Internet
Internet-Draft
Intended status: Informational
Expires: April 17, 2015

D. Yeung
Y. Qu
Cisco Systems
J. Zhang
D. Bogdanovic
Juniper Networks
K. Sreenivasa
Brocade Communications System
October 14, 2014

Yang Data Model for OSPF Protocol
draft-yeung-netmod-ospf-02

Abstract

This document defines a YANG data model that can be used to configure and manage OSPF.

Status of This Memo

Copyright Notice

Table of Contents

1. Overview

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

A core routing data model is defined in [I-D.ietf-netmod-routing-cfg], and it proposes a basis for the development of data models for routing protocols. The interface data model is defined in [RFC7223] and is used for referencing interface from the routing protocol. This document defines a YANG data model that can be used to configure and manage OSPF and it is an augment to the core routing data model.

This document defines a YANG data model that can be used to configure and manage OSPF. Both OSPFv2 [RFC2328] and OSPFv3 [RFC5340] are supported. In additional to the core OSPF protocol, features described in different separate OSPF RFCs are also supported. They includes demand circuit [RFC1793], traffic engineering [RFC3630], multiple address family [RFC5838], graceful restart [RFC3623] [RFC5187], NSSA [RFC3101] and sham link [RFC4577]. Those non-core features are made optional in the data model provided.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Design of Data Model

Although the basis of OSPF configuration elements like routers, areas and interfaces remains the same, the detailed configuration model varies among different vendors. Differences are observed in term of how protocol engine is tied to routing domain, how multiple protocol engines could be instantiated and configuration inheritance, just to name a few.

The goal of this document is to define a data model that is capable of representing these differences. There is very little information that is designated as "mandatory", providing freedom to vendors to adapt this data model to their product implementation.

2.1. Overview

The OSPF YANG module defined in this document has all the common building blocks for OSPF protocol.

The OSPF YANG module augments the routing/routing-instance/routing-protocols/routing-protocol path of the ietf-routing module.

```
   module: ospf
   +--rw routing
    +--rw routing-instance [name]
     +--rw routing-protocols
       +--rw routing-protocol [name]
          +--rw ospf
               .
               .
             +--rw all-instances-inherit {instance-inheritance}?
             |    .
             |    .
             +--rw instance* [routing-instance af]
                   .
                   .
                +--rw all-areas-inherit {area-inheritance}?
                |  .
                |  .
                +--rw area* [area-id]
                |  .
                |  .
                |  +--rw all-interfaces-inherit {interface-inheritance}?
                |  .
                |  .
                |  +--rw interface* [interface]
                |  .
                |  .
                +--rw topology* [name]
```

The ospf is intended to match to the vendor specific OSPF configuration construct which is identified by a local identifier 'name'. The field 'version' allows support for OSPFv2 and OSPFv3.

The ospf container includes one or more OSPF protocol engines, each encapsulated in the instance entity. Each instance includes information for the routing domain it is running on based on the [routing-instance af] specification. There is no default routing domain assumed by the data model. For example, to enable OSPF on the default IPv4 routing domain of the vendor, this model requires an explicit instance entity with the specification like ["default" "ipv4-unicast"]. The instance also contains OSPF router level configuration

The instance/area and instance/area/interface container contain the OSPF configuration for the area and interface level respectively

The instance/topology container contain the OSPF configuration for topology when multi-topology feature is enabled

2.2. OSPFv2 and OSPFv3

The defined data model supports both OSPFv2 and OSPFv3.

The field 'version' is used to indicate the OSPF version and is a mandatory. Based on the version set, the data model change accordingly to accommodate the difference between the two versions.

2.3. Optional Features

Optional features are features beyond the basic of OSPF configurations and it is up to a vendor to decide the support of a particular feature on a particular device.

This module has declared a number of features, such as NSR, max-LSA etc.. It is intended that vendors will extend the features list.

2.4. Inheritance

This defined data model supports configuration inheritance for intances, areas and interfaces.

The all-instances-inherit, all-areas-inherit and all-interfaces-inherit containers provides a consistent way to configure inheritable command. Inheritance is treated as a feature. Vendors are expected to augment the above container to provide the list of inheritance command for their implementation.

2.5. OSPF Router Configuration

The container ospf is the top level container in this data model. It contains shared information among different OSPF instances under the container.

```
module: ospf
   +--rw ospf
      +--rw all-instances-inherit {instance-inheritance}?
      |  +--rw area
      |  +--rw interface
      +--rw operation-mode?          identityref
      +--rw instance* [routing-instance af]
         .
         .
```

2.6. OSPF Instance Configuration

The container instance represents an OSPF protocol engine. Each instance indicates the routing domain it is associated with based on [routing-instance af] and contains the router level configurations.

The all-areas-inherit container contains area configuration that could be inherited to all OSPF areas defined. Similarly, the all-areas-inherit also contains interface configuration that could be inherited to all the OSPF interfaces defined.

```
   module: ospf

      +--rw ospf
           .
           .
         +--rw instance* [routing-instance af]
            +--rw routing-instance     rt:routing-instance-ref
            +--rw af                   identityref
            +--rw router-id?           yang:dotted-quad {router-id}?
            +--rw admin-distance
            |  +--rw (granularity)?
            |  |  +--:(detail)
            |  |  |  +--rw intra-area?   uint8
            |  |  |  +--rw inter-area?   uint8
            |  |  +--:(coarse)
            |  |     +--rw internal?     uint8
            |  +--rw external?     uint8
            +--rw nsr {nsr}?
            |  +--rw enable?   boolean
            +--rw graceful-restart {graceful-restart}?
            |  +--rw enable?                      boolean
            |  +--rw helper-enable?               boolean
            |  +--rw restart-interval?            uint16
            |  +--rw helper-strict-lsa-checking?   boolean
            +--rw protocol-shutdown {protocol-shutdown}?
            |  +--rw shutdown?   boolean
            +--rw auto-cost {auto-cost}?
            |  +--rw enable?                boolean
            |  +--rw reference-bandwidth?   uint32
            +--rw maximum
            |  +--rw paths?     uint16 {max-ecmp}?
            |  +--rw max-lsa?   uint32 {max-lsa}?
            +--rw mpls
            |  +--rw te-rid {te-rid}?
            |  |  +--rw (source)?
            |  |     +--:(interface)
            |  |     |  +--rw interface?   if:interface-ref
            |  |     +--:(explicit)
            |  |        +--rw router-id?   inet:ipv4-address
            |  +--rw ldp
            |     +--rw igp-sync?     boolean {ldp-igp-sync}?
            |     +--rw autoconfig?   boolean {ldp-igp-autoconfig}?
            +--rw all-areas-inherit {area-inheritance}?
            |  +--rw area
            |  +--rw interface
```

