BGPCEP

Release master

OpenDaylight Project

Jul 17, 2020

CONTENTS

1 Developer Guides 3

2 User Guides 19

This documentation provides critical information needed to help you write code for the BGPCEP project.

CHAPTER

ONE

DEVELOPER GUIDES

1.1 BGP Developer Guide

1.1.1 Overview

This section provides an overview of the `odl-bgpcep-bgp-all` Karaf feature. This feature will install everything needed for BGP (Border Gateway Protocol) from establishing the connection, storing the data in RIBs (Route Information Base) and displaying data in network-topology overview.

1.1.2 BGP Architecture

Each feature represents a module in the BGPCEP codebase. The following diagram illustrates how the features are related.

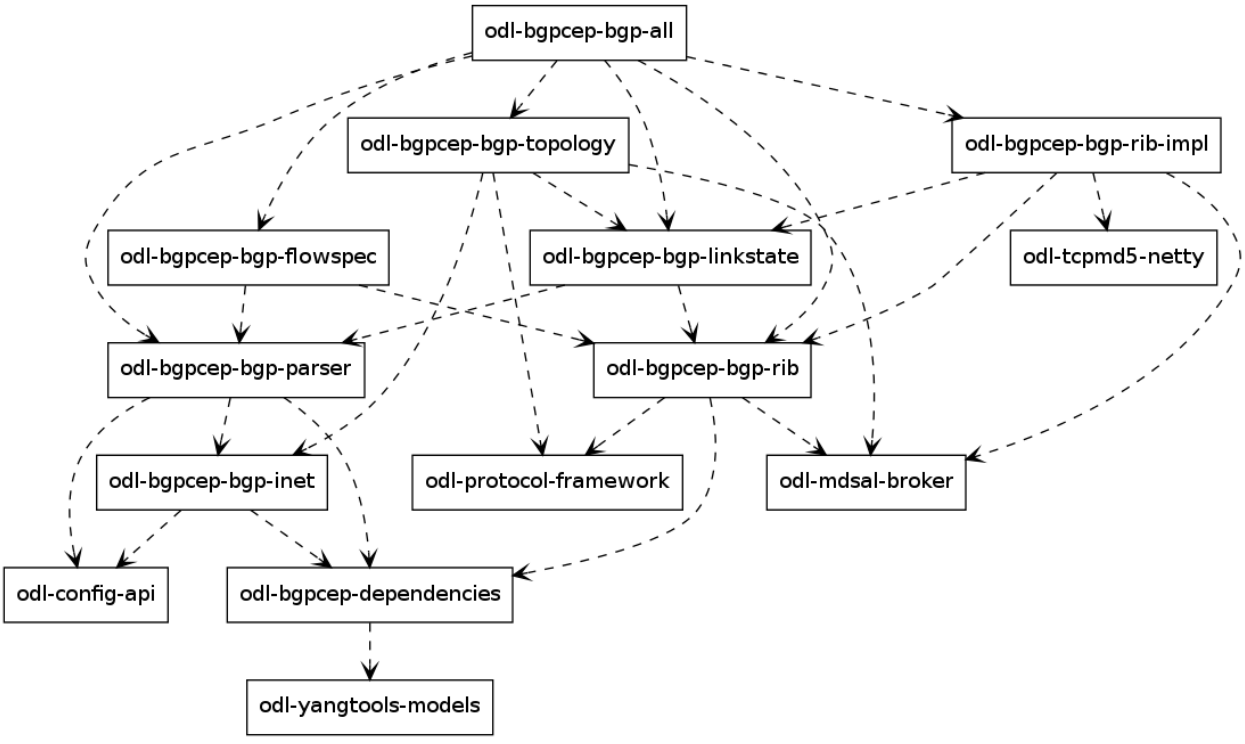


Fig. 1: BGP Dependency Tree

1.1.3 Key APIs and Interfaces

BGP concepts

This module contains the base BGP concepts contained in RFC 4271, RFC 4760, RFC 4456, RFC 1997 and RFC 4360.

All the concepts are described in one yang model: bgp-types.yang.

Outside generated classes, there is just one class NextHopUtil that contains methods for serializing and parsing NextHop.

BGP parser

Base BGP parser includes messages and attributes from RFC 4271, RFC 4760, RFC 1997 and RFC 4360.

API module defines BGP messages in YANG.

IMPL module contains actual parsers and serializers for BGP messages and Activator class

SPI module contains helper classes needed for registering parsers into activators

Registration

All parsers and serializers need to be registered into the *Extension provider*. This *Extension provider* is configured in initial configuration of the parser-spi module (`31-bgp.xml`).

```
<module>
 <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:bgp:parser:spi">
↪prefix:bgp-extensions-impl</type>
 <name>global-bgp-extensions</name>
 <extension>
  <type xmlns:bgpspi="urn:opendaylight:params:xml:ns:yang:controller:bgp:parser:spi">
↪bgpspi:extension</type>
  <name>base-bgp-parser</name>
 </extension>
 <extension>
  <type xmlns:bgpspi="urn:opendaylight:params:xml:ns:yang:controller:bgp:parser:spi">
↪bgpspi:extension</type>
  <name>bgp-linkstate</name>
 </extension>
</module>
```

- *base-bgp-parser* - will register parsers and serializers implemented in the bgp-parser-impl module
- *bgp-linkstate* - will register parsers and serializers implemented in the bgp-linkstate module

The bgp-linkstate module is a good example of a BGP parser extension.

The configuration of bgp-parser-spi specifies one implementation of *Extension provider* that will take care of registering mentioned parser extensions: SimpleBGPExtensionProviderContext. All registries are implemented in package bgp-parser-spi.

Serializing

The serializing of BGP elements is mostly done in the same way as in *PCEP*, the only exception is the serialization of path attributes, which is described here. Path attributes are different from any other BGP element, as path attributes don't implement one common interface, but this interface contains getters for individual path attributes (this structure is because update message can contain exactly one instance of each path attribute). This means, that a given *PathAttributes* object, you can only get to the specific type of the path attribute through checking its presence. Therefore method *serialize()* in *AttributeRegistry*, won't look up the registered class, instead it will go through the registrations and offer this object to the each registered parser. This way the object will be passed also to serializers unknown to module bgp-parser, for example to LinkstateAttributeParser. RFC 4271 recommends ordering path attributes, hence the serializers are ordered in a list as they are registered in the *Activator*. In other words, this is the only case, where registration ordering matters.

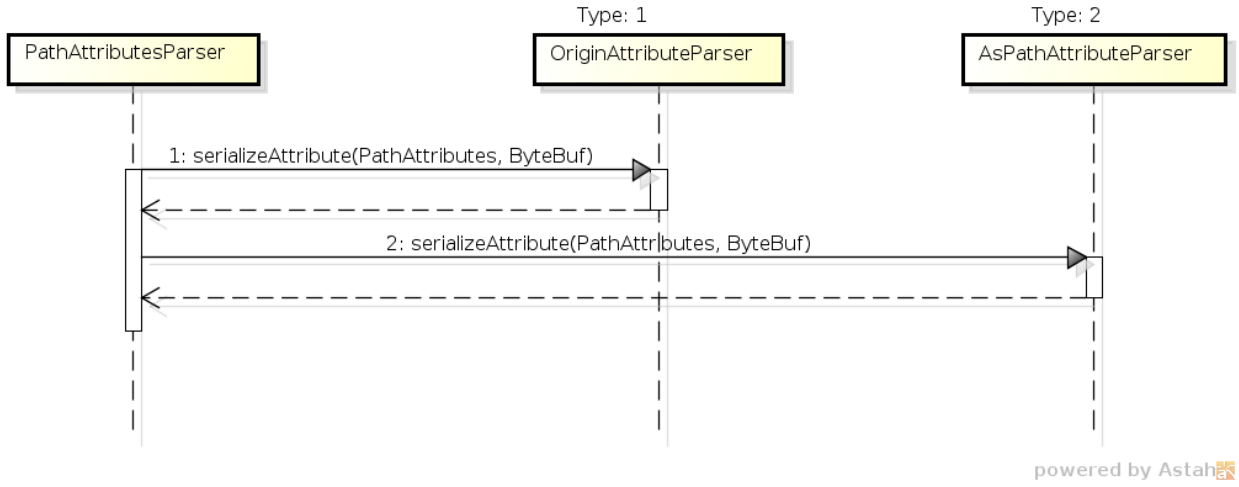


Fig. 2: PathAttributesSerialization

serialize() method in each Path Attribute parser contains check for presence of its attribute in the PathAttributes object, which simply returns, if the attribute is not there:

```
if (pathAttributes.getAtomicAggregate() == null) {
    return;
}
//continue with serialization of Atomic Aggregate
```

1.1.4 BGP RIB

The BGP RIB module can be divided into two parts:

- BGP listener and speaker session handling
- RIB handling.

Session handling

`31-bgp.xml` defines only bgp-dispatcher and the parser it should be using (global-bgp-extensions).

```
<module>
 <type>prefix:bgp-dispatcher-impl</type>
 <name>global-bgp-dispatcher</name>
 <bgp-extensions>
  <type>bgpspi:extensions</type>
  <name>global-bgp-extensions</name>
 </bgp-extensions>
 <boss-group>
  <type>netty:netty-threadgroup</type>
  <name>global-boss-group</name>
 </boss-group>
 <worker-group>
  <type>netty:netty-threadgroup</type>
  <name>global-worker-group</name>
 </worker-group>
</module>
```

For user configuration of BGP, check User Guide.

Synchronization

Synchronization is a phase, where upon connection, a BGP speaker sends all available data about topology to its new client. After the whole topology has been advertised, the synchronization is over. For the listener, the synchronization is over when the RIB receives End-of-RIB (EOR) messages. There is a special EOR message for each AFI (Address Family Identifier).

- IPv4 EOR is an empty Update message.
- Ipv6 EOR is an Update message with empty MP_UNREACH attribute where AFI and SAFI (Subsequent Address Family Identifier) are set to Ipv6. OpenDaylight also supports EOR for IPv4 in this format.
- Linkstate EOR is an Update message with empty MP_UNREACH attribute where AFI and SAFI are set to Linkstate.

For BGP connections, where both peers support graceful restart, the EORs are sent by the BGP speaker and are redirected to RIB, where the specific AFI/SAFI table is set to *true*. Without graceful restart, the messages are generated by OpenDaylight itself and sent after second keepalive for each AFI/SAFI. This is done in BGPSynchronization.

Peers

BGPPeer has various meanings. If you configure BGP listener, *BGPPeer* represents the BGP listener itself. If you are configuring BGP speaker, you need to provide a list of peers, that are allowed to connect to this speaker. Unknown peer represents, in this case, a peer that is allowed to be refused. *BGPPeer* represents in this case peer, that is supposed to connect to your speaker. *BGPPeer* is stored in BGPPeerRegistry. This registry controls the number of sessions. Our strict implementation limits sessions to one per peer.

ApplicationPeer is a special case of peer, that has it's own RIB. This RIB is populated from RESTCONF. The RIB is synchronized with default BGP RIB. Incoming routes to the default RIB are treated in the same way as they were from a BGP peer (speaker or listener) in the network.

RIB handling

RIB (Route Information Base) is defined as a concept in RFC 4271. RFC does not define how it should be implemented. In our implementation, the routes are stored in the MD-SAL datastore. There are four supported routes - *Ipv4Routes*, *Ipv6Routes*, *LinkstateRoutes* and *FlowspecRoutes*.

Each route type needs to provide a RIBSupport.java implementation. *RIBSupport* tells RIB how to parse binding-aware data (BGP Update message) to binding-independent (datastore format).

Following picture describes the data flow from BGP message that is sent to *BGPPeer* to datastore and various types of RIB.

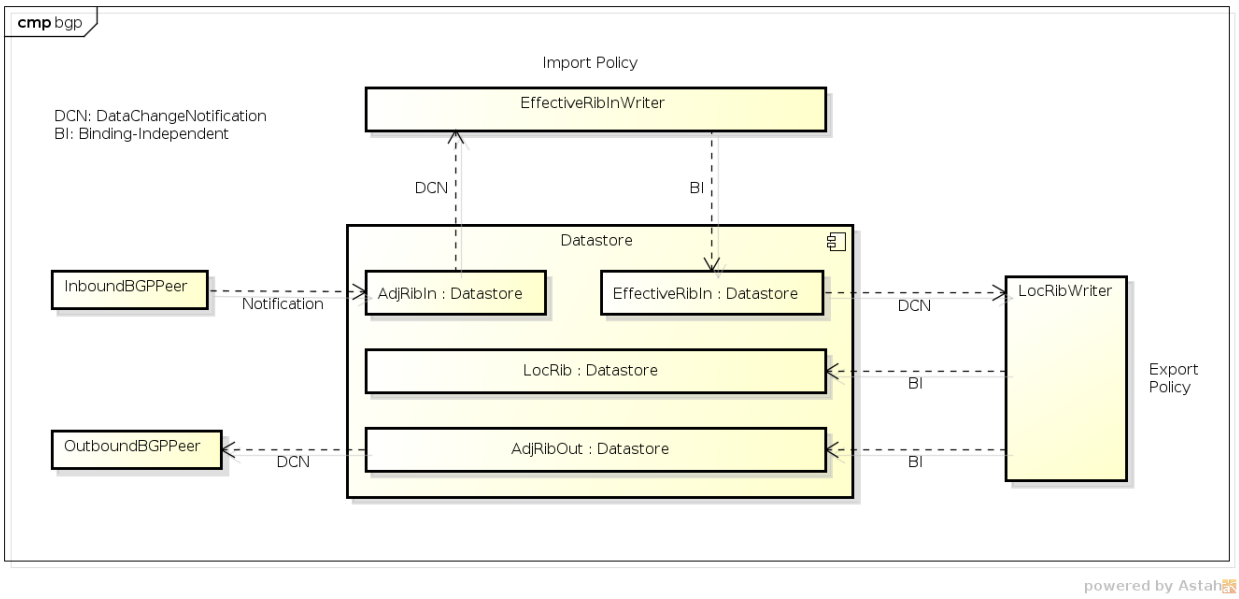


Fig. 3: RIB

AdjRibInWriter - represents the first step in putting data to datastore. This writer is notified whenever a peer receives an Update message. The message is transformed into binding-independent format and pushed into datastore to *adj-rib-in*. This RIB is associated with a peer.

EffectiveRibInWriter - this writer is notified whenever *adj-rib-in* is updated. It applies all configured import policies to the routes and stores them in *effective-rib-in*. This RIB is also associated with a peer.

LocRibWriter - this writer is notified whenever **any** *effective-rib-in* is updated (in any peer). Performs best path selection filtering and stores the routes in *loc-rib*. It also determines which routes need to be advertised and fills in *adj-rib-out* that is per peer as well.

AdjRibOutListener - listens for changes in *adj-rib-out*, transforms the routes into BGPUpdate messages and sends them to its associated peer.

1.1.5 BGP inet

This module contains only one YANG model bgp-inet.yang that summarizes the ipv4 and ipv6 extensions to RIB routes and BGP messages.

1.1.6 BGP flowspec

BGP flowspec is a module that implements RFC 5575 for IPv4 AFI and draft-ietf-idr-flow-spec-v6-06 for IPv6 AFI. The RFC defines an extension to BGP in form of a new subsequent address family, NLRI and extended communities. All of those are defined in the bgp-flowspec.yang model. In addition to generated sources, the module contains parsers for newly defined elements and RIBSupport for flowspec-routes. The route key of flowspec routes is a string representing human-readable flowspec request.

1.1.7 BGP linkstate

BGP linkstate is a module that implements draft-ietf-idr-ls-distribution version 04. The draft defines an extension to BGP in form of a new address family, subsequent address family, NLRI and path attribute. All of those are defined in the bgp-linkstate.yang model. In addition to generated sources, the module contains LinkstateAttributeParser, LinkstateNlriParser, activators for both, parser and RIB, and RIBSupport handler for linkstate address family. As each route needs a key, in case of linkstate, the route key is defined as a binary string, containing all the NLRI serialized to byte format. The BGP linkstate extension also supports distribution of MPLS TE state as defined in draft-ietf-idr-te-lsp-distribution-03, extension for Segment Routing draft-gredler-idr-bgp-ls-segment-routing-ext-00 and Segment Routing Egress Peer Engineering draft-ietf-idr-bgpls-segment-routing-epe-02.

1.1.8 BGP labeled-unicast

BGP labeled unicast is a module that implements RFC 3107. The RFC defines an extension to the BGP MP to carry Label Mapping Information as a part of the NLRI. The AFI indicates, as usual, the address family of the associated route. The fact that the NLRI contains a label is indicated by using SAFI value 4. All of those are defined in bgp-labeled-unicast.yang model. In addition to the generated sources, the module contains new NLRI codec and RIBSupport. The route key is defined as a binary, where whole NLRI information is encoded.

1.1.9 BGP topology provider

BGP data besides RIB, is stored in network-topology view. The format of how the data is displayed there conforms to draft-clemm-netmod-yang-network-topo.

1.1.10 API Reference Documentation

Javadocs are generated while creating mvn:site and they are located in target/ directory in each module.

1.2 BGP Monitoring Protocol Developer Guide

1.2.1 Overview

This section provides an overview of **feature odl-bgpcep-bmp**. This feature will install everything needed for BMP (BGP Monitoring Protocol) including establishing the connection, processing messages, storing information about monitored routers, peers and their Adj-RIB-In (unprocessed routing information) and Post-Policy Adj-RIB-In and displaying data in BGP RIBs overview. The OpenDaylight BMP plugin plays the role of a monitoring station.

1.2.2 Key APIs and Interfaces

Session handling

32-bmp.xml defines only bmp-dispatcher the parser should be using (global-bmp-extensions).

```
<module>
 <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:bmp:impl">
↪prefix:bmp-dispatcher-impl</type>
 <name>global-bmp-dispatcher</name>
  <bmp-extensions>
   <type xmlns:bmp-spi="urn:opendaylight:params:xml:ns:yang:controller:bmp:spi">bmp-
↪spi:extensions</type>
   <name>global-bmp-extensions</name>
  </bmp-extensions>
  <boss-group>
   <type xmlns:netty="urn:opendaylight:params:xml:ns:yang:controller:netty">
↪netty:netty-threadgroup</type>
   <name>global-boss-group</name>
  </boss-group>
  <worker-group>
   <type xmlns:netty="urn:opendaylight:params:xml:ns:yang:controller:netty">
↪netty:netty-threadgroup</type>
   <name>global-worker-group</name>
 </worker-group>
</module>
```

For user configuration of BMP, check User Guide.

Parser

The base BMP parser includes messages and attributes from https://tools.ietf.org/html/draft-ietf-grow-bmp-15

Registration

All parsers and serializers need to be registered into *Extension provider*. This *Extension provider* is configured in initial configuration of the parser (*32-bmp.xml*).

```
<module>
 <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:bmp:spi">
↪prefix:bmp-extensions-impl</type>
 <name>global-bmp-extensions</name>
 <extension>
```

(continues on next page)

(continued from previous page)

```
  <type xmlns:bmp-spi="urn:opendaylight:params:xml:ns:yang:controller:bmp:spi">bmp-
↪spi:extension</type>
  <name>bmp-parser-base</name>
 </extension>
</module>
```

- *bmp-parser-base* - will register parsers and serializers implemented in bmp-impl module

Parsing

Parsing of BMP elements is mostly done equally to BGP. Some of the BMP messages includes wrapped BGP messages.

BMP Monitoring Station

The BMP application (Monitoring Station) serves as message processor incoming from monitored routers. The processed message is transformed and relevant information is stored. Route information is stored in a BGP RIB data structure.

BMP data is displayed only through one URL that is accessible from the base BMP URL:

`http://<controllerIP>:8181/restconf/operational/bmp-monitor:bmp-monitor <http://<controllerIP>:8181/restconf/operational/bmp-monitor:bmp-monitor>`__

Each Monitor station will be displayed and it may contains multiple monitored routers and peers within:

```
<bmp-monitor xmlns="urn:opendaylight:params:xml:ns:yang:bmp-monitor">
 <monitor>
 <monitor-id>example-bmp-monitor</monitor-id>
  <router>
  <router-id>127.0.0.11</router-id>
   <status>up</status>
   <peer>
    <peer-id>20.20.20.20</peer-id>
    <as>72</as>
    <type>global</type>
    <peer-session>
     <remote-port>5000</remote-port>
     <timestamp-sec>5</timestamp-sec>
     <status>up</status>
     <local-address>10.10.10.10</local-address>
     <local-port>220</local-port>
    </peer-session>
    <pre-policy-rib>
     <tables>
      <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-address-
↪family</afi>
      <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-
↪subsequent-address-family</safi>
      <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
       <ipv4-route>
        <prefix>10.10.10.0/24</prefix>
         <attributes>
          ...
         </attributes>
```

(continues on next page)

(continued from previous page)

```
        </ipv4-route>
       </ipv4-routes>
       <attributes>
        <uptodate>true</uptodate>
       </attributes>
      </tables>
     </pre-policy-rib>
     <address>10.10.10.10</address>
     <post-policy-rib>
      ...
     </post-policy-rib>
     <bgp-id>20.20.20.20</bgp-id>
     <stats>
      <timestamp-sec>5</timestamp-sec>
      <invalidated-cluster-list-loop>53</invalidated-cluster-list-loop>
      <duplicate-prefix-advertisements>16</duplicate-prefix-advertisements>
      <loc-rib-routes>100</loc-rib-routes>
      <duplicate-withdraws>11</duplicate-withdraws>
      <invalidated-as-confed-loop>55</invalidated-as-confed-loop>
      <adj-ribs-in-routes>10</adj-ribs-in-routes>
      <invalidated-as-path-loop>66</invalidated-as-path-loop>
      <invalidated-originator-id>70</invalidated-originator-id>
      <rejected-prefixes>8</rejected-prefixes>
     </stats>
    </peer>
    <name>name</name>
    <description>description</description>
    <info>some info;</info>
   </router>
  </monitor>
</bmp-monitor>
</source>
```

1.2.3 API Reference Documentation

Javadocs are generated while creating mvn:site and they are located in target/ directory in each module.

1.3 PCEP Developer Guide

1.3.1 Overview

This section provides an overview of **feature odl-bgpcep-pcep-all** . This feature will install everything needed for PCEP (Path Computation Element Protocol) including establishing the connection, storing information about LSPs (Label Switched Paths) and displaying data in network-topology overview.

1.3.2 PCEP Architecture

Each feature represents a module in the BGPCEP codebase. The following diagram illustrates how the features are related.

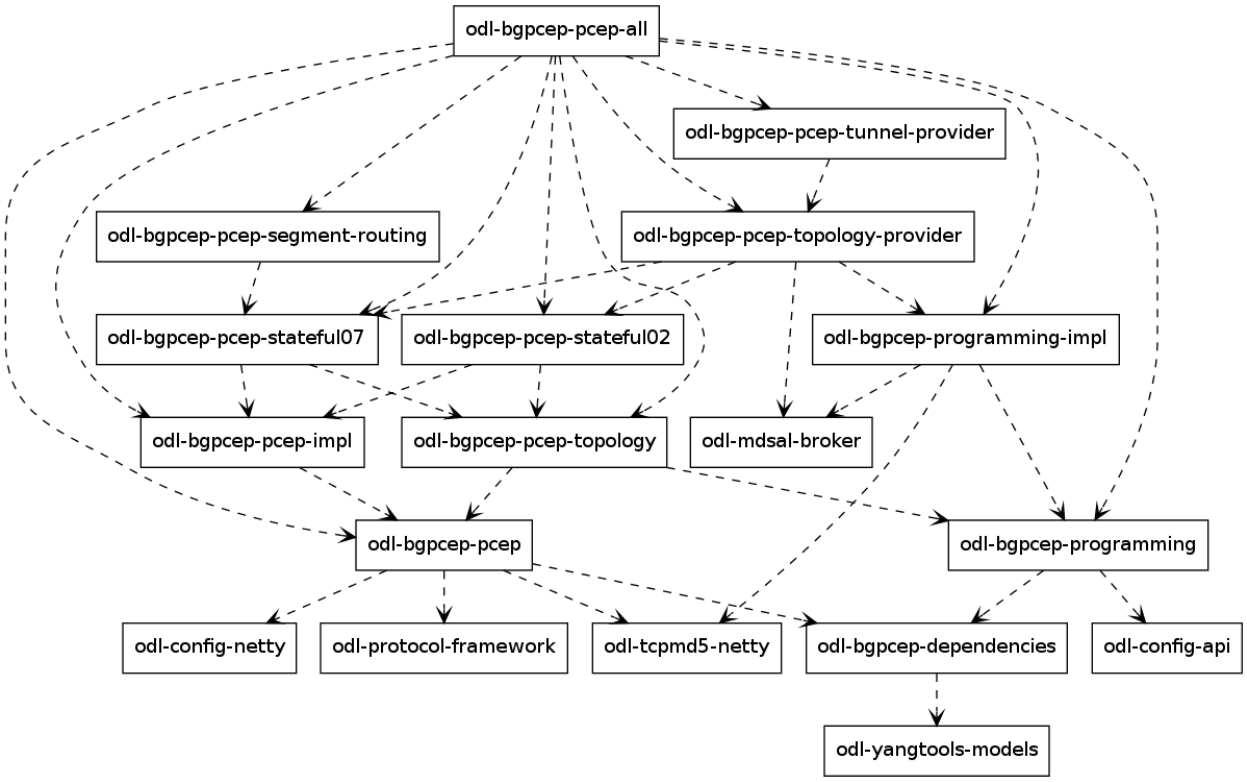


Fig. 4: PCEP Dependency Tree

1.3.3 Key APIs and Interfaces

PCEP

Session handling

32-pcep.xml defines only pcep-dispatcher the parser should be using (global-pcep-extensions), factory for creating session proposals (you can create different proposals for different PCCs (Path Computation Clients)).

```
<module>
 <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:pcep:impl">
↪prefix:pcep-dispatcher-impl</type>
 <name>global-pcep-dispatcher</name>
 <pcep-extensions>
  <type xmlns:pcepspi="urn:opendaylight:params:xml:ns:yang:controller:pcep:spi">
↪pcepspi:extensions</type>
  <name>global-pcep-extensions</name>
 </pcep-extensions>
 <pcep-session-proposal-factory>
  <type xmlns:pcep="urn:opendaylight:params:xml:ns:yang:controller:pcep">pcep:pcep-
↪session-proposal-factory</type>
```

(continues on next page)

(continued from previous page)

```
  <name>global-pcep-session-proposal-factory</name>
 </pcep-session-proposal-factory>
 <boss-group>
  <type xmlns:netty="urn:opendaylight:params:xml:ns:yang:controller:netty">
↪netty:netty-threadgroup</type>
  <name>global-boss-group</name>
 </boss-group>
 <worker-group>
  <type xmlns:netty="urn:opendaylight:params:xml:ns:yang:controller:netty">
↪netty:netty-threadgroup</type>
  <name>global-worker-group</name>
 </worker-group>
</module>
```

For user configuration of PCEP, check User Guide.

Parser

The base PCEP parser includes messages and attributes from RFC5441, RFC5541, RFC5455, RFC5557 and RFC5521.

Registration

All parsers and serializers need to be registered into *Extension provider*. This *Extension provider* is configured in initial configuration of the parser-spi module (*32-pcep.xml*).

```
<module>
 <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:pcep:spi">
↪prefix:pcep-extensions-impl</type>
 <name>global-pcep-extensions</name>
 <extension>
  <type xmlns:pcepspi="urn:opendaylight:params:xml:ns:yang:controller:pcep:spi">
↪pcepspi:extension</type>
  <name>pcep-parser-base</name>
 </extension>
 <extension>
  <type xmlns:pcepspi="urn:opendaylight:params:xml:ns:yang:controller:pcep:spi">
↪pcepspi:extension</type>
  <name>pcep-parser-ietf-stateful07</name>
 </extension>
 <extension>
  <type xmlns:pcepspi="urn:opendaylight:params:xml:ns:yang:controller:pcep:spi">
↪pcepspi:extension</type>
  <name>pcep-parser-ietf-initiated00</name>
 </extension>
 <extension>
  <type xmlns:pcepspi="urn:opendaylight:params:xml:ns:yang:controller:pcep:spi">
↪pcepspi:extension</type>
  <name>pcep-parser-sync-optimizations</name>
 </extension>
</module>
```

- *pcep-parser-base* - will register parsers and serializers implemented in pcep-impl module
- *pcep-parser-ietf-stateful07* - will register parsers and serializers of draft-ietf-pce-stateful-pce-07 implementation

- *pcep-parser-ietf-initiated00* - will register parser and serializer of draft-ietf-pce-pce-initiated-lsp-00 implementation
- *pcep-parser-sync-optimizations* - will register parser and serializers of draft-ietf-pce-stateful-sync-optimizations-03 implementation

Stateful07 module is a good example of a PCEP parser extension.

Configuration of PCEP parsers specifies one implementation of *Extension provider* that will take care of registering mentioned parser extensions: SimplePCEPExtensionProviderContext. All registries are implemented in package pcep-spi.

Parsing

Parsing of PCEP elements is mostly done equally to BGP, the only exception is message parsing, that is described here.

In BGP messages, parsing of first-level elements (path-attributes) can be validated in a simple way, as the attributes should be ordered chronologically. PCEP, on the other hand, has a strict object order policy, that is described in RBNF (Routing Backus-Naur Form) in each RFC. Therefore the algorithm for parsing here is to parse all objects in order as they appear in the message. The result of parsing is a list of *PCEPObjects*, that is put through validation. *validate()* methods are present in each message parser. Depending on the complexity of the message, it can contain either a simple condition (checking the presence of a mandatory object) or a full state machine.

In addition to that, PCEP requires sending error message for each documented parsing error. This is handled by creating an empty list of messages *errors* which is then passed as argument throughout whole parsing process. If some parser encounters *PCEPDocumentedException*, it has the duty to create appropriate PCEP error message and add it to this list. In the end, when the parsing is finished, this list is examined and all messages are sent to peer.

Better understanding provides this sequence diagram:

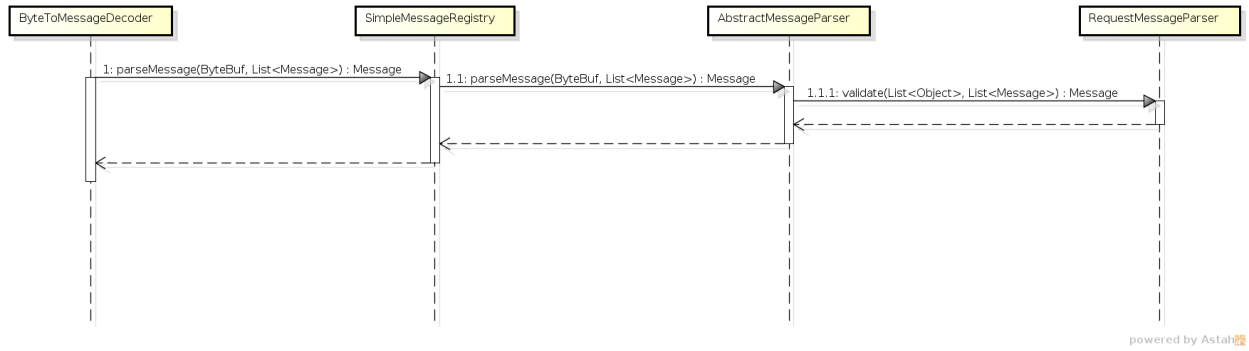


Fig. 5: Parsing

PCEP IETF stateful

This section summarizes module pcep-ietf-stateful07. The term *stateful* refers to draft-ietf-pce-stateful-pce and draft-ietf-pce-pce-initiated-lsp in versions draft-ietf-pce-stateful-pce-07 with draft-ietf-pce-pce-initiated-lsp-00.

We will upgrade our implementation, when the stateful draft gets promoted to RFC.

The stateful module is implemented as extensions to pcep-base-parser. The stateful draft declared new elements as well as additional fields or TLVs (type,length,value) to known objects. All new elements are defined in yang models, that contain augmentations to elements defined in pcep-types.yang. In the case of extending known elements, the *Parser* class merely extends the base class and overrides necessary methods as shown in following diagram:

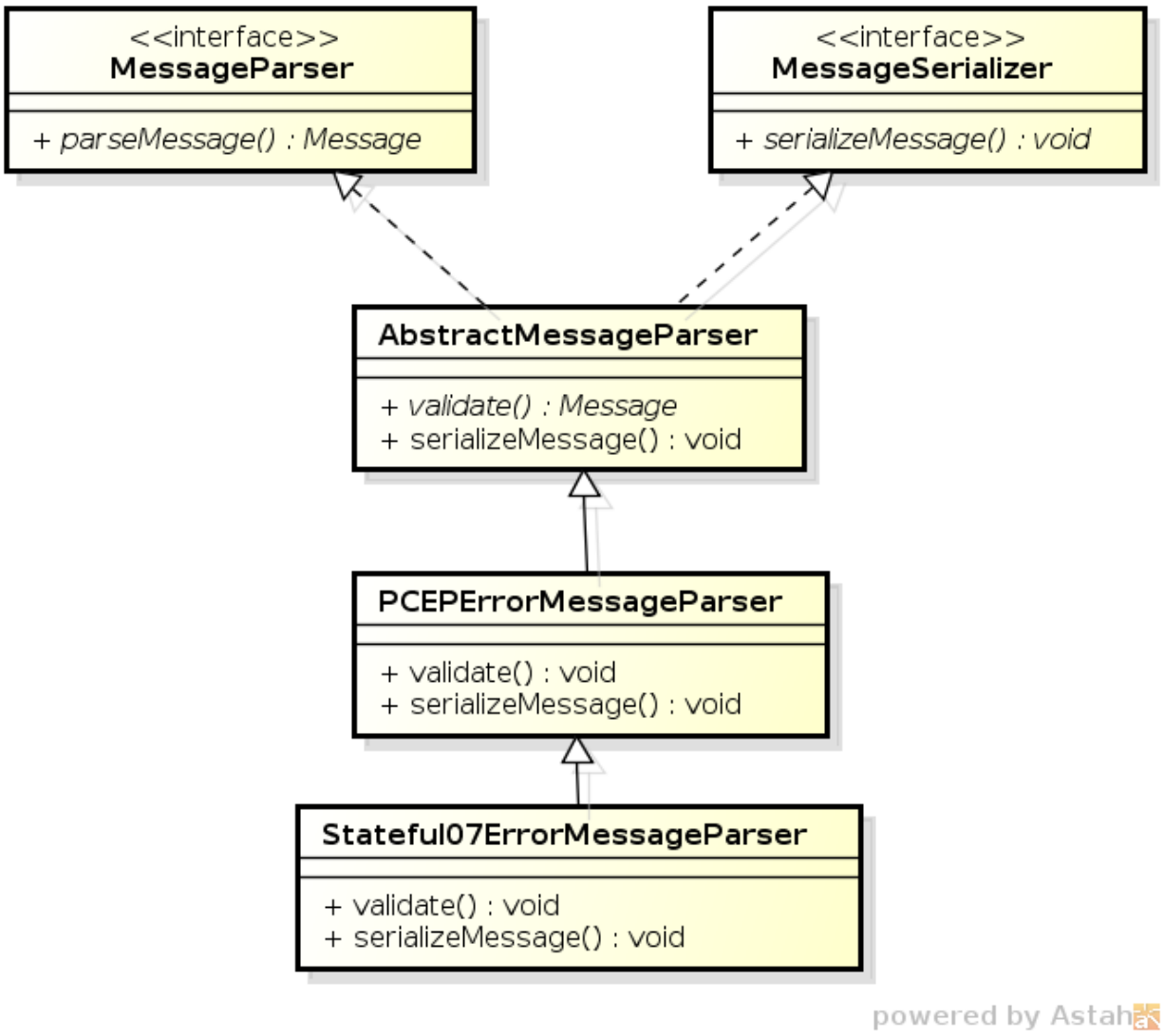


Fig. 6: Extending existing parsers

All parsers (including those for newly defined PCEP elements) have to be registered via the *Activator* class. This class is present in both modules.

In addition to parsers, the stateful module also introduces additional session proposal. This proposal includes new fields defined in stateful drafts for Open object.

PCEP segment routing (SR)

PCEP Segment Routing is an extension of base PCEP and pcep-ietf-stateful-07 extension. The pcep-segment-routing module implements draft-ietf-pce-segment-routing-01.

The extension brings new SR-ERO (Explicit Route Object) and SR-RRO (Reported Route Object) subobject composed of SID (Segment Identifier) and/or NAI (Node or Adjacency Identifier). The segment Routing path is carried in the ERO and RRO object, as a list of SR-ERO/SR-RRO subobjects in an order specified by the user. The draft defines new TLV - SR-PCE-CAPABILITY TLV, carried in PCEP Open object, used to negotiate Segment Routing ability.

The yang models of subobject, SR-PCE-CAPABILITY TLV and appropriate augmentations are defined in odl-pcep-segment-routing.yang.
The pcep-segment-routing module includes parsers/serializers for new subobject (SrEroSubobjectParser) and TLV (SrPceCapabilityTlvParser).

The pcep-segment-routing module implements draft-ietf-pce-lsp-setup-type-01, too. The draft defines new TLV - Path Setup Type TLV, which value indicate path setup signaling technique. The TLV may be included in RP(Request Parameters)/SRP(Stateful PCE Request Parameters) object. For the default RSVP-TE (Resource Reservation Protocol), the TLV is omitted. For Segment Routing, PST = 1 is defined.

The Path Setup Type TLV is modeled with yang in module pcep-types.yang. A parser/serializer is implemented in PathSetupTypeTlvParser and it is overriden in segment-routing module to provide the aditional PST.

PCEP Synchronization Procedures Optimization

Optimizations of Label Switched Path State Synchronization Procedures for a Stateful PCE draft-ietf-pce-stateful-sync-optimizations-03 specifies following optimizations for state synchronization and the corresponding PCEP procedures and extensions:

- **State Synchronization Avoidance:** To skip state synchronization if the state has survived and not changed during session restart.
- **Incremental State Synchronization:** To do incremental (delta) state synchronization when possible.
- **PCE-triggered Initial Synchronization:** To let PCE control the timing of the initial state synchronization. The capability can be applied to both full and incremental state synchronization.
- **PCE-triggered Re-synchronization:** To let PCE re-synchronize the state for sanity check.

PCEP Topology

PCEP data is displayed only through one URL that is accessible from the base network-topology URL:

http://localhost:8181/restconf/operational/network-topology:network-topology/topology/pcep-topology

Each PCC will be displayed as a node:

```
<node>
 <path-computation-client>
  <ip-address>42.42.42.42</ip-address>
  <state-sync>synchronized</state-sync>
  <stateful-tlv>
   <stateful>
    <initiation>true</initiation>
    <lsp-update-capability>true</lsp-update-capability>
   </stateful>
  </stateful-tlv>
 </path-computation-client>
 <node-id>pcc://42.42.42.42</node-id>
</node>
</source>
```

If some tunnels are configured on the network, they would be displayed on the same page, within a node that initiated the tunnel:

```
<node>
 <path-computation-client>
  <state-sync>synchronized</state-sync>
  <stateful-tlv>
   <stateful>
    <initiation>true</initiation>
    <lsp-update-capability>true</lsp-update-capability>
   </stateful>
  </stateful-tlv>
  <reported-lsp>
   <name>foo</name>
   <lsp>
    <operational>down</operational>
    <sync>false</sync>
    <ignore>false</ignore>
    <plsp-id>1</plsp-id>
    <create>false</create>
    <administrative>true</administrative>
    <remove>false</remove>
    <delegate>true</delegate>
    <processing-rule>false</processing-rule>
    <tlvs>
    <lsp-identifiers>
      <ipv4>
       <ipv4-tunnel-sender-address>43.43.43.43</ipv4-tunnel-sender-address>
       <ipv4-tunnel-endpoint-address>0.0.0.0</ipv4-tunnel-endpoint-address>
       <ipv4-extended-tunnel-id>0.0.0.0</ipv4-extended-tunnel-id>
      </ipv4>
      <tunnel-id>0</tunnel-id>
      <lsp-id>0</lsp-id>
     </lsp-identifiers>
     <symbolic-path-name>
```

(continues on next page)

(continued from previous page)

```
      <path-name>Zm9v</path-name>
     </symbolic-path-name>
    </tlvs>
   </lsp>
  </reported-lsp>
  <ip-address>43.43.43.43</ip-address>
 </path-computation-client>
 <node-id>pcc://43.43.43.43</node-id>
</node>
```

Note that, the <*path-name*> tag displays tunnel name in Base64 encoding.

1.3.4 API Reference Documentation

Javadocs are generated while creating mvn:site and they are located in target/ directory in each module.

CHAPTER

TWO

USER GUIDES

2.1 BGP User Guide

This guide contains information on how to use OpenDaylight Border Gateway Protocol (BGP) plugin. The user should learn about BGP basic concepts, supported capabilities, configuration and usage.

2.1.1 Overview

This section provides high-level overview of the Border Gateway Protocol, OpenDaylight implementation and BGP usage in SDN era.

Contents

- *Border Gateway Protocol*
- *BGP in SDN*
- *OpenDaylight BGP plugin*

Border Gateway Protocol

The Border Gateway Protocol (BGP) is an inter-Autonomous System (AS) routing protocol. The primary role of the BGP is an exchange of routes among other BGP systems. The route is an unit of information which pairs destination (IP address prefix) with attributes to the path with the destination. One of the most interesting attributes is a list of ASes that the route traversed - essential when avoiding loop routing. Advertised routes are stored in the Routing Information Bases (RIBs). Routes are later used to forward packets, stored in Routing Table for this purpose. The main advantage of the BGP over other routing protocols is its scalability, thus it has become the standardized Internet routing protocol (Internet is a set of ASes).

BGP in SDN

However BGP evolved long time before SDN was born, it plays a significant role in many SDN use-cases. Also, continuous evolution of the protocol brings extensions that are very well suited for SDN. Nowadays, BGP can carry various types of routing information - L3VPN, L2VPN, IP multicast, linkstate, etc. Here is a brief list of software-based/legacy-network technologies where BGP-based SDN solution get into an action:

- SDN WAN - WAN orchestration and optimization
- SDN router - Turns switch into an Internet router
- Virtual Route Reflector - High-performance server-based BGP Route Reflector
- SDX - A Software Defined Internet Exchange controller
- Large-Scale Data Centers - BGP Data Center Routing, MPLS/SR in DCs, DC interconnection
- DDoS mitigation - Traffic Filtering distribution with BGP

OpenDaylight BGP plugin

The OpenDaylight controller provides an implementation of BGP (RFC 4271) as a south-bound protocol plugin. The implementation renders all basic *BGP speaker capabilities*:

- inter/intra-AS peering
- routes advertising
- routes originating
- routes storage

The plugin's **north-bound API** (`REST/Java`) provides to user:

- fully dynamic runtime standardized BGP configuration
- read-only access to all RIBs
- read-write programmable RIBs
- read-only reachability/linkstate topology view

Note: The BGP plugin is NOT a virtual router - does not construct Routing Tables, nor forward traffic.

2.1.2 List of supported capabilities

In addition to the base protocol implementation, the plugin provides many extensions to BGP, all based on IETF standards.

- RFC4271 - A Border Gateway Protocol 4 (BGP-4)
- RFC4456 - BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)
- RFC1997 - BGP Communities Attribute
- RFC4360 - BGP Extended Communities Attribute
- RFC4486 - Subcodes for BGP Cease Notification Message
- RFC5492 - Capabilities Advertisement with BGP-4
- RFC5004 - Avoid BGP Best Path Transitions from One External to Another

- RFC6286 - Autonomous-System-Wide Unique BGP Identifier for BGP-4
- RFC6793 - BGP Support for Four-Octet Autonomous System (AS) Number Space
- RFC7311 - The Accumulated IGP Metric Attribute for BGP
- RFC5668 - 4-Octet AS Specific BGP Extended Community
- draft-ietf-idr-link-bandwidth - BGP Link Bandwidth Extended Community
- draft-ietf-idr-bgp-extended-messages - Extended Message support for BGP
- **RFC4760 - Multiprotocol Extensions for BGP-4**
  - **RFC7752 - North-Bound Distribution of Link-State and TE Information using BGP**
    - draft-gredler-idr-bgp-ls-segment-routing-ext - BGP Link-State extensions for Segment Routing
    - draft-ietf-idr-bgpls-segment-routing-epe - Segment Routing Egress Peer Engineering BGP-LS Extensions
  - **RFC5575 - Dissemination of Flow Specification Rules**
    - RFC7674 - Clarification of the Flowspec Redirect Extended Community
    - draft-ietf-idr-flow-spec-v6 - Dissemination of Flow Specification Rules for IPv6
    - draft-ietf-idr-flowspec-redirect-ip - BGP Flow-Spec Redirect to IP Action
  - **RFC3107 - Carrying Label Information in BGP-4**
    - draft-ietf-idr-bgp-prefix-sid - Segment Routing Prefix SID extensions for BGP
  - **RFC4364 - BGP/MPLS IP Virtual Private Networks (VPNs)**
    - RFC4659 - BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN
  - **RFC6513 - Multicast in MPLS/BGP IP VPNs (VPNs)**
    - RFC6514 - BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs
    - RFC6515 - IPv4 and IPv6 Infrastructure Addresses in BGP Updates for Multicast VPN
    - RFC4684 - Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)
  - **RFC7432 - BGP MPLS-Based Ethernet VPN**
    - draft-ietf-bess-evpn-overlay - A Network Virtualization Overlay Solution using EVPN
    - draft-ietf-bess-evpn-vpws - VPWS support in EVPN
    - draft-sajassi-bess-evpn-vpws-fxc-01 - EVPN VPWS Flexible Cross-Connect Service
- RFC7911 - Advertisement of Multiple Paths in BGP
- RFC2918 - Route Refresh Capability for BGP-4
- **RFC4724 - Graceful Restart Mechanism for BGP**
  - draft-uttaro-idr-bgp-persistence-04 - Support for Long-lived BGP Graceful Restart
- RFC7606 - Revised Error Handling for BGP UPDATE Messages

2.1.3 Running BGP

This section explains how to install BGP plugin.

1. Install BGP feature - `odl-bgpcep-bgp`. Also, for sake of this sample, it is required to install RESTCONF. In the Karaf console, type command:

```
feature:install odl-restconf odl-bgpcep-bgp
```

2. The BGP plugin contains a default configuration, which is applied after the feature starts up. One instance of BGP plugin is created (named *example-bgp-rib*), and its presence can be verified via REST:

   URL: `/restconf/operational/bgp-rib:bgp-rib`

   Method: `GET`

   Response Body:

```
<bgp-rib xmlns="urn:opendaylight:params:xml:ns:yang:bgp-rib">
   <rib>
       <id>example-bgp-rib</id>
       <loc-rib>
       ....
       </loc-rib>
   </rib>
</bgp-rib>
```

2.1.4 Basic Configuration & Concepts

The following section shows how to configure BGP basics, how to verify functionality and presents essential components of the plugin. Next samples demonstrate the plugin's runtime configuration capability. It shows the way to configure the plugin via REST, using standardized OpenConfig BGP APIs.

Contents

- *Configuration*
- *BGP RIB API*
- *BGP pipeline*
- *References*

Configuration

Protocol Configuration

As a first step, a new protocol instance needs to be configured. It is a very basic configuration conforming with RFC4271.

Note: RIB policy must already be configured and present before configuring the protocol.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
             <apply-policy>
                 <config>
                     <default-export-policy>REJECT-ROUTE</default-export-policy>
                     <default-import-policy>REJECT-ROUTE</default-import-policy>
                     <import-policy>default-odl-import-policy</import-policy>
                     <export-policy>default-odl-export-policy</export-policy>
                 </config>
             </apply-policy>
        </global>
    </bgp>
</protocol>
```

@line 2: The unique protocol instance identifier.

@line 7: BGP Identifier of the speaker.

@line 8: Local autonomous system number of the speaker. Note that, OpenDaylight BGP implementation supports four-octet AS numbers only.

@line 14: Default ODL Import Policy.

@line 15: Default ODL Export Policy.

The new instance presence can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example`

Method: `GET`

Response Body:

```
<rib xmlns="urn:opendaylight:params:xml:ns:yang:bgp-rib">
    <id>bgp-example</id>
    <loc-rib>
        <tables>
            <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-
↪address-family</afi>
            <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-
↪subsequent-address-family</safi>
            <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet"></ipv4-
↪routes>
            <attributes>
                <uptodate>true</uptodate>
            </attributes>
        </tables>
```

(continues on next page)

(continued from previous page)

```
    </loc-rib>
</rib>
```

@line 3: Loc-RIB - Per-protocol instance RIB, which contains the routes that have been selected by local BGP speaker's decision process.

@line 4: The BGP-4 supports carrying IPv4 prefixes, such routes are stored in *ipv4-address-family/unicast-subsequent-address-family* table.

RIB Policy Configuration

The OpenDaylight BGP implementation supports configurable RIB policies that allow the modification of import and export policies.

Note: Default ODL BGP RIB Config Policy is provided. Any config policy to be used by Protocol must be configured and present before than Protocol configuration is added. If policy is reconfigured, protocol must be re configured again.

URL: `/restconf/config/openconfig-routing-policy:routing-policy`

Method: `GET`

Content-Type: `application/xml`

Request Body:

```
 <routing-policy xmlns="http://openconfig.net/yang/routing-policy">
     <defined-sets>
         <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
             <cluster-id-sets xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
                 ...
             </cluster-id-sets>
             <role-sets xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
                 ...
             </role-sets>
             <originator-id-sets xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
                 ...
             </originator-id-sets>
         </bgp-defined-sets>
     </defined-sets>
     <policy-definitions>
         <policy-definition>
             <name>default-odl-export-policy</name>
             <statements>
                 <statement>
                     <name>to-odl-internal</name>
                     <actions>
                         <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
                             ...
                         </bgp-actions>
                     </actions>
                     <conditions>
                         <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy
↪">
```

(continues on next page)

(continued from previous page)

```
                            ...
                        </bgp-conditions>
                    </conditions>
                </statement>
                ...
            </statements>
        </policy-definition>
        <policy-definition>
            <name>default-odl-import-policy</name>
            ...
        </policy-definition>
    </policy-definitions>
</routing-policy>
```

@line 2: BGP defined sets.

@line 15: Policy definitions.

Policy Configuration

Conditions may include multiple match or comparison operations; similarly, actions may consist of a multitude of changes to route attributes or a final disposition regarding the acceptance or rejection of the route.

URL: `/restconf/config/openconfig-routing-policy:routing-policy/openconfig-routing-policy:policy-definitions/`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<policy-definition xmlns="http://openconfig.net/yang/routing-policy">
    <name>odl-policy-example</name>
    <statements>
        <statement>
            <name>reject-all-incoming-routes</name>
            <actions>
                <reject-route/>
            </actions>
            <conditions>
                <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
                    <match-role-set xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
                        <from-role>
                           <role-set>/rpol:routing-policy/rpol:defined-sets/
↪bgppol:bgp-defined-sets/role-sets/role-set[role-set-name="all"]</role-set>
                        </from-role>
                    </match-role-set>
                </bgp-conditions>
            </conditions>
        </statement>
    </statements>
</policy-definition>
```

@line 2: The unique policy instance identifier.

@line 5: Policy Statement Identifier.

@line 7: Actions.

@line 10: BGP Conditions.

The new instance presence can be verified via REST:

URL: `/restconf/config/openconfig-routing-policy:routing-policy/openconfig-routing-policy:policy-definitions/policy-definition/odl-policy-example`

Method: `GET`

Response Body:

```
1  <policy-definition xmlns="http://openconfig.net/yang/routing-policy">
2      <name>odl-policy-example</name>
3      <statements>
4          <statement>
5              <name>reject-all-incoming-routes</name>
6              <actions>
7                  <reject-route></reject-route>
8              </actions>
9              <conditions>
10                 <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
11                     <match-role-set xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
12                         <from-role>
13                             <role-set>/rpol:routing-policy/rpol:defined-sets/
↪bgppol:bgp-defined-sets/role-sets/role-set[role-set-name="all"]</role-set>
14                             <match-set-options>ANY</match-set-options>
15                         </from-role>
16                     </match-role-set>
17                 </bgp-conditions>
18             </conditions>
19         </statement>
20     </statements>
21 </policy-definition>
```

@line 2: Policy definition Identifier.

@line 5: Policy Statement Identifier.

Actions

ODL BGP by default provides support for a group of BGP Actions.