2.7. OSPF Area Configuration

The container area contains configurations of that area and the list of interface container represents all the OSPF interfaces active in the enclosing area.

```
module: ospf
 +--rw ospf
     .
     .
   +--rw instance* [routing-instance af]
       .
       .
     +--rw area* [area-id]
     |  +--rw area-id                  area-id-type
     |  +--rw area-type?               identityref
     |  +--rw summary?                 boolean
     |  +--rw default-cost?            uint32
     |  +--rw virtual-link* [router-id]
     |  |  +--rw router-id              yang:dotted-quad
     |  |  +--rw cost?                  uint16
     |  |  +--rw hello-interval?        uint16
     |  |  +--rw dead-interval?         uint16
     |  |  +--rw retransmit-interval?   uint16
     |  |  +--rw transmit-delay?        uint16
     |  |  +--rw mtu-ignore?            boolean {mtu-ignore}?
     |  |  +--rw lls?                   boolean {lls}?
     |  |  +--rw prefix-suppression?    boolean {prefix-suppression}?
     |  |  +--rw bfd?                   boolean {bfd}?
     |  |  +--rw ttl-security {ttl-security}?
     |  |  |  +--rw enable?   boolean
     |  |  |  +--rw hops?     uint8
     |  |  +--rw protocol-shutdown {protocol-if-shutdown}?
     |  |     +--rw shutdown?   boolean
     |  +--rw sham-link* [local-id remote-id]
     |  |  +--rw local-id               inet:ip-address
     |  |  +--rw remote-id              inet:ip-address
     |  |  +--rw cost?                  uint16
     |  |  +--rw hello-interval?        uint16
     |  |  +--rw dead-interval?         uint16
     |  |  +--rw retransmit-interval?   uint16
     |  |  +--rw transmit-delay?        uint16
     |  |  +--rw mtu-ignore?            boolean {mtu-ignore}?
     |  |  +--rw lls?                   boolean {lls}?
     |  |  +--rw prefix-suppression?    boolean {prefix-suppression}?
     |  |  +--rw bfd?                   boolean {bfd}?
     |  |  +--rw ttl-security {ttl-security}?
     |  |  |  +--rw enable?   boolean
```

```
        |  |  |  +--rw hops?     uint8
        |  |  +--rw protocol-shutdown {protocol-if-shutdown}?
        |  |     +--rw shutdown?   boolean
        |  +--rw range* [prefix]
        |  |  +--rw prefix       inet:ip-prefix
        |  |  +--rw advertise?   boolean
        |  |  +--rw cost?        uint24
        |  +--rw all-interfaces-inherit {interface-inheritance}?
        |  |  +--rw interface
```

2.8. OSPF Interface Configuration

The container interface contains configurations of that interface.

The ospf-interfaces also contain interface configuration that could be inherited to all ospf-interface’s defined.

```
   module: ospf
    +--rw ospf
        .
        .
      +--rw instance* [routing-instance af]
         .
         .
        +--rw area* [area-id]
           .
           .
        |  +--rw interface* [interface]
        |     +--rw interface              if:interface-ref
        |     +--rw network-type?          enumeration
        |     +--rw passive?               boolean
        |     +--rw demand-circuit?        boolean {demand-circuit}?
        |     +--rw multi-area {multi-area-adj}?
        |     |  +--rw multi-area-id?   area-id-type
        |     |  +--rw cost?            uint16
        |     +--rw static-neighbors
        |     |  +--rw neighbor* [address]
        |     |     +--rw address          inet:ip-address
        |     |     +--rw cost?            uint16
        |     |     +--rw poll-interval?   uint16
        |     |     +--rw priority?        uint8
        |     +--rw cost?                  uint16
        |     +--rw hello-interval?        uint16
        |     +--rw dead-interval?         uint16
        |     +--rw retransmit-interval?   uint16
        |     +--rw transmit-delay?        uint16
        |     +--rw mtu-ignore?            boolean {mtu-ignore}?
        |     +--rw lls?                   boolean {lls}?
        |     +--rw prefix-suppression?    boolean {prefix-suppression}?
        |     +--rw bfd?                   boolean {bfd}?
        |     +--rw ttl-security {ttl-security}?
        |     |  +--rw enable?   boolean
        |     |  +--rw hops?     uint8
        |     +--rw protocol-shutdown {protocol-if-shutdown}?
        |     |  +--rw shutdown?   boolean
        |     +--rw topology* [name]
        |        +--rw name    rt:rib-ref
        |        +--rw cost?   uint32
```