Accept

Default policy to accept the route.

```
1  <actions>
2      <accept-route/>
3  </actions>
```

Reject

Default policy to reject the route.

```
<actions>
    <reject-route/>
</actions>
```

As-path prepend

Action to prepend local AS number to the AS-path

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-as-path-prepend/>
    </bgp-actions>
</actions>
```

Originator Id prepend

Action to prepend Originator Id. In case there is non Originator Id present, local Originator Id is prepend.

- Local

```
<bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
    <set-originator-id-prepend xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
</bgp-actions>
```

- By value

```
<bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
    <set-originator-id-prepend xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
        <originator-id>192.0.2.1</originator-id>
    </set-originator-id-prepend>
</bgp-actions>
```

Cluster Id prepend

Action to prepend local Cluster Id to Cluster Id List.

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-cluster-id-prepend xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
    </bgp-actions>
</actions>
```

Set Route Origin

Set the origin attribute to the specified value.

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-route-origin>IGP</set-route-origin>
    </bgp-actions>
</actions>
```

Set Local Preference

Set the local pref attribute on the route update.

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-local-pref>100</set-local-pref>
    </bgp-actions>
</actions>
```

Set NextHop

Set the next-hop attribute in the route update.

- Local

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-next-hop>SELF</set-next-hop>
    </bgp-actions>
</actions>
```

- By value

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-next-hop>4.5.6.7</set-next-hop>
    </bgp-actions>
</actions>
```

Set MED

Set the med metric attribute in the route update.

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-med>15</set-med>
    </bgp-actions>
</actions>
```

Community set prepend

Action to set the community attributes of the route, along with options to modify how the community is modified.

- Inline

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-community>
            <communities>
                <as-number>65</as-number>
                <semantics>10</semantics>
            </communities>
            <communities>
                <as-number>66</as-number>
                <semantics>11</semantics>
            </communities>
            <options>ADD</options>
        </set-community>
    </bgp-actions>
</actions>
```

@line 3: Set Community.

- By reference

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-community>
            <community-set-ref>
                /rpol:routing-policy/rpol:defined-sets/rpol:community-sets/community-
↪set[community-set-name="community-set-name-example"]
            </community-set-ref>
            <options>ADD</options>
        </set-community>
    </bgp-actions>
</actions>
```

@line 3: Set Community.

@line 5: Community set reference.

@line 7: Options are ADD, REMOVE, REPLACE.

Defined set

```
<defined-sets>
    <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
        <community-sets>
            <community-set>
                <community-set-name>community-set-name-test</community-set-name>
                <communities>
                    <as-number>65</as-number>
                    <semantics>10</semantics>
                </communities>
                <communities>
                    <as-number>66</as-number>
```

(continues on next page)

(continued from previous page)

```
                    <semantics>11</semantics>
                </communities>
            </community-set>
        </community-sets>
    </bgp-defined-sets>
</defined-sets>
```

@line 3: Community set.

Extended Community set action

Action to set the extended community attributes of the route, along with options to modify how the community is modified.

- Inline

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-ext-community>
            <ext-community-member>
                <encapsulation-extended-community>
                    <tunnel-type>vxlan</tunnel-type>
                </encapsulation-extended-community>
            </ext-community-member>
            <ext-community-member>
                <as-4-route-origin-extended-community>
                    <as-4-specific-common>
                        <as-number>65000</as-number>
                        <local-administrator>123</local-administrator>
                    </as-4-specific-common>
                </as-4-route-origin-extended-community>
            </ext-community-member>
            <options>ADD</options>
        </set-ext-community>
    </bgp-actions>
</actions>
```

@line 3: Set Extended Community.

- By reference

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <set-ext-community>
            <ext-community-set-ref>
                /rpol:routing-policy/rpol:defined-sets/rpol:ext-community-sets/ext-
↪community-set[ext-community-set-name="ext-community-set-name-example"]
            </ext-community-set-ref>
            <options>REMOVE</options>
        </set-ext-community>
    </bgp-actions>
</actions>
```

@line 3: Set Extended Community.

@line 5: Extended Community set reference.

@line 7: Options are ADD, REMOVE, REPLACE.

Defined set

```
<defined-sets>
    <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
        <ext-community-sets>
            <ext-community-set>
                <ext-community-set-name>ext-community-set-name-test</ext-community-set-name>
                <ext-community-member>
                    <encapsulation-extended-community>
                        <tunnel-type>vxlan</tunnel-type>
                    </encapsulation-extended-community>
                </ext-community-member>
                <ext-community-member>
                    <as-4-route-origin-extended-community>
                        <as-4-specific-common>
                            <as-number>65000</as-number>
                            <local-administrator>123</local-administrator>
                        </as-4-specific-common>
                    </as-4-route-origin-extended-community>
                </ext-community-member>
            </ext-community-set>
        </ext-community-sets>
    </bgp-defined-sets>
</defined-sets>
```

@line 3: Extendend Community set.

@line 5: Extendend Community set name.

Filter Non transitive attributes

Filters attributes, removing non transitive attributes.

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <non-transitive-attributes-filter xmlns="urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
    </bgp-actions>
</actions>
```

Client Attribute Prepend

Replace attributes per any VPN Route attributes from client Peer, if present.

```
<actions>
    <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
        <client-attribute-prepend xmlns="urn:opendaylight:params:xml:ns:yang:bgp:route:target:constrain"/>
    </bgp-actions>
</actions>
```

Conditions

ODL BGP by default provides support for a group of BGP Conditions.

Match BGP Neighbor Set

```
1  <conditions>
2      <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3          <match-bgp-neighbor-set xmlns=
   ↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
4              <from-neighbor>
5                  <neighbor-set>/rpol:routing-policy/rpol:defined-sets/rpol:neighbor-
   ↪sets/neighbor-set[neighbor-set-name="bgp-neighbor-set-example"]</neighbor-set>
6                  <match-set-options>INVERT</match-set-options>
7              </from-neighbor>
8          </match-bgp-neighbor-set>
9      </bgp-conditions>
10 </conditions>
```

@line 3: Match BGP Neighbor Condition set.

@line 4: Match BGP Neighbor from whom we receive the route.

@line 5: Match BGP Neighbor Set reference.

@line 6: Match Set Options (ANY, INVERT)

```
1  <conditions>
2      <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3          <match-bgp-neighbor-set xmlns=
   ↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
4              <to-neighbor>
5                  <neighbor-set>/rpol:routing-policy/rpol:defined-sets/rpol:neighbor-
   ↪sets/neighbor-set[neighbor-set-name="bgp-neighbor-set-example"]</neighbor-set>
6                  <match-set-options>INVERT</match-set-options>
7              </to-neighbor>
8          </match-bgp-neighbor-set>
9      </bgp-conditions>
10 </conditions>
```

@line 3: Match BGP Neighbor Condition set.

@line 4: Match BGP Neighbor to whom we send the route.

@line 5: Match BGP Neighbor Set reference.

@line 6: Match Set Options (ANY, INVERT)

```
1  <conditions>
2      <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3          <match-bgp-neighbor-set xmlns=
   ↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
4              <from-neighbor>
5                  <neighbor-set>/rpol:routing-policy/rpol:defined-sets/rpol:neighbor-
   ↪sets/neighbor-set[neighbor-set-name="bgp-neighbor-set-example"]</neighbor-set>
6              </from-neighbor>
7              <to-neighbor>
8                  <neighbor-set>/rpol:routing-policy/rpol:defined-sets/rpol:neighbor-
   ↪sets/neighbor-set[neighbor-set-name="bgp-neighbor-set-example"]</neighbor-set>
```

(continues on next page)

(continued from previous page)

```
                <match-set-options>INVERT</match-set-options>
            </to-neighbor>
        </match-bgp-neighbor-set>
    </bgp-conditions>
</conditions>
```

@line 3: Match BGP Neighbor Condition set.

@line 4: Match BGP Neighbor from whom we receive the route.

@line 5: Match BGP Neighbor Set reference.

@line 7: Match BGP Neighbor to whom we send the route.

@line 8: Match BGP Neighbor Set reference.

@line 9: Match Set Options (ANY, INVERT)

Defined set

```
<defined-sets>
    <neighbor-sets>
        <neighbor-set>
            <neighbor-set-name>bgp-neighbor-set-example</neighbor-set-name>
            <neighbor>
                <address>127.0.0.1</address>
            </neighbor>
            <neighbor>
                <address>127.0.0.2</address>
            </neighbor>
        </neighbor-set>
    </neighbor-sets>
</defined-sets>
```

@line 3: Originator Id Set.

@line 5: Originator Id Set name.

Match Originator Id Set

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <match-originator-id-set-condition xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
            <originator-id-set>
                /rpol:routing-policy/rpol:defined-sets/bgppol:bgp-defined-sets/
↪originator-id-sets/originator-id-set[originator-set-name="local-originator-id"]
            </originator-id-set>
            <match-set-options>INVERT</match-set-options>
        </match-originator-id-set-condition>
    </bgp-conditions>
</conditions>
```

@line 3: Match Originator Id Condition set.

@line 5: Match Originator Id Set reference.

@line 7: Match Set Options (ANY, INVERT)

Defined set

```
<defined-sets>
    <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
        <originator-id-sets xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
            <originator-id-set>
                <originator-id-set-name>local-originator-id</originator-id-set-name>
                <local/>
            </originator-id-set>
        </originator-id-sets>
    </bgp-defined-sets>
</defined-sets>
```

@line 3: Originator Id Set.

@line 5: Originator Id Set name.

Match Cluster Id Set

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <match-cluster-id-set-condition xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
            <cluster-id-set>
                /rpol:routing-policy/rpol:defined-sets/bgppol:bgp-defined-sets/
↪cluster-id-sets/cluster-id-set[cluster-set-name="local-cluster-id"]
            </cluster-id-set>
            <match-set-options>INVERT</match-set-options>
        </match-cluster-id-set-condition>
    </bgp-conditions>
</conditions>
```

@line 3: Match Cluster Id Condition set.

@line 5: Match Cluster Id Set reference.

Defined set

```
<defined-sets>
    <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
        <cluster-id-sets xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
            <cluster-id-set>
                <cluster-id-set-name>local-cluster-id</cluster-id-set-name>
                <local/>
            </cluster-id-set>
        </cluster-id-sets>
    </bgp-defined-sets>
</defined-sets>
```

@line 3: Cluster Id Set.

@line 5: Cluster Id Set name.

Match Peer Role Set

```
1  <conditions>
2      <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3          <match-role-set xmlns=
   ↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
4              <from-role>
5                  <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-defined-
   ↪sets/role-sets/role-set[role-set-name="only-ibgp"]</role-set>
6                  <match-set-options>INVERT</match-set-options>
7              </from-role>
8              <to-role>
9                  <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-defined-
   ↪sets/role-sets/role-set[role-set-name="all"]</role-set>
10             <to-role>
11         </match-role-set>
12     </bgp-conditions>
13 </conditions>
```

@line 3: Match Role Set.

@line 5: Match Role Set reference.

@line 6: Match Set Options (ANY, INVERT)

Defined set

```
1  <defined-sets>
2      <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
3          <role-set>
4              <role-set-name>all</role-set-name>
5              <role>ebgp</role>
6              <role>ibgp</role>
7              <role>rr-client</role>
8              <role>internal</role>
9          </role-set>
10         <role-set>
11             <role-set-name>only-ibgp</role-set-name>
12             <role>ibgp</role>
13         </role-set>
14     </bgp-defined-sets>
15 </defined-sets>
```

@line 3: Role Set.

@line 4: Role Set name.

@line 10: Role Set.

@line 11: Role Id Set name.

Match AS Path Set

```
1  <conditions>
2      <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3          <match-as-path-set>
4              <as-path-set>
5                  /rpol:routing-policy/rpol:defined-sets/bgp-pol:bgp-defined-sets/bgp-
   ↪pol:as-path-sets/bgp-pol:as-path-set/[as-path-set-name="as-path-set-example"]
6              </as-path-set>
7              <match-set-options>ANY</match-set-options>
8          </match-as-path-set>
9      </bgp-conditions>
10 </conditions>
```

@line 3: Match AS Path Set.

@line 5: AS Path Set reference.

@line 7: Match Set Option(ANY, ALL, INVERT).

Defined set

```
1  <defined-sets>
2      <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
3          <as-path-sets>
4              <as-path-set>
5                  <as-path-set-name>as-path-set-example</as-path-set-name>
6                  <as-path-set-member>65</as-path-set-member>
7                  <as-path-set-member>64</as-path-set-member>
8                  <as-path-set-member>63</as-path-set-member>
9              </as-path-set>
10         </as-path-sets>
11     </bgp-defined-sets>
12 </defined-sets>
```

@line 4: AS Path Set.

@line 5: AS Path Set name.

@line 6: AS Path set member

Match Community Set

```
1  <conditions>
2      <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3          <match-community-set>
4              <community-set>
5                  /rpol:routing-policy/rpol:defined-sets/rpol:community-sets/community-
   ↪set[community-set-name="community-set-name-example"]
6              </community-set>
7              <match-set-options>ANY</match-set-options>
8          </match-community-set>
9      </bgp-conditions>
10 </conditions>
```

@line 3: Match Community Set.

@line 5: Match Community Set reference.

@line 7: Match Set Option(ANY, ALL, INVERT).

Defined set

```
<defined-sets>
    <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
        <community-sets>
            <community-set>
                <community-set-name>community-set-name-example</community-set-name>
                <communities>
                    <as-number>65</as-number>
                    <semantics>10</semantics>
                </communities>
                <communities>
                    <as-number>66</as-number>
                    <semantics>11</semantics>
                </communities>
            </community-set>
        </community-sets>
    </bgp-defined-sets>
</defined-sets>
```

@line 4: Community Set.

@line 5: Community Set name.

@line 6: Communities.

@line 10: Communities.

Match Extended Community Set

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <match-ext-community-set>
            <ext-community-set>
                /rpol:routing-policy/rpol:defined-sets/rpol:ext-community-sets/ext-
↪community-set[ext-community-set-name="ext-community-set-name-test"]
            </ext-community-set>
            <match-set-options>ANY</match-set-options>
        </match-ext-community-set>
    </bgp-conditions>
</conditions>
```

@line 3: Match Extended Community Set.

@line 5: Match Extended Community Set reference.

@line 7: Match Set Option(ANY, ALL, INVERT).

Defined set

```
<defined-sets>
    <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
        <ext-community-sets>
            <ext-community-set>
                <ext-community-set-name>ext-community-set-name-test</ext-community-set-name>
                <ext-community-member>
                    <encapsulation-extended-community>
                        <tunnel-type>vxlan</tunnel-type>
                    </encapsulation-extended-community>
                </ext-community-member>
                <ext-community-member>
                    <as-4-route-origin-extended-community>
                        <as-4-specific-common>
                            <as-number>65000</as-number>
                            <local-administrator>123</local-administrator>
                        </as-4-specific-common>
                    </as-4-route-origin-extended-community>
                </ext-community-member>
            </ext-community-set>
        </ext-community-sets>
    </bgp-defined-sets>
</defined-sets>
```

@line 4: Extended Community Set.

@line 5: Extended Community Set name.

@line 6: Extended Communities.

@line 11: Extended Communities.

Match in Afi Safi

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <afi-safi-in xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-safi-in>
    </bgp-conditions>
</conditions>
```

@line 3: Afi Safi match.

Match not in Afi Safi

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <afi-safi-not-in xmlns="urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"
            xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-safi-not-in>
        <afi-safi-not-in xmlns="urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"
            xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-UNICAST</afi-safi-not-in>
```

(continues on next page)

(continued from previous page)

```
    </bgp-conditions>
</conditions>
```

@line 3: Afi Safi not in match.

Match As Path Length

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <as-path-length>
            <operator xmlns:x="http://openconfig.net/yang/policy-types">x:attribute-
↪eq</operator>
            <value>2</value>
        </as-path-length>
    </bgp-conditions>
</conditions>
```

@line 3: As Path Length match.

Match Local Pref

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <local-pref-eq>100</local-pref-eq>
    </bgp-conditions>
</conditions>
```

@line 3: Local Preference match.

Match Origin

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <origin-eq>IGP</origin-eq>
    </bgp-conditions>
</conditions>
```

@line 3: Origin match.

Match MED

```
<conditions>
    <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
        <med-eq>100</med-eq>
    </bgp-conditions>
</conditions>
```

@line 3: MED match.

Match Next Hop

```
1 <conditions>
2     <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3         <next-hop-in>192.168.2.2</next-hop-in>
4         <next-hop-in>42.42.42.42</next-hop-in>
5     </bgp-conditions>
6 </conditions>
```

@line 3: Next hop match.

Match VPN Non member

True if Route Targets attributes does not match with any Route Target Contrain advertized per Advertized peer.

```
1 <conditions>
2     <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
3         <vpn-non-member xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
4     </bgp-conditions>
5 </conditions>
```

@line 3: VPN Non member match.

BGP Server

BGP uses TCP as its transport protocol, by default listens on port 179. OpenDaylight BGP plugin is configured to listen on port *1790*, due to privileged ports restriction for non-root users. One of the workarounds is to use port redirection. In case other port is desired to be used instead, we can reconfigure it.

Here is a sample of bgp port listening re-configuration:

URL: `/restconf/config/odl-bgp-peer-acceptor-config:bgp-peer-acceptor-config/default`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1 <bgp-peer-acceptor-config xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-peer-
↪acceptor-config">
2     <config-name>default</config-name>
3     <binding-address>0.0.0.0</binding-address>
4     <binding-port>1791</binding-port>
5 </bgp-peer-acceptor-config>
```

@line 3: Binding address: By default is 0.0.0.0, so it is not a mandatory field.

@line 4: Binding Port: Port were BGP Server will listen.

BGP Peering

To exchange routing information between two BGP systems (peers), it is required to configure a peering on both BGP speakers first. This mean that each BGP speaker has a white list of neighbors, representing remote peers, with which the peering is allowed. The TCP connection is established between two peers and they exchange messages to open and confirm the connection parameters followed by routes exchange.

Here is a sample basic neighbor configuration:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1  <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2      <neighbor-address>192.0.2.1</neighbor-address>
3      <timers>
4          <config>
5              <hold-time>90</hold-time>
6              <connect-retry>10</connect-retry>
7          </config>
8      </timers>
9      <transport>
10         <config>
11             <remote-port>179</remote-port>
12             <passive-mode>false</passive-mode>
13             <!--<local-address>192.0.2.5</local-address>-->
14         </config>
15     </transport>
16     <config>
17         <peer-type>INTERNAL</peer-type>
18     </config>
19     <afi-safis>
20         ...
21     </afi-safis>
22 </neighbor>
```

@line 2: IP address of the remote BGP peer. Also serves as an unique identifier of a neighbor in a list of neighbors.

@line 5: Proposed number of seconds for value of the Hold Timer. Default value is **90**.

@line 6: Time interval in seconds between attempts to establish session with the peer. Effective in active mode only. Default value is **30**.

@line 11: Remote port number to which the local BGP is connecting. Effective in active mode only. Default value **179**.

@line 12: Wait for peers to issue requests to open a BGP session, rather than initiating sessions from the local router. Default value is **false**.

@line 13: Optional Local IP (either IPv4 or IPv6) address used to establish connections to the remote peer. Effective in active mode only.

@line 17: Explicitly designate the peer as internal or external. Default value is **INTERNAL**.

@line 19: Enable families.

Once the remote peer is connected and it advertised routes to local BGP system, routes are stored in peer's RIBs. The RIBs can be checked via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/peer/bgp:%2F%2F192.0.2.1`

Method: `GET`

Response Body:

```
<peer xmlns="urn:opendaylight:params:xml:ns:yang:bgp-rib">
    <peer-id>bgp://192.0.2.1</peer-id>
    <supported-tables>
        <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-address-family</afi>
        <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-subsequent-address-family</safi>
    </supported-tables>
    <peer-role>ibgp</peer-role>
    <adj-rib-in>
        <tables>
            <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-address-family</afi>
            <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-subsequent-address-family</safi>
            <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
                <ipv4-route>
                    <path-id>0</path-id>
                    <prefix>10.0.0.10/32</prefix>
                    <attributes>
                        <as-path></as-path>
                        <origin>
                            <value>igp</value>
                        </origin>
                        <local-pref>
                            <pref>100</pref>
                        </local-pref>
                        <ipv4-next-hop>
                            <global>10.10.1.1</global>
                        </ipv4-next-hop>
                    </attributes>
                </ipv4-route>
            </ipv4-routes>
            <attributes>
                <uptodate>true</uptodate>
            </attributes>
        </tables>
    </adj-rib-in>
    <effective-rib-in>
        <tables>
            <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-address-family</afi>
            <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-subsequent-address-family</safi>
            <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
                <ipv4-route>
                    <path-id>0</path-id>
                    <prefix>10.0.0.10/32</prefix>
                    <attributes>
```

(continues on next page)

(continued from previous page)

```
44                          <as-path></as-path>
45                          <origin>
46                              <value>igp</value>
47                          </origin>
48                          <local-pref>
49                              <pref>100</pref>
50                          </local-pref>
51                          <ipv4-next-hop>
52                              <global>10.10.1.1</global>
53                          </ipv4-next-hop>
54                      </attributes>
55                  </ipv4-route>
56              </ipv4-routes>
57              <attributes>
58                  <uptodate>true</uptodate>
59              </attributes>
60          </tables>
61      </effective-rib-in>
62      <adj-rib-out>
63          <tables>
64              <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-
    ↪address-family</afi>
65              <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-
    ↪subsequent-address-family</safi>
66              <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet"></ipv4-
    ↪routes>
67              <attributes></attributes>
68          </tables>
69      </adj-rib-out>
70  </peer>
```

@line 8: **Adj-RIB-In** - Per-peer RIB, which contains unprocessed routes that has been advertised to local BGP speaker by the remote peer.

@line 13: Here is the reported route with destination *10.0.0.10/32* in Adj-RIB-In.

@line 35: **Effective-RIB-In** - Per-peer RIB, which contains processed routes as a result of applying inbound policy to Adj-RIB-In routes.

@line 40: Here is the reported route with destination *10.0.0.10/32*, same as in Adj-RIB-In, as it was not touched by import policy.

@line 62: **Adj-RIB-Out** - Per-peer RIB, which contains routes for advertisement to the peer by means of the local speaker's UPDATE message.

@line 66: The peer's Adj-RIB-Out is empty as there are no routes to be advertise from local BGP speaker.

Also the same route should appeared in Loc-RIB now:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/ipv4-routes`

Method: `GET`

Response Body:

```
1  <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
2      <ipv4-route>
3          <path-id>0</path-id>
4          <prefix>10.0.0.10/32</prefix>
5          <attributes>
6              <as-path></as-path>
7              <origin>
8                  <value>igp</value>
9              </origin>
10             <local-pref>
11                 <pref>100</pref>
12             </local-pref>
13             <ipv4-next-hop>
14                 <global>10.10.1.1</global>
15             </ipv4-next-hop>
16         </attributes>
17     </ipv4-route>
18 </ipv4-routes>
```

@line 4: **Destination** - IPv4 Prefix Address.

@line 6: **AS_PATH** - mandatory attribute, contains a list of the autonomous system numbers through that routing information has traversed.

@line 8: **ORIGIN** - mandatory attribute, indicates an origin of the route - **ibgp**, **egp**, **incomplete**.

@line 11: **LOCAL_PREF** - indicates a degree of preference for external routes, higher value is preferred.

@line 14: **NEXT_HOP** - mandatory attribute, defines IP address of the router that should be used as the next hop to the destination.

There are much more attributes that may be carried along with the destination:

BGP-4 Path Attributes

- **MULTI_EXIT_DISC (MED)** Optional attribute, to be used to discriminate among multiple exit/entry points on external links, lower number is preferred.

```
<multi-exit-disc>
 <med>0</med>
</multi-exit-disc>
```

- **ATOMIC_AGGREGATE** Indicates whether AS_SET was excluded from AS_PATH due to routes aggregation.

```
<atomic-aggregate/>
```

- **AGGREGATOR** Optional attribute, contains AS number and IP address of a BGP speaker which performed routes aggregation.

```
<aggregator>
    <as-number>65000</as-number>
    <network-address>192.0.2.2</network-address>
</aggregator>
```

- **Unrecognised** Optional attribute, used to store optional attributes, unrecognized by a local BGP speaker.

```
<unrecognized-attributes>
    <partial>true</partial>
    <transitive>true</transitive>
    <type>101</type>
    <value>0101010101010101</value>
</unrecognized-attributes>
```

Route Reflector Attributes

- **ORIGINATOR_ID** Optional attribute, carries BGP Identifier of the originator of the route.

```
<originator-id>
    <originator>41.41.41.41</originator>
</originator-id>
```

- **CLUSTER_LIST** Optional attribute, contains a list of CLUSTER_ID values representing the path that the route has traversed.

```
<cluster-id>
    <cluster>40.40.40.40</cluster>
</cluster-id>
```

- **Communities** Optional attribute, may be used for policy routing.

```
<communities>
    <as-number>65000</as-number>
    <semantics>30740</semantics>
</communities>
```

Extended Communities

- **Route Target** Identifies one or more routers that may receive a route.

```
<extended-communities>
    <transitive>true</transitive>
    <route-target-ipv4>
        <global-administrator>192.0.2.2</global-administrator>
        <local-administrator>123</local-administrator>
    </route-target-ipv4>
</extended-communities>
<extended-communities>
    <transitive>true</transitive>
    <as-4-route-target-extended-community>
            <as-4-specific-common>
            <as-number>65000</as-number>
            <local-administrator>123</local-administrator>
        </as-4-specific-common>
    </as-4-route-target-extended-community>
</extended-communities>
```

- **Route Origin** Identifies one or more routers that injected a route.

```
<extended-communities>
    <transitive>true</transitive>
    <route-origin-ipv4>
        <global-administrator>192.0.2.2</global-administrator>
        <local-administrator>123</local-administrator>
    </route-origin-ipv4>
```

(continues on next page)

(continued from previous page)

```
</extended-communities>
<extended-communities>
    <transitive>true</transitive>
    <as-4-route-origin-extended-community>
        <as-4-specific-common>
            <as-number>65000</as-number>
            <local-administrator>123</local-administrator>
        </as-4-origin-common>
    </as-4-route-target-extended-community>
</extended-communities>
```

- **Link Bandwidth** Carries the cost to reach external neighbor.

```
<extended-communities>
    <transitive>true</transitive>
    <link-bandwidth-extended-community>
        <bandwidth>BH9CQAA=</bandwidth>
    </link-bandwidth-extended-community>
</extended-communities>
```

- **AIGP** Optional attribute, carries accumulated IGP metric.

```
<aigp>
    <aigp-tlv>
        <metric>120</metric>
    </aigp-tlv>
</aigp>
```

Note: When the remote peer disconnects, it disappear from operational state of local speaker instance and advertised routes are removed too.

External peering configuration

An example above provided configuration for internal peering only. Following configuration sample is intended for external peering:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

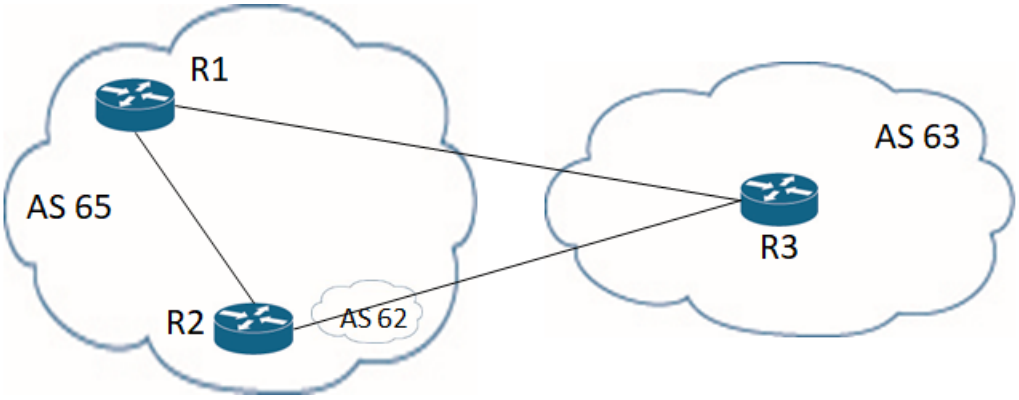
Content-Type: `application/xml`

Request Body:

```
1 <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2     <neighbor-address>192.0.2.3</neighbor-address>
3     <config>
4         <peer-type>EXTERNAL</peer-type>
5         <peer-as>64999</peer-as>
6     </config>
7 </neighbor>
```

@line 5: AS number of the remote peer.

Local AS



The local-AS feature allows a router(eBGP) to appear to be a member of a second autonomous system (AS), in addition to its real AS.

In above figure, R3 is eBGP router with configured local-as of 62, and peer-as of 63.

In updates sent from R3 to R2, the AS_SEQUENCE in the AS_PATH attribute contains "62 63". And updates sent from R2 to R3, the AS_SEQUENCE in the AS_PATH attribute contains "62 65".

AS 62 will be prepended to updates that are sent to and received from R3.

Following configuration sample is intended for external peering with Local AS:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1 <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2     <neighbor-address>192.0.2.3</neighbor-address>
3     <config>
4         <peer-type>EXTERNAL</peer-type>
5         <peer-as>63</peer-as>
6         <local-as>62</local-as>
7     </config>
8 </neighbor>
```

@line 5: AS number of the remote peer.

@line 6: Local AS number of the remote peer.

Route reflector configuration

The local BGP speaker can be configured with a specific *cluster ID*. Following example adds the cluster ID to the existing speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/global/config`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1  <config>
2      <router-id>192.0.2.2</router-id>
3      <as>65000</as>
4      <route-reflector-cluster-id>192.0.2.1</route-reflector-cluster-id>
5  </config>
```

@line 4: Route-reflector cluster id to use when local router is configured as a route reflector. The *router-id* is used as a default value.

Following configuration sample is intended for route reflector client peering:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

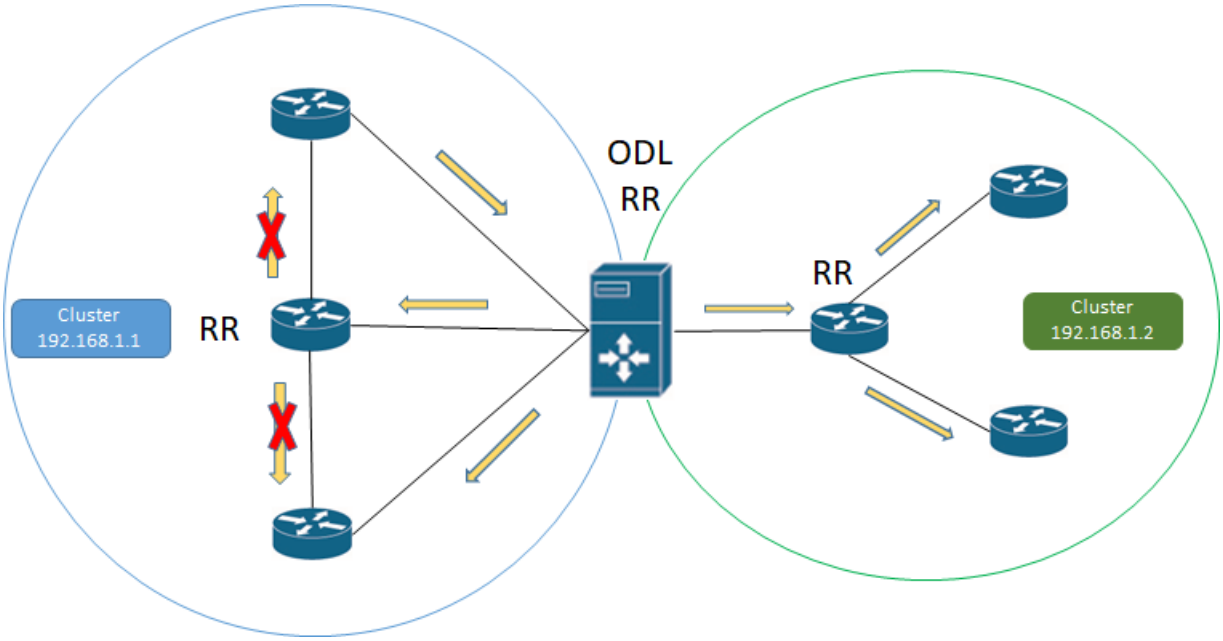
Request Body:

```
1  <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2      <neighbor-address>192.0.2.4</neighbor-address>
3      <config>
4          <peer-type>INTERNAL</peer-type>
5      </config>
6      <route-reflector>
7          <config>
8              <route-reflector-client>true</route-reflector-client>
9          </config>
10     </route-reflector>
11 </neighbor>
```

@line 8: Configure the neighbor as a route reflector client. Default value is *false*.

Route reflector and Multiple Cluster IDs

An optional non-transitive attribute called CLUSTER_LIST is modified when a route reflector reflects a prefix. For loop prevention the route reflector adds its own cluster ID to, and discards any update containing router's own cluster ID. Using multiple cluster IDs allows updates to propagate to nodes that reside in a different cluster.



Following configuration sample is intended for route reflector client peering using specific cluster id:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1  <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2      <neighbor-address>192.0.2.4</neighbor-address>
3      <config>
4          <peer-type>INTERNAL</peer-type>
5      </config>
6      <route-reflector>
7          <config>
8              <route-reflector-client>true</route-reflector-client>
9              <route-reflector-cluster-id>192.0.2.4</route-reflector-cluster-id>
10         </config>
11     </route-reflector>
12 </neighbor>
```

@line 8: Configure the neighbor as a route reflector client. Default value is *false*.

@line 9: Route-reflector cluster id to use for this specific neighbor when local router is configured as a route reflector.

MD5 authentication configuration

The OpenDaylight BGP implementation is supporting TCP MD5 for authentication. Sample configuration below shows how to set authentication password for a peer:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1  <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2      <neighbor-address>192.0.2.5</neighbor-address>
3      <config>
4          <auth-password>topsecret</auth-password>
5      </config>
6  </neighbor>
```

@line 4: Configures an MD5 authentication password for use with neighboring devices.

BGP Peer Group

Allows the creation of a peer group configuration that applies to all peers configured as part of the group.

A sample peer group configuration follows:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/peer-groups`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1  <peer-group xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2      <peer-group-name>internal-neighbor</peer-group-name>
3      <config>
4          <peer-type>INTERNAL</peer-type>
5          <peer-as>64496</peer-as>
6      </config>
7      <transport>
8          <config>
9              <remote-port>179</remote-port>
10             <passive-mode>true</passive-mode>
11         </config>
12     </transport>
13     <timers>
14         <config>
15             <hold-time>180</hold-time>
16             <connect-retry>10</connect-retry>
17         </config>
18     </timers>
19     <route-reflector>
```

(continues on next page)

(continued from previous page)

```
20          <config>
21              <route-reflector-client>false</route-reflector-client>
22          </config>
23      </route-reflector>
24      <afi-safis>
25          <afi-safi>
26              <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-
↪UNICAST</afi-safi-name>
27              <!--Advertise N Paths
28              <receive>true</receive>
29              <send-max>0</send-max>-->
30          </afi-safi>
31          <afi-safi>
32              <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-
↪UNICAST</afi-safi-name>
33          </afi-safi>
34          <afi-safi>
35              <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-
↪LABELLED-UNICAST</afi-safi-name>
36          </afi-safi>
37          <afi-safi>
38              <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-
↪LABELLED-UNICAST</afi-safi-name>
39          </afi-safi>
40          <afi-safi>
41              <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:L3VPN-
↪IPV4-UNICAST</afi-safi-name>
42          </afi-safi>
43          <afi-safi>
44              <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:L3VPN-
↪IPV6-UNICAST</afi-safi-name>
45          </afi-safi>
46          <afi-safi>
47              <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:L2VPN-EVPN
↪</afi-safi-name>
48          </afi-safi>
49          <afi-safi>
50              <afi-safi-name>LINKSTATE</afi-safi-name>
51          </afi-safi>
52          <afi-safi>
53              <afi-safi-name>IPV4-FLOW</afi-safi-name>
54          </afi-safi>
55          <afi-safi>
56              <afi-safi-name>IPV6-FLOW</afi-safi-name>
57          </afi-safi>
58          <afi-safi>
59              <afi-safi-name>IPV4-L3VPN-FLOW</afi-safi-name>
60          </afi-safi>
61          <afi-safi>
62              <afi-safi-name>IPV6-L3VPN-FLOW</afi-safi-name>
63          </afi-safi>
64      </afi-safis>
65  </peer-group>
```

@line 2: Peer Group Identifier.

A sample basic neighbor configuration using a peer group follows:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1 <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2    <neighbor-address>192.0.2.1</neighbor-address>
3    <config>
4        <peer-group>/bgp/neighbors/neighbor/bgp/peer-groups/peer-group[peer-group-name=
   ↪"internal-neighbor"]</peer-group>
5    </config>
6 </neighbor>
```

@line 4: Peer group identifier.

Note: Existing neighbor configuration can be reconfigured (change configuration parameters) anytime. As a result, established connection is dropped, peer instance is recreated with a new configuration settings and connection re-established.

Note: The BGP configuration is persisted on OpendDaylight shutdown and restored after the re-start.

BGP Application Peer and programmable RIB

The OpenDaylight BGP implementation also supports routes injection via *Application Peer*. Such peer has its own programmable RIB, which can be modified by user. This concept allows user to originate new routes and advertise them to all connected peers.

Application Peer configuration

Following configuration sample show a way to configure the *Application Peer*:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>10.25.1.9</neighbor-address>
    <config>
        <peer-group>application-peers</peer-group>
    </config>
</neighbor>
```

@line 2: IP address is uniquely identifying *Application Peer* and its programmable RIB. Address is also used in local BGP speaker decision process.

@line 4: Indicates that peer is associated with *application-peers* group. It serves to distinguish *Application Peer's* from regular neighbors.

The *Application Peer* presence can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/peer/bgp:%2F%2F10.25.1.9`

Method: `GET`

Response Body:

```
1  <peer xmlns="urn:opendaylight:params:xml:ns:yang:bgp-rib">
2      <peer-id>bgp://10.25.1.9</peer-id>
3      <peer-role>internal</peer-role>
4      <adj-rib-in>
5          <tables>
6              <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-
   ↪address-family</afi>
7              <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-
   ↪subsequent-address-family</safi>
8              <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet"></ipv4-
   ↪routes>
9              <attributes>
10                 <uptodate>false</uptodate>
11             </attributes>
12         </tables>
13     </adj-rib-in>
14     <effective-rib-in>
15         <tables>
16             <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-
   ↪address-family</afi>
17             <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-
   ↪subsequent-address-family</safi>
18             <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet"></ipv4-
   ↪routes>
19             <attributes></attributes>
20         </tables>
21     </effective-rib-in>
22 </peer>
```

@line 3: Peer role for *Application Peer* is *internal*.

@line 8: Adj-RIB-In is empty, as no routes were originated yet.

Note: There is no Adj-RIB-Out for *Application Peer*.

Programmable RIB

Next example shows how to inject a route into the programmable RIB.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<ipv4-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <path-id>0</path-id>
    <prefix>10.0.0.11/32</prefix>
    <attributes>
        <as-path></as-path>
        <origin>
            <value>igp</value>
        </origin>
        <local-pref>
            <pref>100</pref>
        </local-pref>
        <ipv4-next-hop>
            <global>10.11.1.1</global>
        </ipv4-next-hop>
    </attributes>
</ipv4-route>
```

Now the injected route appears in *Application Peer's* RIBs and in local speaker's Loc-RIB:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/peer/bgp:%2F%2F10.25.1.9`

Method: `GET`

Response Body:

```
<peer xmlns="urn:opendaylight:params:xml:ns:yang:bgp-rib">
    <peer-id>bgp://10.25.1.9</peer-id>
    <peer-role>internal</peer-role>
    <adj-rib-in>
        <tables>
            <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-address-family</afi>
            <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-subsequent-address-family</safi>
            <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
                <ipv4-route>
                    <path-id>0</path-id>
                    <prefix>10.0.0.11/32</prefix>
                    <attributes>
                        <as-path></as-path>
                        <origin>
                            <value>igp</value>
                        </origin>
```

(continues on next page)

(continued from previous page)

```
                        <local-pref>
                            <pref>100</pref>
                        </local-pref>
                        <ipv4-next-hop>
                            <global>10.11.1.1</global>
                        </ipv4-next-hop>
                    </attributes>
                </ipv4-route>
            </ipv4-routes>
            <attributes>
                <uptodate>false</uptodate>
            </attributes>
        </tables>
    </adj-rib-in>
    <effective-rib-in>
        <tables>
            <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:ipv4-address-family</afi>
            <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">x:unicast-subsequent-address-family</safi>
            <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
                <ipv4-route>
                    <path-id>0</path-id>
                    <prefix>10.0.0.11/32</prefix>
                    <attributes>
                        <as-path></as-path>
                        <origin>
                            <value>igp</value>
                        </origin>
                        <local-pref>
                            <pref>100</pref>
                        </local-pref>
                        <ipv4-next-hop>
                            <global>10.11.1.1</global>
                        </ipv4-next-hop>
                    </attributes>
                </ipv4-route>
            </ipv4-routes>
            <attributes></attributes>
        </tables>
    </effective-rib-in>
</peer>
```

@line 9: Injected route is present in *Application Peer's* Adj-RIB-In and Effective-RIB-In.

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/ipv4-routes`

Method: `GET`

Response Body:

```
<ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <ipv4-route>
        <path-id>0</path-id>
```

(continues on next page)

(continued from previous page)

```
        <prefix>10.0.0.10/32</prefix>
        <attributes>
            <as-path></as-path>
            <origin>
                <value>igp</value>
            </origin>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <ipv4-next-hop>
                <global>10.11.1.1</global>
            </ipv4-next-hop>
        </attributes>
    </ipv4-route>
    <ipv4-route>
        <path-id>0</path-id>
        <prefix>10.0.0.10/32</prefix>
        <attributes>
            <as-path></as-path>
            <origin>
                <value>igp</value>
            </origin>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <ipv4-next-hop>
                <global>10.10.1.1</global>
            </ipv4-next-hop>
        </attributes>
    </ipv4-route>
</ipv4-routes>
```

@line 2: The injected route is now present in Loc-RIB along with a route (destination *10.0.0.10/32*) advertised by remote peer.

This route is also advertised to the remote peer (*192.0.2.1*), hence route appears in its Adj-RIB-Out:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/peer/bgp:%2F%2F192.0.2.1/adj-rib-out/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes`

Method: `GET`

Response Body:

```
<ipv4-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <path-id>0</path-id>
    <prefix>10.0.0.11/32</prefix>
    <attributes>
        <as-path></as-path>
        <origin>
            <value>igp</value>
        </origin>
        <local-pref>
            <pref>100</pref>
        </local-pref>
```

(continues on next page)

(continued from previous page)

```
        <ipv4-next-hop>
            <global>10.11.1.1</global>
        </ipv4-next-hop>
    </attributes>
</ipv4-route>
```

The injected route can be modified (i.e. different path attribute):

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes/ipv4-route/10.0.0.11%2F32/0`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<ipv4-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <path-id>0</path-id>
    <prefix>10.0.0.11/32</prefix>
    <attributes>
        <as-path></as-path>
        <origin>
            <value>igp</value>
        </origin>
        <local-pref>
            <pref>50</pref>
        </local-pref>
        <ipv4-next-hop>
            <global>10.11.1.2</global>
        </ipv4-next-hop>
    </attributes>
</ipv4-route>
```

The route can be removed from programmable RIB in a following way:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes/ipv4-route/10.0.0.11%2F32/0`

Method: `DELETE`

Also it is possible to remove all routes from a particular table at once:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes/`

Method: `DELETE`

Consequently, route disappears from programmable RIB, *Application Peer's* RIBs, Loc-RIB and peer's Adj-RIB-Out (UPDATE message with prefix withdrawal is send).

Note: Routes stored in programmable RIB are persisted on OpendDaylight shutdown and restored after the re-start.

BGP Protocol Configuration Loader

BGP Protocol Configuration Loader allows the user to define the static initial configuration for a BGP protocol instance. This service will detect the creation of new configuration files following the pattern `protocols-*.xml` under the path "etc/opendaylight/bgpcep". Once the file is processed, the defined configuration will be available from the configuration Data Store.

Note: If the BGP instance is already present, no update or configuration will be applied.

PATH: `etc/opendaylight/bgpcep/protocols-config.xml`

```
<protocols xmlns="http://openconfig.net/yang/network-instance">
    <protocol>
        <name>example-bgp-rib</name>
        <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</
↪identifier>
        <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
            <global>
                <config>
                    <router-id>192.0.2.2</router-id>
                    <as>64496</as>
                    <!-- if cluster-id is not present, it's value is the same as bgp-
↪id -->
                    <!-- <route-reflector-cluster-id>192.0.2.3</route-reflector-
↪cluster-id> -->
                    <!-- <read-only-limit>120</read-only-limit>-->
                </config>
                <afi-safis>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV4-UNICAST</afi-safi-name>
                        <!--Advertise N Paths
                        <receive>true</receive>
                        <send-max>2</send-max>-->
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV6-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV4-LABELLED-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV6-LABELLED-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L3VPN-IPV4-UNICAST</afi-safi-name>
                    </afi-safi>
```

(continues on next page)

(continued from previous page)

```
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L3VPN-IPV6-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L2VPN-EVPN</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>LINKSTATE</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV4-FLOW</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV6-FLOW</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV4-L3VPN-FLOW</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV6-L3VPN-FLOW</afi-safi-name>
                    </afi-safi>
                </afi-safis>
            </global>
            <neighbors xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-
↪extensions">
                <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-
↪extensions">
                    <neighbor-address>192.0.2.1</neighbor-address>
                    <config>
                        <peer-type>INTERNAL</peer-type>
                        <peer-as>64496</peer-as>
                    </config>
                    <transport>
                        <config>
                            <remote-port>179</remote-port>
                            <passive-mode>true</passive-mode>
                        </config>
                    </transport>
                    <timers>
                        <config>
                            <hold-time>180</hold-time>
                            <connect-retry>10</connect-retry>
                        </config>
                    </timers>
                    <route-reflector>
                        <config>
                            <route-reflector-client>false</route-reflector-client>
                        </config>
                    </route-reflector>
                    <afi-safis>
                        <afi-safi>
                            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-
↪types">x:IPV4-UNICAST</afi-safi-name>
                            <!--Advertise N Paths
                            <receive>true</receive>
```

(continues on next page)

(continued from previous page)

```
                        <send-max>0</send-max>-->
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-
↪types">x:IPV6-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-
↪types">x:IPV4-LABELLED-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-
↪types">x:IPV6-LABELLED-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-
↪types">x:L3VPN-IPV4-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-
↪types">x:L3VPN-IPV6-UNICAST</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-
↪types">x:L2VPN-EVPN</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>LINKSTATE</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV4-FLOW</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV6-FLOW</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV4-L3VPN-FLOW</afi-safi-name>
                    </afi-safi>
                    <afi-safi>
                        <afi-safi-name>IPV6-L3VPN-FLOW</afi-safi-name>
                    </afi-safi>
                </afi-safis>
            </neighbor>
            <neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-
↪extensions">
                <neighbor-address>192.0.2.6</neighbor-address>
                <config>
                    <peer-group>application-peers</peer-group>
                </config>
            </neighbor>
        </neighbors>
    </bgp>
  </protocol>
</protocols>
```

BGP Configuration Example

BGP provides a feature providing a BGP Protocol and Network Topology configuration file example. Once feature is installed defined configuration will be loaded and setup.

```
feature:install odl-bgpcep-bgp-config-example
```

BGP RIB API

This tree illustrates the BGP RIBs organization in datastore.