2.9. OSPF notification

This YANG model defines a list of notifications to inform client of important events detected during the protocol operation. The notifications defined cover the common set of traps from OSPFv2 MIB [RFC4750] and OSPFv3 MIB [RFC5643].

```
   module: ospf
   notifications:
      +---n if-state-change
      |  +--ro routing-instance?        rt:routing-instance-ref
      |  +--ro routing-protocol-name?   string
      |  +--ro instance-af
      |  |  +--ro af?   identityref
      |  +--ro link-type?               identityref
      |  +--ro interface
      |  |  +--ro interface?   if:interface-ref
      |  +--ro virtual-link
      |  |  +--ro area-id?               uint32
      |  |  +--ro neighbor-router-id?   yang:dotted-quad
      |  +--ro sham-link
      |  |  +--ro area-id?          uint32
      |  |  +--ro local-ip-addr?    inet:ip-address
      |  |  +--ro remote-ip-addr?   inet:ip-address
      |  +--ro state?                   if-state-type
      +---n if-config-error
      |  +--ro routing-instance?        rt:routing-instance-ref
      |  +--ro routing-protocol-name?   string
      |  +--ro instance-af
      |  |  +--ro af?   identityref
      |  +--ro link-type?               identityref
      |  +--ro interface
      |  |  +--ro interface?       if:interface-ref
      |  |  +--ro packet-source?   yang:dotted-quad
      |  +--ro virtual-link
      |  |  +--ro area-id?               uint32
      |  |  +--ro neighbor-router-id?   yang:dotted-quad
      |  +--ro sham-link
      |  |  +--ro area-id?          uint32
      |  |  +--ro local-ip-addr?    inet:ip-address
      |  |  +--ro remote-ip-addr?   inet:ip-address
      |  +--ro packet-type?             packet-type
      |  +--ro error?                   enumeration
      +---n nbr-state-change
      |  +--ro routing-instance?        rt:routing-instance-ref
      |  +--ro routing-protocol-name?   string
      |  +--ro instance-af
      |  |  +--ro af?   identityref
      |  +--ro link-type?               identityref
      |  +--ro interface
      |  |  +--ro interface?            if:interface-ref
      |  |  +--ro neighbor-router-id?   yang:dotted-quad
      |  |  +--ro neighbor-ip-addr?     yang:dotted-quad
      |  +--ro virtual-link
      |  |  +--ro area-id?               uint32
```

```
     |  |  +--ro neighbor-router-id?   yang:dotted-quad
     |  +--ro sham-link
     |  |  +--ro area-id?              uint32
     |  |  +--ro local-ip-addr?        inet:ip-address
     |  |  +--ro neighbor-router-id?   yang:dotted-quad
     |  |  +--ro neighbor-ip-addr?     yang:dotted-quad
     |  +--ro state?                   nbr-state-type
     +---n nbr-restart-helper-status-change
     |  +--ro routing-instance?        rt:routing-instance-ref
     |  +--ro routing-protocol-name?   string
     |  +--ro instance-af
     |  |  +--ro af?   identityref
     |  +--ro link-type?               identityref
     |  +--ro interface
     |  |  +--ro interface?            if:interface-ref
     |  |  +--ro neighbor-router-id?   yang:dotted-quad
     |  |  +--ro neighbor-ip-addr?     yang:dotted-quad
     |  +--ro virtual-link
     |  |  +--ro area-id?              uint32
     |  |  +--ro neighbor-router-id?   yang:dotted-quad
     |  +--ro status?                  restart-helper-status-type
     |  +--ro age?                     uint32
     |  +--ro exit-reason?             restart-exit-reason-type
     +---n rx-bad-packet
     |  +--ro routing-instance?        rt:routing-instance-ref
     |  +--ro routing-protocol-name?   string
     |  +--ro instance-af
     |  |  +--ro af?   identityref
     |  +--ro link-type?               identityref
     |  +--ro interface
     |  |  +--ro interface?       if:interface-ref
     |  |  +--ro packet-source?   yang:dotted-quad
     |  +--ro virtual-link
     |  |  +--ro area-id?              uint32
     |  |  +--ro neighbor-router-id?   yang:dotted-quad
     |  +--ro sham-link
     |  |  +--ro area-id?          uint32
     |  |  +--ro local-ip-addr?    inet:ip-address
     |  |  +--ro remote-ip-addr?   inet:ip-address
     |  +--ro packet-type?             packet-type
     +---n lsdb-approaching-overflow
     |  +--ro routing-instance?        rt:routing-instance-ref
     |  +--ro routing-protocol-name?   string
     |  +--ro instance-af
     |  |  +--ro af?   identityref
     |  +--ro ext-lsdb-limit?          uint32
     +---n lsdb-overflow
     |  +--ro routing-instance?        rt:routing-instance-ref
```

```
      |  +--ro routing-protocol-name?   string
      |  +--ro instance-af
      |  |  +--ro af?   identityref
      |  +--ro ext-lsdb-limit?          uint32
      +---n nssa-translator-status-change
      |  +--ro routing-instance?        rt:routing-instance-ref
      |  +--ro routing-protocol-name?   string
      |  +--ro instance-af
      |  |  +--ro af?   identityref
      |  +--ro area-id?                 uint32
      |  +--ro status?                  nssa-translator-state-type
      +---n restart-status-change
         +--ro routing-instance?        rt:routing-instance-ref
         +--ro routing-protocol-name?   string
         +--ro instance-af
         |  +--ro af?   identityref
         +--ro status?                  restart-status-type
         +--ro restart-interval?        uint16
         +--ro exit-reason?             restart-exit-reason-type
```

3. OSPF Yang Module

```
   <CODE BEGINS>
   module ospf {
     namespace "urn:ietf:params:xml:ns:yang:ospf";
     // replace with IANA namespace when assigned
     prefix ospf;

     import ietf-inet-types {
       prefix "inet";
     }

     import ietf-yang-types {
       prefix "yang";
     }

     import ietf-interfaces {
       prefix "if";
     }

     import ietf-routing {
       prefix "rt";
     }

     organization
       "Cisco Systems
        170 West Tasman Drive
        San Jose, CA 95134-1706
        USA";
     contact
       "Derek Yeung myeung@cisco.com
        Yingzhen Qu yiqu@cisco.com
        Dean Bogdanovic deanb@juniper.net
        Jeffrey Zhang zzhang@juniper.net
        Kiran Agrahara Sreenivasa kkoushik@Brocade.com";

     description
       "This YANG module defines the generic configuration
        data for OSPF, which is common across all of the vendor
        implementations of the protocol. It is intended that the module
        will be extended by vendors to define vendor-specific
        OSPF configuration parameters and policies,
        for example route maps or route policies.


        Terms and Acronyms

        OSPF (ospf): Open Shortest Path First

        IP (ip): Internet Protocol

        IPv4 (ipv4):Internet Protocol Version 4

        IPv6 (ipv6): Internet Protocol Version 6

        MTU (mtu) Maximum Transmission Unit
       ";

     revision 2014-09-17 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for OSPF";
     }

     identity ospfv2 {
       base "rt:routing-protocol";
       description "OSPFv2";
     }

     identity ospfv3 {
       base "rt:routing-protocol";
       description "OSPFv3";
     }

     identity operation-mode {
       description
         "OSPF operation mode.";
     }

     identity ships-in-the-night {
       base operation-mode;
       description
         "Ships-in-the-night operation mode in which
         each OSPF instance carries only one address family";
     }

     identity area-type {
       description "Base identity for OSPF area type.";
     }

     identity normal {
       base area-type;
       description "OSPF normal area.";
     }

     identity stub {
       base area-type;
       description "OSPF stub area.";
     }

     typedef uint24 {
       type uint32 {
         range "0 .. 16777215";
       }
       description
         "24-bit unsigned integer.";
     }

     typedef area-id-type {
       type union {
         type uint32;
         type yang:dotted-quad;
       }
       description
         "Area ID type.";
     }

     typedef if-state-type {
       type enumeration {
         enum Down {
           value "1";
           description
             "Interface down state";
         }
         enum Loopback {
           value "2";
           description
             "Interface loopback state";
         }
         enum Waiting {
           value "3";
           description
             "Interface waiting state";
         }
         enum Point-to-Point {
           value "4";
           description
             "Interface point-to-point state";
         }
         enum DR {
           value "5";
           description
             "Interface Designated Router (DR) state";
         }
         enum BDR {
           value "6";
           description
             "Interface Backup Designated Router (BDR) state";
         }
         enum DR-Other {
           value "7";
           description
             "Interface Other Designated Router state";
         }
       }
       description
         "OSPF interface state type.";
     }

     typedef nbr-state-type {
       type enumeration {
         enum Down {
           value "1";
           description
             "Neighbor down state";
         }
         enum Attempt {
           value "2";
           description
             "Neighbor attempt state";
         }
         enum Init {
           value "3";
           description
             "Neighbor init state";
         }
         enum 2-Way {
           value "4";
           description
             "Neighbor 2-Way state";
         }
         enum ExStart {
           value "5";
           description
             "Neighbor exchange start state";
         }
         enum Exchange {
           value "6";
           description
             "Neighbor exchange state";
         }
         enum Loading {
           value "7";
           description
             "Neighbor loading state";
         }
         enum Full {
           value "8";
           description
             "Neighbor full state";
         }
       }
       description
         "OSPF neighbor state type.";
     }

     typedef restart-helper-status-type {
       type enumeration {
         enum Not-Helping {
           value "1";
           description
             "Restart helper status not helping.";
         }
         enum Helping {
           value "2";
           description
             "Restart helper status helping.";
         }
       }
       description
         "Restart helper status type.";
     }

     typedef restart-exit-reason-type {
       type enumeration {
         enum None {
           value "1";
           description
             "Not attempted.";
         }
         enum InProgress {
           value "2";
           description
             "Restart in progress.";
         }
         enum Completed {
           value "3";
           description
             "Successfully completed.";
         }
         enum TimedOut {
           value "4";
           description
             "Timed out.";
         }
         enum TopologyChanged {
           value "5";
           description
             "Aborted due to topology change.";
         }
       }
       description
         "Describes the outcome of the last attempt at a
         graceful restart, either by itself or acting
         as a helper.";
     }

     typedef packet-type {
       type enumeration {
         enum Hello {
           value "1";
           description
             "OSPF hello packet.";
         }
         enum Database-Descripton {
           value "2";
           description
```

```
            "OSPF database description packet.";
        }
        enum Link-State-Request {
          value "3";
          description
            "OSPF link state request packet.";
        }
        enum Link-State-Update {
          value "4";
          description
            "OSPF link state update packet.";
        }
        enum Link-State-Ack {
          value "5";
          description
            "OSPF link state acknowlegement packet.";
        }
      }
      description
        "OSPF packet type.";
    }

    typedef nssa-translator-state-type {
      type enumeration {
        enum Enabled {
          value "1";
          description
            "NSSA translator enabled state.";
        }
        enum Elected {
          description
            "NSSA translator elected state.";
        }
        enum Disabled {
          value "3";
          description
            "NSSA translator disabled state.";
        }
      }
      description
        "OSPF NSSA translator state type.";
    }

    typedef restart-status-type {
      type enumeration {
        enum Not-Restarting {
          value "1";
          description
```

```
              "Router is not restarting.";
          }
          enum Planned-Restart {
            description
              "Router is going through planned restart.";
          }
          enum Unplanned-Restart {
            value "3";
            description
              "Router is going through unplanned restart.";
          }
        }
        description
          "OSPF graceful restart status type.";
      }

      feature multi-topology {
        description
          "Support MTR.";
      }

      feature multi-area-adj {
        description
          "OSPF multi-area adjacency support as in RFC 5185.";
      }

      feature router-id {
        description
          "Set router ID per instance.";
      }

      feature demand-circuit {
        description
          "OSPF demand circuit support as in RFC 1793.";
      }

      feature mtu-ignore {
        description
          "Disable OSPF MTU mismatch detection on receiving
           DBD packets.";
      }

      feature lls {
        description