```
bgp-rib
  +--ro rib* [id]
     +--ro id         rib-id
     +--ro peer* [peer-id]
     |  +--ro peer-id                  peer-id
     |  +--ro peer-role                peer-role
     |  +--ro simple-routing-policy?   simple-routing-policy
     |  +--ro supported-tables* [afi safi]
     |  |  +--ro afi             identityref
     |  |  +--ro safi            identityref
     |  |  +--ro send-receive?   send-receive
     |  +--ro adj-rib-in
     |  |  +--ro tables* [afi safi]
     |  |     +--ro afi           identityref
     |  |     +--ro safi          identityref
     |  |     +--ro attributes
     |  |     |  +--ro uptodate?   boolean
     |  |     +--ro (routes)?
     |  +--ro effective-rib-in
     |  |  +--ro tables* [afi safi]
     |  |     +--ro afi           identityref
     |  |     +--ro safi          identityref
     |  |     +--ro attributes
     |  |     |  +--ro uptodate?   boolean
     |  |     +--ro (routes)?
     |  +--ro adj-rib-out
     |     +--ro tables* [afi safi]
     |        +--ro afi           identityref
     |        +--ro safi          identityref
     |        +--ro attributes
     |        |  +--ro uptodate?   boolean
     |        +--ro (routes)?
     +--ro loc-rib
        +--ro tables* [afi safi]
           +--ro afi           identityref
           +--ro safi          identityref
           +--ro attributes
           |  +--ro uptodate?   boolean
           +--ro (routes)?
```

BGP pipeline

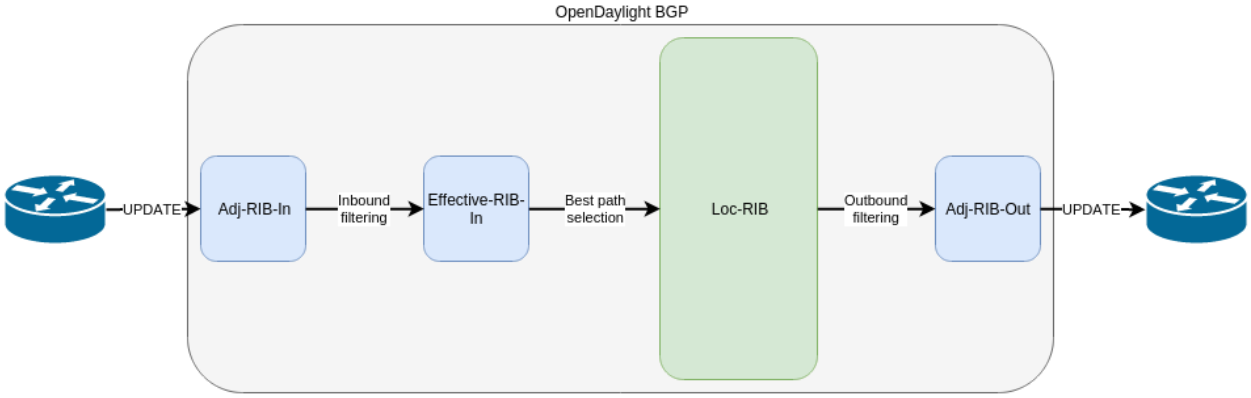


Fig. 1: BGP pipeline - routes re-advertisement.

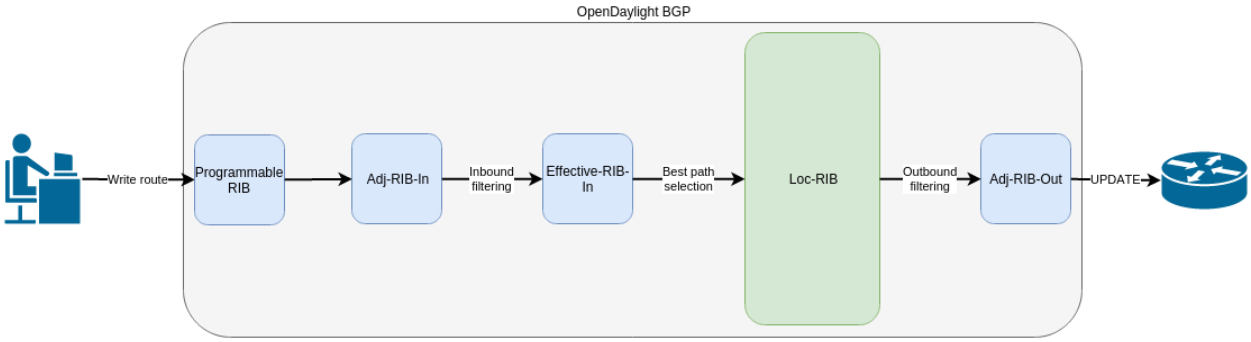


Fig. 2: BGP applcaition peer pipeline - routes injection.

References

- A Border Gateway Protocol 4 (BGP-4)
- BGP Route Reflection
- BGP Communities Attribute
- BGP Support for Four-Octet Autonomous System (AS) Number Space
- The Accumulated IGP Metric Attribute for BGP
- 4-Octet AS Specific BGP Extended Community
- BGP Link Bandwidth Extended Community
- Use of BGP for Routing in Large-Scale Data Centers

2.1.5 IP Unicast Family

The BGP-4 allows to carry IPv4 specific information only. The basic BGP Multiprotocol extension brings *Unicast* Subsequent Address Family (SAFI) - intended to be used for IP unicast forwarding. The combination of IPv4 and IPv6 Address Family (AF) and Unicast SAFI is essential for Internet routing. The IPv4 Unicast routes are interchangeable with BGP-4 routes, as they can carry the same type of routing information.

Contents

- *Configuration*
  - *BGP Speaker*
  - *BGP Peer*
- *IP Unicast API*
  - *IPv4 Unicast Route*
  - *IPv6 Unicast Route*
- *Usage*
  - *IPv4 Unicast*
  - *IPv6 Unicast*
- *Programming*
  - *IPv4 Unicast*
  - *IPv6 Unicast*
- *References*

Configuration

This section shows a way to enable IPv4 and IPv6 Unicast family in BGP speaker and peer configuration.

BGP Speaker

To enable IPv4 and IPv6 Unicast support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
```

(continues on next page)

(continued from previous page)

```
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV4-UNICAST</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV6-UNICAST</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

BGP Peer

Here is an example for BGP peer configuration with enabled IPv4 and IPv6 Unicast family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-
↪UNICAST</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-
↪UNICAST</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

IP Unicast API

Following trees illustrate the BGP IP Unicast routes structures.

IPv4 Unicast Route

```
:(ipv4-routes-case)
   +--ro ipv4-routes
     +--ro ipv4-route* [route-key path-id]
        +--ro route-key     string
        +--ro prefix        inet:ipv4-prefix
        +--ro path-id       path-id
        +--ro attributes
           +--ro origin
           |  +--ro value    bgp-t:bgp-origin
           +--ro as-path
           |  +--ro segments*
           |     +--ro as-sequence*   inet:as-number
           |     +--ro as-set*        inet:as-number
           +--ro (c-next-hop)?
           |  +--:(ipv4-next-hop-case)
           |  |  +--ro ipv4-next-hop
           |  |     +--ro global?   inet:ipv4-address
           |  +--:(ipv6-next-hop-case)
           |  |  +--ro ipv6-next-hop
           |  |     +--ro global?       inet:ipv6-address
           |  |     +--ro link-local?   inet:ipv6-address
           |  +--:(empty-next-hop-case)
           |     +--ro empty-next-hop?            empty
           +--ro multi-exit-disc
           |  +--ro med?   uint32
           +--ro local-pref
           |  +--ro pref?   uint32
           +--ro atomic-aggregate!
           +--ro aggregator
           |  +--ro as-number?         inet:as-number
           |  +--ro network-address?   inet:ipv4-address
           +--ro communities*
           |  +--ro as-number?   inet:as-number
           |  +--ro semantics?   uint16
           +--ro extended-communities*
           |  +--ro transitive?                              boolean
           |  +--ro (extended-community)?
           |     +--:(as-specific-extended-community-case)
           |     |  +--ro as-specific-extended-community
           |     |     +--ro global-administrator?   short-as-number
           |     |     +--ro local-administrator?    binary
           |     +--:(inet4-specific-extended-community-case)
           |     |  +--ro inet4-specific-extended-community
           |     |     +--ro global-administrator?   inet:ipv4-address
           |     |     +--ro local-administrator?    binary
           |     +--:(opaque-extended-community-case)
           |     |  +--ro opaque-extended-community
           |     |     +--ro value?   binary
           |     +--:(route-target-extended-community-case)
           |     |  +--ro route-target-extended-community
           |     |     +--ro global-administrator?   short-as-number
           |     |     +--ro local-administrator?    binary
           |     +--:(route-origin-extended-community-case)
           |     |  +--ro route-origin-extended-community
           |     |     +--ro global-administrator?   short-as-number
```

(continues on next page)

(continued from previous page)

```
|     |     +--ro local-administrator?    binary
|     +--:(route-target-ipv4-case)
|     |  +--ro route-target-ipv4
|     |     +--ro global-administrator?   inet:ipv4-address
|     |     +--ro local-administrator?    uint16
|     +--:(route-origin-ipv4-case)
|     |  +--ro route-origin-ipv4
|     |     +--ro global-administrator?   inet:ipv4-address
|     |     +--ro local-administrator?    uint16
|     +--:(link-bandwidth-case)
|     |  +--ro link-bandwidth-extended-community
|     |     +--ro bandwidth    netc:bandwidth
|     +--:(as-4-generic-spec-extended-community-case)
|     |  +--ro as-4-generic-spec-extended-community
|     |     +--ro as-4-specific-common
|     |        +--ro as-number              inet:as-number
|     |        +--ro local-administrator    uint16
|     +--:(as-4-route-target-extended-community-case)
|     |  +--ro as-4-route-target-extended-community
|     |     +--ro as-4-specific-common
|     |        +--ro as-number              inet:as-number
|     |        +--ro local-administrator    uint16
|     +--:(as-4-route-origin-extended-community-case)
|     |  +--ro as-4-route-origin-extended-community
|     |     +--ro as-4-specific-common
|     |        +--ro as-number              inet:as-number
|     |        +--ro local-administrator    uint16
|     +--:(encapsulation-case)
|        +--ro encapsulation-extended-community
|           +--ro tunnel-type    encapsulation-tunnel-type
+--ro originator-id
|  +--ro originator?   inet:ipv4-address
+--ro cluster-id
|  +--ro cluster*   bgp-t:cluster-identifier
+--ro aigp
|  +--ro aigp-tlv
|     +--ro metric?   netc:accumulated-igp-metric
+--ro unrecognized-attributes* [type]
   +--ro partial       boolean
   +--ro transitive    boolean
   +--ro type          uint8
   +--ro value         binary
```

IPv6 Unicast Route

```
:(ipv6-routes-case)
   +--ro ipv6-routes
      +--ro ipv6-route* [route-key path-id]
         +--ro route-key     string
         +--ro prefix        inet:ipv6-prefix
         +--ro path-id       path-id
         +--ro attributes
         ...
```

Usage

IPv4 Unicast

The IPv4 Unicast table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/ipv4-routes`

Method: `GET`

Response Body:

```
<ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <ipv4-route>
        <route-key>193.0.2.1/32</route-key>
        <path-id>0</path-id>
        <prefix>193.0.2.1/32</prefix>
        <attributes>
            <as-path></as-path>
            <origin>
                <value>igp</value>
            </origin>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <ipv4-next-hop>
                <global>10.0.0.1</global>
            </ipv4-next-hop>
        </attributes>
    </ipv4-route>
</ipv4-routes>
```

IPv6 Unicast

The IPv6 Unicast table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/ipv6-routes`

Method: `GET`

Response Body:

```
<ipv6-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <ipv6-route>
        <route-key>2a02:b80:0:1::/64</route-key>
        <path-id>0</path-id>
        <prefix>2a02:b80:0:1::/64</prefix>
        <attributes>
            <as-path></as-path>
            <origin>
                <value>igp</value>
            </origin>
            <local-pref>
```

(continues on next page)

(continued from previous page)

```
                <pref>200</pref>
            </local-pref>
            <ipv6-next-hop>
                <global>2a02:b80:0:2::1</global>
            </ipv6-next-hop>
        </attributes>
    </ipv6-route>
</ipv6-routes>
```

Note: IPv4/6 routes mapping to topology nodes is supported by BGP Topology Provider.

Programming

IPv4 Unicast

This examples show how to originate and remove IPv4 route via programmable RIB. Make sure the *Application Peer* is configured first.

Note: IPv4 Route Key must be equal to prefix.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<ipv4-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <route-key>10.0.0.11/32</route-key>
    <prefix>10.0.0.11/32</prefix>
    <path-id>0</path-id>
    <attributes>
        <as-path></as-path>
        <origin>
            <value>igp</value>
        </origin>
        <local-pref>
            <pref>100</pref>
        </local-pref>
        <ipv4-next-hop>
            <global>10.11.1.1</global>
        </ipv4-next-hop>
    </attributes>
</ipv4-route>
```

To remove the route added above, following request can be used:

URL: /restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes/ipv4-route/10.0.0.11%2F32/0

Method: DELETE

IPv6 Unicast

This examples show how to originate and remove IPv6 route via programmable RIB:

Note: IPv6 Route Key must be equal to prefix.

URL: /restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv6-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv6-routes

Method: POST

Content-Type: application/xml

Request Body:

```
<ipv6-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <route-key>2001:db8:30::3/128</route-key>
    <prefix>2001:db8:30::3/128</prefix>
    <path-id>0</path-id>
    <attributes>
        <ipv6-next-hop>
            <global>2001:db8:1::6</global>
        </ipv6-next-hop>
        <as-path/>
        <origin>
            <value>igp</value>
        </origin>
        <local-pref>
            <pref>100</pref>
        </local-pref>
    </attributes>
</ipv6-route>
```

To remove the route added above, following request can be used:

URL: /restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv6-address-family/bgp-types:unicast-subsequent-address-family/bgp-inet:ipv6-routes/ipv6-route/2001:db8:30::3%2F128/0

Method: DELETE

References

- Multiprotocol Extensions for BGP-4

2.1.6 IP Labeled Unicast Family

The BGP Labeled Unicast (BGP-LU) Multiprotocol extension is used to distribute a MPLS label that is mapped to a particular route. It can be used to advertise a MPLS transport path between IGP regions and Autonomous Systems. Also, BGP-LU can help to solve the Inter-domain traffic-engineering problem and can be deployed in large-scale data centers along with MPLS and Spring. In addition, IPv6 Labeled Unicast can be used to interconnect IPv6 islands over IPv4/MPLS networks using 6PE.

Contents

- *Configuration*
  - *BGP Speaker*
  - *BGP Peer*
- *IP Labeled Unicast API*
  - *IPv4 Labeled Unicast Route*
  - *IPv6 Labeled Unicast Route*
- *Usage*
- *Programming*
  - *IPv4 Labeled*
  - *IPv6 Labeled*
- *References*

Configuration

This section shows a way to enable IPv4 and IPv6 Labeled Unicast family in BGP speaker and peer configuration.

BGP Speaker

To enable IPv4 and IPv6 Labeled Unicast support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
```

(continues on next page)

(continued from previous page)

```
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV4-LABELLED-UNICAST</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV6-LABELLED-UNICAST</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

BGP Peer

Here is an example for BGP peer configuration with enabled IPv4 and IPv6 Labeled Unicast family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-
↪LABELLED-UNICAST</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-
↪LABELLED-UNICAST</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

IP Labeled Unicast API

Following trees illustrate the BGP IP Labeled Unicast routes structures.

IPv4 Labeled Unicast Route

```
:(labeled-unicast-routes-case)
  +--ro labeled-unicast-routes
     +--ro labeled-unicast-route* [route-key path-id]
        +--ro route-key      string
        +--ro label-stack*
        |  +--ro label-value?   netc:mpls-label
        +--ro prefix?        inet:ip-prefix
        +--ro path-id        path-id
        +--ro attributes
        ...
```

IPv6 Labeled Unicast Route

```
:(labeled-unicast-ipv6-routes-case)
   +--ro labeled-unicast-ipv6-routes
      +--ro labeled-unicast-route* [route-key path-id]
         +--ro route-key      string
         +--ro label-stack*
         |  +--ro label-value?   netc:mpls-label
         +--ro prefix?        inet:ip-prefix
         +--ro path-id        path-id
         +--ro attributes
         ...
```

Usage

The IPv4 Labeled Unicast table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-labeled-unicast:labeled-unicast-subsequent-address-family-bgp-labeled-unicast:labeled-unicast-routes`

Method: `GET`

Response Body:

```
<labeled-unicast-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-labeled-unicast
↪">
    <labeled-unicast-route>
        <path-id>0</path-id>
        <route-key>MAA+gRQAAA==</route-key>
        <attributes>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <ipv4-next-hop>
                <global>200.10.0.101</global>
```

(continues on next page)

(continued from previous page)

```
            </ipv4-next-hop>
            <as-path></as-path>
            <origin>
                <value>igp</value>
            </origin>
        </attributes>
        <label-stack>
            <label-value>1000</label-value>
        </label-stack>
        <prefix>20.0.0.0/24</prefix>
    </labeled-unicast-route>
</labeled-unicast-routes>
```

Programming

IPv4 Labeled

This examples show how to originate and remove IPv4 labeled route via programmable RIB. Make sure the *Application Peer* is configured first.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-labeled-unicast:labeled-unicast-subsequent-address-family bgp-labeled-unicast:labeled-unicast-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<labeled-unicast-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-labeled-unicast
↪">
    <route-key>label1</route-key>
    <prefix>1.1.1.1/32</prefix>
    <path-id>0</path-id>
    <label-stack>
        <label-value>800322</label-value>
    </label-stack>
    <attributes>
        <ipv4-next-hop>
            <global>199.20.160.41</global>
        </ipv4-next-hop>
        <origin>
            <value>igp</value>
        </origin>
        <as-path/>
        <local-pref>
            <pref>100</pref>
        </local-pref>
    </attributes>
</labeled-unicast-route>
```

In addition, BGP-LU Spring extension allows to attach BGP Prefix SID attribute to the route, in order to signal the BGP-Prefix-SID, where the SR is applied to MPLS dataplane.

```
<bgp-prefix-sid>
    <bgp-prefix-sid-tlvs>
        <label-index-tlv xmlns="urn:opendaylight:params:xml:ns:yang:bgp-labeled-
↪unicast">322</label-index-tlv>
    </bgp-prefix-sid-tlvs>
    <bgp-prefix-sid-tlvs>
        <srgb-value xmlns="urn:opendaylight:params:xml:ns:yang:bgp-labeled-unicast">
            <base>800000</base>
            <range>4095</range>
        </srgb-value>
    </bgp-prefix-sid-tlvs>
</bgp-prefix-sid>
```

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-labeled-unicast:labeled-unicast-subsequent-address-family bgp-labeled-unicast:labeled-unicast-routes/bgp-labeled-unicast:labeled-unicast-route/label1/0`

Method: `DELETE`

IPv6 Labeled

This examples show how to originate and remove IPv6 labeled route via programmable RIB.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-labeled-unicast:labeled-unicast-subsequent-address-family bgp-labeled-unicast:labeled-unicast-ipv6-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<labeled-unicast-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-labeled-unicast
↪">
    <route-key>label1</route-key>
    <prefix>2001:db8:30::3/128</prefix>
    <path-id>0</path-id>
    <label-stack>
        <label-value>123</label-value>
    </label-stack>
    <attributes>
        <ipv6-next-hop>
            <global>2003:4:5:6::7</global>
        </ipv6-next-hop>
        <origin>
            <value>igp</value>
        </origin>
        <as-path/>
        <local-pref>
            <pref>100</pref>
        </local-pref>
    </attributes>
</labeled-unicast-route>
```

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-labeled-unicast:labeled-unicast-subsequent-address-family/bgp-labeled-unicast:labeled-unicast-ipv6-routes/bgp-labeled-unicast:labeled-unicast-route/label1/0`

Method: `DELETE`

References

- Carrying Label Information in BGP-4
- Segment Routing Prefix SID extensions for BGP
- Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)
- BGP-Prefix Segment in large-scale data centers
- Egress Peer Engineering using BGP-LU

2.1.7 IP L3VPN Family

The BGP/MPLS IP Virtual Private Networks (BGP L3VPN) Multiprotocol extension can be used to exchange particular VPN (customer) routes among the provider's routers attached to that VPN. Also, routes are distributed to specific VPN remote sites.

Contents

- *Configuration*
  - *BGP Speaker*
  - *BGP Peer*
- *IP L3VPN API*
  - *IPv4 L3VPN Unicast Route*
  - *IPv6 L3VPN Unicast Route*
  - *IPv4 L3VPN Multicast Route*
  - *IPv6 L3VPN Multicast Route*
- *Usage*
  - *IPv4 L3VPN Unicast*
  - *IPv6 L3VPN Unicast*
  - *IPv4 L3VPN Multicast*
  - *IPv6 L3VPN Multicast*
- *Programming*
- *References*

Configuration

This section shows a way to enable IPv4 and IPv6 L3VPN family in BGP speaker and peer configuration.

BGP Speaker

To enable IPv4 and IPv6 L3VPN support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L3VPN-IPV4-UNICAST</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L3VPN-IPV6-UNICAST</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L3VPN-IPV4-MULTICAST</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L3VPN-IPV6-MULTICAST</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

BGP Peer

Here is an example for BGP peer configuration with enabled IPv4 and IPv6 L3VPN family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:L3VPN-
↪IPV4-UNICAST</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:L3VPN-
↪IPV6-UNICAST</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

IP L3VPN API

Following trees illustrate the BGP IP L3VPN routes structures.

IPv4 L3VPN Unicast Route

```
:(vpn-ipv4-routes-case)
   +--ro vpn-ipv4-routes
      +--ro vpn-route* [route-key path-id]
         +--ro route-key              string
         +--ro path-id                path-id
         +--ro label-stack*
         |  +--ro label-value?   netc:mpls-label
         +--ro prefix?                inet:ip-prefix
         +--ro path-id?               path-id
         +--ro route-distinguisher?   bgp-t:route-distinguisher
         +--ro attributes
         ...
```

IPv6 L3VPN Unicast Route

```
:(vpn-ipv6-routes-case)
   +--ro vpn-ipv6-routes
      +--ro vpn-route* [route-key path-id]
         +--ro route-key              string
         +--ro path-id                path-id
         +--ro label-stack*
         |  +--ro label-value?   netc:mpls-label
         +--ro prefix?                inet:ip-prefix
         +--ro path-id?               path-id
         +--ro route-distinguisher?   bgp-t:route-distinguisher
         +--ro attributes
         ...
```

IPv4 L3VPN Multicast Route

```
:(l3vpn-mcast-routes-ipv4-case)
   +--ro l3vpn-mcast-routes-ipv4
        +--ro l3vpn-mcast-route* [route-key path-id]
           +--ro prefix?                inet:ip-prefix
           +--ro route-distinguisher?   bgp-t:route-distinguisher
```

IPv6 L3VPN Multicast Route

```
:(l3vpn-mcast-routes-ipv6-case)
   +--ro l3vpn-mcast-routes-ipv6
        +--ro l3vpn-mcast-route* [route-key path-id]
           +--ro prefix?                inet:ip-prefix
           +--ro route-distinguisher?   bgp-t:route-distinguisher
```

Usage

IPv4 L3VPN Unicast

The IPv4 L3VPN Unicast table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-types:mpls-labeled-vpn-subsequent-address-family/bgp-vpn-ipv4:vpn-ipv4-routes`

Method: `GET`

Response Body:

```
<vpn-ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-vpn-ipv4">
    <vpn-route>
        <path-id>0</path-id>
        <route-key>cAXdYQABrBAALABlCgIi</route-key>
        <label-stack>
            <label-value>24022</label-value>
```

(continues on next page)

(continued from previous page)

```
        </label-stack>
        <attributes>
            <extended-communities>
                <transitive>true</transitive>
                <route-target-extended-community>
                    <global-administrator>65000</global-administrator>
                    <local-administrator>AAAAZQ==</local-administrator>
                </route-target-extended-community>
            </extended-communities>
            <origin>
                <value>igp</value>
            </origin>
            <as-path></as-path>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <ipv4-next-hop>
                <global>127.16.0.44</global>
            </ipv4-next-hop>
        </attributes>
        <route-distinguisher>172.16.0.44:101</route-distinguisher>
        <prefix>10.2.34.0/24</prefix>
    </vpn-route>
</vpn-ipv4-routes>
```

IPv6 L3VPN Unicast

The IPv6 L3VPN Unicast table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv6-address-family/bgp-types:mpls-labeled-vpn-subsequent-address-family/bgp-vpn-ipv6:vpn-ipv6-routes`

Method: `GET`

Response Body:

```
<vpn-ipv6-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-vpn-ipv6">
    <vpn-route>
        <path-id>0</path-id>
        <route-key>mAXdcQABrBAALABlKgILgAAAAAE=</route-key>
        <label-stack>
            <label-value>24023</label-value>
        </label-stack>
        <attributes>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <extended-communities>
                <route-target-extended-community>
                    <global-administrator>65000</global-administrator>
                    <local-administrator>AAAAZQ==</local-administrator>
                </route-target-extended-community>
                <transitive>true</transitive>
            </extended-communities>
            <ipv6-next-hop>
```

(continues on next page)

(continued from previous page)

```
                <global>2a02:b80:0:2::1</global>
            </ipv6-next-hop>
            <origin>
                <value>igp</value>
            </origin>
            <as-path></as-path>
        </attributes>
        <route-distinguisher>172.16.0.44:101</route-distinguisher>
        <prefix>2a02:b80:0:1::/64</prefix>
    </vpn-route>
</vpn-ipv6-routes>
```

IPv4 L3VPN Multicast

The IPv4 L3VPN Multicast table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-types:mcast-mpls-labeled-vpn-subsequent-address-family/bgp-l3vpn-mcast:l3vpn-mcast-routes`

Method: `GET`

Response Body:

```
<l3vpn-mcast-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp:l3vpn:mcast">
    <l3vpn-mcast-route>
        <path-id>0</path-id>
        <route-key>mAXdcQABrBAALABlKgILgAAAAAE=</route-key>
        <route-distinguisher>172.16.0.44:101</route-distinguisher>
        <prefix>10.2.34.0/24</prefix>
        <attributes>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <extended-communities>
                <transitive>true</transitive>
                <vrf-route-import-extended-community>
                    <inet4-specific-extended-community-common>
                        <global-administrator>10.0.0.1</global-administrator>
                        <local-administrator>123=</local-administrator>
                     </inet4-specific-extended-community-common>
                 </vrf-route-import-extended-community>
            </extended-communities>
            <ipv4-next-hop>
                <global>127.16.0.44</global>
            </ipv4-next-hop>
            <origin>
                <value>igp</value>
            </origin>
            <as-path></as-path>
        </attributes>
    </l3vpn-mcast-route>
</l3vpn-mcast-routes>
```

IPv6 L3VPN Multicast

The IPv4 L3VPN Multicast table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv6-address-family/bgp-types:mcast-mpls-labeled-vpn-subsequent-address-family/bgp-l3vpn-mcast:l3vpn-mcast-routes`

Method: `GET`

Response Body:

```
<l3vpn-mcast-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp:l3vpn:mcast">
    <l3vpn-mcast-route>
        <path-id>0</path-id>
        <route-key>mAXdcQABrBAALABlKgILgAAAAAE=</route-key>
        <route-distinguisher>172.16.0.44:101</route-distinguisher>
        <prefix>2a02:b80:0:1::/64</prefix>
        <attributes>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <extended-communities>
                <transitive>true</transitive>
                <vrf-route-import-extended-community>
                    <inet4-specific-extended-community-common>
                        <global-administrator>10.0.0.1</global-administrator>
                        <local-administrator>123=</local-administrator>
                     </inet4-specific-extended-community-common>
                 </vrf-route-import-extended-community>
            </extended-communities>
            <ipv6-next-hop>
                <global>2a02:b80:0:2::1</global>
            </ipv6-next-hop>
            <origin>
                <value>igp</value>
            </origin>
            <as-path></as-path>
        </attributes>
    </l3vpn-mcast-route>
</l3vpn-mcast-routes>
```

Programming

This examples show how to originate and remove IPv4 L3VPN Unicast route via programmable RIB. Make sure the *Application Peer* is configured first.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:mpls-labeled-vpn-subsequent-address-family/bgp-vpn-ipv4:vpn-ipv4-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<vpn-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-vpn-ipv4">
    <path-id>0</path-id>
    <route-key>vpn1</route-key>
    <label-stack>
        <label-value>123</label-value>
    </label-stack>
    <route-distinguisher>429496729:1</route-distinguisher>
    <prefix>2.2.2.2/32</prefix>
    <attributes>
        <ipv4-next-hop>
            <global>199.20.166.41</global>
        </ipv4-next-hop>
        <as-path/>
        <origin>
            <value>igp</value>
        </origin>
        <extended-communities>
            <route-target-extended-community>
                <global-administrator>65000</global-administrator>
                <local-administrator>AAAAZQ==</local-administrator>
            </route-target-extended-community>
            <transitive>true</transitive>
        </extended-communities>
    </attributes>
</vpn-route>
```

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-types:mpls-labeled-vpn-subsequent-address-family/bgp-vpn-ipv4:vpn-ipv4-routes/vpn-route/vpn1/0`

Method: `DELETE`

References

- BGP/MPLS IP Virtual Private Networks (VPNs)
- BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN
- BGP/MPLS VPN Virtual PE

2.1.8 Link-State Family

The BGP Link-State (BGP-LS) Multiprotocol extension allows to distribute Link-State and Traffic Engineering (TE) information. This information is typically distributed by IGP routing protocols with in the network, limiting LSDB or TED visibility to the IGP area. The BGP-LS-enabled routers are capable to collect such information from networks (multiple IGP areas, inter-AS) and share with external components (i.e. OpenDaylight BGP). The information is applicable in ALTO servers and PCEs, as both need to gather information about topologies. In addition, link-state information is extended to carry segment information (Spring).

Contents

- *Configuration*
  - *BGP Speaker*
  - *Linkstate path attribute*
  - *BGP Peer*
- *Link-State Route API*
- *Usage*
- *References*

Configuration

This section shows a way to enable IPv4 and IPv6 Labeled Unicast family in BGP speaker and peer configuration.

BGP Speaker

To enable BGP-LS support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name>LINKSTATE</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

Linkstate path attribute

IANA allocation for BGP-LS path attribute is TYPE 29. Some older BGP-LS implementations might still require earliest asigned allocation TYPE 99. To use TYPE = 99, you need to set value bellow to false.

URL: `/restconf/config/bgp-linkstate-app-config:bgp-linkstate-app-config`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<bgp-linkstate-app-config xmlns=
↪"urn:opendaylight:params:xml:ns:yang:controller:bgp:linkstate-app-config">
    <iana-linkstate-attribute-type>false</iana-linkstate-attribute-type>
</bgp-linkstate-app-config>
```

BGP Peer

Here is an example for BGP peer configuration with enabled BGP-LS family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name>LINKSTATE</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

Link-State Route API

Following tree illustrate the BGP Link-State route structure.

```
:(linkstate-routes-case)
   +--ro linkstate-routes
      +--ro linkstate-route* [route-key path-id]
         +--ro route-key                       string
         +--ro path-id                         path-id
         +--ro protocol-id                     protocol-id
         +--ro identifier                      identifier
         +--ro (object-type)?
         |  +--:(node-case)
         |  |  +--ro node-descriptors
         |  |     +--ro as-number?          inet:as-number
         |  |     +--ro area-id?            area-identifier
```

(continues on next page)

(continued from previous page)

```
|  |     +--ro domain-id?         domain-identifier
|  |     +--ro (c-router-identifier)?
|  |        +--:(isis-node-case)
|  |        |  +--ro isis-node
|  |        |     +--ro iso-system-id    netc:iso-system-identifier
|  |        +--:(isis-pseudonode-case)
|  |        |  +--ro isis-pseudonode
|  |        |     +--ro is-is-router-identifier
|  |        |     |  +--ro iso-system-id    netc:iso-system-identifier
|  |        |     +--ro psn                        uint8
|  |        +--:(ospf-node-case)
|  |        |  +--ro ospf-node
|  |        |     +--ro ospf-router-id    uint32
|  |        +--:(ospf-pseudonode-case)
|  |           +--ro ospf-pseudonode
|  |              +--ro ospf-router-id    uint32
|  |              +--ro lan-interface     ospf-interface-identifier
|  +--:(link-case)
|  |  +--ro local-node-descriptors
|  |  |  +--ro as-number?         inet:as-number
|  |  |  +--ro area-id?           area-identifier
|  |  |  +--ro domain-id?         domain-identifier
|  |  |  +--ro (c-router-identifier)?
|  |  |  |  +--:(isis-node-case)
|  |  |  |  |  +--ro isis-node
|  |  |  |  |     +--ro iso-system-id    netc:iso-system-identifier
|  |  |  |  +--:(isis-pseudonode-case)
|  |  |  |  |  +--ro isis-pseudonode
|  |  |  |  |     +--ro is-is-router-identifier
|  |  |  |  |     |  +--ro iso-system-id    netc:iso-system-identifier
|  |  |  |  |     +--ro psn                        uint8
|  |  |  |  +--:(ospf-node-case)
|  |  |  |  |  +--ro ospf-node
|  |  |  |  |     +--ro ospf-router-id    uint32
|  |  |  |  +--:(ospf-pseudonode-case)
|  |  |  |     +--ro ospf-pseudonode
|  |  |  |        +--ro ospf-router-id    uint32
|  |  |  |        +--ro lan-interface     ospf-interface-identifier
|  |  |  +--ro bgp-router-id?     inet:ipv4-address
|  |  |  +--ro member-asn?        inet:as-number
|  |  +--ro remote-node-descriptors
|  |  |  +--ro as-number?         inet:as-number
|  |  |  +--ro area-id?           area-identifier
|  |  |  +--ro domain-id?         domain-identifier
|  |  |  +--ro (c-router-identifier)?
|  |  |  |  +--:(isis-node-case)
|  |  |  |  |  +--ro isis-node
|  |  |  |  |     +--ro iso-system-id    netc:iso-system-identifier
|  |  |  |  +--:(isis-pseudonode-case)
|  |  |  |  |  +--ro isis-pseudonode
|  |  |  |  |     +--ro is-is-router-identifier
|  |  |  |  |     |  +--ro iso-system-id    netc:iso-system-identifier
|  |  |  |  |     +--ro psn                        uint8
|  |  |  |  +--:(ospf-node-case)
|  |  |  |  |  +--ro ospf-node
|  |  |  |  |     +--ro ospf-router-id    uint32
|  |  |  |  +--:(ospf-pseudonode-case)
```

(continues on next page)

(continued from previous page)

```
|  |  |  |     +--ro ospf-pseudonode
|  |  |  |        +--ro ospf-router-id    uint32
|  |  |  |        +--ro lan-interface     ospf-interface-identifier
|  |  |  +--ro bgp-router-id?     inet:ipv4-address
|  |  |  +--ro member-asn?        inet:as-number
|  |  +--ro link-descriptors
|  |     +--ro link-local-identifier?    uint32
|  |     +--ro link-remote-identifier?   uint32
|  |     +--ro ipv4-interface-address?   ipv4-interface-identifier
|  |     +--ro ipv6-interface-address?   ipv6-interface-identifier
|  |     +--ro ipv4-neighbor-address?    ipv4-interface-identifier
|  |     +--ro ipv6-neighbor-address?    ipv6-interface-identifier
|  |     +--ro multi-topology-id?        topology-identifier
|  +--:(prefix-case)
|  |  +--ro advertising-node-descriptors
|  |  |  +--ro as-number?         inet:as-number
|  |  |  +--ro area-id?           area-identifier
|  |  |  +--ro domain-id?         domain-identifier
|  |  |  +--ro (c-router-identifier)?
|  |  |     +--:(isis-node-case)
|  |  |     |  +--ro isis-node
|  |  |     |     +--ro iso-system-id    netc:iso-system-identifier
|  |  |     +--:(isis-pseudonode-case)
|  |  |     |  +--ro isis-pseudonode
|  |  |     |     +--ro is-is-router-identifier
|  |  |     |     |  +--ro iso-system-id    netc:iso-system-identifier
|  |  |     |     +--ro psn                        uint8
|  |  |     +--:(ospf-node-case)
|  |  |     |  +--ro ospf-node
|  |  |     |     +--ro ospf-router-id    uint32
|  |  |     +--:(ospf-pseudonode-case)
|  |  |        +--ro ospf-pseudonode
|  |  |           +--ro ospf-router-id    uint32
|  |  |           +--ro lan-interface     ospf-interface-identifier
|  |  +--ro prefix-descriptors
|  |     +--ro multi-topology-id?             topology-identifier
|  |     +--ro ospf-route-type?               ospf-route-type
|  |     +--ro ip-reachability-information?   inet:ip-prefix
|  +--:(te-lsp-case)
|     +--ro (address-family)?
|     |  +--:(ipv4-case)
|     |  |  +--ro ipv4-tunnel-sender-address      inet:ipv4-address
|     |  |  +--ro ipv4-tunnel-endpoint-address    inet:ipv4-address
|     |  +--:(ipv6-case)
|     |     +--ro ipv6-tunnel-sender-address      inet:ipv6-address
|     |     +--ro ipv6-tunnel-endpoint-address    inet:ipv6-address
|     +--ro tunnel-id?                        rsvp:tunnel-id
|     +--ro lsp-id?                           rsvp:lsp-id
+--ro attributes
   +--ro (link-state-attribute)?
      +--:(node-attributes-case)
      |  +--ro node-attributes
      |     +--ro topology-identifier*   topology-identifier
      |     +--ro node-flags?            node-flag-bits
      |     +--ro isis-area-id*          isis-area-identifier
      |     +--ro dynamic-hostname?      string
      |     +--ro ipv4-router-id?        ipv4-router-identifier
```

(continues on next page)

(continued from previous page)

```
|     +--ro ipv6-router-id?        ipv6-router-identifier
|     +--ro sr-capabilities
|     |  +--ro mpls-ipv4?      boolean
|     |  +--ro mpls-ipv6?      boolean
|     |  +--ro sr-ipv6?        boolean
|     |  +--ro range-size?     uint32
|     |  +--ro (sid-label-index)?
|     |     +--:(local-label-case)
|     |     |  +--ro local-label?    netc:mpls-label
|     |     +--:(ipv6-address-case)
|     |     |  +--ro ipv6-address?   inet:ipv6-address
|     |     +--:(sid-case)
|     |        +--ro sid?            uint32
|     +--ro sr-algorithm
|        +--ro algorithms*   algorithm
+--:(link-attributes-case)
|  +--ro link-attributes
|     +--ro local-ipv4-router-id?       ipv4-router-identifier
|     +--ro local-ipv6-router-id?       ipv6-router-identifier
|     +--ro remote-ipv4-router-id?      ipv4-router-identifier
|     +--ro remote-ipv6-router-id?      ipv6-router-identifier
|     +--ro mpls-protocol?              mpls-protocol-mask
|     +--ro te-metric?                  netc:te-metric
|     +--ro metric?                     netc:metric
|     +--ro shared-risk-link-groups*    rsvp:srlg-id
|     +--ro link-name?                  string
|     +--ro max-link-bandwidth?         netc:bandwidth
|     +--ro max-reservable-bandwidth?   netc:bandwidth
|     +--ro unreserved-bandwidth* [priority]
|     |  +--ro priority     uint8
|     |  +--ro bandwidth?   netc:bandwidth
|     +--ro link-protection?            link-protection-type
|     +--ro admin-group?                administrative-group
|     +--ro sr-adj-ids*
|     |  +--ro (flags)?
|     |  |  +--:(ospf-adj-flags-case)
|     |  |  |  +--ro backup?           boolean
|     |  |  |  +--ro set?              boolean
|     |  |  +--:(isis-adj-flags-case)
|     |  |     +--ro backup?           boolean
|     |  |     +--ro set?              boolean
|     |  |     +--ro address-family?   boolean
|     |  +--ro weight?           weight
|     |  +--ro (sid-label-index)?
|     |     +--:(local-label-case)
|     |     |  +--ro local-label?      netc:mpls-label
|     |     +--:(ipv6-address-case)
|     |     |  +--ro ipv6-address?     inet:ipv6-address
|     |     +--:(sid-case)
|     |        +--ro sid?              uint32
|     +--ro sr-lan-adj-ids*
|     |  +--ro (flags)?
|     |  |  +--:(ospf-adj-flags-case)
|     |  |  |  +--ro backup?           boolean
|     |  |  |  +--ro set?              boolean
|     |  |  +--:(isis-adj-flags-case)
|     |  |     +--ro backup?           boolean
```

(continues on next page)

(continued from previous page)

```
|     |  |     +--ro set?              boolean
|     |  |     +--ro address-family?   boolean
|     |  +--ro weight?           weight
|     |  +--ro iso-system-id?    netc:iso-system-identifier
|     |  +--ro neighbor-id?      inet:ipv4-address
|     |  +--ro (sid-label-index)?
|     |     +--:(local-label-case)
|     |     |  +--ro local-label?      netc:mpls-label
|     |     +--:(ipv6-address-case)
|     |     |  +--ro ipv6-address?     inet:ipv6-address
|     |     +--:(sid-case)
|     |        +--ro sid?              uint32
|     +--ro peer-node-sid
|     |  +--ro weight?         weight
|     |  +--ro (sid-label-index)?
|     |     +--:(local-label-case)
|     |     |  +--ro local-label?    netc:mpls-label
|     |     +--:(ipv6-address-case)
|     |     |  +--ro ipv6-address?   inet:ipv6-address
|     |     +--:(sid-case)
|     |        +--ro sid?            uint32
|     +--ro peer-adj-sid
|     |  +--ro weight?         weight
|     |  +--ro (sid-label-index)?
|     |     +--:(local-label-case)
|     |     |  +--ro local-label?    netc:mpls-label
|     |     +--:(ipv6-address-case)
|     |     |  +--ro ipv6-address?   inet:ipv6-address
|     |     +--:(sid-case)
|     |        +--ro sid?            uint32
|     +--ro peer-set-sids*
|        +--ro weight?         weight
|        +--ro (sid-label-index)?
|           +--:(local-label-case)
|           |  +--ro local-label?    netc:mpls-label
|           +--:(ipv6-address-case)
|           |  +--ro ipv6-address?   inet:ipv6-address
|           +--:(sid-case)
|              +--ro sid?            uint32
+--:(prefix-attributes-case)
|  +--ro prefix-attributes
|     +--ro igp-bits
|     |  x--ro up-down?               bits
|     |  +--ro is-is-up-down?         boolean
|     |  +--ro ospf-no-unicast?       boolean
|     |  +--ro ospf-local-address?    boolean
|     |  +--ro ospf-propagate-nssa?   boolean
|     +--ro route-tags*                route-tag
|     +--ro extended-tags*             extended-route-tag
|     +--ro prefix-metric?             netc:igp-metric
|     +--ro ospf-forwarding-address?   inet:ip-address
|     +--ro sr-prefix
|     |  +--ro (flags)?
|     |  |  +--:(isis-prefix-flags-case)
|     |  |  |  +--ro no-php?            boolean
|     |  |  |  +--ro explicit-null?     boolean
|     |  |  |  +--ro readvertisement?   boolean
```

(continues on next page)

(continued from previous page)

```
                |      |  |  |  +--ro node-sid?          boolean
                |      |  |  +--:(ospf-prefix-flags-case)
                |      |  |     +--ro no-php?            boolean
                |      |  |     +--ro explicit-null?     boolean
                |      |  |     +--ro mapping-server?    boolean
                |      |  +--ro algorithm?         algorithm
                |      |  +--ro (sid-label-index)?
                |      |     +--:(local-label-case)
                |      |     |  +--ro local-label?       netc:mpls-label
                |      |     +--:(ipv6-address-case)
                |      |     |  +--ro ipv6-address?      inet:ipv6-address
                |      |     +--:(sid-case)
                |      |        +--ro sid?               uint32
                |      +--ro ipv6-sr-prefix
                |      |  +--ro algorithm?   algorithm
                |      +--ro sr-range
                |      |  +--ro inter-area?   boolean
                |      |  +--ro range-size?   uint16
                |      |  +--ro sub-tlvs*
                |      |     +--ro (range-sub-tlv)?
                |      |        +--:(binding-sid-tlv-case)
                |      |        |  +--ro weight?                weight
                |      |        |  +--ro (flags)?
                |      |        |  |  +--:(isis-binding-flags-case)
                |      |        |  |  |  +--ro address-family?        boolean
                |      |        |  |  |  +--ro mirror-context?        boolean
                |      |        |  |  |  +--ro spread-tlv?            boolean
                |      |        |  |  |  +--ro leaked-from-level-2?   boolean
                |      |        |  |  |  +--ro attached-flag?         boolean
                |      |        |  |  +--:(ospf-binding-flags-case)
                |      |        |  |     +--ro mirroring?             boolean
                |      |        |  +--ro binding-sub-tlvs*
                |      |        |     +--ro (binding-sub-tlv)?
                |      |        |        +--:(prefix-sid-case)
                |      |        |        |  +--ro (flags)?
                |      |        |        |  |  +--:(isis-prefix-flags-case)
                |      |        |        |  |  |  +--ro no-php?            boolean
                |      |        |        |  |  |  +--ro explicit-null?     boolean
                |      |        |        |  |  |  +--ro readvertisement?   boolean
                |      |        |        |  |  |  +--ro node-sid?          boolean
                |      |        |        |  |  +--:(ospf-prefix-flags-case)
                |      |        |        |  |     +--ro no-php?            boolean
                |      |        |        |  |     +--ro explicit-null?     boolean
                |      |        |        |  |     +--ro mapping-server?    boolean
                |      |        |        |  +--ro algorithm?         algorithm
                |      |        |        |  +--ro (sid-label-index)?
                |      |        |        |     +--:(local-label-case)
                |      |        |        |     |  +--ro local-label?       netc:mpls-
↪label
                |      |        |        |     +--:(ipv6-address-case)
                |      |        |        |     |  +--ro ipv6-address?      inet:ipv6-
↪address
                |      |        |        |     +--:(sid-case)
                |      |        |        |        +--ro sid?               uint32
                |      |        |        +--:(ipv6-prefix-sid-case)
                |      |        |        |  +--ro algorithm?         algorithm
                |      |        |        +--:(sid-label-case)
```

(continues on next page)

(continued from previous page)

```
                |      |         |         |  +--ro (sid-label-index)?
                |      |         |         |     +--:(local-label-case)
                |      |         |         |     |  +--ro local-label?       netc:mpls-
↪label
                |      |         |         |     +--:(ipv6-address-case)
                |      |         |         |     |  +--ro ipv6-address?      inet:ipv6-
↪address
                |      |         |         |     +--:(sid-case)
                |      |         |         |        +--ro sid?               uint32
                |      |         |        +--:(ero-metric-case)
                |      |         |        |  +--ro ero-metric?        netc:te-metric
                |      |         |        +--:(ipv4-ero-case)
                |      |         |        |  +--ro loose?             boolean
                |      |         |        |  +--ro address            inet:ipv4-address
                |      |         |        +--:(ipv6-ero-case)
                |      |         |        |  +--ro loose?             boolean
                |      |         |        |  +--ro address            inet:ipv6-address
                |      |         |        +--:(unnumbered-interface-id-ero-case)
                |      |         |        |  +--ro loose?             boolean
                |      |         |        |  +--ro router-id?         uint32
                |      |         |        |  +--ro interface-id?      uint32
                |      |         |        +--:(ipv4-ero-backup-case)
                |      |         |        |  +--ro loose?             boolean
                |      |         |        |  +--ro address            inet:ipv4-address
                |      |         |        +--:(ipv6-ero-backup-case)
                |      |         |        |  +--ro loose?             boolean
                |      |         |        |  +--ro address            inet:ipv6-address
                |      |         |        +--:(unnumbered-interface-id-backup-ero-case)
                |      |         |           +--ro loose?             boolean
                |      |         |           +--ro router-id?         uint32
                |      |         |           +--ro interface-id?      uint32
                |      |         +--:(prefix-sid-tlv-case)
                |      |         |  +--ro (flags)?
                |      |         |  |  +--:(isis-prefix-flags-case)
                |      |         |  |  |  +--ro no-php?                boolean
                |      |         |  |  |  +--ro explicit-null?         boolean
                |      |         |  |  |  +--ro readvertisement?       boolean
                |      |         |  |  |  +--ro node-sid?              boolean
                |      |         |  |  +--:(ospf-prefix-flags-case)
                |      |         |  |     +--ro no-php?                boolean
                |      |         |  |     +--ro explicit-null?         boolean
                |      |         |  |     +--ro mapping-server?        boolean
                |      |         |  +--ro algorithm?             algorithm
                |      |         |  +--ro (sid-label-index)?
                |      |         |     +--:(local-label-case)
                |      |         |     |  +--ro local-label?           netc:mpls-label
                |      |         |     +--:(ipv6-address-case)
                |      |         |     |  +--ro ipv6-address?          inet:ipv6-address
                |      |         |     +--:(sid-case)
                |      |         |        +--ro sid?                   uint32
                |      |         +--:(ipv6-prefix-sid-tlv-case)
                |      |         |  +--ro algorithm?             algorithm
                |      |         +--:(sid-label-tlv-case)
                |      |            +--ro (sid-label-index)?
                |      |               +--:(local-label-case)
                |      |               |  +--ro local-label?           netc:mpls-label
                |      |               +--:(ipv6-address-case)
```

(continues on next page)

(continued from previous page)

```
|     |              |  +--ro ipv6-address?          inet:ipv6-address
|     |              +--:(sid-case)
|     |                 +--ro sid?                    uint32
|     +--ro sr-binding-sid-labels*
|        +--ro weight?               weight
|        +--ro (flags)?
|        |  +--:(isis-binding-flags-case)
|        |  |  +--ro address-family?        boolean
|        |  |  +--ro mirror-context?        boolean
|        |  |  +--ro spread-tlv?            boolean
|        |  |  +--ro leaked-from-level-2?   boolean
|        |  |  +--ro attached-flag?         boolean
|        |  +--:(ospf-binding-flags-case)
|        |     +--ro mirroring?             boolean
|        +--ro binding-sub-tlvs*
|           +--ro (binding-sub-tlv)?
|              +--:(prefix-sid-case)
|              |  +--ro (flags)?
|              |  |  +--:(isis-prefix-flags-case)
|              |  |  |  +--ro no-php?            boolean
|              |  |  |  +--ro explicit-null?     boolean
|              |  |  |  +--ro readvertisement?   boolean
|              |  |  |  +--ro node-sid?          boolean
|              |  |  +--:(ospf-prefix-flags-case)
|              |  |     +--ro no-php?            boolean
|              |  |     +--ro explicit-null?     boolean
|              |  |     +--ro mapping-server?    boolean
|              |  +--ro algorithm?         algorithm
|              |  +--ro (sid-label-index)?
|              |     +--:(local-label-case)
|              |     |  +--ro local-label?       netc:mpls-label
|              |     +--:(ipv6-address-case)
|              |     |  +--ro ipv6-address?      inet:ipv6-address
|              |     +--:(sid-case)
|              |        +--ro sid?               uint32
|              +--:(ipv6-prefix-sid-case)
|              |  +--ro algorithm?         algorithm
|              +--:(sid-label-case)
|              |  +--ro (sid-label-index)?
|              |     +--:(local-label-case)
|              |     |  +--ro local-label?       netc:mpls-label
|              |     +--:(ipv6-address-case)
|              |     |  +--ro ipv6-address?      inet:ipv6-address
|              |     +--:(sid-case)
|              |        +--ro sid?               uint32
|              +--:(ero-metric-case)
|              |  +--ro ero-metric?        netc:te-metric
|              +--:(ipv4-ero-case)
|              |  +--ro loose?             boolean
|              |  +--ro address            inet:ipv4-address
|              +--:(ipv6-ero-case)
|              |  +--ro loose?             boolean
|              |  +--ro address            inet:ipv6-address
|              +--:(unnumbered-interface-id-ero-case)
|              |  +--ro loose?             boolean
|              |  +--ro router-id?         uint32
|              |  +--ro interface-id?      uint32
```

(continues on next page)

(continued from previous page)

```
|              +--:(ipv4-ero-backup-case)
|              |  +--ro loose?              boolean
|              |  +--ro address             inet:ipv4-address
|              +--:(ipv6-ero-backup-case)
|              |  +--ro loose?              boolean
|              |  +--ro address             inet:ipv6-address
|              +--:(unnumbered-interface-id-backup-ero-case)
|                 +--ro loose?              boolean
|                 +--ro router-id?          uint32
|                 +--ro interface-id?       uint32
x--:(te-lsp-attributes-case)
   +--ro te-lsp-attributes
```

Usage

The Link-State table in a instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-linkstate:linkstate-address-family/bgp-linkstate:linkstate-subsequent-address-family/linkstate-routes`

Method: `GET`

Response Body:

```
<linkstate-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-linkstate">
   ...
</linkstate-routes>
```

Note: Link-State routes mapping to topology links/nodes/prefixes is supported by BGP Topology Provider.

References

- North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP
- BGP Link-State extensions for Segment Routing
- Segment Routing BGP Egress Peer Engineering BGP-LS Extensions
- BGP Link-State Information Distribution Implementation Report

2.1.9 Flow Specification Family

The BGP Flow Specification (BGP-FS) Multiprotocol extension can be used to distribute traffic flow specifications. For example, the BGP-FS can be used in a case of (distributed) denial-of-service (DDoS) attack mitigation procedures and traffic filtering (BGP/MPLS VPN service, DC).

Contents

- *Configuration*
  - *BGP Speaker*

- *BGP Peer*
- *Flow Specification API*
  - *IPv4 Flow Specification Route*
  - *IPv6 Flow Specification Route*
- *Usage*
  - *IPv4 Flow Specification*
  - *IPv6 Flows Specification*
  - *IPv4 L3VPN Flows Specification*
- *Programming*
  - *IPv4 Flow Specification*
  - *IPv4 L3VPN Flow Specification*
  - *IPv6 Flow Specification*
- *References*

Configuration

This section shows a way to enable BGP-FS family in BGP speaker and peer configuration.

BGP Speaker

To enable BGP-FS support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name>IPV4-FLOW</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name>IPV6-FLOW</afi-safi-name>
                </afi-safi>
                <afi-safi>
```

(continues on next page)

(continued from previous page)

```
                    <afi-safi-name>IPV4-L3VPN-FLOW</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name>IPV6-L3VPN-FLOW</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

BGP Peer

Here is an example for BGP peer configuration with enabled BGP-FS family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name>IPV4-FLOW</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name>IPV6-FLOW</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name>IPV4-L3VPN-FLOW</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name>IPV6-L3VPN-FLOW</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

Flow Specification API

Following trees illustrate the BGP Flow Specification routes structure.