          "OSPF link-local signaling (LLS) as in RFC 5613.";
      }

      feature prefix-suppression {
        description
          "OSPF prefix suppression support as in RFC 6860.";
      }

      feature bfd {
        description
          "OSPF BFD support.";
      }

      feature ttl-security {
        description
          "OSPF ttl security check.";
      }

      feature nsr {
        description
          "Non-Stop-Routing (NSR).";
      }

      feature graceful-restart {
        description
          "Graceful OSPF Restart as defined in RFC3623 and RFC5187.";
      }

      feature protocol-shutdown {
        description
          "Shutdown the protocol.";
      }

      feature auto-cost {
        description
          "Calculate OSPF interface cost according to
            reference bandwidth.";
      }

      feature max-ecmp {
        description
          "Setting maximum number of ECMP paths.";
      }

      feature max-lsa {
        description
          "Setting maximum number of LSAs OSPF will receive.";
      }

      feature te-rid {
        description
          "TE router-id.";
      }

      feature ldp-igp-sync {
        description
          "LDP IGP synchronization.";
      }

      feature ldp-igp-autoconfig {
        description
          "LDP IGP auto-config.";
      }

      feature protocol-if-shutdown {
        description
          "Shutdown the protocol over an interface.";
      }

      feature instance-inheritance {
        description
          "Support inheritance";
      }

      feature af-inheritance {
        description
          "Support inheritance";
      }

      feature area-inheritance {
        description
          "Support area inheritance";
      }

      feature interface-inheritance {
        description
          "Support interface inheritance";
      }

      grouping interface-common-config {
        description "Common configuration for all types of interfaces,
                     including virtual link and sham link";

        leaf cost {
          type uint16 {
            range "1..65535";
          }
          description
            "Interface cost.";
        }
        leaf hello-interval {
          type uint16 {
            range "1..65535";
          }
          units seconds;
          description
            "Time between hello packets.";
        }

        leaf dead-interval {
          type uint16 {
            range "1..65535";
          }
          units seconds;
          must "dead-interval > ../hello-interval" {
            error-message "The dead interval must be "
                        + "larger than the hello interval";
            description
              "The value MUST be greater than 'hello-internval'.";
          }
          description
            "Interval after which a neighbor is declared dead.";
        }

        leaf retransmit-interval {
          type uint16 {
            range "1..65535";
          }
          units seconds;
          description
            "Time between retransmitting unacknowledged Link State
             Advertisements (LSAs).";
        }

        leaf transmit-delay {
          type uint16 {
            range "1..65535";
          }
          units seconds;
          description
            "Estimated time needed to send link-state update.";
        }

        leaf mtu-ignore {
          if-feature mtu-ignore;
          type boolean;
          description
            "Enable/Disable ignoring of MTU in DBD packets.";
        }

        leaf lls {
          if-feature lls;
          type boolean;
          description
            "Enable/Disable link-local signaling (LLS) support.";
        }

        leaf prefix-suppression {
          if-feature prefix-suppression;
          type boolean;
          description
            "Suppress advertisement of the prefixes.";
        }

        leaf bfd {
          if-feature bfd;
          type boolean;
          description
            "Enable/disable bfd.";
        }

        container ttl-security {
          if-feature ttl-security;
          description "TTL security check.";
          leaf enable {
            type boolean;
            description
              "Enable/Disable TTL security check.";
          }
          leaf hops {
            type uint8 {
              range "1..254";
            }
            description
              "Maximum number of hops that a OSPF packet may
               have traveled.";
          }
        }
        container protocol-shutdown {
          if-feature protocol-if-shutdown;
          description
            "Protocol shutdown interface config state.";
          leaf shutdown {
            type boolean;
            description
              "Enable/Disable protocol shutdown on the interface.";
```

```
        }
      }
    } // interface-common-config

    grouping interface-config {
      description "Configuration for real interfaces.";

      leaf network-type {
        type enumeration {
          enum "broadcast" {
            description
              "Specify OSPF broadcast multi-access network.";
          }
          enum "non-broadcast" {
            description
              "Specify OSPF Non-Broadcast Multi-Access
              (NBMA) network.";
          }
          enum "point-to-multipoint" {
            description
              "Specify OSPF point-to-multipoint network.";
          }
          enum "point-to-point" {
            description
              "Specify OSPF point-to-point network.";
          }
        }
        description
          "Network type.";
      }

      leaf passive {
        type boolean;
        description
          "Enable/Disable passive.";
      }

      leaf demand-circuit {
        if-feature demand-circuit;
        type boolean;
        description
          "Enable/Disable demand circuit.";
      }

      container multi-area {
        if-feature multi-area-adj;
        description
          "Configure ospf multi-area.";
        leaf multi-area-id {
          type area-id-type;
          description
            "Multi-area ID";
        }
        leaf cost {
          type uint16;
          description
            "Interface cost for multi-area.";
        }
      }

      container static-neighbors {
        description "Static configured neighbors.";

        list neighbor {
          key "address";
          description
            "Specify a neighbor router.";

          leaf address {
            type inet:ip-address;
            description "Neighbor IP address.";
          }

          leaf cost {
            type uint16 {
              range "1..65535";
            }
            description "Neighbor cost.";
          }
          leaf poll-interval {
            type uint16 {
              range "1..65535";
            }
            units seconds;
            description "Neighbor poll interval.";
          }
          leaf priority {
            type uint8 {
              range "1..255";
            }
            description "Neighbor priority for DR election.";
          }
        }
      }

      uses interface-common-config;
```

```
     } // grouping interface-config

     grouping tlv {
       description
         "TLV";
       leaf type {
         type uint16;
         description "TLV type.";
       }
       leaf length {
         type uint16;
         description "TLV length.";
       }
       leaf value {
         type yang:hex-string;
         description "TLV value.";
       }
     }

     grouping ospfv2-lsa-body {
       description "OSPFv2 LSA body.";
       container router {
         when "../../header/type = 1" {
           description
             "Only apply to Router-LSA.";
         }
         description
           "Router LSA.";
         leaf flags {
           type bits {
             bit V {
               description
                 "When set, the router is an endpoint of one or
                 more virtual links.";
             }
             bit E {
               description
                 "When set, the router is an AS Boundary Router
                 (ASBR).";
             }
             bit B {
               description
                 "When set, the router is an Area Border Router (ABR).";
             }
           }
           description "Flags";
         }
         leaf num-of-links {
           type uint16;