IPv4 Flow Specification Route

```
:(flowspec-routes-case)
  +--ro flowspec-routes
     +--ro flowspec-route* [route-key path-id]
        +--ro route-key     string
        +--ro flowspec*
        |  +--ro (flowspec-type)?
        |     +--:(port-case)
        |     |  +--ro ports*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint16
        |     +--:(destination-port-case)
        |     |  +--ro destination-ports*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint16
        |     +--:(source-port-case)
        |     |  +--ro source-ports*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint16
        |     +--:(icmp-type-case)
        |     |  +--ro types*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint8
        |     +--:(icmp-code-case)
        |     |  +--ro codes*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint8
        |     +--:(tcp-flags-case)
        |     |  +--ro tcp-flags*
        |     |     +--ro op?      bitmask-operand
        |     |     +--ro value?   uint16
        |     +--:(packet-length-case)
        |     |  +--ro packet-lengths*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint16
        |     +--:(dscp-case)
        |     |  +--ro dscps*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   dscp
        |     +--:(fragment-case)
        |     |  +--ro fragments*
        |     |     +--ro op?      bitmask-operand
        |     |     +--ro value?   fragment
        |     +--:(destination-prefix-case)
        |     |  +--ro destination-prefix?   inet:ipv4-prefix
        |     +--:(source-prefix-case)
        |     |  +--ro source-prefix?        inet:ipv4-prefix
        |     +--:(protocol-ip-case)
        |        +--ro protocol-ips*
        |           +--ro op?      numeric-operand
        |           +--ro value?   uint8
```

(continues on next page)

(continued from previous page)

```
        +--ro path-id       path-id
        +--ro attributes
           +--ro extended-communities*
              +--ro transitive?                              boolean
              +--ro (extended-community)?
                 +--:(traffic-rate-extended-community-case)
                 |  +--ro traffic-rate-extended-community
                 |     +--ro informative-as?        bgp-t:short-as-number
                 |     +--ro local-administrator?   netc:bandwidth
                 +--:(traffic-action-extended-community-case)
                 |  +--ro traffic-action-extended-community
                 |     +--ro sample?            boolean
                 |     +--ro terminal-action?   boolean
                 +--:(redirect-extended-community-case)
                 |  +--ro redirect-extended-community
                 |     +--ro global-administrator?   bgp-t:short-as-number
                 |     +--ro local-administrator?    binary
                 +--:(traffic-marking-extended-community-case)
                 |  +--ro traffic-marking-extended-community
                 |     +--ro global-administrator?   dscp
                 +--:(redirect-ipv4-extended-community-case)
                 |  +--ro redirect-ipv4
                 |     +--ro global-administrator?   inet:ipv4-address
                 |     +--ro local-administrator?    uint16
                 +--:(redirect-as4-extended-community-case)
                 |  +--ro redirect-as4
                 |     +--ro global-administrator?   inet:as-number
                 |     +--ro local-administrator?    uint16
                 +--:(redirect-ip-nh-extended-community-case)
                   +--ro redirect-ip-nh-extended-community
                      +--ro next-hop-address?   inet:ip-address
                      +--ro copy?               boolean
```

IPv6 Flow Specification Route

```
:(flowspec-ipv6-routes-case)
  +--ro flowspec-ipv6-routes
     +--ro flowspec-route* [route-key path-id]
        +--ro flowspec*
        |  +--ro (flowspec-type)?
        |     +--:(port-case)
        |     |  +--ro ports*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint16
        |     +--:(destination-port-case)
        |     |  +--ro destination-ports*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint16
        |     +--:(source-port-case)
        |     |  +--ro source-ports*
        |     |     +--ro op?      numeric-operand
        |     |     +--ro value?   uint16
        |     +--:(icmp-type-case)
        |     |  +--ro types*
```

(continues on next page)

(continued from previous page)

```
|      |     +--ro op?      numeric-operand
|      |     +--ro value?   uint8
|      +--:(icmp-code-case)
|      |  +--ro codes*
|      |     +--ro op?      numeric-operand
|      |     +--ro value?   uint8
|      +--:(tcp-flags-case)
|      |  +--ro tcp-flags*
|      |     +--ro op?      bitmask-operand
|      |     +--ro value?   uint16
|      +--:(packet-length-case)
|      |  +--ro packet-lengths*
|      |     +--ro op?      numeric-operand
|      |     +--ro value?   uint16
|      +--:(dscp-case)
|      |  +--ro dscps*
|      |     +--ro op?      numeric-operand
|      |     +--ro value?   dscp
|      +--:(fragment-case)
|      |  +--ro fragments*
|      |     +--ro op?      bitmask-operand
|      |     +--ro value?   fragment
|      +--:(destination-ipv6-prefix-case)
|      |  +--ro destination-prefix?   inet:ipv6-prefix
|      +--:(source-ipv6-prefix-case)
|      |  +--ro source-prefix?        inet:ipv6-prefix
|      +--:(next-header-case)
|      |  +--ro next-headers*
|      |     +--ro op?      numeric-operand
|      |     +--ro value?   uint8
|      +--:(flow-label-case)
|         +--ro flow-label*
|            +--ro op?      numeric-operand
|            +--ro value?   uint32
+--ro path-id       path-id
+--ro attributes
   +--ro extended-communities*
      +--ro transitive?                              boolean
      +--ro (extended-community)?
         +--:(traffic-rate-extended-community-case)
         |  +--ro traffic-rate-extended-community
         |     +--ro informative-as?        bgp-t:short-as-number
         |     +--ro local-administrator?   netc:bandwidth
         +--:(traffic-action-extended-community-case)
         |  +--ro traffic-action-extended-community
         |     +--ro sample?            boolean
         |     +--ro terminal-action?   boolean
         +--:(redirect-extended-community-case)
         |  +--ro redirect-extended-community
         |     +--ro global-administrator?   bgp-t:short-as-number
         |     +--ro local-administrator?    binary
         +--:(traffic-marking-extended-community-case)
         |  +--ro traffic-marking-extended-community
         |     +--ro global-administrator?   dscp
         +--:(redirect-ipv6-extended-community-case)
         |  +--ro redirect-ipv6
         |     +--ro global-administrator?   inet:ipv6-address
```

(continues on next page)

(continued from previous page)

```
|     +--ro local-administrator?    uint16
+--:(redirect-as4-extended-community-case)
|  +--ro redirect-as4
|     +--ro global-administrator?   inet:as-number
|     +--ro local-administrator?    uint16
+--:(redirect-ip-nh-extended-community-case)
   +--ro redirect-ip-nh-extended-community
      +--ro next-hop-address?   inet:ip-address
      +--ro copy?               boolean
```

Usage

The flowspec route represents rules and an action, defined as an extended community.

IPv4 Flow Specification

The IPv4 Flowspec table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-flowspec:flowspec-subsequent-address-family/bgp-flowspec:flowspec-routes`

Method: `GET`

Response Body:

```
<flowspec-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-flowspec">
    <flowspec-route>
        <path-id>0</path-id>
        <route-key>all packets to 192.168.0.1/32 AND from 10.0.0.2/32 AND where IP␣
↪protocol equals to 17 or equals to 6 AND where port equals to 80 or equals to 8080␣
↪AND where destination port is greater than 8080 and is less than 8088 or equals to␣
↪3128 AND where source port is greater than 1024 </route-key>
        <attributes>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <origin>
                <value>igp</value>
            </origin>
            <as-path></as-path>
            <extended-communities>
                <transitive>true</transitive>
                <redirect-extended-community>
                    <local-administrator>AgMWLg==</local-administrator>
                    <global-administrator>258</global-administrator>
                </redirect-extended-community>
            </extended-communities>
        </attributes>
        <flowspec>
            <destination-prefix>192.168.0.1/32</destination-prefix>
        </flowspec>
        <flowspec>
            <source-prefix>10.0.0.2/32</source-prefix>
        </flowspec>
```

(continues on next page)

(continued from previous page)

```
        <flowspec>
            <protocol-ips>
                <op>equals</op>
                <value>17</value>
            </protocol-ips>
            <protocol-ips>
                <op>equals end-of-list</op>
                <value>6</value>
            </protocol-ips>
        </flowspec>
        <flowspec>
            <ports>
                <op>equals</op>
                <value>80</value>
            </ports>
            <ports>
                <op>equals end-of-list</op>
                <value>8080</value>
            </ports>
        </flowspec>
        <flowspec>
            <destination-ports>
                <op>greater-than</op>
                <value>8080</value>
            </destination-ports>
            <destination-ports>
                <op>less-than and-bit</op>
                <value>8088</value>
            </destination-ports>
            <destination-ports>
                <op>equals end-of-list</op>
                <value>3128</value>
            </destination-ports>
        </flowspec>
        <flowspec>
            <source-ports>
                <op>end-of-list greater-than</op>
                <value>1024</value>
            </source-ports>
        </flowspec>
    </flowspec-route>
</flowspec-routes>
```

IPv6 Flows Specification

The IPv6 Flowspec table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv6-address-family/bgp-flowspec:flowspec-subsequent-address-family/bgp-flowspec:flowspec-ipv6-routes`

Method: `GET`

Response Body:

```
<flowspec-ipv6-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-flowspec">
    <flowspec-route>
        <path-id>0</path-id>
        <route-key>all packets to 2001:db8:31::/64 AND from 2001:db8:30::/64 AND␣
↪where next header equals to 17 AND where DSCP equals to 50 AND where flow label␣
↪equals to 2013 </route-key>
        <attributes>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <origin>
                <value>igp</value>
            </origin>
            <as-path></as-path>
            <extended-communities>
                <transitive>true</transitive>
                <traffic-rate-extended-community>
                    <informative-as>0</informative-as>
                    <local-administrator>AAAAAA==</local-administrator>
                </traffic-rate-extended-community>
            </extended-communities>
        </attributes>
        <flowspec>
            <destination-prefix>2001:db8:31::/64</destination-prefix>
        </flowspec>
        <flowspec>
            <source-prefix>2001:db8:30::/64</source-prefix>
        </flowspec>
        <flowspec>
            <next-headers>
                <op>equals end-of-list</op>
                <value>17</value>
            </next-headers>
        </flowspec>
        <flowspec>
            <dscps>
                <op>equals end-of-list</op>
                <value>50</value>
            </dscps>
        </flowspec>
        <flowspec>
            <flow-label>
                <op>equals end-of-list</op>
                <value>2013</value>
            </flow-label>
        </flowspec>
    </flowspec-route>
</flowspec-ipv6-routes>
```

IPv4 L3VPN Flows Specification

The IPv4 L3VPN Flowspec table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-flowspec:flowspec-l3vpn-subsequent-address-family/bgp-flowspec:flowspec-l3vpn-ipv4-routes`

Method: `GET`

Response Body:

```
<flowspec-l3vpn-ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-flowspec">
    <flowspec-l3vpn-route>
        <path-id>0</path-id>
        <route-key>[l3vpn with route-distinguisher 172.16.0.44:101] all packets from␣
↪10.0.0.3/32</route-key>
        <attributes>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <ipv4-next-hop>
                <global>5.6.7.8</global>
            </ipv4-next-hop>
            <origin>
                <value>igp</value>
            </origin>
            <as-path></as-path>
            <extended-communities>
                <transitive>true</transitive>
                <redirect-ip-nh-extended-community>
                    <copy>false</copy>
                    <next-hop-address>0.0.0.0</next-hop-address>
                </redirect-ip-nh-extended-community>
            </extended-communities>
        </attributes>
        <route-distinguisher>172.16.0.44:101</route-distinguisher>
        <flowspec>
            <source-prefix>10.0.0.3/32</source-prefix>
        </flowspec>
    </flowspec-l3vpn-route>
</flowspec-l3vpn-ipv4-routes>
```

Programming

IPv4 Flow Specification

This examples show how to originate and remove IPv4 fowspec route via programmable RIB. Make sure the *Application Peer* is configured first.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-flowspec:flowspec-subsequent-address-family/bgp-flowspec:flowspec-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<flowspec-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-flowspec">
    <route-key>flow1</route-key>
    <path-id>0</path-id>
    <flowspec>
        <destination-prefix>192.168.0.1/32</destination-prefix>
    </flowspec>
    <flowspec>
        <source-prefix>10.0.0.1/32</source-prefix>
    </flowspec>
    <flowspec>
        <protocol-ips>
            <op>equals end-of-list</op>
            <value>6</value>
        </protocol-ips>
    </flowspec>
    <flowspec>
        <ports>
            <op>equals end-of-list</op>
            <value>80</value>
        </ports>
    </flowspec>
    <flowspec>
        <destination-ports>
            <op>greater-than</op>
            <value>8080</value>
        </destination-ports>
        <destination-ports>
            <op>and-bit less-than end-of-list</op>
            <value>8088</value>
        </destination-ports>
    </flowspec>
    <flowspec>
        <source-ports>
            <op>greater-than end-of-list</op>
            <value>1024</value>
        </source-ports>
    </flowspec>
    <flowspec>
        <types>
            <op>equals end-of-list</op>
            <value>0</value>
        </types>
    </flowspec>
    <flowspec>
        <codes>
            <op>equals end-of-list</op>
            <value>0</value>
        </codes>
    </flowspec>
    <flowspec>
        <tcp-flags>
            <op>match end-of-list</op>
            <value>32</value>
        </tcp-flags>
    </flowspec>
    <flowspec>
```

(continues on next page)

(continued from previous page)

```
        <packet-lengths>
            <op>greater-than</op>
            <value>400</value>
        </packet-lengths>
        <packet-lengths>
            <op>and-bit less-than end-of-list</op>
            <value>500</value>
        </packet-lengths>
    </flowspec>
    <flowspec>
        <dscps>
            <op>equals end-of-list</op>
            <value>20</value>
        </dscps>
    </flowspec>
    <flowspec>
        <fragments>
            <op>match end-of-list</op>
            <value>first</value>
        </fragments>
    </flowspec>
    <attributes>
        <origin>
            <value>igp</value>
        </origin>
        <as-path/>
        <local-pref>
            <pref>100</pref>
        </local-pref>
        <extended-communities>
            ....
        </extended-communities>
    </attributes>
</flowspec-route>
```

Extended Communities

- **Traffic Rate**

```
1 <extended-communities>
2     <transitive>true</transitive>
3     <traffic-rate-extended-community>
4         <informative-as>123</informative-as>
5         <local-administrator>AAAAAA==</local-administrator>
6     </traffic-rate-extended-community>
7 </extended-communities>
```

@line 5: A rate in bytes per second, *AAAAAA*== (0) means traffic discard.

- **Traffic Action**

```
<extended-communities>
    <transitive>true</transitive>
    <traffic-action-extended-community>
        <sample>true</sample>
        <terminal-action>false</terminal-action>
```

(continues on next page)

(continued from previous page)

```
    </traffic-action-extended-community>
</extended-communities>
```

- **Redirect to VRF AS 2byte format**

```
<extended-communities>
    <transitive>true</transitive>
    <redirect-extended-community>
        <global-administrator>123</global-administrator>
        <local-administrator>AAAAew==</local-administrator>
    </redirect-extended-community>
</extended-communities>
```

- **Redirect to VRF IPv4 format**

```
<extended-communities>
    <transitive>true</transitive>
    <redirect-ipv4>
        <global-administrator>192.168.0.1</global-administrator>
        <local-administrator>12345</local-administrator>
    </redirect-ipv4>
</extended-communities>
```

- **Redirect to VRF AS 4byte format**

```
<extended-communities>
    <transitive>true</transitive>
    <redirect-as4>
        <global-administrator>64495</global-administrator>
        <local-administrator>12345</local-administrator>
    </redirect-as4>
</extended-communities>
```

- **Redirect to IP**

```
<extended-communities>
    <transitive>true</transitive>
    <redirect-ip-nh-extended-community>
        <copy>false</false>
    </redirect-ip-nh-extended-community>
</extended-communities>
```

- **Traffic Marking**

```
<extended-communities>
    <transitive>true</transitive>
    <traffic-marking-extended-community>
        <global-administrator>20</global-administrator>
    </traffic-marking-extended-community>
</extended-communities>
```

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-flowspec:flowspec-subsequent-address-family/bgp-flowspec:flowspec-routes/bgp-flowspec:flowspec-route/flow1/0`

Method: `DELETE`

IPv4 L3VPN Flow Specification

This examples show how to originate and remove IPv4 L3VPN fowspec route via programmable RIB.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-flowspec:flowspec-l3vpn-subsequent-address-family/bgp-flowspec:flowspec-l3vpn-ipv4-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<flowspec-l3vpn-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-flowspec">
    <path-id>0</path-id>
    <route-key>flow-l3vpn</route-key>
    <route-distinguisher>172.16.0.44:101</route-distinguisher>
    <flowspec>
        <source-prefix>10.0.0.3/32</source-prefix>
    </flowspec>
    <attributes>
        <local-pref>
            <pref>100</pref>
        </local-pref>
        <origin>
            <value>igp</value>
        </origin>
        <as-path></as-path>
           <extended-communities>
               <transitive>true</transitive>
               <redirect-ipv4>
                   <global-administrator>172.16.0.44</global-administrator>
                   <local-administrator>102</local-administrator>
               </redirect-ipv4>
           </extended-communities>
    </attributes>
</flowspec-l3vpn-route>
```

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-flowspec:flowspec-l3vpn-subsequent-address-family/bgp-flowspec:flowspec-l3vpn-ipv4-routes/flowspec-l3vpn-route/flow-l3vpn/0`

Method: `DELETE`

IPv6 Flow Specification

This examples show how to originate and remove IPv6 fowspec route via programmable RIB.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv6-address-family/bgp-flowspec:flowspec-subsequent-address-family/bgp-flowspec:flowspec-ipv6-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<flowspec-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-flowspec">
    <route-key>flow-v6</route-key>
    <path-id>0</path-id>
    <flowspec>
        <destination-prefix>2001:db8:30::3/128</destination-prefix>
    </flowspec>
    <flowspec>
        <source-prefix>2001:db8:31::3/128</source-prefix>
     </flowspec>
    <flowspec>
        <flow-label>
            <op>equals end-of-list</op>
            <value>1</value>
        </flow-label>
    </flowspec>
    <attributes>
        <extended-communities>
            <transitive>true</transitive>
            <redirect-ipv6>
                <global-administrator>2001:db8:1::6</global-administrator>
                <local-administrator>12345</local-administrator>
            </redirect-ipv6>
        </extended-communities>
        <origin>
            <value>igp</value>
        </origin>
        <as-path/>
        <local-pref>
            <pref>100</pref>
        </local-pref>
    </attributes>
</flowspec-route>
```

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv6-address-family/bgp-flowspec:flowspec-subsequent-address-family/bgp-flowspec:flowspec-ipv6-routes/bgp-flowspec:flowspec-route/flow-v6/0`

Method: `DELETE`

References

- Dissemination of Flow Specification Rules
- Dissemination of Flow Specification Rules for IPv6
- BGP Flow-Spec Extended Community for Traffic Redirect to IP Next Hop
- Clarification of the Flowspec Redirect Extended Community
- Revised Validation Procedure for BGP Flow Specifications

2.1.10 MCAST-VPN Family

The BGP Multicast VPN(BGP MCAST-VPN) Multiprotocol extension can be used for MVPN auto-discovery, advertising MVPN to Inclusive P-Multicast Service Interface (I-PMSI) tunnel binding, advertising (C-S,C-G) to Selective PMSI (S-PMSI) tunnel binding, VPN customer multicast routing information exchange among Provider Edge routers (PEs), choosing a single forwarder PE, and for procedures in support of co-locating a Customer Rendezvous Point (C-RP) on a PE.

Contents

- *Configuration*
  - *BGP Speaker*
  - *BGP Peer*
- *Ipv4 MCAST-VPN Route API*
- *Ipv6 MCAST-VPN Route API*
- *Usage*
- *Programming*

Configuration

This section shows a way to enable MCAST-VPN family in BGP speaker and peer configuration.

BGP Speaker

To enable MCAST-VPN support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
```

(continues on next page)

(continued from previous page)

```
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name>IPV4-MCAST-VPN</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name>IPV6-MCAST-VPN</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

BGP Peer

Here is an example for BGP peer configuration with enabled IPV4 MCAST-VPN family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name>IPV4-MCAST-VPN</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

Ipv4 MCAST-VPN Route API

Following tree illustrates the BGP MCAST-VPN route structure.

```
:(mvpn-routes-ipv4-case)
   +--ro mvpn-routes-ipv4
      +--ro mvpn-route* [route-key path-id]
         +--ro (mvpn-choice)
            +--:(intra-as-i-pmsi-a-d-case)
            |  +--ro intra-as-i-pmsi-a-d
            +--:(inter-as-i-pmsi-a-d-case)
            |  +--ro inter-as-i-pmsi-a-d
            |     +--ro source-as    inet:as-number
            +--:(s-pmsi-a-d-case)
```

(continues on next page)

(continued from previous page)

```
|  +--ro s-pmsi-a-d
|     +--ro multicast-source             inet:ip-address
|     +--ro (multicast-group)?
|        +--:(c-g-address-case)
|        |  +--ro c-g-address?           inet:ip-address
|        +--:(ldp-mp-opaque-value-case)
|           +--ro ldp-mp-opaque-value
|              +--ro opaque-type             uint8
|              +--ro opaque-extended-type?   uint16
|              +--ro opaque                  yang:hex-string
+--:(leaf-a-d-case)
|  +--ro leaf-a-d
|     +--ro (leaf-a-d-route-key)
|        +--:(inter-as-i-pmsi-a-d-case)
|        |  +--ro inter-as-i-pmsi-a-d
|        |     +--ro source-as    inet:as-number
|        +--:(s-pmsi-a-d-case)
|           +--ro s-pmsi-a-d
|              +--ro multicast-source             inet:ip-address
|              +--ro (multicast-group)?
|                 +--:(c-g-address-case)
|                 |  +--ro c-g-address?           inet:ip-address
|                 +--:(ldp-mp-opaque-value-case)
|                    +--ro ldp-mp-opaque-value
|                       +--ro opaque-type             uint8
|                       +--ro opaque-extended-type?   uint16
|                       +--ro opaque                  yang:hex-string
+--:(source-active-a-d-case)
|  +--ro source-active-a-d
|     +--ro multicast-source    inet:ip-address
|     +--ro multicast-group     inet:ip-address
+--:(shared-tree-join-case)
|  +--ro shared-tree-join
|     +--ro c-multicast
|        +--ro multicast-source             inet:ip-address
|        +--ro source-as                     inet:as-number
|        +--ro (multicast-group)?
|           +--:(c-g-address-case)
|           |  +--ro c-g-address?           inet:ip-address
|           +--:(ldp-mp-opaque-value-case)
|              +--ro ldp-mp-opaque-value
|                 +--ro opaque-type             uint8
|                 +--ro opaque-extended-type?   uint16
|                 +--ro opaque                  yang:hex-string
+--:(source-tree-join-case)
   +--ro source-tree-join
      +--ro c-multicast
         +--ro multicast-source             inet:ip-address
         +--ro source-as                    inet:as-number
         +--ro (multicast-group)?
            +--:(c-g-address-case)
            |  +--ro c-g-address?           inet:ip-address
            +--:(ldp-mp-opaque-value-case)
               +--ro ldp-mp-opaque-value
                  +--ro opaque-type             uint8
                  +--ro opaque-extended-type?   uint16
                  +--ro opaque                  yang:hex-string
```

(continues on next page)

(continued from previous page)

```
                    ...
```

Ipv6 MCAST-VPN Route API

Following tree illustrates the BGP MCAST-VPN route structure.

```
:(mvpn-routes-ipv6-case)
   +--ro mvpn-routes-ipv6
      +--ro mvpn-route* [route-key path-id]
         +--ro (mvpn-choice)
            +--:(intra-as-i-pmsi-a-d-case)
            |  +--ro intra-as-i-pmsi-a-d
            +--:(inter-as-i-pmsi-a-d-case)
            |  +--ro inter-as-i-pmsi-a-d
            |     +--ro source-as    inet:as-number
            +--:(s-pmsi-a-d-case)
            |  +--ro s-pmsi-a-d
            |     +--ro multicast-source             inet:ip-address
            |     +--ro (multicast-group)?
            |        +--:(c-g-address-case)
            |        |  +--ro c-g-address?           inet:ip-address
            |        +--:(ldp-mp-opaque-value-case)
            |           +--ro ldp-mp-opaque-value
            |              +--ro opaque-type             uint8
            |              +--ro opaque-extended-type?   uint16
            |              +--ro opaque                  yang:hex-string
            +--:(leaf-a-d-case)
            |  +--ro leaf-a-d
            |     +--ro (leaf-a-d-route-key)
            |        +--:(inter-as-i-pmsi-a-d-case)
            |        |  +--ro inter-as-i-pmsi-a-d
            |        |     +--ro source-as    inet:as-number
            |        +--:(s-pmsi-a-d-case)
            |           +--ro s-pmsi-a-d
            |              +--ro multicast-source             inet:ip-address
            |              +--ro (multicast-group)?
            |                 +--:(c-g-address-case)
            |                 |  +--ro c-g-address?           inet:ip-address
            |                 +--:(ldp-mp-opaque-value-case)
            |                    +--ro ldp-mp-opaque-value
            |                       +--ro opaque-type             uint8
            |                       +--ro opaque-extended-type?   uint16
            |                       +--ro opaque                  yang:hex-string
            +--:(source-active-a-d-case)
            |  +--ro source-active-a-d
            |     +--ro multicast-source    inet:ip-address
            |     +--ro multicast-group     inet:ip-address
            +--:(shared-tree-join-case)
            |  +--ro shared-tree-join
            |     +--ro c-multicast
            |        +--ro multicast-source             inet:ip-address
            |        +--ro source-as                    inet:as-number
            |        +--ro (multicast-group)?
            |           +--:(c-g-address-case)
            |           |  +--ro c-g-address?           inet:ip-address
```

(continues on next page)

(continued from previous page)

```
            |           +--:(ldp-mp-opaque-value-case)
            |              +--ro ldp-mp-opaque-value
            |                 +--ro opaque-type             uint8
            |                 +--ro opaque-extended-type?   uint16
            |                 +--ro opaque                  yang:hex-string
            +--:(source-tree-join-case)
               +--ro source-tree-join
                  +--ro c-multicast
                     +--ro multicast-source             inet:ip-address
                     +--ro source-as                    inet:as-number
                     +--ro (multicast-group)?
                        +--:(c-g-address-case)
                        |  +--ro c-g-address?           inet:ip-address
                        +--:(ldp-mp-opaque-value-case)
                           +--ro ldp-mp-opaque-value
                              +--ro opaque-type             uint8
                              +--ro opaque-extended-type?   uint16
                              +--ro opaque                  yang:hex-string
                  ...
```

Usage

The Ipv4 Multicast VPN table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-mvpn:mcast-vpn-subsequent-address-family/bgp-mvpn-ipv4:mvpn-routes`

Method: `GET`

Response Body:

```
<mvpn-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp:mvpn:ipv4">
   <mvpn-route>
      <route-key>flow1</route-key>
      <path-id>0</path-id>
      <intra-as-i-pmsi-a-d>
         <route-distinguisher>172.16.0.44:101</route-distinguisher>
         <orig-route-ip>192.168.100.1</orig-route-ip>
      </intra-as-i-pmsi-a-d>
      <attributes>
         <ipv4-next-hop>
            <global>199.20.166.41</global>
         </ipv4-next-hop>
         <as-path/>
         <origin>
            <value>igp</value>
         </origin>
      </attributes>
   </mvpn-route>
</mvpn-routes>
```

The Ipv6 Multicast VPN table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-mvpn:mcast-vpn-subsequent-address-family/bgp-mvpn-ipv6:mvpn-routes`

Method: `GET`

Response Body:

```
<mvpn-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp:mvpn:ipv6">
   <mvpn-route>
      <route-key>flow1</route-key>
      <path-id>0</path-id>
      <intra-as-i-pmsi-a-d>
         <route-distinguisher>172.16.0.44:101</route-distinguisher>
         <orig-route-ip>192.168.100.1</orig-route-ip>
      </intra-as-i-pmsi-a-d>
      <attributes>
         <ipv6-next-hop>
            <global>2001:db8:1::6</global>
         </ipv6-next-hop>
         <as-path/>
         <origin>
            <value>igp</value>
         </origin>
      </attributes>
   </mvpn-route>
</mvpn-routes>
```

Programming

These examples show how to originate and remove MCAST-VPN routes via programmable RIB. There are seven different types of MCAST-VPN routes, and multiples extended communities. Routes can be used for variety of use-cases supported by BGP/MPLS MCAST-VPN. Make sure the *Application Peer* is configured first.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-mvpn:mcast-vpn-subsequent-address-family/bgp-mvpn-ipv4:mvpn-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<mvpn-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp:mvpn:ipv4">
   <route-key>mvpn</route-key>
   <path-id>0</path-id>
   <intra-as-i-pmsi-a-d>
      <route-distinguisher>172.16.0.44:101</route-distinguisher>
      <orig-route-ip>192.168.100.1</orig-route-ip>
   </intra-as-i-pmsi-a-d>
    ....
    <attributes>
        <ipv4-next-hop>
            <global>199.20.166.41</global>
        </ipv4-next-hop>
        <as-path/>
        <origin>
            <value>igp</value>
        </origin>
        <extended-communities>
        ....
```

(continues on next page)

(continued from previous page)

```
19          </extended-communities>
20      </attributes>
21  </mvpn-route>
```

@line 4: One of the MCAST-VPN route must be set here.

@line 15: In some cases, specific extended community presence is required.

MVPN Routes:

- **Intra-AS I-PMSI A-D**

```
<intra-as-i-pmsi-a-d>
    <route-distinguisher>0:5:3</route-distinguisher>
    <orig-route-ip>10.10.10.10</orig-route-ip>
</intra-as-i-pmsi-a-d>
```

- **Inter-AS I-PMSI A-D**

```
<inter-as-i-pmsi-a-d>
    <route-distinguisher>1.2.3.4:258</route-distinguisher>
    <source-as>64496</source-as>
</inter-as-i-pmsi-a-d>
```

- **S-PMSI A-D**

```
<s-pmsi-a-d>
    <route-distinguisher>1.2.3.4:258</route-distinguisher>
    <multicast-source>10.0.0.10</multicast-source>
    <c-g-address>12.0.0.12</c-g-address>
    <orig-route-ip>1.0.0.1</orig-route-ip>
</s-pmsi-a-d>
```

```
<s-pmsi-a-d>
    <route-distinguisher>1.2.3.4:258</route-distinguisher>
    <multicast-source>10.0.0.10</multicast-source>
    <ldp-mp-opaque-value>
        <opaque-type>1</opaque-type>
        <opaque-extended-type>0</opaque-extended-type>
        <opaque>das75das48bvxc</opaque>
    </ldp-mp-opaque-value>
    <orig-route-ip>1.0.0.1</orig-route-ip>
</s-pmsi-a-d>
```

- **Leaf A-D**

```
<leaf-a-d>
    <inter-as-i-pmsi-a-d>
        <route-distinguisher>1.2.3.4:258</route-distinguisher>
        <source-as>1</source-as>
    </inter-as-i-pmsi-a-d>
    <orig-route-ip>1.0.0.1</orig-route-ip>
</leaf-a-d>
```

```
<leaf-a-d>
    <s-pmsi-a-d>
        <route-distinguisher>1.2.3.4:258</route-distinguisher>
        <multicast-source>10.0.0.10</multicast-source>
        <ldp-mp-opaque-value>
            <opaque-type>1</opaque-type>
            <opaque-extended-type>0</opaque-extended-type>
            <opaque>das75das48bvxc</opaque>
        </ldp-mp-opaque-value>
        <orig-route-ip>1.0.0.1</orig-route-ip>
    </s-pmsi-a-d>
    <orig-route-ip>1.0.0.1</orig-route-ip>
</leaf-a-d>
```

- **Source Active A-D**

```
<source-active-a-d>
    <route-distinguisher>1.2.3.4:258</route-distinguisher>
    <multicast-source>1.0.0.1</multicast-source>
    <multicast-group>2.0.0.2</multicast-group>
</source-active-a-d>
```

- **Shared Tree Join**

```
<shared-tree-join>
    <c-multicast>
        <route-distinguisher>1.2.3.4:258</route-distinguisher>
        <source-as>64415</source-as>
        <multicast-source>1.0.0.1</multicast-source>
        <c-g-address>2.0.0.2</c-g-address>
    </c-multicast>
</shared-tree-join>
```

```
<shared-tree-join>
    <c-multicast>
        <route-distinguisher>1.2.3.4:258</route-distinguisher>
        <source-as>64415</source-as>
        <multicast-source>1.0.0.1</multicast-source>
        <ldp-mp-opaque-value>
            <opaque-type>1</opaque-type>
            <opaque-extended-type>0</opaque-extended-type>
            <opaque>das75das48bvxc</opaque>
        </ldp-mp-opaque-value>
    </c-multicast>
</shared-tree-join>
```

- **Source Tree Join**

```
<source-tree-join>
    <c-multicast>
        <route-distinguisher>1.2.3.4:258</route-distinguisher>
        <source-as>64415</source-as>
        <multicast-source>1.0.0.1</multicast-source>
        <c-g-address>2.0.0.2</c-g-address>
    </c-multicast>
</source-tree-join>
```

```
<source-tree-join>
    <c-multicast>
        <route-distinguisher>1.2.3.4:258</route-distinguisher>
        <source-as>64415</source-as>
        <multicast-source>1.0.0.1</multicast-source>
        <ldp-mp-opaque-value>
            <opaque-type>1</opaque-type>
            <opaque-extended-type>0</opaque-extended-type>
            <opaque>das75das48bvxc</opaque>
        </ldp-mp-opaque-value>
    </c-multicast>
</source-tree-join>
```

Attributes:

2.1.11 PSMI Attribute

- **P-Multicast Service Interface Tunnel (PMSI) attribute:**
  - **RSVP-TE P2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <rsvp-te-p2mp-lsp>
        <p2mp-id>1111111111</p2mp-id>
        <tunnel-id>11111</tunnel-id>
        <extended-tunnel-id>10.10.10.10</extended-tunnel-id>
    </rsvp-te-p2mp-lsp>
</pmsi-tunnel>
```

  - **mLDP P2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <mldp-p2mp-lsp>
        <address-family xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types
↪">x:ipv4-address-family</address-family>
        <root-node-address>10.10.10.10</root-node-address>
        <opaque-value>
            <opaque-type>255</opaque-type>
            <opaque-extended-type>11111</opaque-extended-type>
            <opaque>aa:aa:aa</opaque>
        </opaque-value>
    </mldp-p2mp-lsp>
</pmsi-tunnel>
```

  - **PIM-SSM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <pim-ssm-tree>
        <p-address>11.12.13.14</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
```

(continues on next page)

(continued from previous page)

```
    </pim-ssm-tree>
</pmsi-tunnel>
```

– **PIM-SM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <pim-sm-tree>
        <p-address>1.0.0.1</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </pim-sm-tree>
</pmsi-tunnel>
```

– **BIDIR-PIM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <bidir-pim-tree>
        <p-address>1.0.0.1</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </bidir-pim-tree>
</pmsi-tunnel>
```

– **Ingress Replication**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <ingress-replication>
        <receiving-endpoint-address>172.12.123.3</receiving-endpoint-address>
    </ingress-replication>
</pmsi-tunnel>
```

– **mLDP MP2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <mldp-mp2mp-lsp>
        <opaque-type>255</opaque-type>
        <opaque-extended-type>11111</opaque-extended-type>
        <opaque>aa:aa</opaque>
    </mldp-mp2mp-lsp>
</pmsi-tunnel>
```

- **PE Distinguisher Labels Attribute**

```
<pe-distinguisher-labels-attribute>
    <pe-distinguisher-label-attribute>
        <pe-address>10.10.10.1</pe-address>
        <mpls-label>20024</mpls-label>
    </pe-distinguisher-label-attribute>
    <pe-distinguisher-label-attribute>
        <pe-address>10.10.20.2</pe-address>
```

(continues on next page)

(continued from previous page)

```
        <mpls-label>20028</mpls-label>
    </pe-distinguisher-label-attribute>
</pe-distinguisher-labels-attribute>
```

Extended Communities:

- **Source AS Extended Community**

```
<extended-communities>
    <transitive>true</transitive>
    <source-as-extended-community>
        <global-administrator>65</global-administrator>
    </source-as-extended-community>
</extended-communities>
```

- **Source AS 4 Octets Extended Community**

```
<extended-communities>
    <transitive>true</transitive>
    <source-as-4-extended-community>
        <global-administrator>65555</global-administrator>
    </source-as-4-extended-community>
</extended-communities>
```

- **ES-Import Route Target**

```
<extended-communities>
    <transitive>true</transitive>
    <vrf-route-import-extended-community>
        <inet4-specific-extended-community-common>
            <global-administrator>10.0.0.1</global-administrator>
            <local-administrator>123=</local-administrator>
         </inet4-specific-extended-community-common>
     </vrf-route-import-extended-community>
</extended-communities>
```

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/bgp-mvpn:mcast-vpn-subsequent-address-family/bgp-mvpn-ipv4:mvpn-routes/mvpn-route/mvpn/0`

Method: `DELETE`

References

- Multicast in MPLS/BGP IP VPNs
- BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs
- IPv4 and IPv6 Infrastructure Addresses in BGP Updates for Multicast VPN

2.1.12 EVPN Family

The BGP MPLS-Based Ethernet VPN (BGP EVPN) Multiprotocol extension can be used to distribute Ethernet L2VPN service related routes in order to support a concept of MAC routing. A major use-case for BGP EVPN is data-center interconnection (DCI), where advantage of BGP EVPN are MAC/IP address advertising across MPLS network, Multihoming functionality including Fast Convergence, Split Horizon and Aliasing support, VM (MAC) Mobility, support Multicast and Broadcast traffic. In addition to MPLS, IP tunnelling encapsulation techniques like VXLAN, NVGRE, MPLSoGRE and others can be used for packet transportation. Also, Provider Backbone Bridging (PBB) can be combined with EVPN in order to reduce a number of MAC Advertisement routes.

Contents

- *Configuration*
  - *BGP Speaker*
  - *BGP Peer*
- *EVPN Route API*
- *Usage*
- *Programming*
- *EVPN Routes*

Configuration

This section shows a way to enable EVPN family in BGP speaker and peer configuration.

BGP Speaker

To enable EVPN support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:L2VPN-EVPN</afi-safi-name>
```

(continues on next page)

(continued from previous page)

```
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

BGP Peer

Here is an example for BGP peer configuration with enabled EVPN family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:L2VPN-EVPN
↪</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

EVPN Route API

Following tree illustrate the BGP EVPN route structure.

```
:(evpn-routes-case)
   +--ro evpn-routes
      +--ro evpn-route* [route-key path-id]
         +--ro route-key                      string
         +--ro path-id                        path-id
         +--ro (evpn-choice)
         |  +--:(ethernet-a-d-route-case)
         |  |  +--ro ethernet-a-d-route
         |  |     +--ro (esi)
         |  |     |  +--:(arbitrary-case)
         |  |     |  |  +--ro arbitrary
         |  |     |  |     +--ro arbitrary    binary
         |  |     |  +--:(lacp-auto-generated-case)
         |  |     |  |  +--ro lacp-auto-generated
         |  |     |  |     +--ro ce-lacp-mac-address    yang:mac-address
         |  |     |  |     +--ro ce-lacp-port-key       uint16
         |  |     |  +--:(lan-auto-generated-case)
         |  |     |  |  +--ro lan-auto-generated
         |  |     |  |     +--ro root-bridge-mac-address    yang:mac-address
         |  |     |  |     +--ro root-bridge-priority       uint16
```

(continues on next page)

(continued from previous page)

```
|  |     |  +--:(mac-auto-generated-case)
|  |     |  |  +--ro mac-auto-generated
|  |     |  |     +--ro system-mac-address     yang:mac-address
|  |     |  |     +--ro local-discriminator    uint24
|  |     |  +--:(router-id-generated-case)
|  |     |  |  +--ro router-id-generated
|  |     |  |     +--ro router-id              inet:ipv4-address
|  |     |  |     +--ro local-discriminator    uint32
|  |     |  +--:(as-generated-case)
|  |     |     +--ro as-generated
|  |     |        +--ro as                     inet:as-number
|  |     |        +--ro local-discriminator    uint32
|  |     +--ro ethernet-tag-id
|  |     |  +--ro vlan-id    uint32
|  |     +--ro mpls-label             netc:mpls-label
|  +--:(mac-ip-adv-route-case)
|  |  +--ro mac-ip-adv-route
|  |     +--ro (esi)
|  |     |  +--:(arbitrary-case)
|  |     |  |  +--ro arbitrary
|  |     |  |     +--ro arbitrary    binary
|  |     |  +--:(lacp-auto-generated-case)
|  |     |  |  +--ro lacp-auto-generated
|  |     |  |     +--ro ce-lacp-mac-address    yang:mac-address
|  |     |  |     +--ro ce-lacp-port-key       uint16
|  |     |  +--:(lan-auto-generated-case)
|  |     |  |  +--ro lan-auto-generated
|  |     |  |     +--ro root-bridge-mac-address    yang:mac-address
|  |     |  |     +--ro root-bridge-priority       uint16
|  |     |  +--:(mac-auto-generated-case)
|  |     |  |  +--ro mac-auto-generated
|  |     |  |     +--ro system-mac-address     yang:mac-address
|  |     |  |     +--ro local-discriminator    uint24
|  |     |  +--:(router-id-generated-case)
|  |     |  |  +--ro router-id-generated
|  |     |  |     +--ro router-id              inet:ipv4-address
|  |     |  |     +--ro local-discriminator    uint32
|  |     |  +--:(as-generated-case)
|  |     |     +--ro as-generated
|  |     |        +--ro as                     inet:as-number
|  |     |        +--ro local-discriminator    uint32
|  |     +--ro ethernet-tag-id
|  |     |  +--ro vlan-id    uint32
|  |     +--ro mac-address            yang:mac-address
|  |     +--ro ip-address?            inet:ip-address
|  |     +--ro mpls-label1            netc:mpls-label
|  |     +--ro mpls-label2?           netc:mpls-label
|  +--:(inc-multi-ethernet-tag-res-case)
|  |  +--ro inc-multi-ethernet-tag-res
|  |     +--ro ethernet-tag-id
|  |     |  +--ro vlan-id    uint32
|  |     +--ro orig-route-ip?     inet:ip-address
|  +--:(es-route-case)
|     +--ro es-route
|        +--ro (esi)
|        |  +--:(arbitrary-case)
|        |  |  +--ro arbitrary
```

(continues on next page)

(continued from previous page)

```
|        |  |     +--ro arbitrary    binary
|        |  +--:(lacp-auto-generated-case)
|        |  |  +--ro lacp-auto-generated
|        |  |     +--ro ce-lacp-mac-address    yang:mac-address
|        |  |     +--ro ce-lacp-port-key       uint16
|        |  +--:(lan-auto-generated-case)
|        |  |  +--ro lan-auto-generated
|        |  |     +--ro root-bridge-mac-address    yang:mac-address
|        |  |     +--ro root-bridge-priority       uint16
|        |  +--:(mac-auto-generated-case)
|        |  |  +--ro mac-auto-generated
|        |  |     +--ro system-mac-address     yang:mac-address
|        |  |     +--ro local-discriminator    uint24
|        |  +--:(router-id-generated-case)
|        |  |  +--ro router-id-generated
|        |  |     +--ro router-id              inet:ipv4-address
|        |  |     +--ro local-discriminator    uint32
|        |  +--:(as-generated-case)
|        |     +--ro as-generated
|        |        +--ro as                     inet:as-number
|        |        +--ro local-discriminator    uint32
|        +--ro orig-route-ip          inet:ip-address
+--ro route-distinguisher           bgp-t:route-distinguisher
+--ro attributes
   +--ro extended-communities*
   |  +--ro transitive?                            boolean
   |  +--ro (extended-community)?
   |     +--:(encapsulation-case)
   |     |  +--ro encapsulation-extended-community
   |     |     +--ro tunnel-type    encapsulation-tunnel-type
   |     +--:(esi-label-extended-community-case)
   |     |  +--ro esi-label-extended-community
   |     |     +--ro single-active-mode?   boolean
   |     |     +--ro esi-label             netc:mpls-label
   |     +--:(es-import-route-extended-community-case)
   |     |  +--ro es-import-route-extended-community
   |     |     +--ro es-import    yang:mac-address
   |     +--:(mac-mobility-extended-community-case)
   |     |  +--ro mac-mobility-extended-community
   |     |     +--ro static?       boolean
   |     |     +--ro seq-number    uint32
   |     +--:(default-gateway-extended-community-case)
   |     |  +--ro default-gateway-extended-community!
   |     +--:(layer-2-attributes-extended-community-case)
   |        +--ro layer-2-attributes-extended-community
   |           +--ro primary-pe?     boolean
   |           +--ro backup-pe?      boolean
   |           +--ro control-word?   boolean
   |           +--ro l2-mtu          uint16
   +--ro pmsi-tunnel!
      +--ro leaf-information-required    boolean
      +--ro mpls-label?                  netc:mpls-label
      +--ro (tunnel-identifier)?
         +--:(rsvp-te-p2mp-lsp)
         |  +--ro rsvp-te-p2mp-lps
         |     +--ro p2mp-id               uint32
         |     +--ro tunnel-id             uint16
```

(continues on next page)

(continued from previous page)

```
|     +--ro extended-tunnel-id    inet:ip-address
+--:(mldp-p2mp-lsp)
|  +--ro mldp-p2mp-lsp
|     +--ro address-family       identityref
|     +--ro root-node-address    inet:ip-address
|     +--ro opaque-value*
|        +--ro opaque-type             uint8
|        +--ro opaque-extended-type?   uint16
|        +--ro opaque                  yang:hex-string
+--:(pim-ssm-tree)
|  +--ro pim-ssm-tree
|     +--ro p-address            inet:ip-address
|     +--ro p-multicast-group    inet:ip-address
+--:(pim-sm-tree)
|  +--ro pim-sm-tree
|     +--ro p-address            inet:ip-address
|     +--ro p-multicast-group    inet:ip-address
+--:(bidir-pim-tree)
|  +--ro bidir-pim-tree
|     +--ro p-address            inet:ip-address
|     +--ro p-multicast-group    inet:ip-address
+--:(ingress-replication)
|  +--ro ingress-replication
|     +--ro receiving-endpoint-address?   inet:ip-address
+--:(mldp-mp2mp-lsp)
   +--ro mldp-mp2mp-lsp
      +--ro opaque-type             uint8
      +--ro opaque-extended-type?   uint16
      +--ro opaque
...
```

Usage

The L2VPN EVPN table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/odl-bgp-evpn:l2vpn-address-family/odl-bgp-evpn:evpn-subsequent-address-family/evpn-routes`

Method: `GET`

Response Body:

```
<evpn-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-evpn">
   <evpn-route>
      <route-key>AxEAAcCoZAED6AAAAQAgwKhkAQ==</route-key>
      <path-id>0</path-id>
      <route-distinguisher>192.168.100.1:1000</route-distinguisher>
      <inc-multi-ethernet-tag-res>
         <ethernet-tag-id>
            <vlan-id>256</vlan-id>
         </ethernet-tag-id>
         <orig-route-ip>192.168.100.1</orig-route-ip>
      </inc-multi-ethernet-tag-res>
      <attributes>
         <ipv4-next-hop>
```

(continues on next page)

(continued from previous page)

```
            <global>172.23.29.104</global>
         </ipv4-next-hop>
         <as-path/>
         <origin>
            <value>igp</value>
         </origin>
         <extended-communities>
            <extended-communities>
                <transitive>true</transitive>
                <route-target-extended-community>
                    <global-administrator>65504</global-administrator>
                    <local-administrator>AAAD6A==</local-administrator>
                </route-target-extended-community>
            </extended-communities>
         </extended-communities>
         <pmsi-tunnel>
             <leaf-information-required>true</leaf-information-required>
             <mpls-label>20024</mpls-label>
             <ingress-replication>
                 <receiving-endpoint-address>192.168.100.1</receiving-endpoint-
↪address>
             </ingress-replication>
         </pmsi-tunnel>
      </attributes>
   </evpn-route>
</evpn-routes>
```

Programming

This examples show how to originate and remove EVPN routes via programmable RIB. There are four different types of EVPN routes, and several extended communities. Routes can be used for variety of use-cases supported by BGP/MPLS EVPN, PBB EVPN and NVO EVPN. Make sure the *Application Peer* is configured first.

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/odl-bgp-evpn:l2vpn-address-family/odl-bgp-evpn:evpn-subsequent-address-family/odl-bgp-evpn:evpn-routes`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<evpn-route xmlns="urn:opendaylight:params:xml:ns:yang:bgp-evpn">
    <route-key>evpn</route-key>
    <path-id>0</path-id>
    <route-distinguisher>172.12.123.3:200</route-distinguisher>
    ....
    <attributes>
        <ipv4-next-hop>
            <global>199.20.166.41</global>
        </ipv4-next-hop>
        <as-path/>
        <origin>
            <value>igp</value>
        </origin>
```

(continues on next page)

(continued from previous page)

```
14          <extended-communities>
15          ....
16          </extended-communities>
17      </attributes>
18  </evpn-route>
```

@line 4: Route Distinguisher (RD) - set to RD of the MAC-VRF advertising the NLRI, recommended format *<IP>:<VLAN_ID>*

@line 5: One of the EVPN route must be set here.

@line 15: In some cases, specific extended community presence is required. The route may carry one or more Route Target attributes.

EVPN Routes

- **Ethernet AD per ESI**

```
<ethernet-a-d-route>
    <mpls-label>0</mpls-label>
    <ethernet-tag-id>
        <vlan-id>4294967295</vlan-id>
    </ethernet-tag-id>
    <arbitrary>
        <arbitrary>AAAAAAAAAAAA</arbitrary>
    </arbitrary>
</ethernet-a-d-route>
```

- **Ethernet AD per EVI**

```
<ethernet-a-d-route>
    <mpls-label>24001</mpls-label>
    <ethernet-tag-id>
        <vlan-id>2200</vlan-id>
    </ethernet-tag-id>
    <arbitrary>
        <arbitrary>AAAAAAAAAAAA</arbitrary>
    </arbitrary>
</ethernet-a-d-route>
```

- **MAC/IP Advertisement**

```
<mac-ip-adv-route>
    <arbitrary>
        <arbitrary>AAAAAAAAAAAA</arbitrary>
    </arbitrary>
    <ethernet-tag-id>
        <vlan-id>2100</vlan-id>
    </ethernet-tag-id>
    <mac-address>f2:0c:dd:80:9f:f7</mac-address>
    <ip-address>10.0.1.12</ip-address>
    <mpls-label1>299776</mpls-label1>
</mac-ip-adv-route>
```

- **Inclusive Multicast Ethernet Tag**

```
<inc-multi-ethernet-tag-res>
    <ethernet-tag-id>
        <vlan-id>2100</vlan-id>
    </ethernet-tag-id>
    <orig-route-ip>43.43.43.43</orig-route-ip>
</inc-multi-ethernet-tag-res>
```

- **Ethernet Segment**

```
<es-route>
    <orig-route-ip>43.43.43.43</orig-route-ip>
    <arbitrary>
        <arbitrary>AAAAAAAAAAAA</arbitrary>
    </arbitrary>
</es-route>
```

EVPN Ethernet Segment Identifier (ESI):

- **Type 0** Indicates an arbitrary 9-octet ESI.

```
<arbitrary>
    <arbitrary>AAAAAAAAAAAA</arbitrary>
</arbitrary>
```

- **Type 1** IEEE 802.1AX LACP is used.

```
<lacp-auto-generated>
    <ce-lacp-mac-address>f2:0c:dd:80:9f:f7</ce-lacp-mac-address>
    <ce-lacp-port-key>22</ce-lacp-port-key>
</lacp-auto-generated>
```

- **Type 2** Indirectly connected hosts via a bridged LAN.

```
<lan-auto-generated>
    <root-bridge-mac-address>f2:0c:dd:80:9f:f7</root-bridge-mac-address>
    <root-bridge-priority>20</root-bridge-priority>
</lan-auto-generated>
```

- **Type 3** MAC-based ESI.

```
<mac-auto-generated>
    <system-mac-address>f2:0c:dd:80:9f:f7</system-mac-address>
    <local-discriminator>2000</local-discriminator>
</mac-auto-generated>
```

- **Type 4** Router-ID ESI

```
<router-id-generated>
    <router-id>43.43.43.43</router-id>
    <local-discriminator>2000</local-discriminator>
</router-id-generated>
```

- **Type 5** AS-based ESI

```
<as-generated>
    <as>16843009</as>
    <local-discriminator>2000</local-discriminator>
</as-generated>
```

Attributes:

2.1.13 PSMI Attribute

- **P-Multicast Service Interface Tunnel (PMSI) attribute:**
  - **RSVP-TE P2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <rsvp-te-p2mp-lsp>
        <p2mp-id>1111111111</p2mp-id>
        <tunnel-id>11111</tunnel-id>
        <extended-tunnel-id>10.10.10.10</extended-tunnel-id>
    </rsvp-te-p2mp-lsp>
</pmsi-tunnel>
```

  - **mLDP P2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <mldp-p2mp-lsp>
        <address-family xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types
↪">x:ipv4-address-family</address-family>
        <root-node-address>10.10.10.10</root-node-address>
        <opaque-value>
            <opaque-type>255</opaque-type>
            <opaque-extended-type>11111</opaque-extended-type>
            <opaque>aa:aa:aa</opaque>
        </opaque-value>
    </mldp-p2mp-lsp>
</pmsi-tunnel>
```

  - **PIM-SSM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <pim-ssm-tree>
        <p-address>11.12.13.14</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </pim-ssm-tree>
</pmsi-tunnel>
```

  - **PIM-SM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <pim-sm-tree>
        <p-address>1.0.0.1</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </pim-sm-tree>
</pmsi-tunnel>
```

  - **BIDIR-PIM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <bidir-pim-tree>
        <p-address>1.0.0.1</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </bidir-pim-tree>
</pmsi-tunnel>
```

– **Ingress Replication**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <ingress-replication>
        <receiving-endpoint-address>172.12.123.3</receiving-endpoint-address>
    </ingress-replication>
</pmsi-tunnel>
```

– **mLDP MP2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <mldp-mp2mp-lsp>
        <opaque-type>255</opaque-type>
        <opaque-extended-type>11111</opaque-extended-type>
        <opaque>aa:aa</opaque>
    </mldp-mp2mp-lsp>
</pmsi-tunnel>
```

Extended Communities:

- **ESI Label Extended Community**

```
<extended-communities>
    <transitive>true</transitive>
    <esi-label-extended-community>
        <single-active-mode>false</single-active-mode>
        <esi-label>24001</esi-label>
    </esi-label-extended-community>
</extended-communities>
```

- **ES-Import Route Target**

```
<extended-communities>
    <transitive>true</transitive>
    <es-import-route-extended-community>
        <es-import>f2:0c:dd:80:9f:f7</es-import>
    </es-import-route-extended-community>
</extended-communities>
```

- **MAC Mobility Extended Community**

```
<extended-communities>
    <transitive>true</transitive>
    <mac-mobility-extended-community>
```

(continues on next page)

(continued from previous page)

```
        <static>true</static>
        <seq-number>200</seq-number>
    </mac-mobility-extended-community>
</extended-communities>
```

- **Default Gateway Extended Community**

```
<extended-communities>
    <transitive>true</transitive>
    <default-gateway-extended-community>
    </default-gateway-extended-community>
</extended-communities>
```

- **EVPN Layer 2 attributes extended community**

```
<extended-communities>
    <transitive>false</transitive>
    <layer-2-attributes-extended-community>
        <primary-pe>true</primary-pe>
        <backup-pe>true</backup-pe>
        <control-word >true</control-word>
        <l2-mtu>200</l2-mtu>
    </layer-2-attributes-extended-community>
</extended-communities>
```

- **BGP Encapsulation extended community**

```
1 <extended-communities>
2     <transitive>false</transitive>
3     <encapsulation-extended-community>
4         <tunnel-type>vxlan</tunnel-type>
5     </encapsulation-extended-community>
6 </extended-communities>
```

@line 4: full list of tunnel types

To remove the route added above, following request can be used:

URL: `/restconf/config/bgp-rib:application-rib/10.25.1.9/tables/bgp-types:ipv4-address-family/odl-bgp-evpn:l2vpn-address-family/odl-bgp-evpn:evpn-subsequent-address-family/odl-bgp-evpn:evpn-routes/evpn-route/evpn/0`

Method: `DELETE`

Table 1: EVPN Routes Usage.

EVN Route Type	Extended Communities	Usage
Ethernet Auto-discovery	ESI Label, BGP EncapsulationEVPN Layer 2 attributes	Fast Convergence, Split Horizon, Aliasing
MAC/IP Advertisement	BGP Encapsulation, MAC Mobility, Default Gateway	MAC address reachability
Inclusive Multicast Ethernet Tag	PMSI Tunnel, BGP Encapsulation	Handling of Multi-destination traffic
Ethernet Segment	BGP Encapsulation, ES-Import Route Target	Designated Forwarder Election

References

- BGP MPLS-Based Ethernet VPN
- Provider Backbone Bridging Combined with Ethernet VPN
- VPWS support in EVPN
- A Network Virtualization Overlay Solution using EVPN
- Interconnect Solution for EVPN Overlay networks
- Usage and applicability of BGP MPLS based Ethernet VPN

2.1.14 Route Target Constrain Family

The BGP Multicast Route Target (RT) Constrain Multiprotocol extension can be used to restrict advertisement of VPN NLRI to peers that have advertised their respective Route Targets, effectively building a route distribution graph.

Contents

- *Configuration*
  - *BGP Speaker*
  - *BGP Peer*
- *ROUTE-TARGET-CONSTRAIN Route API*
- *Usage*
- *Routing Policies*
- *References*

Configuration

This section shows a way to enable ROUTE-TARGET-CONSTRAIN family in BGP speaker and peer configuration.

BGP Speaker

To enable ROUTE-TARGET-CONSTRAIN support in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
```

(continues on next page)

(continued from previous page)

```
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name>ROUTE-TARGET-CONSTRAIN</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

BGP Peer

Here is an example for BGP peer configuration with enabled ROUTE-TARGET-CONSTRAIN family.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name>ROUTE-TARGET-CONSTRAIN</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

ROUTE-TARGET-CONSTRAIN Route API

Following tree illustrates the BGP ROUTE-TARGET-CONSTRAIN route structure.

```
:(route-target-constrain-routes-case)
  +--rw route-target-constrain-routes
     +--rw route-target-constrain-route* [route-key path-id]
        +--rw origin-as                  inet:as-number
        +--rw (route-target-constrain-choice)
           +--:(route-target-constrain-default-case)
           |  +--rw route-target-constrain-default-route!
           +--:(route-target-constrain-route-case)
           |  +--rw route-target-extended-community
           |     +--rw global-administrator?   short-as-number
           |     +--rw local-administrator?    binary
           +--:(route-target-constrain-ipv4-route-case)
           |  +--rw route-target-ipv4
           |     +--rw global-administrator?   inet:ipv4-address
```

(continues on next page)

(continued from previous page)

```
|     +--rw local-administrator?    uint16
+--:(route-target-constrain-as-4-extended-community-case)
   +--rw as-4-route-target-extended-community
      +--rw as-4-specific-common
         +--rw as-number              inet:as-number
         +--rw local-administrator    uint16
```

Usage

The ROUTE TARGET CONSTRAIN table in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-route-target-constrain:route-target-constrain-subsequent-bgp-route-target-constrain:route-target-constrain-routes`

Method: `GET`

Response Body:

```
<route-target-constrain-routes xmlns=
↪"urn:opendaylight:params:xml:ns:yang:bgp:route:target:constrain">
   <route-target-constrain-route>
      <route-key>flow1</route-key>
      <path-id>0</path-id>
      <origin-as>64511</origin-as>
      <route-target-extended-community>
          <global-administrator>64511</global-administrator>
          <local-administrator>AAAAZQ==</local-administrator>
      </route-target-extended-community>
      <attributes>
         <ipv4-next-hop>
            <global>199.20.166.41</global>
         </ipv4-next-hop>
         <as-path/>
         <origin>
            <value>igp</value>
         </origin>
         <local-pref>
            <pref>100</pref>
         </local-pref>
      </attributes>
   </route-target-constrain-route>
</route-target-constrain-routes>
```

Routing Policies

```
<policy-definition>
    <name>default-odl-export-policy</name>
    <statement>
    ...
    <statement>
        <name>from-external-to-external-RTC</name>
        <conditions>
            <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
```

(continues on next page)

(continued from previous page)

```
                <afi-safi-in xmlns:x=
↪"urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">x:ROUTE-TARGET-
↪CONSTRAIN</afi-safi-in>
                <match-role-set xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
                    <from-role>
                        <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-
↪defined-sets/role-sets/role-set[role-set-name="only-ebgp"]</role-set>
                    </from-role>
                    <to-role>
                        <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-
↪defined-sets/role-sets/role-set[role-set-name="only-ebgp"]</role-set>
                    </to-role>
                </match-role-set>
            </bgp-conditions>
        </conditions>
        <actions>
            <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
                <client-attribute-prepend xmlns=
↪"urn:opendaylight:params:xml:ns:yang:bgp:route:target:constrain"/>
            </bgp-actions>
        </actions>
    </statement>
    ...
    </statement>
    <statement>
        <name>from-internal-or-rr-client-to-route-reflector</name>
        <conditions>
            <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
                <afi-safi-not-in xmlns:x=
↪"urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions"
                                 xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">x:ROUTE-TARGET-
↪CONSTRAIN
                </afi-safi-not-in>
                <match-role-set xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
                    <from-role>
                        <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-
↪defined-sets/role-sets/role-set[role-set-name="ibgp-rr-client"]</role-set>
                    </from-role>
                    <to-role>
                        <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-
↪defined-sets/role-sets/role-set[role-set-name="only-rr-client"]</role-set>
                    </to-role>
                </match-role-set>
            </bgp-conditions>
        </conditions>
        <actions>
            <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
                <set-cluster-id-prepend xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
                <set-originator-id-prepend xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
            </bgp-actions>
        </actions>
    </statement>
```

(continues on next page)

(continued from previous page)

```
    <statement>
        <name>from-internal-or-rr-client-to-route-RTC</name>
        <conditions>
            <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
                <afi-safi-in xmlns:x=
↪"urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">x:ROUTE-TARGET-
↪CONSTRAIN</afi-safi-in>
                <match-role-set xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy">
                    <from-role>
                        <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-
↪defined-sets/role-sets/role-set[role-set-name="ibgp-rr-client"]</role-set>
                    </from-role>
                    <to-role>
                        <role-set>/rpol:routing-policy/rpol:defined-sets/bgppol:bgp-
↪defined-sets/role-sets/role-set[role-set-name="only-rr-client"]</role-set>
                    </to-role>
                </match-role-set>
            </bgp-conditions>
        </conditions>
        <actions>
            <bgp-actions xmlns="http://openconfig.net/yang/bgp-policy">
                <set-originator-id-prepend xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
                <set-next-hop>SELF</set-next-hop>
            </bgp-actions>
        </actions>
    </statement>
    <statement>
        <name>vpn-membership-RTC</name>
        <conditions>
            <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
                <afi-safi-in xmlns:x="http://openconfig.net/yang/bgp-types">x:L3VPN-
↪IPV4-UNICAST</afi-safi-in>
                <afi-safi-in xmlns:x="http://openconfig.net/yang/bgp-types">x:L3VPN-
↪IPV6-UNICAST</afi-safi-in>
                <vpn-non-member xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl:bgp:default:policy"/>
            </bgp-conditions>
        </conditions>
        <actions>
            <reject-route/>
        </actions>
    </statement>
    ...
    ...
</policy-definition>
```

References

- Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)

2.1.15 Additional Path Capability

The ADD-PATH capability allows to advertise multiple paths for the same address prefix. It can help with optimal routing and routing convergence in a network by providing potential alternate or backup paths.

Contents

- *Configuration*
  - *BGP Speaker*
  - *BGP Peer*
- *Usage*
- *References*

Configuration

This section shows a way to enable ADD-PATH capability in BGP speaker and peer configuration.

Note: The capability is applicable for IP Unicast, IP Labeled Unicast and Flow Specification address families.

BGP Speaker

To enable ADD-PATH capability in BGP plugin, first configure BGP speaker instance:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>bgp-example</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>192.0.2.2</router-id>
                <as>65000</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">
↪x:IPV4-UNICAST</afi-safi-name>
```

(continues on next page)

(continued from previous page)

```
                    <receive>true</receive>
                    <send-max>2</send-max>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

@line 14: Defines path selection strategy: *send-max* > 1 -> Advertise N Paths or *send-max* = 0 -> Advertise All Paths

Here is an example for update a specific family with enable ADD-PATH capability

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/global/afi-safis/afi-safi/openconfig-bgp-types:IPV4%2DUNICAST`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<afi-safi xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
   <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-
↪safi-name>
   <receive>true</receive>
   <send-max>0</send-max>
</afi-safi>
```

BGP Peer

Here is an example for BGP peer configuration with enabled ADD-PATH capability.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>192.0.2.1</neighbor-address>
    <afi-safis>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-
↪LABELLED-UNICAST</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-
↪UNICAST</afi-safi-name>
            <receive>true</receive>
            <send-max>0</send-max>
        </afi-safi>
```

(continues on next page)

(continued from previous page)

```
    </afi-safis>
</neighbor>
```

Note: The path selection strategy is not configurable on per peer basis. The send-max presence indicates a willingness to send ADD-PATH NLRIs to the neighbor.

Here is an example for update specific family BGP peer configuration with enabled ADD-PATH capability.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/afi-safis/afi-safi/openconfig-bgp-types:IPV4%2DUNICAST`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<afi-safi xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
   <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-
↪safi-name>
   <receive>true</receive>
   <send-max>0</send-max>
</afi-safi>
```

Usage

The IPv4 Unicast table with enabled ADD-PATH capability in an instance of the speaker's Loc-RIB can be verified via REST:

URL: `/restconf/operational/bgp-rib:bgp-rib/rib/bgp-example/loc-rib/tables/bgp-types:ipv4-address-family/bgp-types:unicast-subsequent-address-family/ipv4-routes`

Method: `GET`

Response Body:

```
<ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet">
    <ipv4-route>
        <path-id>1</path-id>
        <prefix>193.0.2.1/32</prefix>
        <attributes>
            <as-path></as-path>
            <origin>
                <value>igp</value>
            </origin>
            <local-pref>
                <pref>100</pref>
            </local-pref>
            <ipv4-next-hop>
                <global>10.0.0.1</global>
            </ipv4-next-hop>
        </attributes>
```

(continues on next page)

(continued from previous page)

```
17      </ipv4-route>
18      <ipv4-route>
19          <path-id>2</path-id>
20          <prefix>193.0.2.1/32</prefix>
21          <attributes>
22              <as-path></as-path>
23              <origin>
24                  <value>igp</value>
25              </origin>
26              <local-pref>
27                  <pref>100</pref>
28              </local-pref>
29              <ipv4-next-hop>
30                  <global>10.0.0.2</global>
31              </ipv4-next-hop>
32          </attributes>
33      </ipv4-route>
34  </ipv4-routes>
```

@line 3: The routes with the same destination are distinguished by *path-id* attribute.

References

- Advertisement of Multiple Paths in BGP
- Best Practices for Advertisement of Multiple Paths in IBGP

2.1.16 Route Refresh

The Route Refresh Capability allows to dynamically request a re-advertisement of the Adj-RIB-Out from a BGP peer. This is useful when the inbound routing policy for a peer changes and all prefixes from a peer must be reexamined against a new policy.

Contents

- *Configuration*
- *Usage*
- *References*
  - *Peer Session Release*
- *Configuration*
- *Usage*

Configuration

The capability is enabled by default, no additional configuration is required.

Usage

To send a Route Refresh request from OpenDaylight BGP speaker instance to its neighbor, invoke RPC:

URL: `/restconf/operations/bgp-peer-rpc:route-refresh-request`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:bgp-peer-rpc">
    <afi xmlns:types="urn:opendaylight:params:xml:ns:yang:bgp-types">types:ipv4-
↪address-family</afi>
    <safi xmlns:types="urn:opendaylight:params:xml:ns:yang:bgp-types">types:unicast-
↪subsequent-address-family</safi>
    <peer-ref xmlns:rib="urn:opendaylight:params:xml:ns:yang:bgp-rib">/rib:bgp-rib/
↪rib:rib[rib:id="bgp-example"]/rib:peer[rib:peer-id="bgp://10.25.1.9"]</peer-ref>
</input>
```

References

- Route Refresh Capability for BGP-4

Peer Session Release

BGP provides a RPC feature to release a Neighbor session.

Configuration

The capability is enabled by default, no additional configuration is required.

Usage

To release neighbor session, invoke RPC:

URL: `/restconf/operations/bgp-peer-rpc:reset-session`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:bgp-peer-rpc">
    <peer-ref xmlns:rib="urn:opendaylight:params:xml:ns:yang:bgp-rib">/rib:bgp-rib/
↪rib:rib[rib:id="bgp-example"]/rib:peer[rib:peer-id="bgp://10.25.1.9"]</peer-ref>
</input>
```

2.1.17 Graceful Restart Capability

The Graceful Restart Capability helps us to minimize the negative effects on routing caused by BGP restart by allowing BGP speaker to express its ability to preserve forwarding state during BGP restart. New capability is advertised in OPEN message which contains information about Graceful Restart timer value, supported families and their forwarding state.

Contents

- *Configuration*
  - *Graceful Restart Timer*
  - *BGP Neighbor Families Graceful Restart Configuration*
- *Usage*
- *References*

Configuration

This section shows a way how to configure Graceful Restart Timer and enable Graceful Restart support for specific families.

Note: Graceful Restart capability is enabled by default even when no families are advertised. In that case only receiving speaker procedures apllies.

Graceful Restart Timer

Routing information for configured families are preserved for time given by Graceful Restart timer in seconds. This can be configured in *graceful-restart* section of *neighbor* or *peer-group* configuration.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/graceful-restart`

or

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/peer-groups/peer-group/external-neighbors/graceful-restart`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<graceful-restart xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions
↪">
    <config>
        <restart-time>60</restart-time>
```

(continues on next page)

(continued from previous page)

```
    </config>
</graceful-restart>
```

@line 3: value of Graceful Restart timer in seconds

Note: If case that Graceful Restart timer is configured for both neighbor and peer-group, the one from peer-group is used. If no Graceful Restart timer is configured value of HOLD timer is used.

BGP Neighbor Families Graceful Restart Configuration

Preserving specific family during Graceful Restart must be enabled in *graceful-restart* section of family configuration for *neighbor* or *peer-group*.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/afi-safis/afi-safi/openconfig-bgp-types:IPV4%2DUNICAST/graceful-restart`

or

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/peer-groups/peer-group/external-neighbors/afi-safis/afi-safi/openconfig-bgp-types:IPV4%2DUNICAST/graceful-restart`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<graceful-restart xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions
↪">
   <config>
      <enable>true</enable>
    </config>
</graceful-restart>
```

@line 3: True if we want to preserve family routing information during Graceful Restart

Usage

In case when we are invoking Graceful Restart we act as Restarting Speaker and we are additionally postponing path selection process until end-of-rib is received for all families or Selection Deferral timer expires, whichever happens first. To perform Graceful Restart with peer, invoke RPC:

URL: `/restconf/operations/bgp-peer-rpc:restart-gracefully`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:bgp-peer-rpc">
    <peer-ref xmlns:rib="urn:opendaylight:params:xml:ns:yang:bgp-rib">/rib:bgp-rib/
↪rib:rib[rib:id="bgp-example"]/rib:peer[rib:peer-id="bgp://10.25.1.9"]</peer-ref>
    <selection-deferral-time>60</slection-deferral-time>
</input>
```

@line 3: Value of Selection Deferral timer in seconds

References

- Graceful Restart Mechanism for BGP

2.1.18 Long-Lived Graceful Restart Capability

The Long-Lived Graceful Restart Capability is extension to Graceful Restart that provides tools to retain stale routes for longer time upon session failure. New capability is advertised in OPEN message in conjunction with Graceful Restart Capability, which contains list of supported families and their stale timer. After session failure and Graceful Restart timer expire routing information is retained for value of Long-Lived Stale Timer.

Contents

- *Configuration*
- *References*

Configuration

Long-Live Graceful Restart is enabled and configured per family in *ll-graceful-restart* section of *neighbor* or *peer-group* family configuration.

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/afi-safis/afi-safi/openconfig-bgp-types:IPV4%2DUNICAST/graceful-restart`

or

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/peer-groups/peer-group/external-neighbors/afi-safis/afi-safi/openconfig-bgp-types:IPV4%2DUNICAST/graceful-restart`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<graceful-restart xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions
↪">
   <config>
      <enable>true</enable>
```

(continues on next page)

(continued from previous page)

```
4       <ll-graceful-restart xmlns="urn:opendaylight:params:xml:ns:yang:bgp:ll-graceful-
↪restart">
5           <config>
6               <long-lived-stale-time>180</long-lived-stale-time>
7           </config>
8       </ll-graceful-restart>
9     </config>
10 </graceful-restart>
```

@line 3: if Graceful Restart support is disabled for family only procedures for Long-Lived Graceful Restart applies

@line 6: value of Long-Lived Stale timer in seconds

References

- Support for Long-lived BGP Graceful Restart

2.1.19 Operational State

The OpenDaylight BGP implementation provides a set of APIs (described below), that give its operational state refreshed periodically, by default every 5 seconds. The following APIs describe what is available starting with how to change the default refresh rate.

Contents

- *Operational State Configuration*
- *BGP RIB Operational State*
- *BGP RIB Families Operational State*
- *BGP Neighbors Operational State*
- *BGP Neighbor Operational State*
- *BGP Neighbor Families Operational State*
- *BGP Neighbor Family Operational State*
- *BGP Neighbor Timers Operational State*
- *BGP Neighbor Transport Operational State*
- *BGP Neighbor Error Handling Operational State*
- *BGP Neighbor Graceful Restart Operational State*
- *BGP Peer Groups Operational State*
  - *CLI*

Operational State Configuration

URL: `/restconf/config/bgp-state-config:bgp-state-config`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<bgp-state-config xmlns="urn:opendaylight:params:xml:ns:yang:controller:config">
    <config-name xmlns="urn:opendaylight:params:xml:ns:yang:bgp-state-config">
↪operationalState</config-name>
    <timer xmlns="urn:opendaylight:params:xml:ns:yang:bgp-state-config">1</timer>
</bgp-state-config>
```

@line 3: Time in seconds between operational state update.

BGP RIB Operational State

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/global/state`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<state xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <as>65000</as>
    <router-id>192.0.2.2</router-id>
    <total-paths>0</total-paths>
    <total-prefixes>0</total-prefixes>
</state>
```

@line 2: AS number of the remote peer.

@line 3: The unique protocol instance identifier.

@line 4: Total number of Paths installed on RIB (Loc-RIB)

@line 5: Total number of Prefixes installed on RIB (Loc-RIB)

BGP RIB Families Operational State

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/global/afi-safis`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<afi-safis xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <afi-safi>
        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</
↪afi-safi-name>
        <state>
            <total-paths>0</total-paths>
            <total-prefixes>0</total-prefixes>
        </state>
    </afi-safi>
    <afi-safi>
        <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-UNICAST</
↪afi-safi-name>
        <state>
            <total-paths>0</total-paths>
            <total-prefixes>0</total-prefixes>
        </state>
    </afi-safi>
    ....
</afi-safis>
```

@line 3: Family Identifier.

@line 5: Total number of Paths installed on RIB (Loc-RIB) per specific family.

@line 6: Total number of Prefixes installed on RIB (Loc-RIB) per specific family.

BGP Neighbors Operational State

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<neighbors xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor>
        <neighbor-address>192.0.2.1</neighbor-address>
        .....
    </neighbor>
    <neighbor>
        <neighbor-address>192.0.2.2</neighbor-address>
        .....
    </neighbor>
</neighbors>
```

@line 3: IP address of the remote BGP peer. Also serves as an unique identifier of a neighbor in a list of neighbors.

BGP Neighbor Operational State

Note: Supported Capabilities only provided when session has been established.

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/127.0.0.2/state`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<state xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <session-state>ESTABLISHED</session-state>
    <supported-capabilities xmlns:x="http://openconfig.net/yang/bgp-types">x:ASN32</supported-capabilities>
    <supported-capabilities xmlns:x="http://openconfig.net/yang/bgp-types">x:MPBGP</supported-capabilities>
    <messages>
        <sent>
            <UPDATE>0</UPDATE>
            <NOTIFICATION>0</NOTIFICATION>
        </sent>
        <received>
            <UPDATE>4</UPDATE>
            <NOTIFICATION>0</NOTIFICATION>
        </received>
    </messages>
</state>
```

@line 2: Session status

@line 3-4: BGP capabilities supported (ASN32 / MPBGP / ROUTE_REFRESH / GRACEFUL_RESTART / ADD_PATHS)

@line 7: Total count of Update Messages sent

@line 8: Total count of Notification Messages sent

@line 11: Total count of Update Messages received

@line 12: Total count of Notification Messages received

BGP Neighbor Families Operational State

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/afi-safis`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<afi-safis xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
     <afi-safi>
         <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST
↪</afi-safi-name>
         <state>
             <active>false</active>
         </state>
         <graceful-restart>
            <state>
                <received>true</received>
                <ll-received>true</ll-received>
                <ll-advertised>true</ll-advertised>
                <ll-stale-timer>180</ll-stale-timer>
                <advertised>true</advertised>
            </state>
        </graceful-restart>
     </afi-safi>
     <afi-safi>
         <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-UNICAST
↪</afi-safi-name>
         <state>
             <active>false</active>
         </state>
         <graceful-restart>
            <state>
                <received>true</received>
                <ll-received>true</ll-received>
                <ll-advertised>true</ll-advertised>
                <ll-stale-timer>100</ll-stale-timer>
                <advertised>true</advertised>
            </state>
        </graceful-restart>
     </afi-safi>
</afi-safis>
```

@line 3: Family Identifier.

@line 5: True if family is advertized by peer.

@line 7: Graceful Restart Operational State per specific family.

@line 9: True if the peer supports graceful restart.

@line 10: True if peer supports Long-Lived graceful restart.

@line 11: True if we supports Long-Lived graceful restart.

@line 12: Value of Long-Lived stale timer in seconds for specific family

@line 13: True if we support graceful restart.

BGP Neighbor Family Operational State

Note: Prefixes state is only provided once session is established.

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/afi-safis/afi-safi/openconfig-bgp-types:IPV4%2DUNICAST`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
1  <afi-safi xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2      <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-
   ↪safi-name>
3      <state>
4          <active>true</active>
5          <prefixes>
6              <installed>3</installed>
7              <sent>0</sent>
8              <received>3</received>
9          </prefixes>
10     </state>
11     <graceful-restart>
12         <state>
13             <received>true</received>
14             <ll-received>true</ll-received>
15             <ll-advertised>true</ll-advertised>
16             <ll-stale-timer>180</ll-stale-timer>
17             <advertised>true</advertised>
18         </state>
19     </graceful-restart>
20 </afi-safi>
```

@line 2: Family Identifier.

@line 4: True if family is advertized to and by peer.

@line 6: Total count of prefixes advertized by peer and installed (effective-rib-in).

@line 7: Total count of prefixes advertized to peer (adj-rib-out).

@line 8: Total count of prefixes advertized by peer (adj-rib-in).

BGP Neighbor Timers Operational State

Note: State is only provided once session is established.

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/timers`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<timers xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <state>
        <negotiated-hold-time>180</negotiated-hold-time>
        <uptime>1580676</uptime>
    </state>
</timers>
```

@line 3: The negotiated hold-time for the BGP session in seconds.

@line 4: Session duration since establishment in timeticks (hundredths of a second).

BGP Neighbor Transport Operational State

Note: State is only provided once session is established.

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/transport`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<transport xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <state>
        <remote-address>127.0.0.2</remote-address>
        <remote-port>44718</remote-port>
        <local-port>1790</local-port>
    </state>
</transport>
```

@line 3: IP address of the remote BGP peer.

@line 4: Port of the remote BGP peer.

@line 5: Local port.

BGP Neighbor Error Handling Operational State

Note: State is only provided once session is established.

Note: Error handling not supported yet. Planned for Carbon.

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/error-handling`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<error-handling xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <state>
        <erroneous-update-messages>0</erroneous-update-messages>
    </state>
</error-handling>
```

@line 3: The number of BGP UPDATE messages for which the treat-as-withdraw mechanism has been applied based on erroneous message contents

BGP Neighbor Graceful Restart Operational State

Note: Graceful Restart not supported yet. Planned for Carbon.

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/graceful-restart`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<graceful-restart xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions
↪">
    <state>
        <peer-restarting>false</peer-restarting>
        <local-restarting>false</local-restarting>
        <peer-restart-time>5</peer-restart-time>
        <mode>BILATERAL</mode>
    </state>
</graceful-restart>
```

@line 3: This flag indicates whether the remote neighbor is currently in the process of restarting, and hence received routes are currently stale.

@line 4: This flag indicates whether the local neighbor is currently restarting. The flag is unset after all NLRI have been advertised to the peer, and the End-of-RIB (EOR) marker has been unset.

@line 5: The period of time (advertised by the peer) in seconds that the peer expects a restart of a BGP session to take.

@line 6: Mode of Graceful Restart operation, depending on family support advertising to peer and receiving from peer can be HELPER-ONLY (only remote peers support some families), REMOTE-HELPER (only we advertise support), BILATERAL (two-side support).

BGP Peer Groups Operational State

URL: `/restconf/operational/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/peer-groups`

Method: `GET`

Content-Type: `application/xml`

Response Body:

```
<peer-groups>
    <peer-group>
        <peer-group-name>application-peers</peer-group-name>
        <state>
            <total-paths>0</total-paths>
            <total-prefixes>0</total-prefixes>
        </state>
    </peer-group>
</peer-groups>
```

@line 3: Peer Group Identifier.

@line 5: At this moment the cost for count path under effect-rib-in is to high. Therefore the value is the same as total prefixes.

@line 6: Total Prefixes installed under by peers pertaining to this peer group (effective-rib-in). This count doesn't differentiate repeated prefixes.

CLI

BGP Karaf Console (odl-bgpcep-bgp-cli) provides a CLI feature to read operational state per RIB, Neighbor and Peer Group.

```
opendaylight-user@root> bgp:operational-state -rib example-bgp-rib
```

```
opendaylight-user@root> bgp:operational-state -rib example-bgp-rib -neighbor 192.0.2.1
```

```
opendaylight-user@root> bgp:operational-state -rib -peer-group application-peers
```

2.1.20 High Availability

Running OpenDaylight BGP in clustered environment brings an advantage of the plugin's high availability (HA). This section illustrates a basic scenario for HA, also presents a configuration for clustered OpenDaylight BGP.

Contents

- *Configuration*
- *Failover scenario*

Configuration

Following example shows a configuration for running BGP in clustered environment.

1. As the first step, configure (replicated *deafult* shard and *topology* shard if needed) and run OpenDaylight in clustered environment, install BGP and RESTCONF.
2. On one node (OpenDaylight instance), configure BGP speaker instance and neighbor. In addition, configure BGP topology exporter if required. The configuration is shared across all interconnected cluster nodes, however BGP become active only on one node. Other nodes with configured BGP serves as stand-by backups.
3. Remote peer should be configured to accept/initiate connection from/to all OpenDaylight cluster nodes with configured BGP plugin.
4. Connect remote peer, let it advertise some routes. Verify routes presence in Loc-RIB and/or BGP topology exporter instance on all nodes of the OpenDaylight cluster.

Warning: Replicating RIBs across the cluster nodes is causing severe scalability issue and overall performance degradation. To avoid this problems, configure BGP RIB module as a separate shard without enabled replication. Change configuration on all nodes as following (*configuration/initial*):

- **In `modules.conf` add a new module:**

```
{
    name = "bgp_rib"
    namespace = "urn:opendaylight:params:xml:ns:yang:bgp-rib"
    shard-strategy = "module"
}
```

- **In `module-shards.conf` define a new module shard:**

```
{
    name = "bgp_rib"
    shards = [
        {
            name="bgp_rib"
            replicas = [
                "member-1"
            ]
        }
    ]
}
```

Note: Use correct member name in module shard configuration.

Failover scenario

Following section presents a basic BGP speaker failover scenario on 3-node OpenDaylight cluster setup.

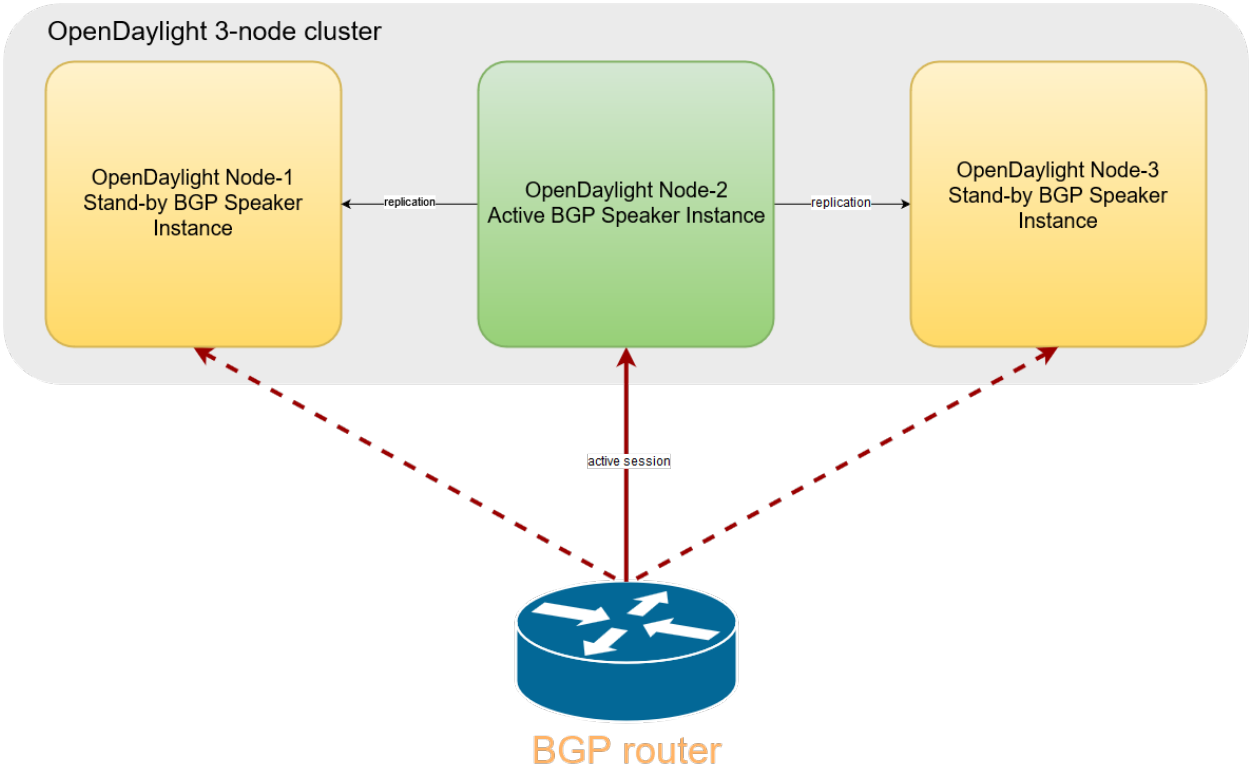


Fig. 3: Once the OpenDaylight BGP is configured, the speaker become active on one of the cluster nodes. Remote peer can establish connection with this BGP instance. Routes advertised by remote peer are replicated, hence RIBs state on all nodes is the same.

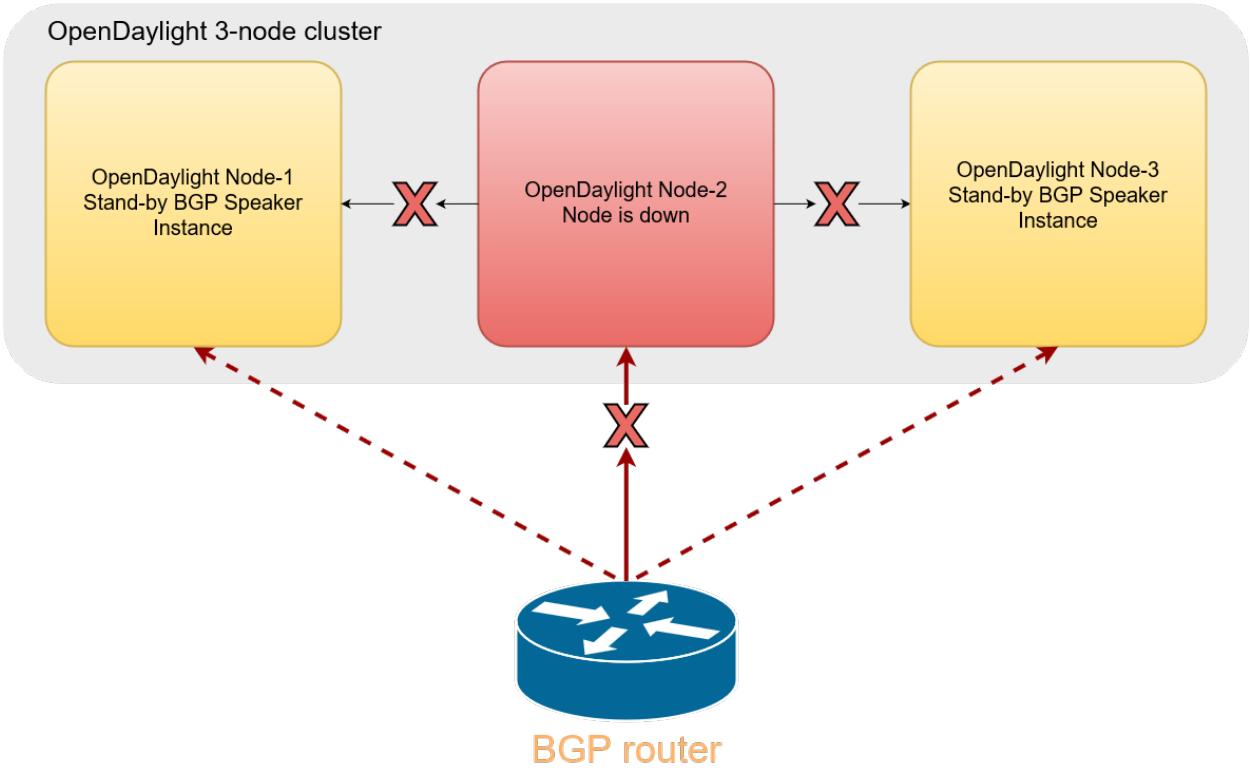


Fig. 4: In a case a cluster node, where BGP instance is running, goes down (unexpected failure, restart), active BGP session is dropped.

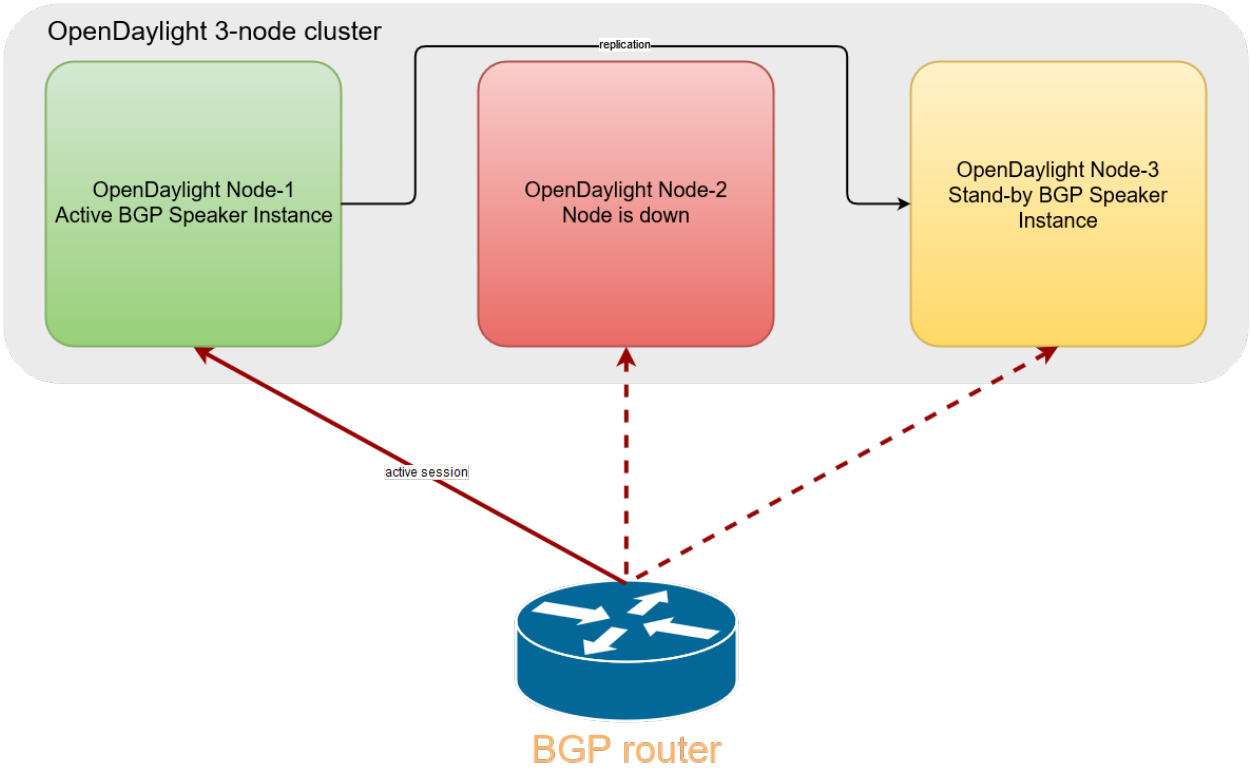


Fig. 5: Now, one of the stand-by BGP speaker instances become active. Remote peer establishes new connection and advertises routes again.

2.1.21 Topology Provider

This section provides an overview of the BGP topology provider service. It shows how to configure and use all available BGP topology providers. Providers are building topology view of BGP routes stored in local BGP speaker's Loc-RIB. Output topologies are rendered in a form of standardised IETF network topology model.

Contents

- *Inet Reachability Topology*
  - *Configuration*
  - *Usage*
- *BGP Linkstate Topology*
  - *Configuration*
  - *Usage*
- *BGP Network Topology Configuration Loader*

Inet Reachability Topology

Inet reachability topology exporter offers a mapping service from IPv4/6 routes to network topology nodes.

Configuration

Following example shows how to create a new instance of IPv4 BGP topology exporter:

URL: `/restconf/config/network-topology:network-topology`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1 <topology xmlns="urn:TBD:params:xml:ns:yang:network-topology">
2     <topology-id>bgp-example-ipv4-topology</topology-id>
3     <topology-types>
4         <bgp-ipv4-reachability-topology xmlns=
  ↪"urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-types"></bgp-ipv4-
  ↪reachability-topology>
5     </topology-types>
6     <rib-id xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-config">bgp-
  ↪example</rib-id>
7 </topology>
```

@line 2: An identifier for a topology.

@line 4: Used to identify type of the topology. In this case BGP IPv4 reachability topology.

@line 6: A name of the local BGP speaker instance.

The topology exporter instance can be removed in a following way:

URL: `/restconf/config/network-topology:network-topology/topology/bgp-example-ipv4-topology`

Method: `DELETE`

Following example shows how to create a new instance of IPv6 BGP topology exporter:

URL: `/restconf/config/network-topology:network-topology`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<topology xmlns="urn:TBD:params:xml:ns:yang:network-topology">
    <topology-id>bgp-example-ipv6-topology</topology-id>
    <topology-types>
        <bgp-ipv6-reachability-topology xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-types"></bgp-ipv6-
↪reachability-topology>
    </topology-types>
    <rib-id xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-config">bgp-
↪example</rib-id>
</topology>
```

Usage

Operational state of the topology can be verified via REST:

URL: `/restconf/operational/network-topology:network-topology/topology/bgp-example-ipv4-topology`

Method: `GET`

Response Body:

```
1  <topology xmlns="urn:TBD:params:xml:ns:yang:network-topology">
2      <topology-id>bgp-example-ipv4-topology</topology-id>
3      <server-provided>true</server-provided>
4      <topology-types>
5          <bgp-ipv4-reachability-topology xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-types"></bgp-ipv4-
↪reachability-topology>
6      </topology-types>
7      <node>
8          <node-id>10.10.1.1</node-id>
9          <igp-node-attributes xmlns="urn:TBD:params:xml:ns:yang:nt:l3-unicast-igp-
↪topology">
10             <prefix>
11                 <prefix>10.0.0.10/32</prefix>
12             </prefix>
13         </igp-node-attributes>
14     </node>
15 </topology>
```

@line 8: The identifier of a node in a topology. Its value is mapped from route's NEXT_HOP attribute.

@line 11: The IP prefix attribute of the node. Its value is mapped from routes's destination IP prefix.

BGP Linkstate Topology

BGP linkstate topology exporter offers a mapping service from BGP-LS routes to network topology nodes and links.

Configuration

Following example shows how to create a new instance of linkstate BGP topology exporter:

URL: `/restconf/config/network-topology:network-topology`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<topology  xmlns="urn:TBD:params:xml:ns:yang:network-topology">
    <topology-id>bgp-example-linkstate-topology</topology-id>
    <topology-types>
        <bgp-linkstate-topology xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-
↪topology-types"></bgp-linkstate-topology>
    </topology-types>
    <rib-id xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-config">bgp-
↪example</rib-id>
</topology>
```

Usage

Operational state of the topology can be verified via REST. A sample output below represents a two node topology with two unidirectional links interconnecting those nodes.

URL: `/restconf/operational/network-topology:network-topology/topology/bgp-example-linkstate-topology`

Method: `GET`

Response Body:

```
<topology xmlns="urn:TBD:params:xml:ns:yang:network-topology">
    <topology-id>bgp-example-linkstate-topology</topology-id>
    <server-provided>true</server-provided>
    <topology-types>
        <bgp-linkstate-topology xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-
↪topology-types"></bgp-linkstate-topology>
    </topology-types>
    <node>
        <node-id>bgpls://IsisLevel2:1/type=node&amp;as=65000&amp;domain=673720360&amp;
↪router=0000.0000.0040</node-id>
        <termination-point>
            <tp-id>bgpls://IsisLevel2:1/type=tp&amp;ipv4=203.20.160.40</tp-id>
            <igp-termination-point-attributes xmlns="urn:TBD:params:xml:ns:yang:nt:l3-
↪unicast-igp-topology"/>
        </termination-point>
        <igp-node-attributes xmlns="urn:TBD:params:xml:ns:yang:nt:l3-unicast-igp-
↪topology">
            <prefix>
                <prefix>40.40.40.40/32</prefix>
```

(continues on next page)

(continued from previous page)

```
                <metric>10</metric>
            </prefix>
            <prefix>
                <prefix>203.20.160.0/24</prefix>
                <metric>10</metric>
            </prefix>
            <name>node1</name>
            <router-id>40.40.40.40</router-id>
            <isis-node-attributes xmlns="urn:TBD:params:xml:ns:yang:network:isis-
↪topology">
                <ted>
                    <te-router-id-ipv4>40.40.40.40</te-router-id-ipv4>
                </ted>
                <iso>
                    <iso-system-id>MDAwMDAwMDAwMDY0</iso-system-id>
                </iso>
            </isis-node-attributes>
        </igp-node-attributes>
    </node>
    <node>
        <node-id>bgpls://IsisLevel2:1/type=node&amp;as=65000&amp;domain=673720360&amp;
↪router=0000.0000.0039</node-id>
        <termination-point>
            <tp-id>bgpls://IsisLevel2:1/type=tp&amp;ipv4=203.20.160.39</tp-id>
            <igp-termination-point-attributes xmlns="urn:TBD:params:xml:ns:yang:nt:l3-
↪unicast-igp-topology"/>
        </termination-point>
        <igp-node-attributes xmlns="urn:TBD:params:xml:ns:yang:nt:l3-unicast-igp-
↪topology">
            <prefix>
                <prefix>39.39.39.39/32</prefix>
                <metric>10</metric>
            </prefix>
            <prefix>
                <prefix>203.20.160.0/24</prefix>
                <metric>10</metric>
            </prefix>
            <name>node2</name>
            <router-id>39.39.39.39</router-id>
            <isis-node-attributes xmlns="urn:TBD:params:xml:ns:yang:network:isis-
↪topology">
                <ted>
                    <te-router-id-ipv4>39.39.39.39</te-router-id-ipv4>
                </ted>
                <iso>
                    <iso-system-id>MDAwMDAwMDAwMDg3</iso-system-id>
                </iso>
            </isis-node-attributes>
        </igp-node-attributes>
    </node>
    <link>
        <destination>
            <dest-node>bgpls://IsisLevel2:1/type=node&amp;as=65000&amp;
↪domain=673720360&amp;router=0000.0000.0039</dest-node>
            <dest-tp>bgpls://IsisLevel2:1/type=tp&amp;ipv4=203.20.160.39</dest-tp>
        </destination>
        <link-id>bgpls://IsisLevel2:1/type=link&amp;local-as=65000&amp;local-
↪domain=673720360&amp;local-router=0000.0000.0040&amp;remote-as=65000&amp;remote-as-
↪domain=673720360&amp;remote-router=0000.0000.0039&amp;ipv4-iface=203.20.160.40&amp;
↪ipv4-neigh=203.20.160.39</link-id>
```

(continues on next page)

(continued from previous page)

```
        <source>
            <source-node>bgpls://IsisLevel2:1/type=node&amp;as=65000&amp;
↪domain=673720360&amp;router=0000.0000.0040</source-node>
            <source-tp>bgpls://IsisLevel2:1/type=tp&amp;ipv4=203.20.160.40</source-tp>
        </source>
        <igp-link-attributes xmlns="urn:TBD:params:xml:ns:yang:nt:l3-unicast-igp-
↪topology">
            <metric>10</metric>
            <isis-link-attributes xmlns="urn:TBD:params:xml:ns:yang:network:isis-
↪topology">
                <ted>
                    <color>0</color>
                    <max-link-bandwidth>1250000.0</max-link-bandwidth>
                    <max-resv-link-bandwidth>12500.0</max-resv-link-bandwidth>
                    <te-default-metric>0</te-default-metric>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>0</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>1</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>2</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>3</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>4</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>5</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>6</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>7</priority>
                    </unreserved-bandwidth>
                </ted>
            </isis-link-attributes>
        </igp-link-attributes>
    </link>
    <link>
        <destination>
            <dest-node>bgpls://IsisLevel2:1/type=node&amp;as=65000&amp;
↪domain=673720360&amp;router=0000.0000.0040</dest-node>
            <dest-tp>bgpls://IsisLevel2:1/type=tp&amp;ipv4=203.20.160.40</dest-tp>
        </destination>
```

(continues on next page)

(continued from previous page)

```
        <link-id>bgpls://IsisLevel2:1/type=link&amp;local-as=65000&amp;local-
↪domain=673720360&amp;local-router=0000.0000.0039&amp;remote-as=65000&amp;remote-
↪domain=673720360&amp;remote-router=0000.0000.0040&amp;ipv4-iface=203.20.160.39&amp;
↪ipv4-neigh=203.20.160.40</link-id>
        <source>
            <source-node>bgpls://IsisLevel2:1/type=node&amp;as=65000&amp;
↪domain=673720360&amp;router=0000.0000.0039</source-node>
            <source-tp>bgpls://IsisLevel2:1/type=tp&amp;ipv4=203.20.160.39</source-tp>
        </source>
        <igp-link-attributes xmlns="urn:TBD:params:xml:ns:yang:nt:l3-unicast-igp-
↪topology">
            <metric>10</metric>
            <isis-link-attributes xmlns="urn:TBD:params:xml:ns:yang:network:isis-
↪topology">
                <ted>
                    <color>0</color>
                    <max-link-bandwidth>1250000.0</max-link-bandwidth>
                    <max-resv-link-bandwidth>12500.0</max-resv-link-bandwidth>
                    <te-default-metric>0</te-default-metric>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>0</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>1</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>2</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>3</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>4</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>5</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>6</priority>
                    </unreserved-bandwidth>
                    <unreserved-bandwidth>
                        <bandwidth>12500.0</bandwidth>
                        <priority>7</priority>
                    </unreserved-bandwidth>
                </ted>
            </isis-link-attributes>
        </igp-link-attributes>
    </link>
</topology>
```

BGP Network Topology Configuration Loader

BGP Network Topology Configuration Loader allows user to define static initial configuration for a BGP protocol instance. This service will detect the creation of new configuration files following the pattern `network-topology-*.xml` under the path `etc/opendaylight/bgpcep`. Once the file is processed, the defined configuration will be available from the configuration Data Store.

Note: If the BGP topology instance is already present, no update or configuration will be applied.

PATH: `etc/opendaylight/bgpcep/network-topology-config.xml`

```
<network-topology xmlns="urn:TBD:params:xml:ns:yang:network-topology">
    <topology>
        <topology-id>example-ipv4-topology</topology-id>
        <topology-types>
            <bgp-ipv4-reachability-topology xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-types"/>
        </topology-types>
        <rib-id xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-config">
↪example-bgp-rib</rib-id>
    </topology>
    <topology>
        <topology-id>example-ipv6-topology</topology-id>
        <topology-types>
            <bgp-ipv6-reachability-topology xmlns=
↪"urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-types"/>
        </topology-types>
        <rib-id xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-config">
↪example-bgp-rib</rib-id>
    </topology>
    <topology>
        <topology-id>example-linkstate-topology</topology-id>
        <topology-types>
            <bgp-linkstate-topology xmlns="urn:opendaylight:params:xml:ns:yang:odl-
↪bgp-topology-types"/>
        </topology-types>
        <rib-id xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-topology-config">
↪example-bgp-rib</rib-id>
    </topology>
</network-topology>
```

2.1.22 Test Tools

BGP test tools serves to test basic BGP functionality, scalability and performance.