           description "Number of links.";
         }
         list link {
           key "link-id link-data";
           description "Router LSA link.";
           leaf link-id {
             type union {
               type inet:ipv4-address;
               type yang:dotted-quad;
             }
             description "Link ID";
           }
           leaf link-data {
             type union {
               type inet:ipv4-address;
               type uint32;
             }
             description "Link data.";
           }
           leaf type {
             type uint8;
             description "Link type.";
           }
           list topology {
             key "mt-id";
             description
               "Topology specific information.";
             leaf mt-id {
               type uint8;
               description
                 "The MT-ID for topology enabled on the link.";
             }
             leaf metric {
               type uint16;
               description "Metric for the topology.";
             }
           }
         }
       }
       container network {
         when "../../header/type = 2" {
           description
             "Only apply to network LSA.";
         }
         description
           "Network LSA.";
         leaf network-mask {
           type inet:ipv4-address;
           description
             "The IP address mask for the network";
         }
         leaf-list attached-router {
           type yang:dotted-quad;
           description
             "List of the routers attached to the network.";
         }
       }
       container summary {
         when "../../header/type = 3 or "
            + "../../header/type = 4" {
           description
             "Only apply to Summary-LSA.";
         }
         description
           "Summary LSA.";
         leaf network-mask {
           type inet:ipv4-address;
           description
             "The IP address mask for the network";
         }
         list topology {
           key "mt-id";
           description
             "Topology specific information.";
           leaf mt-id {
             type uint8;
             description
               "The MT-ID for topology enabled on the link.";
           }
           leaf metric {
             type uint24;
             description "Metric for the topology.";
           }
         }
       }
       container external {
         when "../../header/type = 5 or "
            + "../../header/type = 7" {
           description
             "Only apply to AS-external-LSA and NSSA-LSA.";
         }
         description
           "External LSA.";
         leaf network-mask {
           type inet:ipv4-address;
           description
             "The IP address mask for the network";
         }
         list topology {
           key "mt-id";
           description
             "Topology specific information.";
           leaf mt-id {
             type uint8;
             description
               "The MT-ID for topology enabled on the link.";
           }
           leaf flags {
             type bits {
               bit E {
                 description
                   "When set, the metric specified is a Type 2
                   external metric.";
               }
             }
             description "Flags.";
           }
           leaf metric {
             type uint24;
             description "Metric for the topology.";
           }
           leaf forwarding-address {
             type inet:ipv4-address;
             description
               "Forwarding address.";
           }
           leaf external-route-tag {
             type uint32;
             description
               "Route tag.";
           }
         }
       }
       container opaque {
         when "../../header/type = 9 or "
            + "../../header/type = 10 or "
            + "../../header/type = 11" {
           description
             "Only apply to opaque LSA.";
         }
         description
           "Opaque LSA.";
         list unknown-tlv {
           key "type";
           description "Unknown TLV.";
           uses tlv;
         }

         container router-address-tlv {
           leaf router-address {
             type inet:ipv4-address;
             description
               "Router address.";
           }
           description
             "Router address TLV.";
         }

         container link-tlv {
           leaf link-type {
             type uint8;
             mandatory true;
             description "Link type.";
           }
           leaf link-id {
             type union {
               type inet:ipv4-address;
               type yang:dotted-quad;
             }
             mandatory true;
             description "Link ID.";
           }
           leaf-list local-if-ipv4-addr {
             type inet:ipv4-address;
             description
               "List of local interface IPv4 addresses.";
           }
           leaf-list local-remote-ipv4-addr {
             type inet:ipv4-address;
             description
               "List of remote interface IPv4 addresses.";
           }
           leaf te-metric {
             type uint32;
             description "TE metric.";
           }
           leaf max-bandwidth {
             type decimal64 {
               fraction-digits 2;
             }
             description "Maximum bandwidth.";
           }
           leaf max-reservable-bandwidth {
             type decimal64 {
               fraction-digits 2;
             }
             description "Maximum reservable bandwidth.";
           }
           leaf unreserved-bandwidth {
             type decimal64 {
               fraction-digits 2;
             }
             description "Unreserved bandwidth.";
           }
           leaf admin-group {
             type uint32;
             description "Administrative group/Resource class/Color.";
           }
           list unknown-subtlv {
             key "type";
             description "Unknown sub-TLV.";
             uses tlv;
           }
           description
             "Link TLV.";
         }
       }
     }

     grouping ospfv3-lsa-options {
       description "OSPFv3 LSA options";
       leaf options {
         type bits {
           bit DC {
             description
               "When set, the router support demand circuits.";
           }
           bit R {
             description
               "When set, the originator is an active router.";
           }
           bit N {
             description
               "If set, the router is attached to an NSSA";
           }
           bit E {
             description
               "This bit describes the way AS-external-LSAs
```

```
                are flooded";
            }
            bit V6 {
              description
                "If clear, the router/link should be excluded
                 from IPv6 routing calculaton";
            }
          }
          mandatory true;
          description "OSPFv3 LSA options.";
        }
      }

      grouping ospfv3-lsa-prefix {
        description
          "OSPFv3 LSA prefix.";

        leaf prefix {
          type inet:ip-prefix;
          description
            "Prefix";
        }
        leaf prefix-options {
          type bits {
            bit NU {
              description
                "When set, the prefix should be excluded
                from IPv6 unicast calculations.";
            }
            bit LA {
              description
                "When set, the prefix is actually an IPv6 interface
                address of the Advertising Router.";
            }
            bit P {
              description
                "When set, the NSSA area prefix should be
                readvertised by the translating NSSA area border.";
            }
            bit DN {
              description
                "When set, the inter-area-prefix-LSA or
                AS-external-LSA prefix has been advertised in a VPN
                environment.";
            }
          }
          mandatory true;
          description "Prefix options.";
```

```
       }
     }

     grouping ospfv3-lsa-external {
       description
         "AS-External and NSSA LSA.";
       leaf metric {
         type uint24;
         description "Metric";
       }