Contents

- *BGP Test Tool*
  - *Usage*
- *BGP Application Peer Benchmark*
  - *Configuration*

- *Inject routes*
- *Remove routes*

BGP Test Tool

The BGP Test Tool is a stand-alone Java application purposed to simulate remote BGP peers, that are capable to advertise sample routes. This application is not part of the OpenDaylight Karaf distribution, however it can be downloaded from OpenDaylight's Nexus (use latest release version):

`https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/bgpcep/bgp-testtool`

Usage

The application can be run from command line:

```
java -jar bgp-testtool-*-executable.jar
```

with optional input parameters:

```
-i <BOOLEAN>, --active <BOOLEAN>
   Active initialisation of the connection, by default false.

-ho <N>, --holdtimer <N>
   In seconds, value of the desired holdtimer, by default 90.

-sc <N>, --speakersCount <N>
   Number of simulated BGP speakers, when creating each speaker, uses incremented␣
↪local-address for binding, by default 0.

-ra <IP_ADDRESS:PORT,...>, --remote-address <IP_ADDRESS:PORT,...>
   A list of IP addresses of remote BGP peers, that the tool can accept or initiate␣
↪connect to that address (based on the mode), by default 192.0.2.2:1790.

-la <IP_ADDRESS:PORT>, --localAddress <IP_ADDRESS:PORT>
   IP address of BGP speakers which the tools simulates, by default 192.0.2.2:0.

-pr <N>, --prefixes <N>
   Number of prefixes to be advertised by each simulated speaker

-mp <BOOLEAN>, --multiPathSupport <BOOLEAN>
   Active ADD-PATH support, by default false.

-as <N>, --as <N>
   Local AS Number, by default 64496.

-ec <EXTENDED_COMMUNITIES>, --extended_communities <EXTENDED_COMMUNITIES>
   Extended communities to be send. Format: x,x,x  where  x  is  each  extended ␣
↪community from bgp-types.yang, by default empty.

-ll <LOG_LEVEL>, --log_level <LOG_LEVEL>
   Log level for console output, by default INFO.
```

BGP Application Peer Benchmark

It is a simple OpenDaylight application which is capable to inject and remove specific amount of IPv4 routes. This application is part of the OpenDaylight Karaf distribution.

Configuration

As a first step install BGP and RESTCONF, then configure *Application Peer*. Install `odl-bgpcep-bgp-benchmark` feature and reconfigure BGP Application Peer Benchmark application as per following:

URL: `/restconf/config/odl-bgp-app-peer-benchmark-config:config`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<odl-bgp-app-peer-benchmark-config xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-app-peer-benchmark-config">
   <app-peer-id xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-app-peer-benchmark-config">10.25.1.9</app-peer-id>
</odl-bgp-app-peer-benchmark-config>
```

@line 2: The *Application Peer* identifier.

Inject routes

Routes injection can be invoked via RPC:

URL: `/restconf/operations/odl-bgp-app-peer-benchmark:add-prefix`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-app-peer-benchmark">
    <prefix>1.1.1.1/32</prefix>
    <count>100000</count>
    <batchsize>2000</batchsize>
    <nexthop>192.0.2.2</nexthop>
</input>
```

@line 2: A initial IPv4 prefix carried in route. Value is incremented for following routes.

@line 3: An amount of routes to be added to *Application Peer's* programmable RIB.

@line 4: A size of the transaction batch.

@line 5: A NEXT_HOP attribute value used in all injected routes.

Response Body:

```
<output xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-app-peer-benchmark">
    <result>
        <duration>4301</duration>
        <rate>25000</rate>
        <count>100000</count>
    </result>
</output>
```

@line 3: Request duration in milliseconds.

@line 4: Writes per second rate.

@line 5: An amount of routes added to *Application Peer's* programmable RIB.

Remove routes

Routes deletion can be invoked via RPC:

URL: `/restconf/operations/odl-bgp-app-peer-benchmark:delete-prefix`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-app-peer-benchmark">
    <prefix>1.1.1.1/32</prefix>
    <count>100000</count>
    <batchsize>2000</batchsize>
</input>
```

@line 2: A initial IPv4 prefix carried in route to be removed. Value is incremented for following routes.

@line 3: An amount of routes to be removed from *Application Peer's* programmable RIB.

@line 4: A size of the transaction batch.

Response Body:

```
<output xmlns="urn:opendaylight:params:xml:ns:yang:odl-bgp-app-peer-benchmark">
    <result>
        <duration>1837</duration>
        <rate>54500</rate>
        <count>100000</count>
    </result>
</output>
```

2.1.23 PSMI Attribute

- **P-Multicast Service Interface Tunnel (PMSI) attribute:**
  - **RSVP-TE P2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <rsvp-te-p2mp-lsp>
```

(continues on next page)

(continued from previous page)

```
        <p2mp-id>1111111111</p2mp-id>
        <tunnel-id>11111</tunnel-id>
        <extended-tunnel-id>10.10.10.10</extended-tunnel-id>
    </rsvp-te-p2mp-lsp>
</pmsi-tunnel>
```

– **mLDP P2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <mldp-p2mp-lsp>
        <address-family xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types
↪">x:ipv4-address-family</address-family>
        <root-node-address>10.10.10.10</root-node-address>
        <opaque-value>
            <opaque-type>255</opaque-type>
            <opaque-extended-type>11111</opaque-extended-type>
            <opaque>aa:aa:aa</opaque>
        </opaque-value>
    </mldp-p2mp-lsp>
</pmsi-tunnel>
```

– **PIM-SSM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <pim-ssm-tree>
        <p-address>11.12.13.14</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </pim-ssm-tree>
</pmsi-tunnel>
```

– **PIM-SM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <pim-sm-tree>
        <p-address>1.0.0.1</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </pim-sm-tree>
</pmsi-tunnel>
```

– **BIDIR-PIM Tree**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <bidir-pim-tree>
        <p-address>1.0.0.1</p-address>
        <p-multicast-group>10.10.10.10</p-multicast-group>
    </bidir-pim-tree>
</pmsi-tunnel>
```

– **Ingress Replication**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <ingress-replication>
        <receiving-endpoint-address>172.12.123.3</receiving-endpoint-address>
    </ingress-replication>
</pmsi-tunnel>
```

– **mLDP MP2MP LSP**

```
<pmsi-tunnel>
    <leaf-information-required>true</leaf-information-required>
    <mpls-label>20024</mpls-label>
    <mldp-mp2mp-lsp>
        <opaque-type>255</opaque-type>
        <opaque-extended-type>11111</opaque-extended-type>
        <opaque>aa:aa</opaque>
    </mldp-mp2mp-lsp>
</pmsi-tunnel>
```

2.1.24 Troubleshooting

This section offers advices in a case OpenDaylight BGP plugin is not working as expected.

Contents

- *BGP is not working...*
- *Bug reporting*

BGP is not working...

- First of all, ensure that all required features are installed, local and remote BGP configuration is correct.
- Check OpenDaylight Karaf logs:

From Karaf console:

```
log:tail
```

or open log file: `data/log/karaf.log`

Possibly, a reason/hint for a cause of the problem can be found there.

- Try to minimise effect of other OpenDaylight features, when searching for a reason of the problem.
- Try to set DEBUG severity level for BGP logger via Karaf console commands, in order to collect more information:

```
log:set DEBUG org.opendaylight.protocol.bgp
```

```
log:set DEBUG org.opendaylight.bgpcep.bgp
```

Bug reporting

Before you report a bug, check BGPCEP Jira to ensure same/similar bug is not already filed there.

Write an e-mail to bgpcep-users@lists.opendaylight.org and provide following information:

1. State OpenDaylight version
2. Describe your use-case and provide as much details related to BGP as possible
3. Steps to reproduce
4. Attach Karaf log files, optionally packet captures, REST input/output

2.1.25 Revised Error Handling for BGP UPDATE Messages

According to RFC4271 a BGP speaker that receives an UPDATE message containing a malformed attribute is required to reset the session over which the offending attribute was received. Revised Error Handling procedures defined in RFC7606 is introducing a ways to avoid negative effects of session restart. This document provides guide to configure specific approach called *treat-as-withdraw* which is treating malformed UPDATE messages as their withdrawal equivalent.

Configuration

Treat-as-withdraw procedures are disabled by default. There are two ways to enable it. One via *peer-group* to affect all neighbors in that group and one via *neighbor*.

For *neighbor* configuration:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/neighbors/neighbor/192.0.2.1/error-handling`

For *peer-group* configuration:

URL: `/restconf/config/openconfig-network-instance:network-instances/network-instance/global-bgp/openconfig-network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-example/bgp/peer-groups/peer-group/external-neighbor/error-handling`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1 <error-handling xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
2     <config>
3         <treat-as-withdraw>true</treat-as-withdraw>
4     </config>
5 </error-handling>
```

@line 3: *True* to enable *treat-as-withdraw* procedures, *False* to disabled it

Note: If neighbor Error handling configuration in *neighbor* ALWAYS supersed *peer-group* configuration. That means if *peer-group* have error handling enabled and *neighbor* disabled, result is disabled error handling.

References

- Revised Error Handling for BGP UPDATE Messages

2.2 BGP Monitoring Protocol User Guide

This guide contains information on how to use the OpenDaylight BGP Monitoring Protocol (BMP) plugin. It covers BMP basic concepts, supported capabilities, configuration and operations.

2.2.1 Overview

This section provides high-level overview of the BMP plugin, OpenDaylight implementation and BMP usage for SDN.

Contents

- *BGP Monitoring Protocol*
- *BMP in SDN*
- *OpenDaylight BMP plugin*

BGP Monitoring Protocol

The BGP Monitoring Protocol (BMP) serves to monitor BGP sessions. The BMP can be used to obtain route view instead of screen scraping. The BMP provides access to unprocessed routing information (Adj-RIB-In) and processed routes (applied inbound policy) of monitored router's peer. In addition, monitored router can provide periodic dump of statistics.

The BMP runs over TCP. Both monitored router and monitoring station can be configured as active or passive party of the connection. The passive party listens at particular port. The router can be monitored by multiple monitoring stations. BMP messages are sent by monitored router only, monitoring station supposed to collect and process data received over BMP.

BMP in SDN

The main concept of BMP is to monitor BGP sessions - monitoring station is aware of monitored peer's status, collects statistics and analyzes them in order to provide valuable information for network operators.

Moreover, BMP provides provides peer RIBs visibility, without need to establish BGP sessions. Unprocessed routes may serve as a source of information for software-driven routing optimization. In this case, SDN controller, a BMP monitoring station, collects routing information from monitored routers. The routes are used in subsequent optimization procedures.

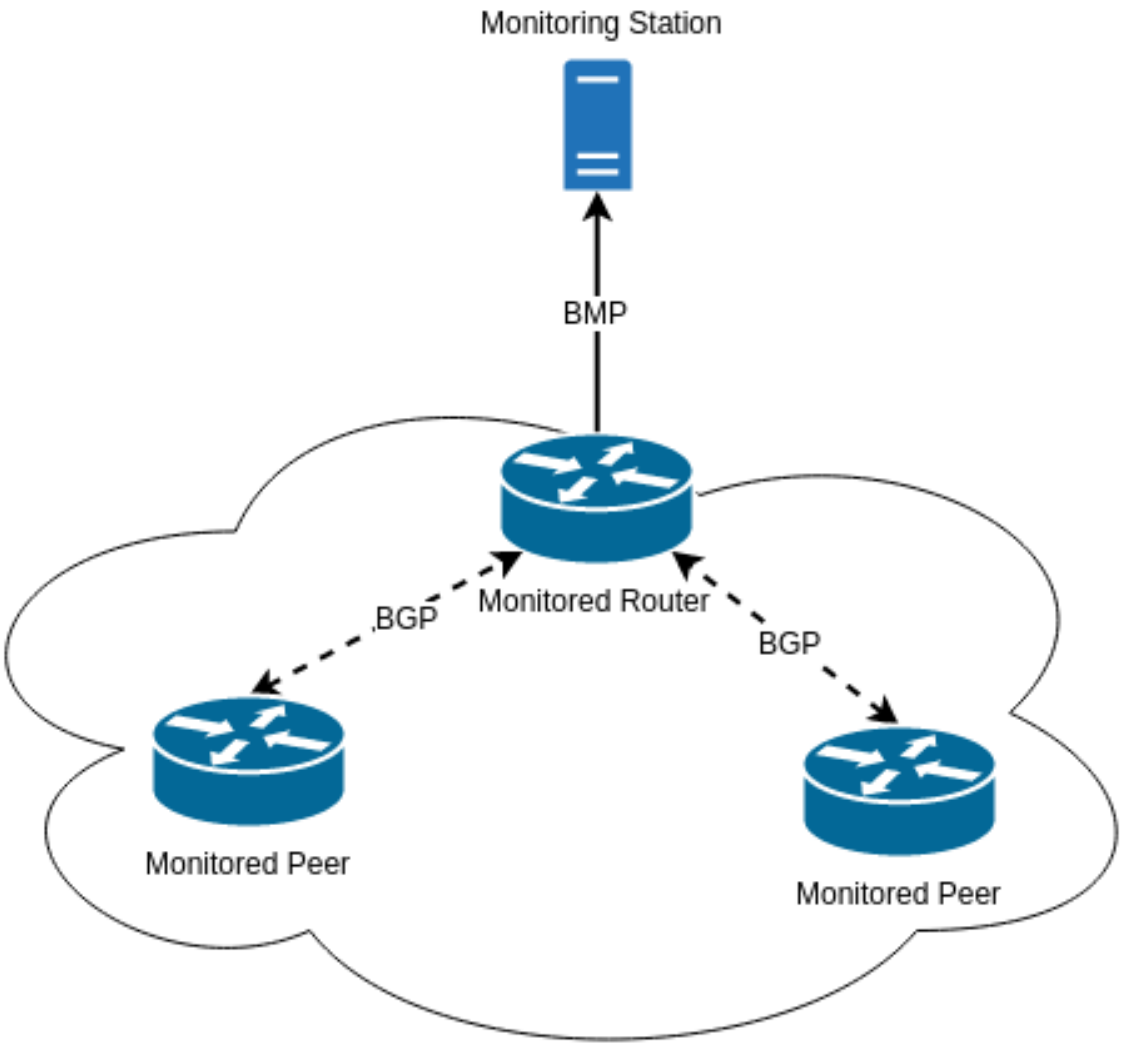


Fig. 6: The BMP overview - Monitoring Station, Monitored Router and Monitored Peers.

OpenDaylight BMP plugin

The OpenDaylight BMP plugin provides monitoring station implementation. The plugin can establish BMP session with one or more monitored routers in order to collect routing and statistical information.

- Runtime configurable monitoring station
- Read-only routes and statistics view
- Supports various routing information types

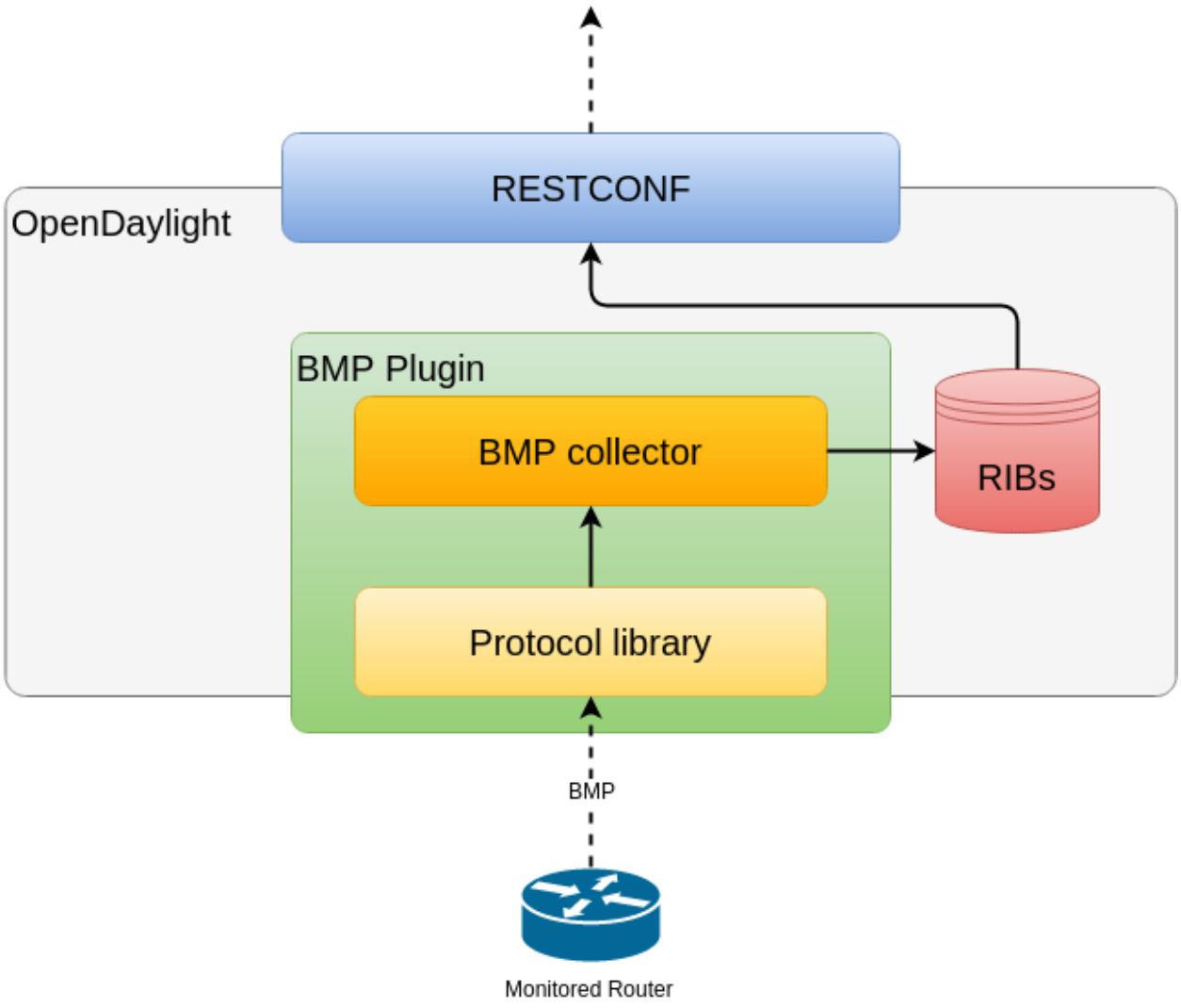


Fig. 7: OpenDaylight BMP plugin overview.

Important: The BMP plugin is not storing historical data, it provides current snapshot only.

2.2.2 List of supported capabilities

The BMP plugin implementation is based on Internet standards:

- RFC7854 - BGP Monitoring Protocol (BMP)

Note: The BMP plugin is capable to process various types of routing information (IP Unicast, EVPN, L3VPN, Link-State,...). Please, see complete list in BGP user guide.

2.2.3 Running BMP

This section explains how to install BMP plugin.

1. Install BMP feature - `odl-bgpcep-bmp`. Also, for sake of this sample, it is required to install RESTCONF. In the Karaf console, type command:

```
feature:install odl-restconf odl-bgpcep-bmp
```

2. The BMP plugin contains a default configuration, which is applied after the feature starts up. One instance of BMP monitoring station is created (named *example-bmp-monitor*), and its presence can be verified via REST:

   URL: `/restconf/config/odl-bmp-monitor-config:odl-bmp-monitors/bmp-monitor-config/example-bmp-monitor`

   Method: `GET`

   Response Body:

```
<bmp-monitor-config xmlns="urn:opendaylight:params:xml:ns:yang:bmp-monitor-config
↪">
   <monitor-id>example-bmp-monitor</monitor-id>
   <server>
      <binding-port>12345</binding-port>
      <binding-address>0.0.0.0</binding-address>
   </server>
</bmp-monitor-config>
```

2.2.4 BMP Monitoring Station

The following section shows how to configure BMP basics, how to verify functionality and presents essential components of the plugin. Next samples demonstrate the plugin's runtime configuration capability.

The monitoring station is responsible for received BMP PDUs processing and storage. The default BMP server is listening at port *12345*.

Contents

- *Configuration*
  - *Monitoring station configuration*
  - *Active mode configuration*
  - *MD5 authentication configuration*

- *BMP Monitors Configuration Loader*
  - *BMP Monitor Configuration Example*
- *Collector DB Tree*
- *Operations*

Configuration

This section shows the way to configure the BMP monitoring station via REST API.

Monitoring station configuration

In order to change default's BMP monitoring station configuration, use following request.

URL: `/restconf/config/odl-bmp-monitor-config:odl-bmp-monitors/bmp-monitor-config/example-bmp-monitor`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1 <bmp-monitor-config xmlns="urn:opendaylight:params:xml:ns:yang:bmp-monitor-config">
2    <monitor-id>example-bmp-monitor</monitor-id>
3    <server>
4       <binding-port>12345</binding-port>
5       <binding-address>0.0.0.0</binding-address>
6    </server>
7 </bmp-monitor-config>
```

@line 4: **binding-port** - The BMP server listening port.

@line 5: **binding-address** - The BMP server biding address.

Note: User may create multiple BMP monitoring station instances at runtime.

Active mode configuration

In order to enable active connection, use following request.

URL: `/restconf/config/odl-bmp-monitor-config:odl-bmp-monitors/bmp-monitor-config/example-bmp-monitor`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1 <bmp-monitor-config xmlns="urn:opendaylight:params:xml:ns:yang:bmp-monitor-config">
2    <monitor-id>example-bmp-monitor</monitor-id>
3    <server>
```

(continues on next page)

(continued from previous page)

```
4        <binding-port>12345</binding-port>
5        <binding-address>0.0.0.0</binding-address>
6     </server>
7     <monitored-router>
8        <address>192.0.2.2</address>
9        <port>1234</port>
10       <active>true</active>
11    </monitored-router>
12 </bmp-monitor-config>
```

@line 8: **address** - The monitored router's IP address.

@line 9: **port** - The monitored router's port.

@line 10: **active** - Active mode set.

Note: User may configure active session establishment for multiple monitored routers.

MD5 authentication configuration

In order to enable active connection, use following request.

URL: `/restconf/config/odl-bmp-monitor-config:odl-bmp-monitors/bmp-monitor-config/example-bmp-monitor`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1  <bmp-monitor-config xmlns="urn:opendaylight:params:xml:ns:yang:bmp-monitor-config">
2     <monitor-id>example-bmp-monitor</monitor-id>
3     <server>
4        <binding-port>12345</binding-port>
5        <binding-address>0.0.0.0</binding-address>
6     </server>
7     <monitored-router>
8        <address>192.0.2.2</address>
9        <password>changeme</password>
10    </monitored-router>
11 </bmp-monitor-config>
```

@line 8: **address** - The monitored router's IP address.

@line 9: **password** - The TCP MD5 signature.

BMP Monitors Configuration Loader

BMP Monitors Configuration Loader allows user to define static initial configuration for a BMP protocol instance. This service will detect the creation of new configuration files following the pattern `odl-bmp-monitors-*.xml` under the path `etc/opendaylight/bgpcep`. Once the file is processed, the defined configuration will be available from the configuration Data Store.

Note: If the BMP Monitor instance is already present, no update or configuration will be applied.

PATH: `etc/opendaylight/bgpcep/odl-bmp-monitors-config.xml`

```
<odl-bmp-monitors xmlns="urn:opendaylight:params:xml:ns:yang:bmp-monitor-config">
    <bmp-monitor-config>
        <monitor-id>example-bmp-monitor</monitor-id>
        <server>
            <binding-port>12345</binding-port>
            <binding-address>0.0.0.0</binding-address>
        </server>
    </bmp-monitor-config>
</odl-bmp-monitors>
```

BMP Monitor Configuration Example

BGP provides a feature providing a BMP Monitor configuration file example. Once feature is installed defined configuration will be loaded and setup.

```
feature:install odl-bgpcep-bmp-config-example
```

Collector DB Tree

```
module: bmp-monitor
   +--rw bmp-monitor
      +--ro monitor* [monitor-id]
         +--ro monitor-id    monitor-id
         +--ro router* [router-id]
            +--ro name?          string
            +--ro description?   string
            +--ro info?          string
            +--ro router-id      router-id
            +--ro status?        status
            +--ro peer* [peer-id]
               +--ro peer-id                rib:peer-id
               +--ro type                   peer-type
               x--ro distinguisher
               |  +--ro distinguisher-type?   distinguisher-type
               |  +--ro distinguisher?        string
               +--ro peer-distinguisher?    union
               +--ro address                inet:ip-address
               +--ro as                     inet:as-number
               +--ro bgp-id                 inet:ipv4-address
               +--ro router-distinguisher?  string
               +--ro peer-session
```

(continues on next page)

(continued from previous page)

```
|  +--ro local-address      inet:ip-address
|  +--ro local-port         inet:port-number
|  +--ro remote-port        inet:port-number
|  +--ro sent-open
|  |  +--ro version?          protocol-version
|  |  +--ro my-as-number?     uint16
|  |  +--ro hold-timer        uint16
|  |  +--ro bgp-identifier    inet:ipv4-address
|  |  +--ro bgp-parameters*
|  |     +--ro optional-capabilities*
|  |        +--ro c-parameters
|  |           +--ro as4-bytes-capability
|  |           |  +--ro as-number?   inet:as-number
|  |           +--ro bgp-extended-message-capability!
|  |           +--ro multiprotocol-capability
|  |           |  +--ro afi?    identityref
|  |           |  +--ro safi?   identityref
|  |           +--ro graceful-restart-capability
|  |           |  +--ro restart-flags    bits
|  |           |  +--ro restart-time     uint16
|  |           |  +--ro tables* [afi safi]
|  |           |     +--ro afi          identityref
|  |           |     +--ro safi         identityref
|  |           |     +--ro afi-flags    bits
|  |           +--ro add-path-capability
|  |           |  +--ro address-families*
|  |           |     +--ro afi?            identityref
|  |           |     +--ro safi?           identityref
|  |           |     +--ro send-receive?   send-receive
|  |           +--ro route-refresh-capability!
|  +--ro received-open
|  |  +--ro version?          protocol-version
|  |  +--ro my-as-number?     uint16
|  |  +--ro hold-timer        uint16
|  |  +--ro bgp-identifier    inet:ipv4-address
|  |  +--ro bgp-parameters*
|  |     +--ro optional-capabilities*
|  |        +--ro c-parameters
|  |           +--ro as4-bytes-capability
|  |           |  +--ro as-number?   inet:as-number
|  |           +--ro bgp-extended-message-capability!
|  |           +--ro multiprotocol-capability
|  |           |  +--ro afi?    identityref
|  |           |  +--ro safi?   identityref
|  |           +--ro graceful-restart-capability
|  |           |  +--ro restart-flags    bits
|  |           |  +--ro restart-time     uint16
|  |           |  +--ro tables* [afi safi]
|  |           |     +--ro afi          identityref
|  |           |     +--ro safi         identityref
|  |           |     +--ro afi-flags    bits
|  |           +--ro add-path-capability
|  |           |  +--ro address-families*
|  |           |     +--ro afi?            identityref
|  |           |     +--ro safi?           identityref
|  |           |     +--ro send-receive?   send-receive
|  |           +--ro route-refresh-capability!
```

(continues on next page)

(continued from previous page)

```
|  +--ro information
|  |  +--ro string-information*
|  |     +--ro string-tlv
|  |        +--ro string-info?   string
|  +--ro status?            status
|  +--ro timestamp-sec?     yang:timestamp
|  +--ro timestamp-micro?   yang:timestamp
+--ro stats
|  +--ro rejected-prefixes?                 yang:counter32
|  +--ro duplicate-prefix-advertisements?   yang:counter32
|  +--ro duplicate-withdraws?               yang:counter32
|  +--ro invalidated-cluster-list-loop?     yang:counter32
|  +--ro invalidated-as-path-loop?          yang:counter32
|  +--ro invalidated-originator-id?         yang:counter32
|  +--ro invalidated-as-confed-loop?        yang:counter32
|  +--ro adj-ribs-in-routes?                yang:gauge64
|  +--ro loc-rib-routes?                    yang:gauge64
|  +--ro per-afi-safi-adj-rib-in-routes
|  |  +--ro afi-safi* [afi safi]
|  |     +--ro afi      identityref
|  |     +--ro safi     identityref
|  |     +--ro count?   yang:gauge64
|  +--ro per-afi-safi-loc-rib-routes
|  |  +--ro afi-safi* [afi safi]
|  |     +--ro afi      identityref
|  |     +--ro safi     identityref
|  |     +--ro count?   yang:gauge64
|  +--ro updates-treated-as-withdraw?       yang:counter32
|  +--ro prefixes-treated-as-withdraw?      yang:counter32
|  +--ro duplicate-updates?                 yang:counter32
|  +--ro timestamp-sec?                     yang:timestamp
|  +--ro timestamp-micro?                   yang:timestamp
+--ro pre-policy-rib
|  +--ro tables* [afi safi]
|     +--ro afi           identityref
|     +--ro safi          identityref
|     +--ro attributes
|     |  +--ro uptodate?   boolean
|     +--ro (routes)?
+--ro post-policy-rib
|  +--ro tables* [afi safi]
|     +--ro afi           identityref
|     +--ro safi          identityref
|     +--ro attributes
|     |  +--ro uptodate?   boolean
|     +--ro (routes)?
+--ro mirrors
   +--ro information?       bmp-msg:mirror-information-code
   +--ro timestamp-sec?     yang:timestamp
   +--ro timestamp-micro?   yang:timestamp
```

Operations

The BMP plugin offers view of collected routes and statistical information from monitored peers. To get top-level view of monitoring station:

URL: `/restconf/operational/bmp-monitor:bmp-monitor/monitor/example-bmp-monitor`

Method: `GET`

Response Body:

```
<bmp-monitor xmlns="urn:opendaylight:params:xml:ns:yang:bmp-monitor">
   <monitor>
      <monitor-id>example-bmp-monitor</monitor-id>
         <router>
            <router-id>10.10.10.10</router-id>
            <name>name</name>
            <description>monitored-router</description>
            <info>monitored router;</info>
            <status>up</status>
            <peer>
                <peer-id>20.20.20.20</peer-id>
                <address>20.20.20.20</address>
                <bgp-id>20.20.20.20</bgp-id>
                <as>65000</as>
                <type>global</type>
                <peer-session>
                  <remote-port>1790</remote-port>
                  <timestamp-sec>0</timestamp-sec>
                  <status>up</status>
                  <local-address>10.10.10.10</local-address>
                  <local-port>2200</local-port>
                  <received-open>
                     <hold-timer>180</hold-timer>
                     <my-as-number>65000</my-as-number>
                     <bgp-identifier>20.20.20.20</bgp-identifier>
                  </received-open>
                  <sent-open>
                     <hold-timer>180</hold-timer>
                     <my-as-number>65000</my-as-number>
                     <bgp-identifier>65000</bgp-identifier>
                  </sent-open>
                </peer-session>
                <pre-policy-rib>
                  <tables>
                     <afi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">
↪x:ipv4-address-family</afi>
                     <safi xmlns:x="urn:opendaylight:params:xml:ns:yang:bgp-types">
↪x:unicast-subsequent-address-family</safi>
                     <ipv4-routes xmlns="urn:opendaylight:params:xml:ns:yang:bgp-inet
↪">
                        <ipv4-route>
                           <prefix>10.10.10.0/24</prefix>
                           <attributes>
                           ...
                           </attributes>
                        </ipv4-route>
                     </ipv4-routes>
                     <attributes>
```

(continues on next page)

(continued from previous page)

```
46                          <uptodate>true</uptodate>
47                        </attributes>
48                      </tables>
49                    </pre-policy-rib>
50                    <post-policy-rib>
51                      ...
52                    </post-policy-rib>
53                    <stats>
54                      <timestamp-sec>0</timestamp-sec>
55                      <invalidated-cluster-list-loop>0</invalidated-cluster-list-loop>
56                      <duplicate-prefix-advertisements>0</duplicate-prefix-advertisements>
57                      <loc-rib-routes>100</loc-rib-routes>
58                      <duplicate-withdraws>0</duplicate-withdraws>
59                      <invalidated-as-confed-loop>0</invalidated-as-confed-loop>
60                      <adj-ribs-in-routes>10</adj-ribs-in-routes>
61                      <invalidated-as-path-loop>0</invalidated-as-path-loop>
62                      <invalidated-originator-id>0</invalidated-originator-id>
63                      <rejected-prefixes>8</rejected-prefixes>
64                    </stats>
65                  </peer>
66              </router>
67          </monitor>
68      </bmp-monitor>
```

@line 3: **monitor-id** - The BMP monitoring station instance identifier.

@line 5: **router-id** - The monitored router IP address, serves as an identifier.

@line 11: **peer-id** - The monitored peer's BGP identifier, serves a an identifier.

@line 12: **address** - The IP address of the peer, associated with the TCP session.

@line 13: **bgp-id** - The BGP Identifier of the peer.

@line 14: **as** - The Autonomous System number of the peer.

@line 15: **type** - Identifies type of the peer - *Global Instance*, *RD Instance* or *Local Instance*

@line 17: **remote-port** - The peer's port number associated with TCP session.

@line 20: **local-address** - The IP address of the monitored router associated with the peering TCP session.

@line 21: **local-port** - The port number of the monitored router associated with the peering TCP session.

@line 22: **received-open** - The full OPEN message received by monitored router from the peer.

@line 27: **sent-open** - The full OPEN message send by monitored router to the peer.

@line 33: **pre-policy-rib** - The Adj-RIB-In that contains unprocessed routing information.

@line 50: **post-policy-rib** - The Post-Policy Ad-RIB-In that contains routes filtered by inbound policy.

@line 53: **stats** - Contains various statistics, periodically updated by the router.

- **To view collected information from particular monitored router: URL:** `/restconf/operational/bmp-monitor:bmp-monitor/monitor/example-bmp-monitor/router/10.10.10.10`
- **To view collected information from particular monitored peer: URL:** `/restconf/operational/bmp-monitor:bmp-monitor/monitor/example-bmp-monitor/router/10.10.10.10/peer/20.20.20.20`

2.2.5 Test tools

BMP test tool serves to test basic BMP functionality, scalability and performance.

BMP mock

The BMP mock is a stand-alone Java application purposed to simulate a BMP-enabled router(s) and peers. The simulator is capable to report dummy routes and statistics. This application is not part of the OpenDaylight Karaf distribution, however it can be downloaded from OpenDaylight's Nexus (use latest release version):

`https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/bgpcep/bgp-bmp-mock`

Usage

The application can be run from command line:

```
java -jar bgp-bmp-mock-*-executable.jar
```

with optional input parameters:

```
--local_address <address> (optional, default 127.0.0.1)
   The IPv4 address where BMP mock is bind to.

-ra <IP_ADDRESS:PORT,...>, --remote_address <IP_ADDRESS:PORT,...>
   A list of IP addresses of BMP monitoring station, by default 127.0.0.1:12345.

--passive (optional, not present by default)
   This flags enables passive mode for simulated routers.

--routers_count <0..N> (optional, default 1)
    An amount of BMP routers to be connected to the BMP monitoring station.

--peers_count <0..N> (optional, default 0)
   An amount of peers reported by each BMP router.

--pre_policy_routes <0..N> (optional, default 0)
   An amount of "pre-policy" simple IPv4 routes reported by each peer.

--post_policy_routes <0..N> (optional, default 0)
   An amount of "post-policy" simple IPv4 routes reported by each peer.

--log_level <FATAL|ERROR|INFO|DEBUG|TRACE> (optional, default INFO)
   Set logging level for BMP mock.
```

2.2.6 Troubleshooting

This section offers advices in a case OpenDaylight BMP plugin is not working as expected.

Contents

- *BMP is not working...*
- *Bug reporting*

BMP is not working...

- First of all, ensure that all required features are installed, local monitoring station and monitored router/peers configuration is correct.

  To list all installed features in OpenDaylight use the following command at the Karaf console:

  ```
  feature:list -i
  ```

- Check OpenDaylight Karaf logs:

  From Karaf console:

  ```
  log:tail
  ```

  or open log file: `data/log/karaf.log`

  Possibly, a reason/hint for a cause of the problem can be found there.

- Try to minimize effect of other OpenDaylight features, when searching for a reason of the problem.
- Try to set DEBUG severity level for BMP logger via Karaf console commands, in order to collect more information:

  ```
  log:set DEBUG org.opendaylight.protocol.bmp
  ```

Bug reporting

Before you report a bug, check BGPCEP Jira to ensure same/similar bug is not already filed there.

Write an e-mail to bgpcep-users@lists.opendaylight.org and provide following information:

1. State OpenDaylight version
2. Describe your use-case and provide as much details related to BMP as possible
3. Steps to reproduce
4. Attach Karaf log files, optionally packet captures, REST input/output

2.3 PCEP User Guide

This guide contains information on how to use the OpenDaylight Path Computation Element Configuration Protocol (PCEP) plugin. The user should learn about PCEP basic concepts, supported capabilities, configuration and operations.

2.3.1 Overview

This section provides a high-level overview of the PCEP, SDN use-cases and OpenDaylight implementation.

Contents

- *Path Computation Element Communication Protocol*
- *PCEP in SDN*
- *OpenDaylight PCEP plugin*

Path Computation Element Communication Protocol

The Path Computation Element (PCE) Communication Protocol (PCEP) is used for communication between a Path Computation Client (PCC) and a PCE in context of MPLS and GMPLS Traffic Engineering (TE) Label Switched Paths (LSPs). This interaction include path computation requests and computation replies. The PCE operates on a network graph, built from the (Traffic Engineering Database) TED, in order to compute paths based on the path computation request issued by the PCC. The path computation request includes the source and destination of the path and set of constrains to be applied during the computation. The PCE response contains the computed path or the computation failure reason. The PCEP operates on top the TCP, which provides reliable communication.

PCEP in SDN

The Path Computation Element perfectly fits into the centralized SDN controller architecture. The PCE's knowledge of the availability of network resources (i.e. TED) and active LSPs awareness (LSP-DB) allows to perform automated application-driven network operations:

- LSP Re-optimization
- Resource defragmentation
- Link failure restoration
- Auto-bandwidth adjustment
- Bandwidth scheduling
- Shared Risk Link Group (SRLG) diversity maintenance

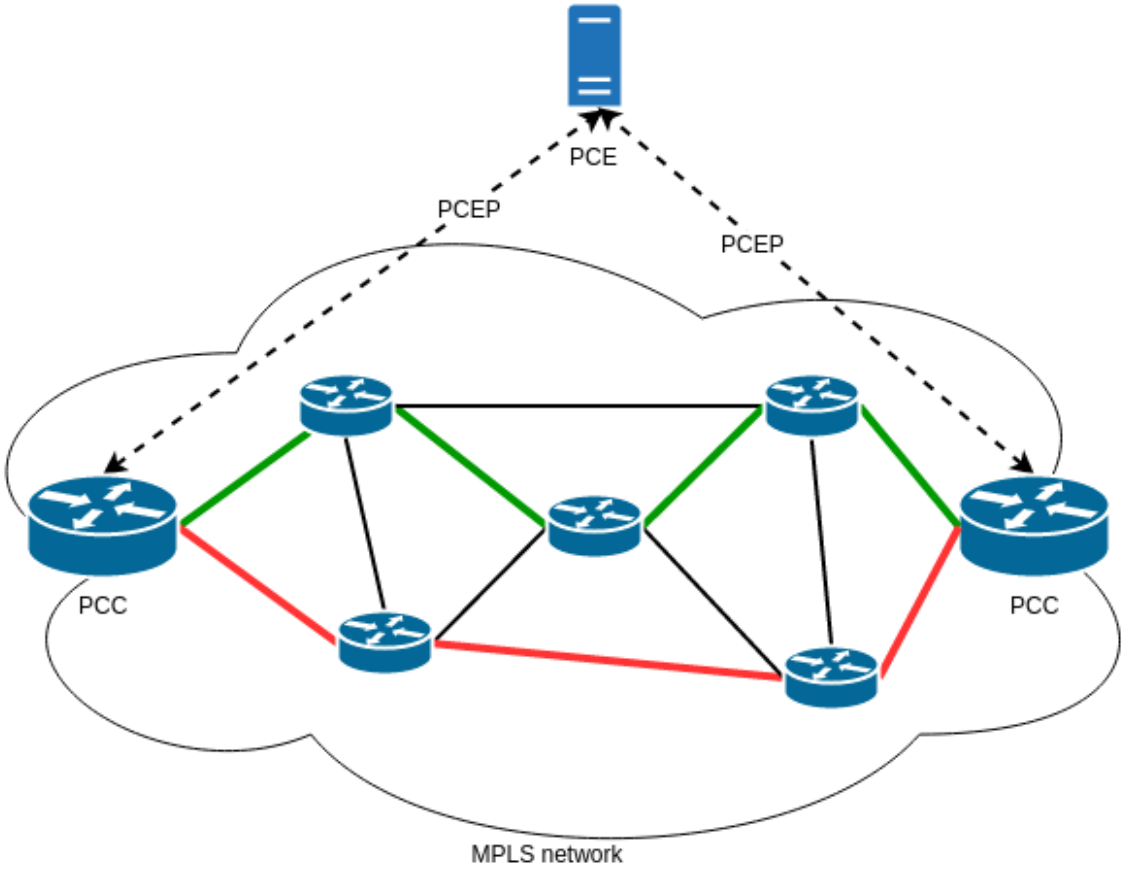


Fig. 8: PCE-based architecture.

OpenDaylight PCEP plugin

The OpenDaylight PCEP plugin provides all basic service units necessary to build-up a PCE-based controller. In addition, it offers LSP management functionality for Active Stateful PCE - the cornerstone for majority of PCE-enabled SDN solutions. It consists of the following components:

- Protocol library
- PCEP session handling
- Stateful PCE LSP-DB
- Active Stateful PCE LSP Operations

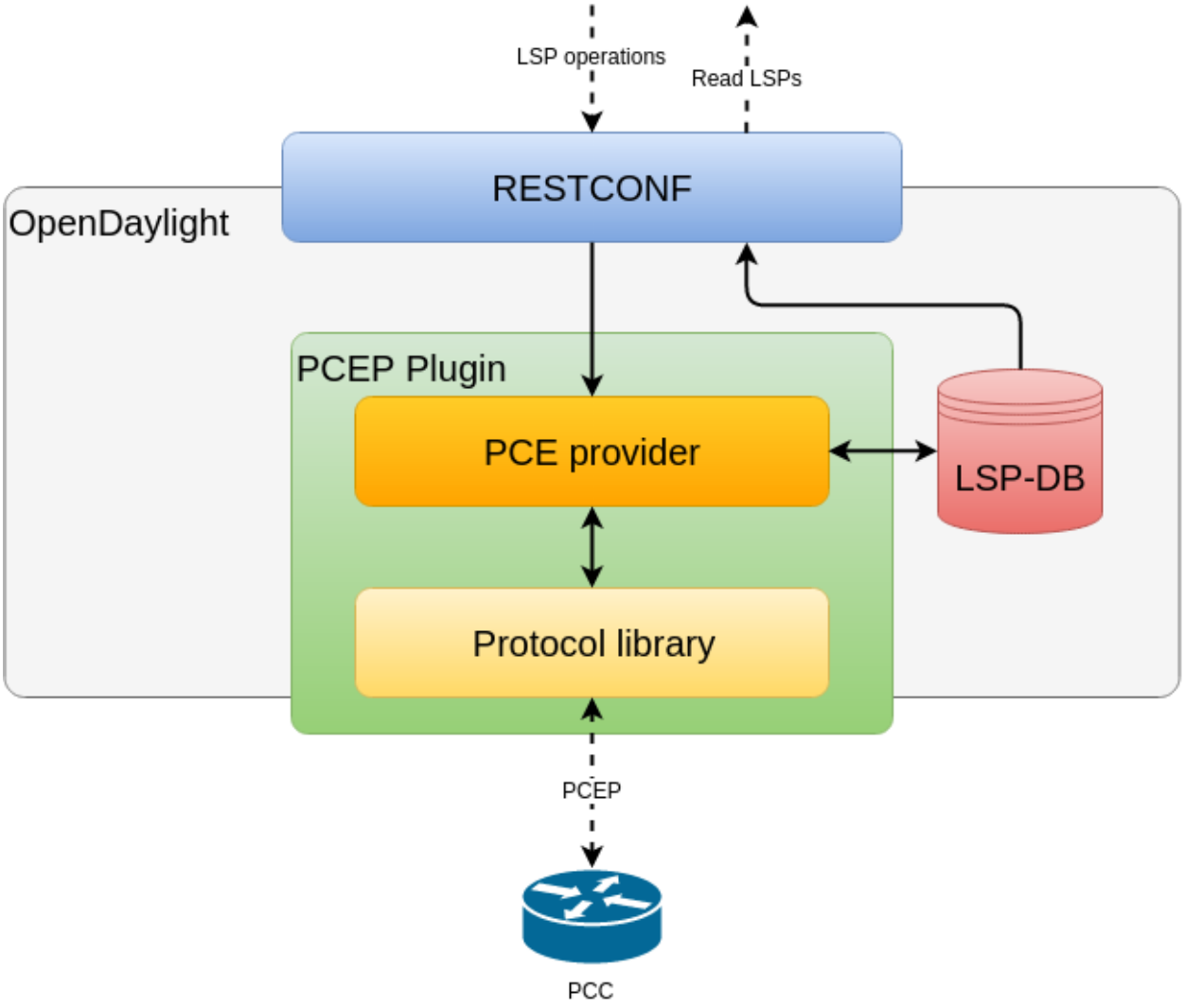


Fig. 9: OpenDaylight PCEP plugin overview.

Important: The PCEP plugin does not provide path computational functionality and does not build TED.

2.3.2 List of supported capabilities

- RFC5440 - Path Computation Element (PCE) Communication Protocol (PCEP)
- RFC5455 - Diffserv-Aware Class-Type Object for the Path Computation Element Communication Protocol
- RFC5520 - Preserving Topology Confidentiality in Inter-Domain Path Computation Using a Path-Key-Based Mechanism
- RFC5521 - Extensions to the Path Computation Element Communication Protocol (PCEP) for Route Exclusions
- RFC5541 - Encoding of Objective Functions in the Path Computation Element Communication Protocol (PCEP)
- RFC5557 - Path Computation Element Communication Protocol (PCEP) Requirements and Protocol Extensions in Support of Global Concurrent Optimization
- RFC5886 - A Set of Monitoring Tools for Path Computation Element (PCE)-Based Architecture
- RFC7470 - Conveying Vendor-Specific Constraints in the Path Computation Element Communication Protocol
- RFC7896 - Update to the Include Route Object (IRO) Specification in the Path Computation Element Communication Protocol (PCEP)
- **draft-ietf-pce-stateful-pce - PCEP Extensions for Stateful PCE**
  - draft-ietf-pce-pce-initiated-lsp - PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model
  - draft-ietf-pce-segment-routing - PCEP Extension for segment routing
  - draft-ietf-pce-lsp-setup-type - PCEP Extension for path setup type
  - draft-ietf-pce-stateful-sync-optimizations - Optimizations of Label Switched Path State Synchronization Procedures for a Stateful PCE
  - draft-sivabalan-pce-binding-label-sid - Carrying Binding Label/Segment-ID in PCE-based Networks
- draft-ietf-pce-pceps - Secure Transport for PCEP
- RFC8306 - Extensions to the Path Computation Element Communication Protocol (PCEP) for Point-to-Multipoint Traffic Engineering Label Switched Paths

2.3.3 Running PCEP

This section explains how to install PCEP plugin.

1. Install PCEP feature - `odl-bgpcep-pcep`. Also, for sake of this sample, it is required to install RESTCONF. In the Karaf console, type command:

   ```
   feature:install odl-restconf odl-bgpcep-pcep
   ```

2. The PCEP plugin contains a default configuration, which is applied after the feature starts up. One instance of PCEP plugin is created (named *pcep-topology*), and its presence can be verified via REST:

   URL: `restconf/operational/network-topology:network-topology/topology/pcep-topology`

   Method: `GET`

   Response Body:

```
<topology xmlns="urn:TBD:params:xml:ns:yang:network-topology">
    <topology-id>pcep-topology</topology-id>
    <topology-types>
        <topology-pcep xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
↪</topology-pcep>
    </topology-types>
</topology>
```

2.3.4 Active Stateful PCE

The PCEP extension for Stateful PCE brings a visibility of active LSPs to PCE, in order to optimize path computation, while considering individual LSPs and their interactions. This requires state synchronization mechanism between PCE and PCC. Moreover, Active Stateful PCE is capable to address LSP parameter changes to the PCC.

Contents

- *Configuration*
  - *Speaker Entity identifier*
  - *MD5 authentication configuration*
- *LSP State Database*
  - *LSP-DB API*
  - *LSP Delegation*
  - *LSP Update*
- *PCE-initiated LSP Setup*
  - *Configuration*
  - *LSP Instantiation*
  - *LSP Deletion*
  - *PCE-initiated LSP Delegation*
- *Segment Routing*
  - *Configuration*
  - *IANA code points*
  - *LSP Operations for PCEP SR*
- *LSP State Synchronization Optimization Procedures*
  - *Configuration*
  - *State Synchronization Avoidance*
  - *Incremental State Synchronization*
  - *PCE-triggered Initial Synchronization*
  - *PCE-triggered Re-synchronization*

Configuration

This capability is enabled by default. No additional configuration is required.

Speaker Entity identifier

The Speaker Entity Identifier is an optional TLV that may be included in the OPEN Object when a PCEP speaker wishes to determine if state synchronization can be skipped when a PCEP session is restarted.

URL: `/restconf/config/network-topology:network-topology/topology/pcep-topology/node/43.43.43.43`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1 <node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
2     <node-id>43.43.43.43</node-id>
3     <session-config xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep:config">
4         <speaker-entity-id-value xmlns=
↪"urn:opendaylight:params:xml:ns:yang:topology:pcep:sync:optimizations:config">
↪AQIDBA==</speaker-entity-id-value>
5     </session-config>
6 </node>
```

@line 2: **address** - A PCC IP address.

@line 4: **Speaker Entity Identifier** - The Speaker Entity identifier assigned to PCEP Node.

MD5 authentication configuration

The OpenDaylight PCEP implementation supports TCP MD5 for authentication. The sample configuration below shows how to set authentication password for a particular PCC.

URL: `/restconf/config/network-topology:network-topology/topology/pcep-topology/node/43.43.43.43`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1 <node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
2     <node-id>43.43.43.43</node-id>
3     <session-config xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep:config">
4         <password>topsecret</password>
5     </session-config>
6 </node>
```

@line 2: **address** - A PCC IP address.

@line 4: **password** - MD5 authentication phrase.

LSP State Database

The *LSP State Database* (LSP-DB) contains an information about all LSPs and their attributes. The LSP state is synchronized between the PCC and PCE. First, initial LSP state synchronization is performed once the session between PCC and PCE is established in order to learn PCC's LPSs. This step is a prerequisite to following LSPs manipulation operations.

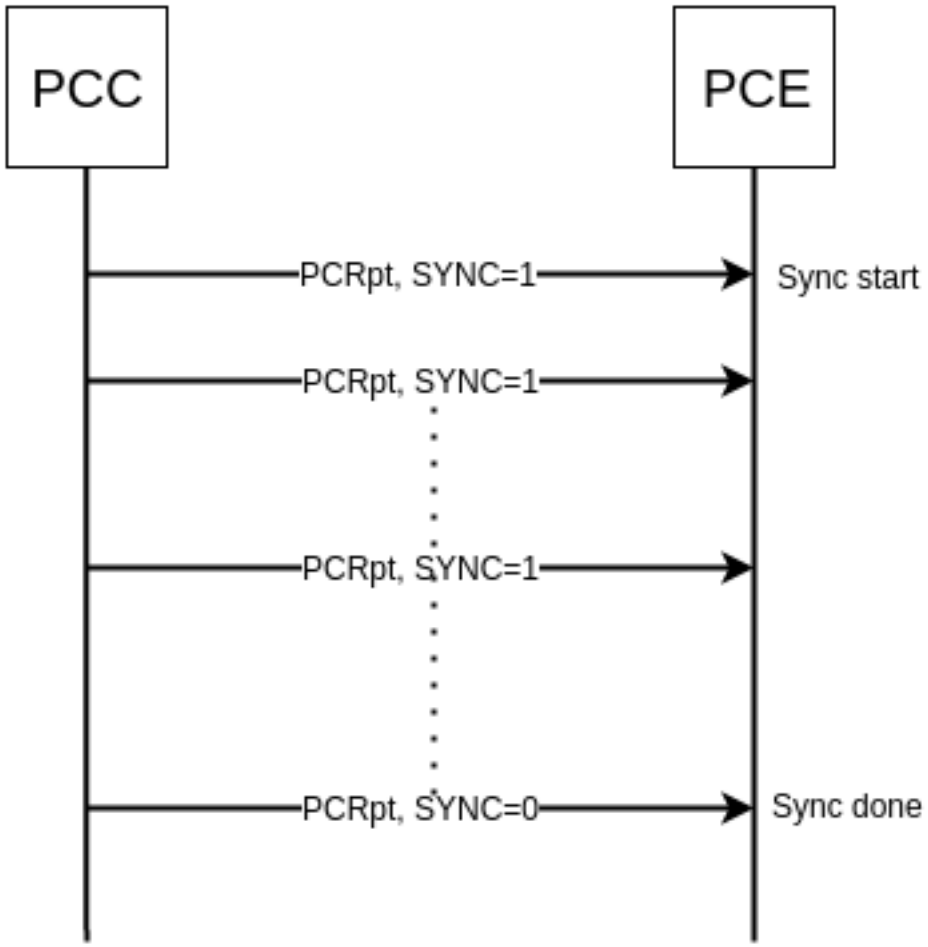


Fig. 10: LSP State Synchronization.

LSP-DB API

```
path-computation-client
   +--ro reported-lsp* [name]
      +--ro name         string
      +--ro path* [lsp-id]
      |  +--ro lsp-id                      rsvp:lsp-id
      |  +--ro ero
      |  |  +--ro processing-rule?   boolean
      |  |  +--ro ignore?            boolean
      |  |  +--ro subobject*
      |  |     +--ro loose         boolean
      |  |     +--ro (subobject-type)?
```

(continues on next page)

(continued from previous page)

```
      |  |        +--:(as-number-case)
      |  |        |  +--ro as-number
      |  |        |     +--ro as-number    inet:as-number
      |  |        +--:(ip-prefix-case)
      |  |        |  +--ro ip-prefix
      |  |        |     +--ro ip-prefix    inet:ip-prefix
      |  |        +--:(label-case)
      |  |        |  +--ro label
      |  |        |     +--ro uni-directional             boolean
      |  |        |     +--ro (label-type)?
      |  |        |        +--:(type1-label-case)
      |  |        |        |  +--ro type1-label
      |  |        |        |     +--ro type1-label    uint32
      |  |        |        +--:(generalized-label-case)
      |  |        |        |  +--ro generalized-label
      |  |        |        |     +--ro generalized-label    binary
      |  |        |        +--:(waveband-switching-label-case)
      |  |        |           +--ro waveband-switching-label
      |  |        |              +--ro end-label      uint32
      |  |        |              +--ro start-label    uint32
      |  |        |              +--ro waveband-id    uint32
      |  |        +--:(srlg-case)
      |  |        |  +--ro srlg
      |  |        |     +--ro srlg-id    srlg-id
      |  |        +--:(unnumbered-case)
      |  |        |  +--ro unnumbered
      |  |        |     +--ro router-id       uint32
      |  |        |     +--ro interface-id    uint32
      |  |        +--:(exrs-case)
      |  |        |  +--ro exrs
      |  |        |     +--ro exrs*
      |  |        |        +--ro mandatory?    boolean
      |  |        |        +--ro attribute     enumeration
      |  |        |        +--ro (subobject-type)?
      |  |        |           +--:(as-number-case)
      |  |        |           |  +--ro as-number
      |  |        |           |     +--ro as-number    inet:as-number
      |  |        |           +--:(ip-prefix-case)
      |  |        |           |  +--ro ip-prefix
      |  |        |           |     +--ro ip-prefix    inet:ip-prefix
      |  |        |           +--:(label-case)
      |  |        |           |  +--ro label
      |  |        |           |     +--ro uni-directional             boolean
      |  |        |           |     +--ro (label-type)?
      |  |        |           |        +--:(type1-label-case)
      |  |        |           |        |  +--ro type1-label
      |  |        |           |        |     +--ro type1-label    uint32
      |  |        |           |        +--:(generalized-label-case)
      |  |        |           |        |  +--ro generalized-label
      |  |        |           |        |     +--ro generalized-label    binary
      |  |        |           |        +--:(waveband-switching-label-case)
      |  |        |           |           +--ro waveband-switching-label
      |  |        |           |              +--ro end-label      uint32
      |  |        |           |              +--ro start-label    uint32
      |  |        |           |              +--ro waveband-id    uint32
      |  |        |           +--:(srlg-case)
      |  |        |           |  +--ro srlg
```

(continues on next page)

(continued from previous page)

```
|  |        |           |     +--ro srlg-id    srlg-id
|  |        |           +--:(unnumbered-case)
|  |        |              +--ro unnumbered
|  |        |                 +--ro router-id       uint32
|  |        |                 +--ro interface-id    uint32
|  |        +--:(path-key-case)
|  |           +--ro path-key
|  |              +--ro pce-id      pce-id
|  |              +--ro path-key    path-key
|  +--ro lspa
|  |  +--ro processing-rule?           boolean
|  |  +--ro ignore?                    boolean
|  |  +--ro hold-priority?             uint8
|  |  +--ro setup-priority?            uint8
|  |  +--ro local-protection-desired?  boolean
|  |  +--ro label-recording-desired?   boolean
|  |  +--ro se-style-desired?          boolean
|  |  +--ro session-name?              string
|  |  +--ro include-any?               attribute-filter
|  |  +--ro exclude-any?               attribute-filter
|  |  +--ro include-all?               attribute-filter
|  |  +--ro tlvs
|  |     +--ro vendor-information-tlv*
|  |        +--ro enterprise-number?   iana:enterprise-number
|  |        +--ro (enterprise-specific-information)?
|  +--ro bandwidth
|  |  +--ro processing-rule?   boolean
|  |  +--ro ignore?            boolean
|  |  +--ro bandwidth?         netc:bandwidth
|  +--ro reoptimization-bandwidth
|  |  +--ro processing-rule?   boolean
|  |  +--ro ignore?            boolean
|  |  +--ro bandwidth?         netc:bandwidth
|  +--ro metrics*
|  |  +--ro metric
|  |     +--ro processing-rule?   boolean
|  |     +--ro ignore?            boolean
|  |     +--ro metric-type        uint8
|  |     +--ro bound?             boolean
|  |     +--ro computed?          boolean
|  |     +--ro value?             ieee754:float32
|  +--ro iro
|  |  +--ro processing-rule?   boolean
|  |  +--ro ignore?            boolean
|  |  +--ro subobject*
|  |     +--ro loose         boolean
|  |     +--ro (subobject-type)?
|  |        +--:(as-number-case)
|  |        |  +--ro as-number
|  |        |     +--ro as-number    inet:as-number
|  |        +--:(ip-prefix-case)
|  |        |  +--ro ip-prefix
|  |        |     +--ro ip-prefix    inet:ip-prefix
|  |        +--:(label-case)
|  |        |  +--ro label
|  |        |     +--ro uni-directional            boolean
|  |        |     +--ro (label-type)?
```

(continues on next page)

(continued from previous page)

```
      |  |        |        +--:(type1-label-case)
      |  |        |        |  +--ro type1-label
      |  |        |        |     +--ro type1-label    uint32
      |  |        |        +--:(generalized-label-case)
      |  |        |        |  +--ro generalized-label
      |  |        |        |     +--ro generalized-label    binary
      |  |        |        +--:(waveband-switching-label-case)
      |  |        |           +--ro waveband-switching-label
      |  |        |              +--ro end-label      uint32
      |  |        |              +--ro start-label    uint32
      |  |        |              +--ro waveband-id    uint32
      |  |        +--:(srlg-case)
      |  |        |  +--ro srlg
      |  |        |     +--ro srlg-id    srlg-id
      |  |        +--:(unnumbered-case)
      |  |        |  +--ro unnumbered
      |  |        |     +--ro router-id       uint32
      |  |        |     +--ro interface-id    uint32
      |  |        +--:(exrs-case)
      |  |        |  +--ro exrs
      |  |        |     +--ro exrs*
      |  |        |        +--ro mandatory?    boolean
      |  |        |        +--ro attribute     enumeration
      |  |        |        +--ro (subobject-type)?
      |  |        |           +--:(as-number-case)
      |  |        |           |  +--ro as-number
      |  |        |           |     +--ro as-number    inet:as-number
      |  |        |           +--:(ip-prefix-case)
      |  |        |           |  +--ro ip-prefix
      |  |        |           |     +--ro ip-prefix    inet:ip-prefix
      |  |        |           +--:(label-case)
      |  |        |           |  +--ro label
      |  |        |           |     +--ro uni-directional             boolean
      |  |        |           |     +--ro (label-type)?
      |  |        |           |        +--:(type1-label-case)
      |  |        |           |        |  +--ro type1-label
      |  |        |           |        |     +--ro type1-label    uint32
      |  |        |           |        +--:(generalized-label-case)
      |  |        |           |        |  +--ro generalized-label
      |  |        |           |        |     +--ro generalized-label    binary
      |  |        |           |        +--:(waveband-switching-label-case)
      |  |        |           |           +--ro waveband-switching-label
      |  |        |           |              +--ro end-label      uint32
      |  |        |           |              +--ro start-label    uint32
      |  |        |           |              +--ro waveband-id    uint32
      |  |        |           +--:(srlg-case)
      |  |        |           |  +--ro srlg
      |  |        |           |     +--ro srlg-id    srlg-id
      |  |        |           +--:(unnumbered-case)
      |  |        |              +--ro unnumbered
      |  |        |                 +--ro router-id       uint32
      |  |        |                 +--ro interface-id    uint32
      |  |        +--:(path-key-case)
      |  |           +--ro path-key
      |  |              +--ro pce-id      pce-id
      |  |              +--ro path-key    path-key
      |  +--ro rro
```

(continues on next page)

(continued from previous page)

```
      |  |  +--ro processing-rule?   boolean
      |  |  +--ro ignore?            boolean
      |  |  +--ro subobject*
      |  |     +--ro protection-available?   boolean
      |  |     +--ro protection-in-use?      boolean
      |  |     +--ro (subobject-type)?
      |  |        +--:(ip-prefix-case)
      |  |        |  +--ro ip-prefix
      |  |        |     +--ro ip-prefix    inet:ip-prefix
      |  |        +--:(label-case)
      |  |        |  +--ro label
      |  |        |     +--ro uni-directional             boolean
      |  |        |     +--ro (label-type)?
      |  |        |     |  +--:(type1-label-case)
      |  |        |     |  |  +--ro type1-label
      |  |        |     |  |     +--ro type1-label    uint32
      |  |        |     |  +--:(generalized-label-case)
      |  |        |     |  |  +--ro generalized-label
      |  |        |     |  |     +--ro generalized-label    binary
      |  |        |     |  +--:(waveband-switching-label-case)
      |  |        |     |     +--ro waveband-switching-label
      |  |        |     |        +--ro end-label      uint32
      |  |        |     |        +--ro start-label    uint32
      |  |        |     |        +--ro waveband-id    uint32
      |  |        |     +--ro global?                       boolean
      |  |        +--:(unnumbered-case)
      |  |        |  +--ro unnumbered
      |  |        |     +--ro router-id       uint32
      |  |        |     +--ro interface-id    uint32
      |  |        +--:(path-key-case)
      |  |           +--ro path-key
      |  |              +--ro pce-id      pce-id
      |  |              +--ro path-key    path-key
      |  +--ro xro
      |  |  +--ro processing-rule?   boolean
      |  |  +--ro ignore?            boolean
      |  |  +--ro flags              bits
      |  |  +--ro subobject*
      |  |     +--ro mandatory?    boolean
      |  |     +--ro attribute     enumeration
      |  |     +--ro (subobject-type)?
      |  |        +--:(as-number-case)
      |  |        |  +--ro as-number
      |  |        |     +--ro as-number    inet:as-number
      |  |        +--:(ip-prefix-case)
      |  |        |  +--ro ip-prefix
      |  |        |     +--ro ip-prefix    inet:ip-prefix
      |  |        +--:(label-case)
      |  |        |  +--ro label
      |  |        |     +--ro uni-directional             boolean
      |  |        |     +--ro (label-type)?
      |  |        |        +--:(type1-label-case)
      |  |        |        |  +--ro type1-label
      |  |        |        |     +--ro type1-label    uint32
      |  |        |        +--:(generalized-label-case)
      |  |        |        |  +--ro generalized-label
      |  |        |        |     +--ro generalized-label    binary
```

(continues on next page)

(continued from previous page)

```
      |  |        |        +--:(waveband-switching-label-case)
      |  |        |           +--ro waveband-switching-label
      |  |        |              +--ro end-label      uint32
      |  |        |              +--ro start-label    uint32
      |  |        |              +--ro waveband-id    uint32
      |  |        +--:(srlg-case)
      |  |        |  +--ro srlg
      |  |        |     +--ro srlg-id    srlg-id
      |  |        +--:(unnumbered-case)
      |  |           +--ro unnumbered
      |  |              +--ro router-id       uint32
      |  |              +--ro interface-id    uint32
      |  +--ro of
      |  |  +--ro processing-rule?   boolean
      |  |  +--ro ignore?            boolean
      |  |  +--ro code               of-id
      |  |  +--ro tlvs
      |  |     +--ro vendor-information-tlv*
      |  |        +--ro enterprise-number?   iana:enterprise-number
      |  |        +--ro (enterprise-specific-information)?
      |  +--ro class-type
      |     +--ro processing-rule?   boolean
      |     +--ro ignore?            boolean
      |     +--ro class-type         class-type
      +--ro metadata
      +--ro lsp
      |  +--ro processing-rule?   boolean
      |  +--ro ignore?            boolean
      |  +--ro tlvs
      |  |  +--ro lsp-error-code
      |  |  |  +--ro error-code?   uint32
      |  |  +--ro lsp-identifiers
      |  |  |  +--ro lsp-id?      rsvp:lsp-id
      |  |  |  +--ro tunnel-id?   rsvp:tunnel-id
      |  |  |  +--ro (address-family)?
      |  |  |     +--:(ipv4-case)
      |  |  |     |  +--ro ipv4
      |  |  |     |     +--ro ipv4-tunnel-sender-address      inet:ipv4-address
      |  |  |     |     +--ro ipv4-extended-tunnel-id         rsvp:ipv4-extended-
↪tunnel-id
      |  |  |     |     +--ro ipv4-tunnel-endpoint-address    inet:ipv4-address
      |  |  |     +--:(ipv6-case)
      |  |  |        +--ro ipv6
      |  |  |           +--ro ipv6-tunnel-sender-address      inet:ipv6-address
      |  |  |           +--ro ipv6-extended-tunnel-id         rsvp:ipv6-extended-
↪tunnel-id
      |  |  |           +--ro ipv6-tunnel-endpoint-address    inet:ipv6-address
      |  |  +--ro rsvp-error-spec
      |  |  |  +--ro (error-type)?
      |  |  |     +--:(rsvp-case)
      |  |  |     |  +--ro rsvp-error
      |  |  |     +--:(user-case)
      |  |  |        +--ro user-error
      |  |  +--ro symbolic-path-name
      |  |  |  +--ro path-name?   symbolic-path-name
      |  |  o--ro vs-tlv
      |  |  |  +--ro enterprise-number?   iana:enterprise-number
```

(continues on next page)

(continued from previous page)

```
      |  |  |  +--ro (vendor-payload)?
      |  |  +--ro vendor-information-tlv*
      |  |  |  +--ro enterprise-number?   iana:enterprise-number
      |  |  |  +--ro (enterprise-specific-information)?
      |  |  +--ro path-binding
      |  |     x--ro binding-type?      uint8
      |  |     x--ro binding-value?     binary
      |  |     +--ro (binding-type-value)?
      |  |        +--:(mpls-label)
      |  |        |  +--ro mpls-label?        netc:mpls-label
      |  |        +--:(mpls-label-entry)
      |  |           +--ro label?             netc:mpls-label
      |  |           +--ro traffic-class?     uint8
      |  |           +--ro bottom-of-stack?   boolean
      |  |           +--ro time-to-live?      uint8
      |  +--ro plsp-id?           plsp-id
      |  +--ro delegate?          boolean
      |  +--ro sync?              boolean
      |  +--ro remove?            boolean
      |  +--ro administrative?    boolean
      |  +--ro operational?       operational-status
      +--ro path-setup-type
         +--ro pst?   uint8
```

The LSP-DB is accessible via RESTCONF. The PCC's LSPs are stored in the `pcep-topology` while the session is active. In a next example, there is one PCEP session with PCC identified by its IP address (*43.43.43.43*) and one reported LSP (*foo*).

URL: `/restconf/operational/network-topology:network-topology/topology/pcep-topology/node/pcc:%2F%2F43.43.43.43`

Method: `GET`

Response Body:

```
<node>
   <node-id>pcc://43.43.43.43</node-id>
   <path-computation-client>
      <ip-address>43.43.43.43</ip-address>
      <state-sync>synchronized</state-sync>
      <stateful-tlv>
         <stateful>
            <lsp-update-capability>true</lsp-update-capability>
         </stateful>
      </stateful-tlv>
      <reported-lsp>
         <name>foo</name>
         <lsp>
            <operational>up</operational>
            <sync>true</sync>
            <plsp-id>1</plsp-id>
            <create>false</create>
            <administrative>true</administrative>
            <remove>false</remove>
            <delegate>true</delegate>
            <tlvs>
```

(continues on next page)

(continued from previous page)

```
22                <lsp-identifiers>
23                   <ipv4>
24                      <ipv4-tunnel-sender-address>43.43.43.43</ipv4-tunnel-sender-
↪address>
25                      <ipv4-tunnel-endpoint-address>39.39.39.39</ipv4-tunnel-endpoint-
↪address>
26                      <ipv4-extended-tunnel-id>39.39.39.39</ipv4-extended-tunnel-id>
27                   </ipv4>
28                   <tunnel-id>1</tunnel-id>
29                   <lsp-id>1</lsp-id>
30                </lsp-identifiers>
31                <symbolic-path-name>
32                   <path-name>Zm9v</path-name>
33                </symbolic-path-name>
34             </tlvs>
35          </lsp>
36          <ero>
37             <subobject>
38                <loose>false</loose>
39                <ip-prefix>
40                   <ip-prefix>201.20.160.40/32</ip-prefix>
41                </ip-prefix>
42             </subobject>
43             <subobject>
44                <loose>false</loose>
45                <ip-prefix>
46                   <ip-prefix>195.20.160.39/32</ip-prefix>
47                </ip-prefix>
48             </subobject>
49             <subobject>
50                <loose>false</loose>
51                <ip-prefix>
52                   <ip-prefix>39.39.39.39/32</ip-prefix>
53                </ip-prefix>
54             </subobject>
55          </ero>
56       </reported-lsp>
57    </path-computation-client>
58 </node>
```

@line 2: **node-id** The PCC identifier.

@line 4: **ip-address** IP address of the PCC.

@line 5: **state-sync** Synchronization status of the PCC's LSPs. The *synchronized* indicates the State Synchronization is done.

@line 8: **lsp-update-capability** - Indicates that PCC allows LSP modifications.

@line 12: **name** - Textual representation of LPS's name.

@line 14: **operational** - Represent operational status of the LSP:

- *down* - not active.
- *up* - signaled.
- *active* - up and carrying traffic.
- *going-down* - LSP is being torn down, resources are being released.

- *going-up* - LSP is being signaled.

@line 15: **sync** - The flag set by PCC during LSPs State Synchronization.

@line 16: **plsp-id** - A PCEP-specific identifier for the LSP. It is assigned by PCC and it is constant for a lifetime of a PCEP session.

@line 17: **create** - The *false* indicates that LSP is PCC-initiated.

@line 18: **administrative** - The flag indicates target operational status of the LSP.

@line 20: **delegate** - The delegate flag indicates that the PCC is delegating the LSP to the PCE.

@line 24: **ipv4-tunnel-sender-address** - Contains the sender node's IP address.

@line 25: **ipv4-tunnel-endpoint-address** - Contains the egress node's IP address.

@line 26: **ipv4-extended-tunnel-id** - The *Extended Tunnel ID* identifier.

@line 28: **tunnel-id** - The *Tunnel ID* identifier.

@line 29: **lsp-id** - The *LSP ID* identifier.

@line 32: **path-name** - The symbolic name for the LSP.

@line 36: **ero** - The *Explicit Route Object* is encoding the path of the TE LSP through the network.

LSP Delegation

The LSP control delegations is an mechanism, where PCC grants to a PCE the temporary right in order to modify LSP attributes. The PCC can revoke the delegation or the PCE may waive the delegation at any time. The LSP control is delegated to at most one PCE at the same time.

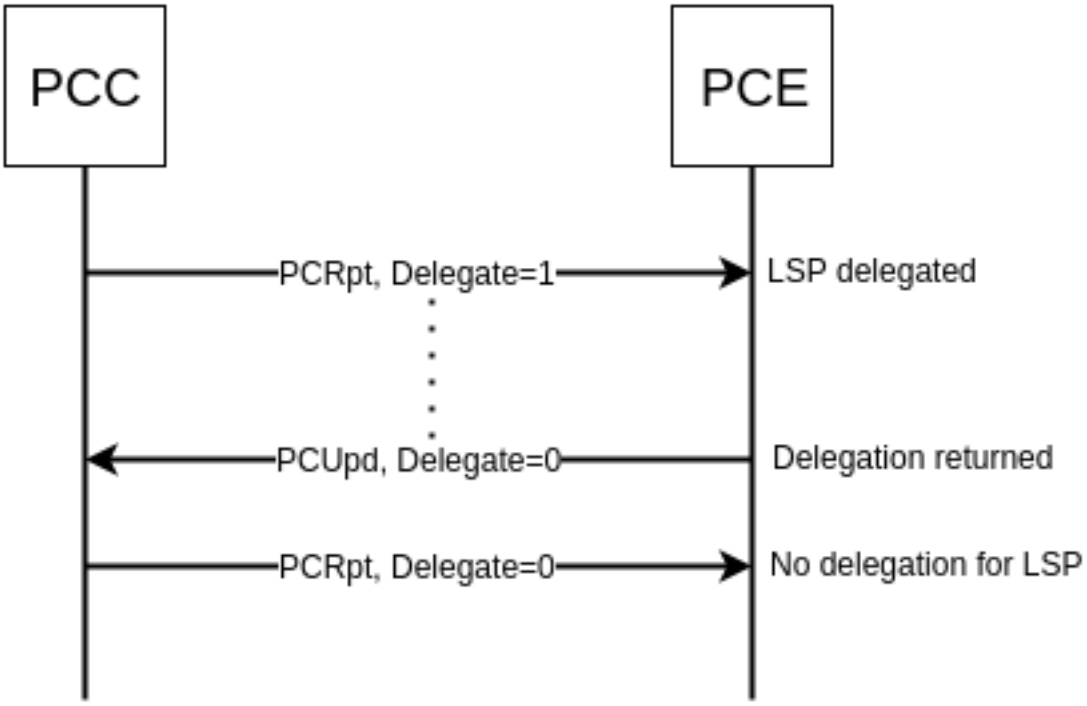


Fig. 11: Returning a Delegation.

Following RPC example illustrates a request for the LSP delegation give up:

URL: `/restconf/operations/network-topology-pcep:update-lsp`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input>
   <node>pcc://43.43.43.43</node>
   <name>foo</name>
   <arguments>
      <lsp xmlns:stateful="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
         <delegate>false</delegate>
         <administrative>true</administrative>
         <tlvs>
            <symbolic-path-name>
               <path-name>Zm9v</path-name>
            </symbolic-path-name>
         </tlvs>
      </lsp>
   </arguments>
   <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
↪topology-ref>
</input>
```

@line 2: **node** The PCC identifier.

@line 3: **name** The name of the LSP.

@line 6: **delegate** - Delegation flag set *false* in order to return the LSP delegation.

@line 10: **path-name** - The Symbolic Path Name TLV must be present when sending a request to give up the delegation.

LSP Update

The LSP Update Request is an operation where a PCE requests a PCC to update attributes of an LSP and to rebuild the LSP with updated attributes. In order to update LSP, the PCE must hold a LSP delegation. The LSP update is done in *make-before-break* fashion - first, new LSP is initiated and then the old LSP is torn down.

Following RPC example shows a request for the LSP update:

URL: `/restconf/operations/network-topology-pcep:update-lsp`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
   <node>pcc://43.43.43.43</node>
   <name>foo</name>
   <arguments>
      <lsp xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
```

(continues on next page)

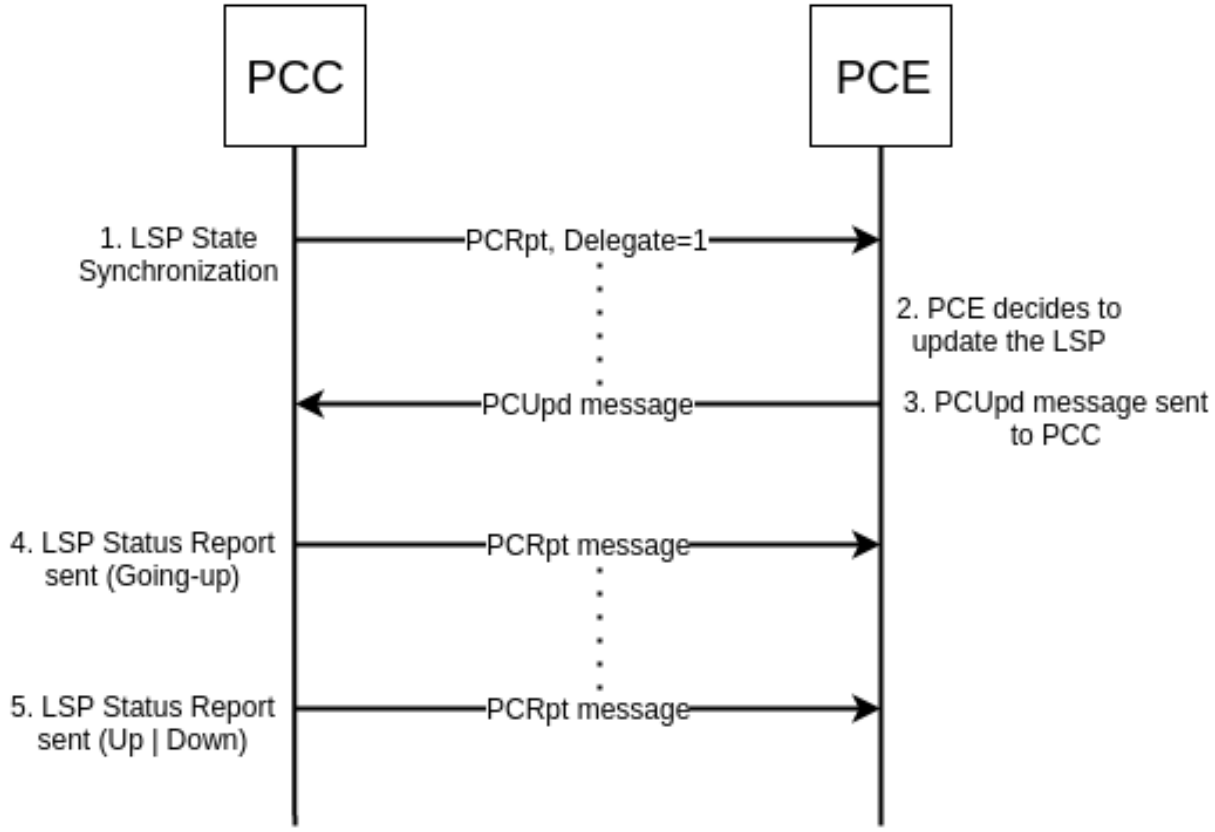


Fig. 12: Active Stateful PCE LSP Update.

(continued from previous page)

```
6           <delegate>true</delegate>
7           <administrative>true</administrative>
8        </lsp>
9        <ero>
10          <subobject>
11             <loose>false</loose>
12             <ip-prefix>
13                <ip-prefix>200.20.160.41/32</ip-prefix>
14             </ip-prefix>
15          </subobject>
16          <subobject>
17             <loose>false</loose>
18             <ip-prefix>
19                <ip-prefix>196.20.160.39/32</ip-prefix>
20             </ip-prefix>
21          </subobject>
22          <subobject>
23             <loose>false</loose>
24             <ip-prefix>
25                <ip-prefix>39.39.39.39/32</ip-prefix>
26             </ip-prefix>
27          </subobject>
28       </ero>
29    </arguments>
30    <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
 ↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
 ↪topology-ref>
31 </input>
```

@line 2: **node** The PCC identifier.

@line 3: **name** The name of the LSP to be updated.

@line 6: **delegate** - Delegation flag set *true* in order to keep the LSP control.

@line 7: **administrative** - Desired administrative status of the LSP is active.

@line 9: **ero** - This LSP attribute is changed.

PCE-initiated LSP Setup

The PCEP Extension for PCE-initiated LSP Setup allows PCE to request a creation and deletion of LSPs.

Configuration

This capability is enabled by default. No additional configuration is required.

LSP Instantiation

The PCE can request LSP creation. The LSP instantiation is done by sending an LSP Initiate Message to PCC. The PCC assign delegation to PCE which triggered creation. PCE-initiated LSPs are identified by *Create* flag.

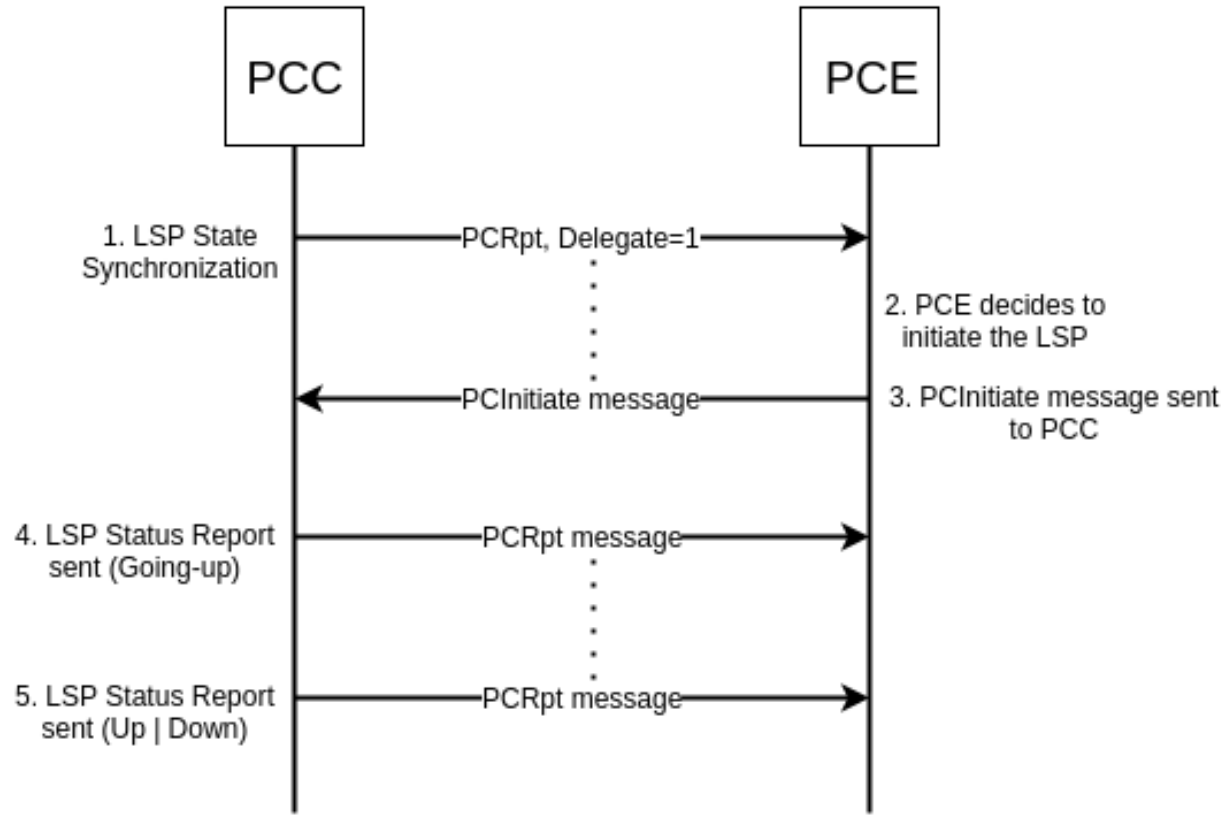


Fig. 13: LSP instantiation.

Following RPC example shows a request for the LSP initiation:

URL: `/restconf/operations/network-topology-pcep:add-lsp`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
   <node>pcc://43.43.43.43</node>
   <name>update-tunel</name>
      <arguments>
         <lsp xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
            <delegate>true</delegate>
            <administrative>true</administrative>
         </lsp>
         <endpoints-obj>
            <ipv4>
               <source-ipv4-address>43.43.43.43</source-ipv4-address>
```

(continues on next page)

(continued from previous page)

```
                <destination-ipv4-address>39.39.39.39</destination-ipv4-address>
            </ipv4>
        </endpoints-obj>
        <ero>
            <subobject>
                <loose>false</loose>
                <ip-prefix>
                    <ip-prefix>201.20.160.40/32</ip-prefix>
                </ip-prefix>
            </subobject>
            <subobject>
                <loose>false</loose>
                <ip-prefix>
                    <ip-prefix>195.20.160.39/32</ip-prefix>
                </ip-prefix>
            </subobject>
            <subobject>
                <loose>false</loose>
                <ip-prefix>
                    <ip-prefix>39.39.39.39/32</ip-prefix>
                </ip-prefix>
            </subobject>
        </ero>
    </arguments>
  <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
↪topology-ref>
</input>
```

@line 2: **node** The PCC identifier.

@line 3: **name** The name of the LSP to be created.

@line 8: **endpoints-obj** - The *END-POINT* Object is mandatory for an instantiation request of an RSVP-signaled LSP. It contains source and destination addresses for provisioning the LSP.

@line 14: **ero** - The *ERO* object is mandatory for LSP initiation request.

LSP Deletion

The PCE may request a deletion of PCE-initiated LSPs. The PCE must be delegation holder for this particular LSP.

Following RPC example shows a request for the LSP deletion:

URL: `/restconf/operations/network-topology-pcep:remove-lsp`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
    <node>pcc://43.43.43.43</node>
    <name>update-tunel</name>
```

(continues on next page)

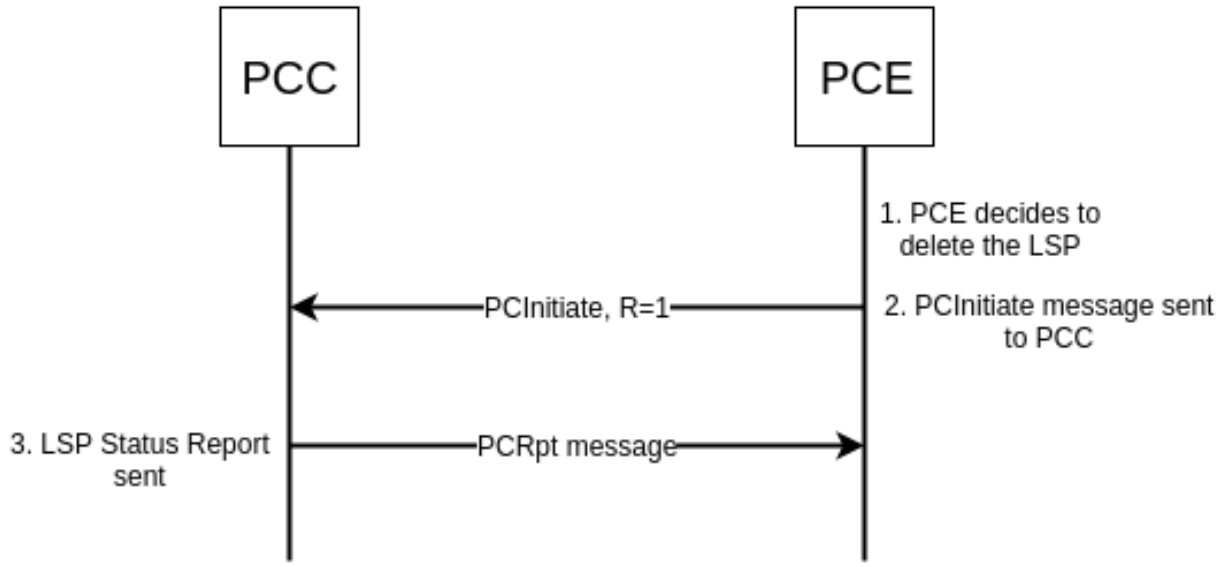


Fig. 14: LSP deletion.

(continued from previous page)

```
4     <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
  ↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
  ↪topology-ref>
5  </input>
```

@line 2: **node** The PCC identifier.

@line 3: **name** The name of the LSP to be removed.

PCE-initiated LSP Delegation

The PCE-initiated LSP control is delegated to the PCE which requested the initiation. The PCC cannot revoke delegation of PCE-initiated LSP. When PCE returns delegation for such LSP or PCE fails, then the LSP become orphan and can be removed by a PCC after some time. The PCE may ask for a delegation of the orphan LSP.

Following RPC example illustrates a request for the LSP delegation:

URL: `/restconf/operations/network-topology-pcep:update-lsp`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1  <input>
2     <node>pcc://43.43.43.43</node>
3     <name>update-tunel</name>
4     <arguments>
5        <lsp xmlns:stateful="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
6           <delegate>true</delegate>
7           <administrative>true</administrative>
```

(continues on next page)

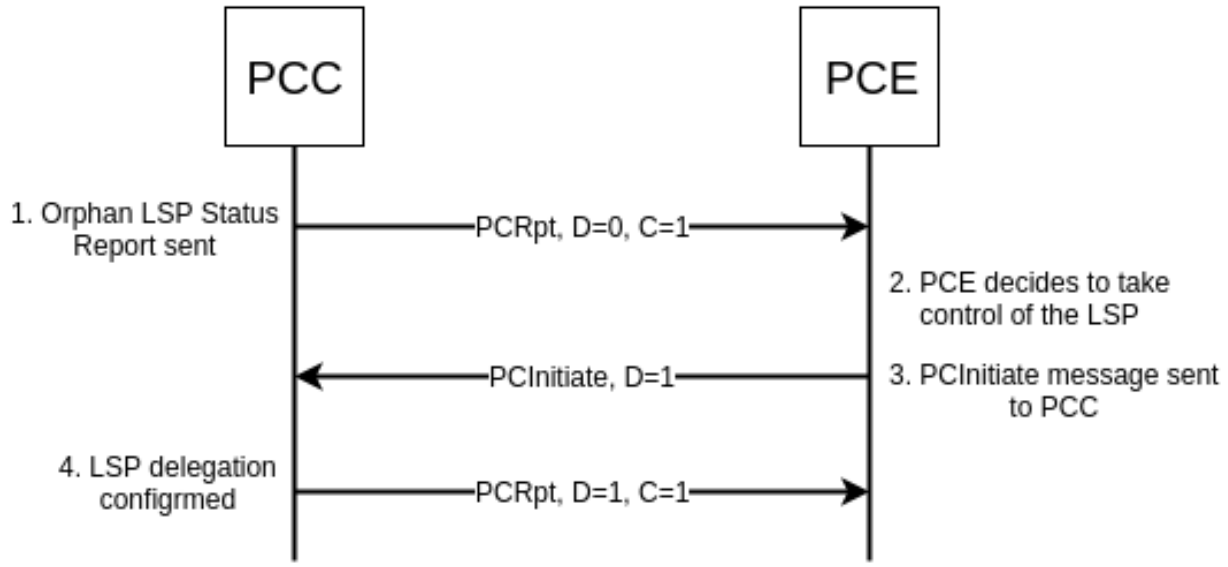


Fig. 15: Orphan PCE-initiated LSP - control taken by PCE.

(continued from previous page)

```
8           <tlvs>
9              <symbolic-path-name>
10                <path-name>dXBkYXRlLXR1bmVs</path-name>
11             </symbolic-path-name>
12          </tlvs>
13       </lsp>
14    </arguments>
15    <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
 ↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
 ↪topology-ref>
16 </input>
```

@line 2: **node** The PCC identifier.

@line 3: **name** The name of the LSP.

@line 6: **delegate** - *Delegation* flag set *true* in order to take the LSP delegation.

@line 10: **path-name** - The *Symbolic Path Name* TLV must be present when sending a request to take a delegation.

Segment Routing

The PCEP Extensions for Segment Routing (SR) allow a stateful PCE to compute and initiate TE paths in SR networks. The SR path is defined as an order list of *segments*. Segment Routing architecture can be directly applied to the MPLS forwarding plane without changes. Segment Identifier (SID) is encoded as a MPLS label.

Configuration

This capability is enabled by default. In order to disable it, a configuration should be changed as follows:

URL: `/restconf/config/pcep-segment-routing-app-config:pcep-segment-routing-app-config`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
1 <pcep-segment-routing-config xmlns=
  ↪"urn:opendaylight:params:xml:ns:yang:controller:pcep:segment-routing-app-config">
2    <sr-capable>false</sr-capable>
3 </pcep-segment-routing-config>
```

@line 2: **sr-capable** - True if capability is supported.

IANA code points

In PCEP-SR draft version 6, SR Explicit Route Object/Record Route Object subobjects IANA code points change was proposed. In order to use the latest code points, a configuration should be changed as follows:

URL: `/restconf/config/pcep-segment-routing-app-config:pcep-segment-routing-config`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<pcep-segment-routing-config xmlns=
↪"urn:opendaylight:params:xml:ns:yang:controller:pcep:segment-routing-app-config">
   <iana-sr-subobjects-type>true</iana-sr-subobjects-type>
</pcep-segment-routing-config>
```

@line 2: **iana-sr-subobjects-type** - True if IANA code points should be used.

LSP Operations for PCEP SR

The PCEP SR extension defines new ERO subobject - *SR-ERO subobject* capable of carrying a SID.

```
sr-ero-type
   +---- c-flag?                boolean
   +---- m-flag?                boolean
   +---- sid-type?              sid-type
   +---- sid?                   uint32
   +---- (nai)?
      +--:(ip-node-id)
      |  +---- ip-address             inet:ip-address
      +--:(ip-adjacency)
      |  +---- local-ip-address       inet:ip-address
      |  +---- remote-ip-address      inet:ip-address
      +--:(unnumbered-adjacency)
         +---- local-node-id          uint32
         +---- local-interface-id     uint32
         +---- remote-node-id         uint32
         +---- remote-interface-id    uint32
```

Following RPC example illustrates a request for the SR-TE LSP creation:

URL: `/restconf/operations/network-topology-pcep:add-lsp`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
   <node>pcc://43.43.43.43</node>
   <name>sr-path</name>
   <arguments>
      <lsp xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
         <delegate>true</delegate>
```

(continues on next page)

(continued from previous page)

```
         <administrative>true</administrative>
      </lsp>
      <endpoints-obj>
         <ipv4>
            <source-ipv4-address>43.43.43.43</source-ipv4-address>
            <destination-ipv4-address>39.39.39.39</destination-ipv4-address>
         </ipv4>
      </endpoints-obj>
      <path-setup-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
         <pst>1</pst>
      </path-setup-type>
      <ero>
         <subobject>
            <loose>false</loose>
            <sid-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing
↪">ipv4-node-id</sid-type>
            <m-flag xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">
↪true</m-flag>
            <sid xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">
↪24001</sid>
            <ip-address xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">39.39.39.39</ip-address>
         </subobject>
      </ero>
   </arguments>
   <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
↪topology-ref>
</input>
```

@line 16: **path-setup-type** - Set *1* for SR-TE LSP

@line 21: **ipv4-node-id** - The SR-ERO subobject represents *IPv4 Node ID* NAI.

@line 22: **m-flag** - The SID value represents an MPLS label.

@line 23: **sid** - The Segment Identifier.

Following RPC example illustrates a request for the SR-TE LSP update including modified path:

URL: `/restconf/operations/network-topology-pcep:update-lsp`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
   <node>pcc://43.43.43.43</node>
   <name>update-tunnel</name>
   <arguments>
      <lsp xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
         <delegate>true</delegate>
         <administrative>true</administrative>
      </lsp>
      <path-setup-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
         <pst>1</pst>
```

(continues on next page)

(continued from previous page)

```
      </path-setup-type>
      <ero>
         <subobject>
            <loose>false</loose>
            <sid-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing
↪">ipv4-node-id</sid-type>
            <m-flag xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">
↪true</m-flag>
            <sid xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">
↪24002</sid>
            <ip-address xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">200.20.160.41</ip-
↪address>
         </subobject>
         <subobject>
            <loose>false</loose>
            <sid-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing
↪">ipv4-node-id</sid-type>
            <m-flag xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">
↪true</m-flag>
            <sid xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">
↪24001</sid>
            <ip-address xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">39.39.39.39</ip-address>
         </subobject>
      </ero>
   </arguments>
   <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
↪topology-ref>
</input>
```

LSP State Synchronization Optimization Procedures

This extension bring optimizations for state synchronization:

- State Synchronization Avoidance
- Incremental State Synchronization
- PCE-triggered Initial Synchronization
- PCE-triggered Re-synchronization

Configuration

This capability is enabled by default. No additional configuration is required.

State Synchronization Avoidance

The State Synchronization Avoidance procedure is intended to skip state synchronization if the state has survived and not changed during session restart.

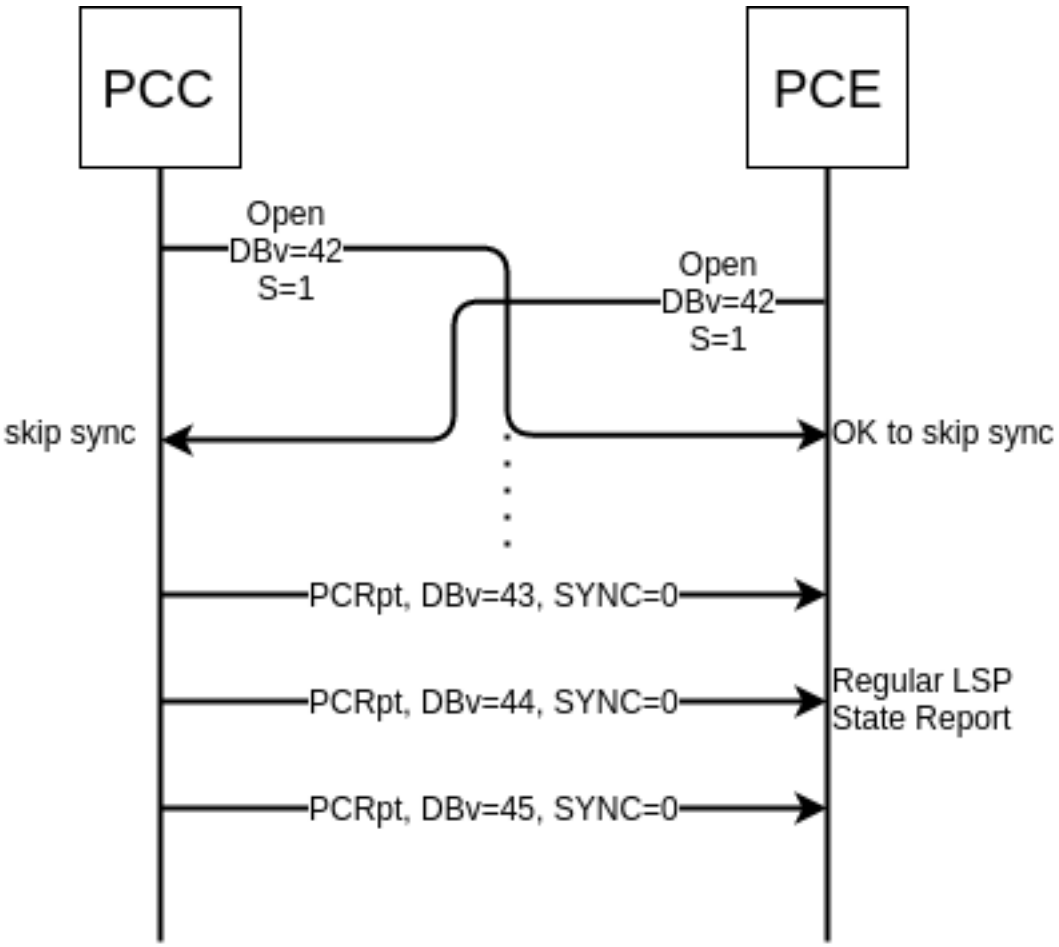


Fig. 16: State Synchronization Skipped.

Incremental State Synchronization

The Incremental State Synchronization procedure is intended to do incremental (delta) state synchronization when possible.

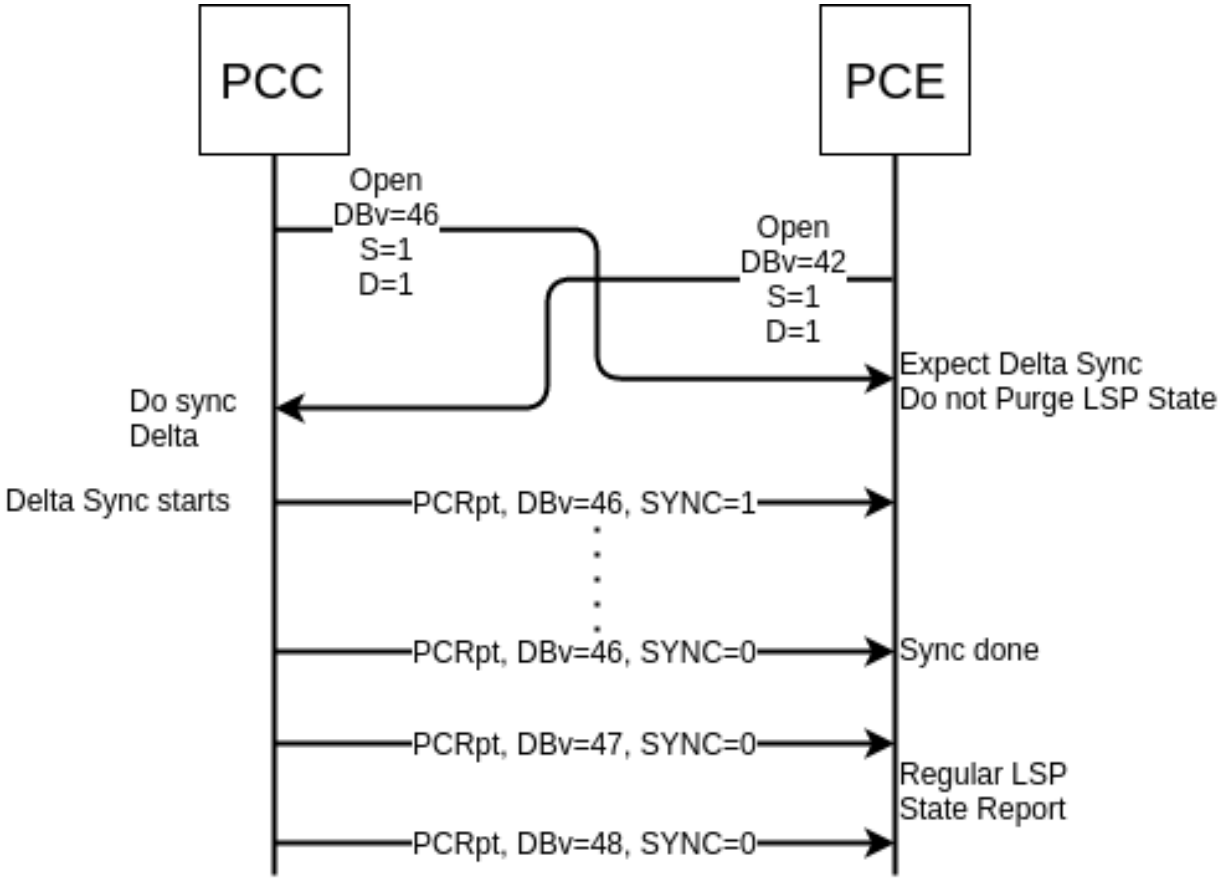


Fig. 17: Incremental Synchronization Procedure.

PCE-triggered Initial Synchronization

The PCE-triggered Initial Synchronization procedure is intended to do let PCE control the timing of the initial state synchronization.