       leaf flags {
         type bits {
           bit E {
             description
               "When set, the metric specified is a Type 2
               external metric.";
           }
         }
         description "Flags.";
       }

       leaf referenced-ls-type {
         type uint16;
         description "Referenced Link State type.";
       }

       uses ospfv3-lsa-prefix;

       leaf forwarding-address {
         type inet:ipv6-address;
         description
           "Forwarding address.";
       }

       leaf external-route-tag {
         type uint32;
         description
           "Route tag.";
       }
       leaf referenced-link-state-id {
         type uint32;
         description
           "Referenced Link State ID.";
       }
     }

     grouping ospfv3-lsa-body {
       description "OSPFv3 LSA body.";
       container router {
         when "../../header/type = 8193" { // 0x2001
           description
             "Only apply to Router-LSA.";
         }
         description "Router LSA.";
         leaf flags {
           type bits {
             bit V {
               description
                 "When set, the router is an endpoint of one or
                 more virtual links.";
             }
             bit E {
               description
                 "When set, the router is an AS Boundary Router
                 (ASBR).";
             }
             bit B {
               description
                 "When set, the router is an Area Border Router (ABR).";
             }
             bit Nt {
               description
                 "When set, the router is an NSSA border router
                 that is unconditionally translating NSSA-LSAs
                 into AS-external-LSAs.";
             }
           }
           mandatory true;
           description "LSA option.";
         }

         uses ospfv3-lsa-options;

         list link {
           key "interface-id neighbor-interface-id neighbor-router-id";
           description "Router LSA link.";
           leaf interface-id {
             type uint32;
             description "Interface ID.";
           }
           leaf neighbor-interface-id {
             type uint32;
             description "Neighbor Interface ID.";
           }
           leaf neighbor-router-id {
             type yang:dotted-quad;
             description "Neighbor Router ID";
           }
           leaf type {
             type uint8;
             description "Link type.";
           }
           leaf metric {
             type uint16;
               description "Metric.";
           }
         }
       }
       container network {
         when "../../header/type = 8194" { // 0x2002
           description
             "Only apply to network LSA.";
         }
         description "Network LSA.";

         uses ospfv3-lsa-options;

         leaf-list attached-router {
           type yang:dotted-quad;
           description
             "List of the routers attached to the network.";
         }
       }
       container inter-area-prefix {
         when "../../header/type = 8195" { // 0x2003
           description
             "Only apply to inter-area-prefix LSA.";
         }
         leaf metric {
           type uint24;
           description "Metric";
         }

         uses ospfv3-lsa-prefix;
         description "Inter-Area-Prefix LSA.";
       }
       container inter-area-router {
         when "../../header/type = 8196" { // 0x2004
           description
             "Only apply to inter-area-router LSA.";
         }
         uses ospfv3-lsa-options;
         leaf metric {
           type uint24;
           description "Metric";
         }
         leaf destination-router-id {
           type yang:dotted-quad;
           description
             "The Router ID of the router being described by the LSA.";
         }
         description "Inter-Area-Router LSA.";
       }
       container as-external {
         when "../../header/type = 16389" { // 0x2007
           description
             "Only apply to as-external LSA.";
         }

         uses ospfv3-lsa-external;

         description "AS-External LSA.";
       }
       container nssa {
         when "../../header/type = 8199" { // 0x2007
           description
             "Only apply to nssa LSA.";
         }
         uses ospfv3-lsa-external;

         description "NSSA LSA.";
       }
       container link {
         when "../../header/type = 8" { // 0x0008
           description
             "Only apply to link LSA.";
         }
         leaf rtr-priority {
           type uint8;
           description "Router Priority of the interface.";
         }

         uses ospfv3-lsa-options;

         leaf link-local-interface-address {
           type inet:ipv6-address;
           description
             "The originating router’s link-local
             interface address on the link.";
         }
         leaf num-of-prefixes {
           type uint32;
           description "Number of prefixes.";
         }

         list prefix {
           key "prefix";
           description "List of prefixes associated with the link.";
           uses ospfv3-lsa-prefix;
         }
         description "Link LSA.";
       }
       container intra-area-prefix {
         when "../../header/type = 8201" { // 0x2009
           description
             "Only apply to intra-area-prefix LSA.";
         }
         description "Intra-Area-Prefix LSA.";

         leaf referenced-ls-type {
           type uint16;
           description "Referenced Link State type.";
         }
         leaf referenced-link-state-id {
           type uint32;
           description
             "Referenced Link State ID.";
         }
         leaf referenced-adv-router {
           type inet:ipv4-address;
           description
             "Referenced Advertising Router.";
         }

         leaf num-of-prefixes {
           type uint16;
           description "Number of prefixes.";
         }
         list prefix {
           key "prefix";
           description "List of prefixes associated with the link.";
           uses ospfv3-lsa-prefix;
           leaf metric {
             type uint24;
             description "Metric";
           }
         }
       }
```

```
     }

     grouping lsa-header {
       description
           "Common LSA for OSPFv2 and OSPFv3";
       leaf age {
         type uint16;
         mandatory true;
         description "LSA age.";
       }
       leaf type {
         type uint16;
         mandatory true;
         description "LSA type.";
       }
       leaf adv-router {
         type yang:dotted-quad;
         mandatory true;
         description "LSA advertising router.";
       }
       leaf seq-num {
         type uint32;
         mandatory true;
         description "LSA sequence number.";
       }
       leaf checksum {
         type uint16;
         mandatory true;
         description "LSA checksum.";
       }
       leaf length {
         type uint16;
         mandatory true;
         description "LSA length.";
       }
     }

     grouping ospfv2-lsa {
       description
           "OSPFv2 LSA.";
       container header {
         description
           "Decoded OSPFv2 LSA header data.";
         leaf option {
           type bits {
             bit DC {
               description
                 "When set, the router support demand circuits.";
             }
             bit P {
               description
                 "Only used in type-7 LSA. When set, the NSSA
                 border router should translate the type-7 LSA
                 to type-5 LSA.";
             }
             bit MC {
               description
                 "When set, the router support MOSPF.";
             }
             bit E {
               description
                 "This bit describes the way AS-external-LSAs
                 are flooded";
             }
           }
           mandatory true;
           description "LSA option.";
         }
         leaf lsa-id {
           type inet:ipv4-address;
           mandatory true;
           description "LSA ID.";
         }

         leaf opaque-type {
           when "../../header/type = 9 or "
              + "../../header/type = 10 or "
              + "../../header/type = 11" {
             description