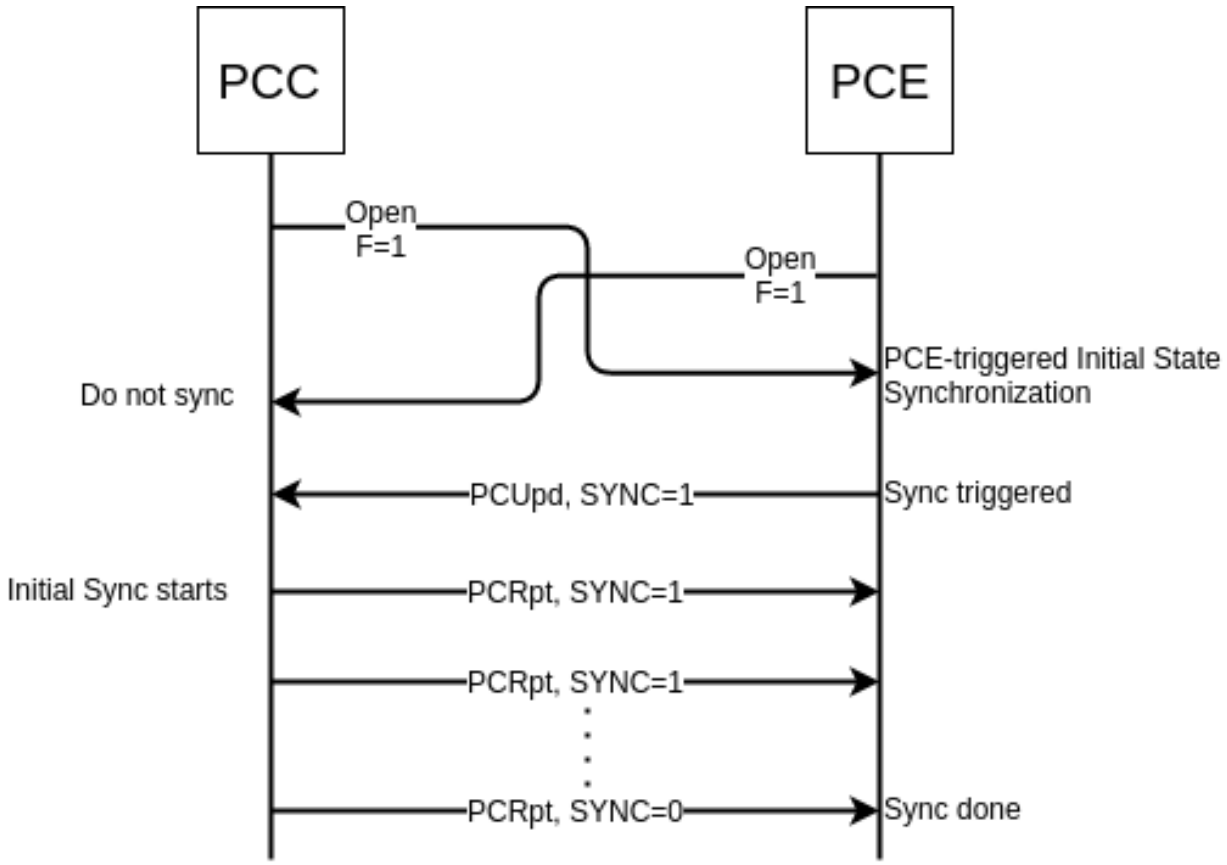


Fig. 18: PCE-triggered Initial State Synchronization Procedure.

Following RPC example illustrates a request for the initial synchronization:

URL: `/restconf/operations/network-topology-pcep:trigger-sync`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
   <node>pcc://43.43.43.43</node>
   <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
↪topology-ref>
</input>
```

PCE-triggered Re-synchronization

The PCE-triggered Re-synchronization: To let PCE re-synchronize the state for sanity check.

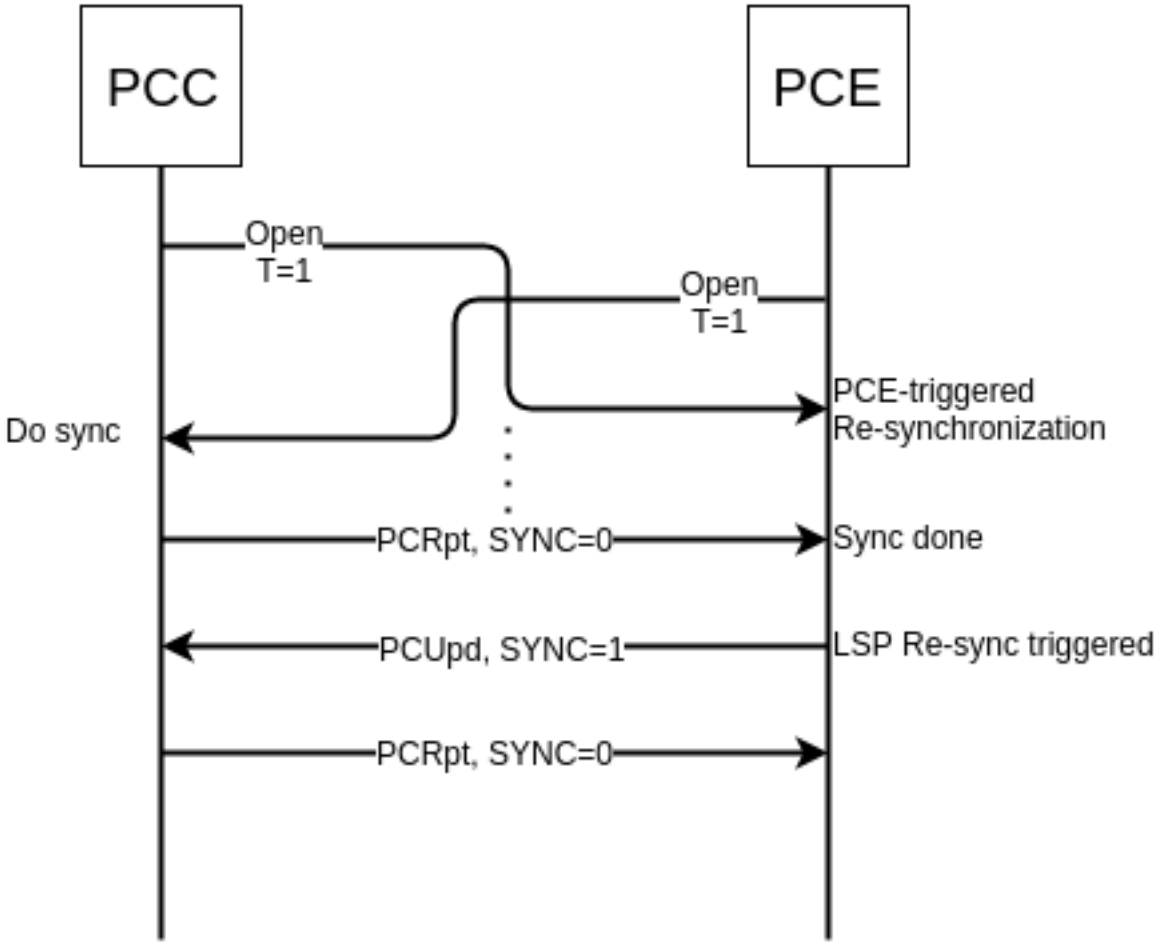


Fig. 19: PCE-triggered Re-synchronization Procedure.

Following RPC example illustrates a request for the LSP re-synchronization:

URL: `/restconf/operations/network-topology-pcep:trigger-sync`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
1 <input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
2    <node>pcc://43.43.43.43</node>
3    <name>update-lsp</name>
4    <network-topology-ref xmlns:topo="urn:TBD:params:xml:ns:yang:network-topology">/
  ↪topo:network-topology/topo:topology[topo:topology-id="pcep-topology"]</network-
  ↪topology-ref>
5 </input>
```

@line 3: **name** - The LSP name. If this parameter is omitted, re-synchronization is requested for all PCC's LSPs.

2.3.5 Path Computation Server

This section describes how to use Path Computation Server bundle in conjunction with the Path Computation Algorithm and Graph plugins. This Server provides a full PCE component that fully supports RFC5440 including PcRequest PcResponse messages exchanges from a PCC requesting a valid path to the PCE.

Contents

- *Installation*
- *Graph Setup*
  - *Manual activation*
  - *BGP-LS activation*
- *Usage*
- *Known limitations*

Installation

Check that the feature `features-algo` is installed. Normally it will be installed with the pcep feature `features-pcep`. Otherwise, install it with the following command:

```
feature:install features-algo
```

Graph Setup

The Path Computation Server uses the Path Computation Algorithm plugin which needs a graph to be able to compute constrained paths. Thus, a valid graph must be provided manually or automatically.

Manual activation

Create a new graph with the Rest API `Create Graph`:

```
PUT: restconf/config/graph:graph-topology
```

Refer to Graph documentation for details.

There is a restriction on the name of the graph due to the Path Computation Algorithm integration. It must be started by **"ted://"** string in order to be learn automatically by the Path Computation Server bundle.

Note that this kind of graph remains static. Thus, resources, mostly bandwidth, are not updated after deploying some RSVP-TE tunnels which consume bandwidth.

BGP-LS activation

To achieve better experience, notably in conjunction with RSVP-TE and in order to work on an up-to-date graph, an integration is provided with the BGP Link State protocol. This allows to automatically fulfil a graph with network traffic engineering information conveyed by the BGP-LS protocol. The resources, mostly bandwidth, are automatically updated in the graph after deploying an RSVP-TE tunnel. Note that this is not the case with Segment Routing.

For that purpose, just setup a BGP peering with a router that is BGP-LS speaker and report traffic engineering network topology from IS-IS-TE or OSPF-TE routing protocol. Refer to BGP documentation for the detail about how to setup a BGP peering with Link-State family.

Once done, verify that the graph is correctly fulfil with the Rest API:

```
GET: restconf/operational/graph:graph-topology
```

Usage

There is two ways to use the Path Computation Server: through PcRequest and with PcInitiate.

With PcRequest, just create a new tunnel on the router with an external PCE for path computation. Once PcRequest received, the Path Computation Server launches the path computation algorithm with requested parameters and in turn sends back to the PCC a PcResponse message with the computed path in the ERO. A NO-PATH object is returned in case of failure with the reason (e.g. source or destination unknown, constraints not met . . .). Check on the router that the tunnel is up and running. Wireshark capture will help to determine if the exchanges between the PCC and the PCE went well. Setting log debug for algo and pcep plugins and looking to the log will also ease debugging.

With PcInitiate message, just use the PCEP Rest API to setup an LSP

```
POST: /restconf/operations/network-topology-pcep:add-lsp
```

by omitting the ERO Object. Indeed, an automatic call to the Path Computation Algorithm will be triggered when the ERO is absent or empty with the given end-points and metrics objects as input paramters. Address family is automatically deduced from the IP address family of the end-points object. The same behaviour applies for Segment Routing: just add the *PST=1* indication in the json or xml payload will force the address family of path computation to Segment Routing.

To verify the result, just check the LSP-Database. The new LSP must have an ERO automatically computed as well as an RRO. Again, setting log debug for algo and pcep plugins and looking to the log will also help to verify that all is conform as expected.

Known limitations

As the Path Computation Server is in its initial release, there are some limitations mentioned hereinafter:

- Following PCEP Objects that may be present in the PcRequest message are not yet supported, and right now, ignored:
  - Include Route Object (IRO)
  - Exclude Route Object (XRO)
  - Objective Function (OF)
- LSP-Update Rest API with an empty ERO will not trigger Path Computation Algorithm. Use Path Computation Algorithm Rest API to get a new path, and then use the LSP-Update Rest API as usual with the computed ERO.
- For Segment Routing, ERO is provided with Node SID for NAI and SID index.

- Due to the integration with BGP-LS, the graph name must start with *ted://* tag in order to be automatically used by the pcep plugin.
- For Segment Routing, as network resources are not updated due to the lack of signaling, the resources consumed by the new segment path are not updated in the graph.

All these limitations will be solved in future releases.

2.3.6 Session statistics

The PCEP statistics provides information about PCE <-> PCC session and its stateful listener (topology-provider).

Usage

URL: `/restconf/operational/network-topology:network-topology/topology/pcep-topology/node/pcc:%2F%2F43.43.43.43/pcep-session-state`

Method: `GET`

Response Body:

```
<pcep-session-state xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep:stats">
   <messages>
      <last-received-rpt-msg-timestamp xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">1512640592</last-received-rpt-msg-timestamp>
      <sent-upd-msg-count xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">0</sent-upd-msg-count>
      <received-rpt-msg-count xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">2</received-rpt-msg-count>
      <sent-init-msg-count xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">0</sent-init-msg-count>
      <sent-msg-count>0</sent-msg-count>
      <last-sent-msg-timestamp>0</last-sent-msg-timestamp>
      <unknown-msg-received>0</unknown-msg-received>
      <received-msg-count>2</received-msg-count>
      <error-messages>
         <last-sent-error></last-sent-error>
         <received-error-msg-count>0</received-error-msg-count>
         <sent-error-msg-count>0</sent-error-msg-count>
         <last-received-error></last-received-error>
      </error-messages>
      <reply-time>
         <average-time>0</average-time>
         <min-time>0</min-time>
         <max-time>0</max-time>
      </reply-time>
   </messages>
   <peer-pref>
      <keepalive>30</keepalive>
      <deadtimer>120</deadtimer>
      <ip-address>127.0.0.1</ip-address>
      <session-id>0</session-id>
   </peer-pref>
   <local-pref>
      <keepalive>30</keepalive>
      <deadtimer>120</deadtimer>
```

(continues on next page)

(continued from previous page)

```
32      <ip-address>127.0.0.1</ip-address>
33      <session-id>0</session-id>
34    </local-pref>
35    <peer-capabilities>
36      <stateful xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">true</
      ↪stateful>
37      <instantiation xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">
      ↪true</instantiation>
38      <active xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">true</
      ↪active>
39    </peer-capabilities>
40    <session-duration>0:00:00:18</session-duration>
41    <delegated-lsps-count>1</delegated-lsps-count>
42    <synchronized>true</synchronized>
43  </pcep-session-state>
```

@line 3: **last-received-rpt-msg-timestamp** - The timestamp of last received PCRpt message.

@line 4: **sent-upd-msg-count** - The number of sent PCUpd messages.

@line 5: **received-rpt-msg-count** - The number of received PcRpt messages.

@line 6: **sent-init-msg-count** - The number of sent PCInitiate messages.

@line 7: **sent-msg-count** - Total number of sent PCEP messages.

@line 8: **last-sent-msg-timestamp** - The timestamp of last sent message.

@line 9: **unknown-msg-received** - The number of received unknown messages.

@line 10: **received-msg-count** - Total number of received PCEP messages.

@line 13: **last-sent-error** - Type/value tuple of last sent error.

@line 14: **received-error-msg-count** - Total number of received PCErr messages.

@line 15: **sent-error-msg-count** - Total number of sent PCErr messages.

@line 16: **last-received-error** - Type/value tuple of last sent error.

@line 19: **keepalive** - Advertised keep-alive value.

@line 20: **deadtimer** - Advertised deadtimer value.

@line 21: **ip-address** - Peer's IP address.

@line 22: **session-id** - Peer's session identifier.

@line 25: **keepalive** - Advertised keep-alive value.

@line 26: **deadtimer** - Advertised deadtimer value.

@line 27: **ip-address** - Peer's IP address.

@line 28: **session-id** - Peer's session identifier.

@line 31: **stateful** - Represents peer's stateful/stateless capability.

@line 32: **instantiation** - Represents peer's instantiation capability.

@line 33: **active** - Represents peer's LSP update capability.

@line 35: **session-duration** - Elapsed time (in d:H:m:s) from session-up until last statistic update.

@line 36: **delegated-lsps-count** - The number of delegated LSPs (tunnels) from PCC.

@line 37: **synchronized** - Represents synchronization status.

Following RPC can be used to fetch PCEP session statistics. If PCEP topology and/or PCC node is not specified in input, statistics for all PCEP sessions under the context are returned.

Usage

URL: `/restconf/operations/pcep-topology-stats-rpc:get-stats`

Method: `POST`

Content-Type: `application/xml`

Request Body:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:pcep:topology:stats:rpc">
   <topology>
      <topology-id>pcep-topology</topology-id>
      <node>
         <node-id>pcc://43.43.43.43</node-id>
      </node>
   </topology>
</input>
```

Response Body:

```
<output xmlns="urn:opendaylight:params:xml:ns:yang:pcep:topology:stats:rpc">
   <topology>
      <topology-id>pcep-topology</topology-id>
      <node>
         <node-id>pcc://43.43.43.43</node-id>
         <pcep-session-state>
            <synchronized>true</synchronized>
            <peer-capabilities>
               <stateful xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">true</stateful>
               <instantiation xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">true</instantiation>
               <active xmlns="urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats
↪">true</active>
            </peer-capabilities>
            <local-pref>
               <keepalive>30</keepalive>
               <deadtimer>120</deadtimer>
               <session-id>1</session-id>
               <ip-address>127.0.0.1</ip-address>
            </local-pref>
            <session-duration>4:01:59:46</session-duration>
            <messages>
               <unknown-msg-received>0</unknown-msg-received>
               <received-msg-count>11752</received-msg-count>
               <error-messages>
                  <last-sent-error>
                     <error-type>0</error-type>
                     <error-value>0</error-value>
                  </last-sent-error>
                  <received-error-msg-count>0</received-error-msg-count>
                  <last-received-error>
```

(continues on next page)

(continued from previous page)

```
                    <error-type>0</error-type>
                    <error-value>0</error-value>
                 </last-received-error>
                 <sent-error-msg-count>0</sent-error-msg-count>
              </error-messages>
              <sent-msg-count>11759</sent-msg-count>
              <last-sent-msg-timestamp>1553547804</last-sent-msg-timestamp>
              <reply-time>
                 <average-time>0</average-time>
                 <min-time>0</min-time>
                 <max-time>0</max-time>
              </reply-time>
              <received-rpt-msg-count xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">1</received-rpt-msg-count>
              <sent-init-msg-count xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">0</sent-init-msg-count>
              <last-received-rpt-msg-timestamp xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">1553195032</last-received-
↪rpt-msg-timestamp>
              <sent-upd-msg-count xmlns=
↪"urn:opendaylight:params:xml:ns:yang:pcep:stateful:stats">0</sent-upd-msg-count>
           </messages>
           <peer-pref>
              <keepalive>30</keepalive>
              <deadtimer>120</deadtimer>
              <session-id>8</session-id>
              <ip-address>127.0.0.1</ip-address>
           </peer-pref>
           <delegated-lsps-count>0</delegated-lsps-count>
        </pcep-session-state>
     </node>
  </topology>
</output>
```

2.3.7 CLI

PCEP Karaf Console (odl-bgpcep-pcep-cli) provides a CLI feature to read session statistics per node.

```
opendaylight-user@root> pcep:node-state -topology-id pcep-topology -node-id pcc://43.
↪43.43.43
```

2.3.8 Test tools

PCC Mock

The PCC Mock is a stand-alone Java application purposed to simulate a PCC(s). The simulator is capable to report sample LSPs, respond to delegation, LSP management operations and synchronization optimization procedures. This application is not part of the OpenDaylight Karaf distribution, however it can be downloaded from OpenDaylight's Nexus (use latest release version):

`https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/bgpcep/pcep-pcc-mock`

Usage

The application can be run from command line:

```
java -jar pcep-pcc-mock-*-executable.jar
```

with optional input parameters:

```
--local-address <Address:Port> (optional, default 127.0.0.1)
   The first PCC IP address. If more PCCs are required, the IP address will be␣
↪incremented. Port number can be optionally specified.

--remote-address <Address1:Port1,Address2:Port2,Address3:Port3,...> (optional,␣
↪default 127.0.0.1:4189)
   The list of IP address for the PCE servers. Port number can be optionally␣
↪specified, otherwise default port number 4189 is used.

--pcc <N> (optional, default 1)
   Number of mocked PCC instances.

--lsp <N> (optional, default 1)
   Number of tunnels (LSPs) reported per PCC, might be zero.

--pcerr (optional flag)
   If the flag is present, response with PCErr, otherwise PCUpd.

--log-level <LEVEL> (optional, default INFO)
   Set logging level for pcc-mock.

-d, --deadtimer <0..255> (optional, default 120)
   DeadTimer value in seconds.

-ka, --keepalive <0.255> (optional, default 30)
   KeepAlive timer value in seconds.

--password <password> (optional)
   If the password is present, it is used in TCP MD5 signature, otherwise plain TCP␣
↪is used.

--reconnect <seconds> (optional)
   If the argument is present, the value in seconds, is used as a delay before each␣
↪new reconnect (initial connect or connection re-establishment) attempt.
   The number of reconnect attempts is unlimited. If the argument is omitted, pcc-
↪mock is not trying to reconnect.

--redelegation-timeout <seconds> (optional, default 0)
   The timeout starts when LSP delegation is returned or PCE fails, stops when LSP is␣
↪re-delegated to PCE.
   When timeout expires, LSP delegation is revoked and held by PCC.

--state-timeout <seconds> (optional, default -1 (disabled))
   The timeout starts when LSP delegation is returned or PCE fails, stops when LSP is␣
↪re-delegated to PCE.
   When timeout expires, PCE-initiated LSP is removed.

--state-sync-avoidance <disconnect_after_x_seconds> <reconnect_after_x_seconds>
↪<dbVersion>
```

(continues on next page)

(continued from previous page)

```
   Synchronization avoidance capability enabled.
      - disconnect_after_x_seconds: seconds that will pass until disconnections is␣
↪forced. If set to smaller number than 1, disconnection wont be performed.
      - reconnect_after_x_seconds: seconds that will pass between disconnection and␣
↪new connection attempt. Only happens if disconnection has been performed.
      - dbVersion: dbVersion used in new Open and must be always equal or bigger than␣
↪LSP. If equal than LSP skip synchronization will be performed,
        if not full synchronization will be performed taking in account new starting␣
↪dbVersion desired.
 --incremental-sync-procedure <disconnect_after_x_seconds> <reconnect_after_x_seconds>
↪ <dbVersion>
   Incremental synchronization capability enabled.
      - dbVersion: dbVersion used in new Open and must be always bigger than LSP.␣
↪Incremental synchronization will be performed taking in account new starting␣
↪dbVersion desired.

 --triggered-initial-sync
   PCE-triggered synchronization capability enabled. Can be combined combined with␣
↪state-sync-avoidance/incremental-sync-procedure.

 --triggered-re-sync
   PCE-triggered re-synchronization capability enabled.
```

Data Change Counter Tool

Data Change Counter tool registers a Data Change Listener to a specified topology's subtree. This will allow us to know the quantity of changes produced under it, with each data change event counter will be incremented.

Installation

Installing data change counter tool

```
feature:install odl-restconf odl-bgpcep-data-change-counter
```

Configuration

Once we set the configuration, a new data change counter will be created and registers to example-linkstate-topology.

Important: Clustering - Each Counter Identifier should be unique.

URL: `/restconf/config/odl-data-change-counter-config:data-change-counter-config/data-change-counter`

Method: `PUT`

Content-Type: `application/xml`

Request Body:

```
<data-change-counter-config xmlns="urn:opendaylight:params:xml:ns:yang:bgpcep:data-change-counter-config">
    <counter-id>data-change-counter</counter-id>
    <topology-name>example-linkstate-topology</topology-name>
</data-change-counter-config>
```

@line 2: **Counter Id** - Unique counter change identifier.

@line 3: **Topology Name** - An identifier for a topology.

Usage

Counter state for topology

URL: `/restconf/operational/data-change-counter:data-change-counter/counter/data-change-counter`

Method: `GET`

Response Body:

```
<counter xmlns="urn:opendaylight:params:xml:ns:yang:bgp-data-change-counter">
    <id>data-change-counter</id>
    <count>0</count>
</counter>
```

@line 2: **Counter Id** - Unique counter change identifier.

@line 3: **Count** - Number of changes under registered topology's subtree.

2.3.9 Troubleshooting

This section offers advices in a case OpenDaylight PCEP plugin is not working as expected.

Contents

- *PCEP is not working...*
- *Bug reporting*
  - *References*

PCEP is not working...

- First of all, ensure that all required features are installed, local PCE and remote PCC configuration is correct.

  To list all installed features in OpenDaylight use the following command at the Karaf console:

  ```
  feature:list -i
  ```

- Check OpenDaylight Karaf logs:

  From Karaf console:

```
log:tail
```

or open log file: `data/log/karaf.log`

Possibly, a reason/hint for a cause of the problem can be found there.

- Try to minimize effect of other OpenDaylight features, when searching for a reason of the problem.
- Try to set DEBUG severity level for PCEP logger via Karaf console commands, in order to collect more information:

```
log:set DEBUG org.opendaylight.protocol.pcep
```

```
log:set DEBUG org.opendaylight.bgpcep.pcep
```

and for Path Computation Algorithm

```
log:set DEBUG org.opendaylight.algo
```

Bug reporting

Before you report a bug, check BGPCEP Jira to ensure same/similar bug is not already filed there.

Write an e-mail to bgpcep-users@lists.opendaylight.org and provide following information:

1. State OpenDaylight version
2. Describe your use-case and provide as much details related to PCEP as possible
3. Steps to reproduce
4. Attach Karaf log files, optionally packet captures, REST input/output

References

- A Path Computation Element (PCE)-Based Architecture
- Path Computation Element (PCE) Communication Protocol Generic Requirements
- Unanswered Questions in the Path Computation Element Architecture
- A PCE-Based Architecture for Application-Based Network Operations
- Framework for PCE-Based Inter-Layer MPLS and GMPLS Traffic Engineering
- Applicability of a Stateful Path Computation Element (PCE)

2.4 GRAPH Model User Guide

This guide contains information on how to use the OpenDaylight Graph Model plugin that is the basis for Path Computation Algorithms. Graph and algorithms are used in the PCE Server part to compute path. This path will then serves to fulfil Explicit Route Object (ERO) for PCResponse message in turn of a PCRequest message.

Note: The user should learn about Graph Theory and (Constraint) Shortest Path First algorithms.

2.4.1 Graph Model Overview

This section provides a high-level overview of the Graph Model.

Contents

- *Graph Theory and Path Computation*
- *Yang Model for Graph*
- *Java Class for Connected Graph*

Graph Theory and Path Computation

The primary goal of Graph and Algorithms features, is to be able to compute constrainted path among a given IP/MPLS network. Such path could be used to enforce connectivity by means of RSVP-TE or Segment Routing TE through PCEP protocol or simply provide a solution to answer a path request (coming from RESTCONF API or a PcRequest message).

In IP/MPLS networks, these path computation algorithms need to access to a representation of the network topology which is, in general, denoted as *Traffic Engineering Database (TED)* in reference to information convey by the IGP Traffic Engineering routing protocols such as OSPF-TE or IS-IS-TE. There is many way to store this TED and the IETF is in the process of defining the corresponding yang model (draft-ietf-teas-yang-te-topo-22.txt). But, most of these path computation algorithms have been designed with a particular representation of network model for perfomrance efficiency: a Graph. This latter is denoted in literature as *G(V, E)* where V, the Vertices, represent the nodes (e.g. routers) and E, the Edges, represent the link between nodes. There is numerous graph type, but for path computation, a Directed and Connected Graph is recommended, again for performance efficiency.

A Connected Graph is a particular graph type where each pair of vertices forms the endpoints of a path. It is used because path computation algorithms need to progress smoothly in the graph without searching for the next hops in the whole graph (mostly for performance issue) and also because some algorithms need to mark Vertices as visited once processed to avoid loop. Connected Graph is a particular graph type where each pair of vertices forms the endpoints of a path.

In general, for modern networks (i.e. with optical links), links are considered as full-duplex. Thus, from a resources (mostly bandwidth) point of view, going from node *A to B* will consumes resources (mostly bandwidth) on the link from A to B while keeping intact resources on the link from B to A. So, the graph model needs to reflect this behaviour. The graph that represents the network will used Directed Edges for that purposes. So, the recommended graph is a Directed Graph (sometimes also named Oriented Graph). As a consequence, it is necessary to create 2 edges between 2 vertices: *A to B* and *B to A*.

For more information, reader could refer to Graph Theory e.g. https://en.wikipedia.org/wiki/Graph_theory and Path Computation algorithms e.g. Shortest Path First (SPF) https://en.wikipedia.org/wiki/Shortest_path_problem and Constrainted Shortest Path First (CSPF) https://en.wikipedia.org/wiki/Constrained_Shortest_Path_First.

Yang Model for Graph

In order to manipulate the Graph within the OpenDaylight Data Store, the given yang model has been provided. It defines Edges and Vertices. Edges are enhanced by Edges Attributes which are define all link attributes we could collect from real network (e.g. through BGP Link State). Vertices are enhanced by Prefixes in order to find quickly where End Points of a path are attached in the Graph. It also serves to store Segment Routing information when computing path for Segment Routing TE setup.

A new yang model is provided as an alternative to the IETF yang-te-topo model and IETF RFC8345 for several reasons:

- Some link and node parameters (e.g. Segment Routing, TE Metric extensions) which are available in IP/MPLS networks, through the IGP-TE or BGP-LS protocols (see Linkstate Routes in BGP RIB) are not present in the IETF ted model
- Node and link identifiers are represented as strings in the IETF ted model, which it is not efficient when looking into a HashMap() to find a node or a link by its identifier
- Even if LinkstateTopologyBuilder() provided mechanism to fulfil IETF ted model in the datastore, the NodeHolder an TpHolder classes have been defined as private, and thus could not be used outside the LinkstateTopologyBuilder() class

Graph and Algorithm have been also designed to be used by other projects (e.g. openflow) which not control IP/MPLS network. Thus, even if the graph model addresses in first case the IP/MPLS network, its genericity allows it to be suitable for other network types.

The yand model for the Graph is as follow:

```
module: graph
+--rw graph-topology
    +--rw graph* [name]
        +--rw name             string
        +--rw domain-scope?    enumeration
        +--rw asn?             uint32
        +--rw vertex* [vertex-id]
        |  +--rw vertex-id       uint64
        |  +--rw name?           string
        |  +--rw router-id?      inet:ip-address
        |  +--rw vertex-type?    enumeration
        |  +--rw srgb
        |  |  +--rw lower-bound?   uint32
        |  |  +--rw range-size?    uint32
        |  +--rw asn?            uint32
        +--rw edge* [edge-id]
        |  +--rw edge-id              uint64
        |  +--rw local-vertex-id?     uint64
        |  +--rw remote-vertex-id?    uint64
        |  +--rw name?                string
        |  +--rw edge-attributes
        |     +--rw metric?                    uint32
        |     +--rw te-metric?                 uint32
        |     +--rw admin-group?               uint32
        |     +--rw local-address?             inet:ip-address
        |     +--rw remote-address?            inet:ip-address
        |     +--rw local-identifier?          uint32
        |     +--rw remote-identifier?         uint32
        |     +--rw max-link-bandwidth?        decimal-bandwidth
        |     +--rw max-resv-link-bandwidth?   decimal-bandwidth
        |     +--rw unreserved-bandwidth* [class-type]
```

(continues on next page)

(continued from previous page)

```
|      |  +--rw class-type    uint8
|      |  +--rw bandwidth?    decimal-bandwidth
|      +--rw delay?                    delay
|      +--rw min-max-delay
|      |  +--rw min-delay?   delay
|      |  +--rw max-delay?   delay
|      +--rw jitter?                   delay
|      +--rw loss?                     loss
|      +--rw residual-bandwidth?       decimal-bandwidth
|      +--rw available-bandwidth?      decimal-bandwidth
|      +--rw utilized-bandwidth?       decimal-bandwidth
|      +--rw adj-sid?                  uint32
|      +--rw backup-adj-sid?           uint32
|      +--rw srlgs*                    uint32
+--rw prefix* [prefix]
+--rw prefix        inet:ip-prefix
+--rw prefix-sid?   uint32
+--rw node-sid?     boolean
+--rw vertex-id?    uint64
```

Java Class for Connected Graph

Yang model represents data as a Flat Tree hierarchy. However, this particular graph representation (without a specific storage engine capabilities) is not very useful for Path Computation due to lower performance compared to other Graph types. Of course path computation algorithms could play with a such Graph, but at the cost of performance issue as algorithms need to search the neighbours of a vertices at each step when progressing in the graph. This will decrease the performance by a factor of N to N^2 depending of the algorithms. For large scale network, say 1000+ nodes, it is too high.

Yang syntax authorizes reference to other grouping or leaf with 'leafref'. This could allows from a Vertex to access to Edges. However, it is not possible to achieve a cross reference between Vertex and Edge. In Connected Graph, both Vertex and Edge must reference each together: from Vertex it is needed to access directly at the list of Edges connected to this Vertex, and from Edge, it is need to access directly at the source and destination Vertex.

So, to overcome this limitation, the implemented Graph is composed of two pieces:

- A standard Graph modeled in yang and stored in the Data Store
- A Connected Graph version based on the yang model but stored in memory only

The connected version of Vertex is composed of:

```
/* Reference to input and output Connected Edge within the Connected Graph */
private ArrayList<ConnectedEdgeImpl> input = new ArrayList<>();
private ArrayList<ConnectedEdgeImpl> output = new ArrayList<>();

/* List of Prefixes announced by this Vertex */
private ArrayList<Prefix> prefixes = new ArrayList<>();

/* Reference to the Vertex of the standard Graph associated to the Connected Graph */
private Vertex vertex = null;
```

Where distinction is made between input and output Edges in order to respect the Directed Graph behviour.

The connected version of Edges is composed of:

```
/* Reference to Source and Destination Connected Vertex within the Connected Graph */
private ConnectedVertexImpl source;
private ConnectedVertexImpl destination;

/* Reference to the Edge within the Graph associated to the Connected Graph */
private Edge edge;
```

Where source and destination Vertices also ease to implement the Directed Graph.

And finally, the connected version of Graph is composed of:

```
/* List of Connected Vertics that composed this Connected Graph */
private final HashMap<Long, ConnectedVertexImpl> vertices = new HashMap<>();

/* List of Connected Edges that composed this Connected Graph */
private final HashMap<Long, ConnectedEdgeImpl> edges = new HashMap<>();

/* List of IP prefix attached to Vertices */
private final HashMap<IpPrefix, Prefix> prefixes = new HashMap<>();

/* Reference to the non connected Graph stored in DataStore */
private Graph graph;
```

Where Vertices, Edges and Prefixes are stored in *HashMap* to speed up the access of a given element of the Graph.

Note that the Unique Key identifier for Connected Edge and Connected Vertex must not be equal to zero (and as a consequence the Edge and Vertex key). This restriction is due to some algorithms that used the value 0 as a special indication during the path computation.

2.4.2 Running Graph

This section explains how to install Graph plugin.

1. Install Graph feature - `features-graph`. Also, for the sake of this sample, it is required to install RESTCONF. In the Karaf console, type command:

   ```
   feature:install features-restconf features-graph
   ```

2. The Graph plugin contains a default empty configuration, which is applied after the feature starts up. One instance of Graph plugin is created (named *graph-topology*), and its presence can be verified via REST:

   URL: `restconf/config/graph:graph-topology`

   Method: `GET`

   Response Body:

   ```
   {}
   ```

   It is also posible to access to the operational graph topology which is also empty by default via REST:

   URL: `restconf/operational/graph:graph-topology`

   Method: `GET`

   Response Body:

   ```
   {}
   ```

2.4.3 Manage Graph

This section explains how to manipulate the Graph through REST API.

Contents

- *Concept*
- *JAVA API*
- *REST API*
  - *Get Graph*
  - *Create Graph*
  - *Add Graph*
  - *Delete Graph*
  - *Add Vertices*
  - *Delete Vertex*
  - *Add Edges*
  - *Delete Edge*
  - *Add Prefixes*
  - *Delete Prefix*

Concept

The connected Graph is not accessible through the REST API as it is not posssible to represent connected Edges and Vertices with yang model (see Graph Overview). Thus, Graph feature provides a new service named ConnectedGraphProvider published in Karaf. This service maintains an up-to-date list of Connected Graph stored in memory and allows to:

- Get Connected Graph by name or GraphKey
- Create Connected Graph
- Add existing graph to create associated Connected Graph
- Delete existing Graph identify by its GraphKey

Then, Connected Graph provides method to manage Vertices, Edges and Prefix. The ConnectedGraphProvider is also in charge to maintain up to date the Graph associated to the Connected Graph in the OpenDaylight operational Data Store.

In fact, two graphs are stored in the Data Store:

- Operational Graph in `restconf/operational` which is the graph associated with the Connected Graph stored in memory
- Configuration Graph in `restconf/config` which is the graph that could be create / modify / delete in order to produce the Connected Graph and thus, the associated Graph stored in operational Data Store

It is also possible to add / delete Vertices, Edges and Prefix on an existing Graph through the REST API.

JAVA API

Developper will refer to the Java Doc to manage Connected and standard Graph. The main principle is as follow:

1. First Create a Connected Graph through the ConnectedGraphProvider which is a new service provided by the Graph feature through Karaf:

```
ConnectedGraph cgraph = graphProvider.createConnectedGraph("example", GraphType.
↪IntraDomain);
```

Or by adding an initial graph:

```
Graph graph = new GraphBuilder().setName("example").setGraphType(GraphType.
↪IntraDomain).build();
ConnectedGraph cgraph = graphProvider.addGraph(graph);
```

Where graphProvider is obtained from blueprint.

2. Once created, the Connected Graph offers various method to manage Vertices and Edges. For example:

```
/* Add a Vertex */
Vertex vertex = new VertexBuilder().setVertexId(Uint64.ValueOf(1)).build();
cgraph.addVertex(vertex);

/* Add an Edge */
Edge edge = new EdgeBuilder().setEdgeId(Uint64.ValueOf(1)).build();
cgraph.addEdge(edge);

...
```

REST API

This section provides the list of REST method that could be used to manage Graph.

Get Graph

Graphs are stored in operation and config Data Store. Thus there are accessible through the `graph:graph-topology` namespace as follow:

URL: `restconf/operational/graph:graph-topology`

Method: `GET`

Response Body:

```
{
    "graph-topology": {
        "graph": [
            {
                "name": "example",
                "vertex": [
                    {
                        "vertex-id": 2,
                        "name": "r2",
```

(continues on next page)

(continued from previous page)

```
                        "vertex-type": "standard"
                    },
                    {
                        "vertex-id": 1,
                        "name": "r1",
                        "vertex-type": "standard"
                    }
                ],
                "domain-scope": "intra-domain"
            }
        ]
    }
}
```

Graphs publish in the configuration Data Store are also accessible through REST API with the same namespace as follow:

URL: `restconf/config/graph:graph-topology`

Method: `GET`

Response Body:

```
{
    "graph-topology": {
        "graph": [
            {
                "name": "example",
                "vertex": [
                    {
                        "vertex-id": 2,
                        "name": "r2",
                        "vertex-type": "standard"
                    },
                    {
                        "vertex-id": 1,
                        "name": "r1",
                        "vertex-type": "standard"
                    }
                ],
                "domain-scope": "intra-domain"
            }
        ]
    }
}
```

Create Graph

Graphs could be created with PUT method. In this case, all previously configured graphs are removed from both the configuration and operational Data Store. This includes all modification and associated Connected Graphs.

URL: `restconf/config/graph:graph-topology`

Method: `PUT`

Content-Type: `application/json`

Request Body:

```
1  {
2      "graph-topology": {
3          "graph": [
4              {
5                  "name": "example",
6                  "domain-scope": "intra-domain",
7                  "vertex": [
8                      {
9                          "vertex-id": 1,
10                         "name": "r1"
11                     },
12                     {
13                         "vertex-id": 2,
14                         "name": "r2"
15                     }
16                 ],
17                 "edge": [
18                     {
19                         "edge-id": 1,
20                         "name": "r1 - r2",
21                         "local-vertex-id": 1,
22                         "remote-vertex-id": 2
23                     },
24                     {
25                         "edge-id": 2,
26                         "name": "r2 - r1",
27                         "local-vertex-id": 2,
28                         "remote-vertex-id": 1
29                     }
30                 ]
31             }
32         ]
33     }
34 }
```

@line 5: **name** The Graph identifier. Must be unique.

@line 6: **domain-scope** The type of the Graph: intra-domain or inter-domain.

@line 7: **vertex** - List of Vertices. Each Vertex ID must be unique.

@line 17: **edges** - List of Edges. Each Edge ID must be unique.

@line 21: **local-vertex-id** - Vertex ID where the Edge is connected from. The vertex ID must correspond to vertex that is present in the vertex list, otherwise, the connection will not be estabished in the Connected Graph.

@line 22: **remote-vertex-id** - Vertex ID where the Edge is connected to. The vertex ID must correspond to vertex that is present in the vertex list, otherwise, the connection will not be estabished in the Connected Graph.

Add Graph

It is also possible to add a Graph to the existing list. POST method will be used instead of PUT. Body and URL remains the same.

Delete Graph

Removing a graph used the DELETE method as follow:

URL: `restconf/config/graph:graph-topology/graph/example`

Method: `DELETE`

The name of the graph i.e. the Graph Key to be deleted must be provide within the URL.

Add Vertices

One or more vertex could be added to a Graph. If the graph doesn't exist, it will be automatically created. Only POST method must be used.

URL: `restconf/config/graph:graph-topology/graph/example`

Method: `POST`

Content-Type: `application/json`

Request Body:

```
{
    "vertex": [
        {
            "vertex-id": 100,
            "name": "r100",
            "router-id": "192.168.1.100"
        }
    ]
}
```

Delete Vertex

Removing a vertex used the DELETE method as follow:

URL: `restconf/config/graph:graph-topology/graph/example/vertex/10`

Method: `DELETE`

The Vertex to be deleted is identified by its Vertex Id and must be provide within the URL.

Add Edges

One or more edges could be added to a Graph. If the graph doesn't exist, it will be automatically created. Only POST method must be used.

URL: `restconf/config/graph:graph-topology/graph/example`

Method: `POST`

Content-Type: `application/json`

Request Body:

```
{
    "edge": [
        {
            "edge-id": 10,
            "name": "r1 - r2",
            "local-vertex-id": 1,
            "remote-vertex-id": 2
        },
        {
            "edge-id": 20,
            "name": "r2 - r1",
            "local-vertex-id": 2,
            "remote-vertex-id": 1
        }
    ]
}
```

Delete Edge

Removing an edge used the DELETE method as follow:

URL: `restconf/config/graph:graph-topology/graph/example/edge/10`

Method: `DELETE`

The Edge to be deleted is identified by its Edge Id and must be provide within the URL.

Add Prefixes

One or more prefixe could be added to a Graph. If the graph doesn't exist, it will be automatically created. Only POST method must be used.

URL: `restconf/config/graph:graph-topology/graph/example`

Method: `POST`

Content-Type: `application/json`

Request Body:

```
{
    "prefix": [
        {
            "prefix": "192.168.1.0/24",
            "vertex-id": 1
        }
    ]
}
```

Delete Prefix

Removing a prefix used the DELETE method as follow:

URL: `restconf/config/graph:graph-topology/graph/example/prefix/192%2e168%2e1%2e0%2f24`

Method: `DELETE`

The Prefix to be deleted is identified by its Prefix Id and must be provide within the URL. As the prefix identifier is the ip prefix, '.' and '/' must be replace by their respective ASCII representation i.e. '%2e' for dot and '%2f' for slash.

2.5 Path Computation Algorithms User Guide

This guide contains information on how to use the OpenDaylight Path Computation Algorithms plugin. They are used in the PCE Server part to compute path in order to fulfil Explicit Route Object (ERO) for PcResponse message in turn of a PcRequest message.

Note: As Path Computation Algorithms use the Graph plugin, user should read the previous chapter about the OpenDaylight Graph plugin as well as learn about (Constrained) Shortest Path First algorithms.

2.5.1 Path Computation Algorithms Overview

This section provides a high-level overview about Path Computation Algorithms.

Contents

- *Path Computation Theory*
- *Path Computation Overview*

Path Computation Theory

Path computation in network has for objective to find a path between two end points (p2p) or between a point and a multiple destination points (p2mp). The well known algorithm is the Djikstra one whch aims to find the shortest path by taking into account the number of hop as key objective.

In addition, path computation aims also to take into account various constraints to find optimal path in a network. The constraints are various and may include standard routing protcol metric (IGP metric), Traffic Enginnering metic (TE metric), end to end delay (Delay), end to end delay variation (Jitter) ... all referenced as *Additive Metric* because the metric carried by each link is added together to check that the end-to-end constraint is respected. The second category of metric is named *Concave Metric* and concerns the Bandwidth and Loss. Indeed, these metrics are not added together but checked over each link to verify that the constraints are met.

For more information, reader could refer to Path Computation algorithms e.g. Shortest Path First (SPF) https://en.wikipedia.org/wiki/Shortest_path_problem and Constrainted Shortest Path First (CSPF) https://en.wikipedia.org/wiki/Constrained_Shortest_Path_First.

Path Computation Overview

This features provides three Path Computation algorithms:

- Shortest Path First (SPF) a.k.a. Djikstra
- Constrainted Shortest Path First (CSPF)
- Self Adaptive Multiple Constraints Routing Algorithm (SAMCRA)

All of these algorithms use the same principles:

- A priority Queue where all potential paths are stored
- A pruning function that validate / invalidate edge to next vertex regarding the constraints

The priority queue sort elements based on a key and outpout the element that presents the smallest key value. Here, depedning of the algorithm, the key will represents the standard metric (SPF), the Traffic Engineering Metric (CSPF) or the TE Metric and Delay as composite metric. The key represents only *Additive Metrics* to be optimized.

The pruning function will check if edge to the next vertex respects the given constraints. This concerns both *Concave Metrics*, bandwidth and loss, and *Additive Metrics*. For the latter, current metrics value are added to the edge metrics value and check against given constraints. In addition, address family (IPv4, IPv6, Segment Routing for IPv4 or IPv6) is checked to avoid validate a path that is not capable of the given requested address family (e.g. an IPv4 vertex / edge for an IPv6 path).

The pseudo code below shows how the various algorithms are working:

```
/* Initialize the algorithms */
initialize pathSource and pathDestination with vertex source and Destination;
visitedVertexList.clear();
processedPathList.clear();
priorityQueue.clear();
priorityQueue.add(pathSource);
currentMetric = Integer.MAX_VALUE;
computedPath = null;

/* Loop until Priority Queue becomes empty */
while (!priorityQueue.empty()) {
    /* Get currentPath with lowest accumulated metric from the Priority Queue */
    currentPath = priorityQueue.poll();
```

(continues on next page)

(continued from previous page)

```
    /* For all Edges from the current vertex, check if next Vertex is acceptable or
↪not */
    for (edge : currentPath.getvertex().getAllEdges()) {
        if (pruneEdge(edge, currentPath)) {
            continue;
        }
        /* If we reach destination with a better Metric, store the path */
        if (relax(edge, currentPath) && (pathDestination.getMetric() < currentMetric))
            computedPath = pathDestination;
        }
    }
}

/* Example of relax function that checks the standard routing Metric */
private boolean relax(edge, currentPath) {
    /* Verify if we have not visited this Vertex to avoid loop */
    if (visitedVerticeList.contains(edge.getDestination())) {
        return false;
    }

    /* Get Next Vertex from processedPathList or create a new one if it has not yet
↪processed */
    nextPath = processedPathList.get(edge.getDestination());
    if (nextPath == null) {
        nextPath = new (edge.getDestination());
        processedPathList.add(nextPath);
    }

    /* Compute Metric from source to this next Vertex and add or update it in the
↪Priority Queue
     * if total path Metric is lower than metric associated to this next Vertex.
     * This could occurs if we process a Vertex that as not yet been visited in the
↪Graph
     * or if we found a shortest path up to this Vertex. */
    int totalMetric = edge.getMetric() + currentPath.getMetric();
    if (nextPath.getMetric() > totalMetric) {
        nextPath = currentPath;
        nextPath.setMetric(totalMetric);
        nextPath.addEdgeToPath(edge);
        /* Here, we set the path key with the total Metric for the  Priority Queue
         * At next iteration, Priority Queue will consider this new Path in the
↪collection
         * to provide the path with the lowest total Metric */
        nextPath.setKey(totalMetric);
        priorityQueue.add(nextPath);
    }
    /* Return True if we reach the destination, false otherwise */
    return pathDestination.equals(nextPath);
}

/* Example of prune function that checks bandwidth and standard metric */
boolean pruneEdge(edge, currentPath) {
    if (edge.getBandwidth() < constraints.getBandwidth()) {
        return true;
    }
    if (edge.getMetric() + currentPath.getMetric() > constraints.getMetric()) {
        return true;
```

(continues on next page)

(continued from previous page)

```
    }
}
```

This pseudo code corresponds to the ShortestPathFist.java class.

Note: Details of SAMCRA algorithm could be found in the article **Concepts of Exact QoS Routing Algorithms**, *Piet Van Mieghem and Fernando A. Kuipers, IEEE/ACM Transactions on Networking, Volume 12, Number 5, October 2004.*

2.5.2 Running Path Computation

This section details how to install and use the Path Computation Algorithm feature.

Contents

- *Installation*
- *Yang Model*
- *REST API*
  - *Get Constrained Path*
- *Troubleshooting*

Installation

Install feature - `features-algo`. Also, for the sake of this sample, it is required to install RESTCONF in order to use the Path Computation service.

In the Karaf console, type command:

```
feature:install features-restconf features-algo
```

Yang Model

Path Computation algorithm used Graph plugin, thus graph API and yang models. A new yang model has been introduced in order to define constrained path model as well as a new RPC service to call Path Computation Algorithm plugin. The model is given below:

```
module: path-computation
+--rw constrained-path
   +--rw metric?             uint32
   +--rw te-metric?          uint32
   +--rw delay?              gr:delay
   +--rw jitter?             gr:delay
   +--rw loss?               gr:loss
   +--rw admin-group?        uint32
   +--rw address-family?     enumeration
   +--rw class-type?         uint8
   +--rw bandwidth?          gr:decimal-bandwidth
   +--rw source?             uint64
```

(continues on next page)

(continued from previous page)

```
   +--rw destination?        uint64
   +--rw path-description* []
   |  +--rw ipv4?    inet:ipv4-address
   |  +--rw ipv6?    inet:ipv6-address
   |  +--rw label?   netc:mpls-label
   +--rw status?             computation-status

rpcs:
   +---x get-constrained-path
      +---w input
      |  +---w graph-name     string
      |  +---w source?        uint64
      |  +---w destination?   uint64
      |  +---w constraints
      |  |  +---w metric?           uint32
      |  |  +---w te-metric?        uint32
      |  |  +---w delay?            gr:delay
      |  |  +---w jitter?           gr:delay
      |  |  +---w loss?             gr:loss
      |  |  +---w admin-group?      uint32
      |  |  +---w address-family?   enumeration
      |  |  +---w class-type?       uint8
      |  |  +---w bandwidth?        gr:decimal-bandwidth
      |  +---w algorithm?     algorithm-type
      +--ro output
         +--ro path-description* []
         |  +--ro ipv4?    inet:ipv4-address
         |  +--ro ipv6?    inet:ipv6-address
         |  +--ro label?   netc:mpls-label
         +--ro status?               computation-status
         +--ro computed-metric?      uint32
         +--ro computed-te-metric?   uint32
         +--ro computed-delay?       gr:delay
```

REST API

This section details how to use the Path Computation Service RPC that could be used to request a path computation from a source to a destination with given constraints over a given graph.

Get Constrained Path

Path Computation algorithms are accessible through the RPC described below:

URL: `restconf/operations/path-computation:get-constrained-path`

Method: `POST`

Content-Type: `application/json`

Request Body:

```
1  {
2     "input": {
```

(continues on next page)

(continued from previous page)

```
3           "graph-name": "example",
4           "source": 9,
5           "destination": 4,
6           "constraints": {
7              "address-family": "ipv4",
8              "te-metric": 250,
9              "bandwidth": 100000000,
10             "class-type": 0
11          },
12          "algorithm": "cspf"
13      }
14  }
```

@line 3: **graph-name** The *name* of the graph that must exist.

@line 4: **source** The *source* as vertex ID in the graph.

@line 5: **destination** - The *destination* as vertex ID in the graph.

@line 6: **constraints** - List of *Constraints*. Possible values are:

- *address-family* (ipv4, ipv6, sr-ipv4 and sr-ipv6) - default ipv4
- *te-metric* as integer value
- *bandwidth* (byte/sec) as integer value
- *class-type* for the bandwidth - default 0
- *delay* (micro-second) as integer value

@line 12: **algorithm** - *Type of Path Computation Algorithm* Valid options are `spf`, `cspf` and `samcra` - default `spf`.

Response Body:

```
1   {
2       "output": {
3           "computed-metric": 210,
4           "status": "completed",
5           "path-description": [
6                   {
7                      "ipv4": "10.194.77.143"
8                   },
9                   {
10                     "ipv4": "10.194.77.155"
11                  },
12                  {
13                     "ipv4": "10.194.77.161"
14                  }
15          ]
16      }
17  }
```

Troubleshooting

Debug message could be activated with:

```
log:set DEBUG org.opendaylight.algo
```

Then check log with `log:tail` command.

In particular, if answer is `failed` check that source and destination vertices are known in the graph and that constraints are not too huge. A good advice is to start first by relaxing some constraints to see if algorithm could find a valid path or not, and then re-enable constraints one by one to find which one could not be met. Logs will also provide information about constraints that are not met during the path computation.