               "Only apply to opaque LSA.";
           }
           type uint8;
           mandatory true;
           description "Opaque type.";
         }

         leaf opaque-id {
           when "../../header/type = 9 or "
              + "../../header/type = 10 or "
              + "../../header/type = 11" {
             description
               "Only apply to opaque LSA.";
           }
           type uint24;
           mandatory true;
           description "Opaque id.";
         }
         uses lsa-header;
       }
       container body {
         description
           "Decoded OSPFv2 LSA body data.";
         uses ospfv2-lsa-body;
       }
     }

     grouping ospfv3-lsa {
       description
           "Decoded OSPFv3 LSA.";
       container header {
         description
           "Decoded OSPFv3 LSA header data.";
         leaf lsa-id {
           type uint32;
           mandatory true;
           description "LSA ID.";
         }
         uses lsa-header;
       }
       container body {
         description
           "Decoded OSPF LSA body data.";
         uses ospfv3-lsa-body;
       }
     }

     grouping lsa {
       description
           "OSPF LSA.";
       leaf decoded-completed {
         type boolean;
         description
           "The OSPF LSA body is fully decoded.";
       }
       leaf raw-data {
         type yang:hex-string;
         description
           "The complete LSA in network byte
           order as received/sent over the wire.";
       }
       choice version {
         description
           "OSPFv2 or OSPFv3 LSA body.";
         container ospfv2 {
           when "../../../rt:type = 'ospfv2'" {
             description "Applied to OSPFv2 only";
           }
           description "OSPFv2 LSA";
           uses ospfv2-lsa;
         }
         container ospfv3 {
           when "../../../rt:type = 'ospfv3'" {
             description "Applied to OSPFv3 only";
           }
           description "OSPFv3 LSA";
           uses ospfv3-lsa;
         }
       }
     }

     grouping lsa-key {
       description
         "OSPF LSA key.";
       leaf lsa-id {
         type union {
           type inet:ipv4-address;
           type uint32;
         }
         description
           "LSA ID.";
       }
       leaf adv-router {
         type inet:ipv4-address;
         description
           "Advertising router.";
       }
     }

     grouping af-area-config {
       description
         "OSPF address-family specific area config state.";
       list range {
         key "prefix";
         description
           "Summarize routes matching address/mask (border
            routers only)";
         leaf prefix {
           type inet:ip-prefix;
           description
             "IPv4 or IPv6 prefix";
         }
         leaf advertise {
           type boolean;
           description
             "Advertise or hide.";
         }
         leaf cost {
           type uint24 {
             range "0..16777214";
           }
           description
             "Cost of summary route.";
         }
       }
     }

     grouping area-config {
       description
         "OSPF area config state.";
       leaf area-type {
         type identityref {
           base area-type;
         }
         default normal;
         description
           "Area type.";
       }

       leaf summary {
         when "area-type = ’stub’ or area-type = ’nssa’" {
           description
             "Summary generation valid for stub/NSSA area.";
         }
         type boolean;
         description
           "Enable/Disable summary generation to the stub or
           NSSA area.";
       }

       leaf default-cost {
         when "area-type = ’stub’ or area-type = ’nssa’" {
             description
               "Default cost for LSA advertised into stub or
               NSSA area.";
         }
         type uint32 {
           range "1..16777215";
         }
         description
           "Set the summary default-cost for a stub or NSSA area.";
       }

       list virtual-link {
         key "router-id";
         description
           "OSPF virtual link";
         leaf router-id {
           type yang:dotted-quad;
           description
             "Virtual link router ID.";
         }

         uses interface-common-config;
       }

       list sham-link {
         key "local-id remote-id";
         description
           "OSPF sham link";
         leaf local-id {
           type inet:ip-address;
           description
             "Address of the local end-point.";
         }
         leaf remote-id {
           type inet:ip-address;
           description
             "Address of the remote end-point.";
         }
         uses interface-common-config;
       }

       uses af-area-config {
         when "../../operation-mode = 'ospf:ships-in-the-night'" {
           description
             "Ships in the night configuration.";
         }
       }
     }

     grouping instance-config {
       description
         "OSPF instance config state.";
       leaf router-id {
         if-feature router-id;
         type yang:dotted-quad;
         description
           "Defined in RFC 2328. A 32-bit number
```

```
            that uniquely identifies the router.";
       }

       container admin-distance {
         description "Admin distance config state.";
         choice granularity {
           description
             "Options for expressing admin distance
              for intra-area and inter-area route";
           case detail {
             leaf intra-area {
               type uint8;
               description
                 "Admin distance for intra-area route.";
             }
             leaf inter-area {
               type uint8;
               description
                 "Admin distance for inter-area route.";
             }
           }
           case coarse {
             leaf internal {
               type uint8;
               description
                 "Admin distance for both intra-area and
                  inter-area route.";
             }
           }
         }
         leaf external {
           type uint8;
           description
             "Admin distance for both external route.";
         }
       }

       container nsr {
         if-feature nsr;
         description
           "NSR config state.";
         leaf enable {
           type boolean;
           description
             "Enable/Disable NSR.";
         }
       }
       container graceful-restart {
         if-feature graceful-restart;
         description
           "Graceful restart config state.";
         leaf enable {
           type boolean;
           description
             "Enable/Disable graceful restart as defined in RFC 3623.";
         }
         leaf helper-enable {
           type boolean;
           description
             "Enable RestartHelperSupport in RFC 3623 Section B.2.";
         }
         leaf restart-interval {
           type uint16 {
             range "1..1800";  // Range is defined in RFC 3623.
           }
           units seconds;
           default "120";  // Default is defined in RFC 3623.
           description
             "RestartInterval option in RFC 3623 Section B.1.";
         }
         leaf helper-strict-lsa-checking {
           type boolean;
           description
             "RestartHelperStrictLSAChecking option in RFC 3623
             Section B.2.";
         }
       }

       container protocol-shutdown {
         if-feature protocol-shutdown;
         description
           "Protocol shutdown config state.";
         leaf shutdown {
           type boolean;
           description
             "Enable/Disable protocol shutdown.";
         }
       }

       container auto-cost {
         if-feature auto-cost;
         description
           "Auto cost config state.";
         leaf enable {
           type boolean;
```

```
            description
              "Enable/Disable auto cost.";
          }
          leaf reference-bandwidth {
            type uint32 {
              range "1..4294967";
            }
            units Mbits;
            description
              "Configure reference bandwidth in term of Mbits";
          }
        }

        container maximum {
          description
            "OSPF limits settings.";
          leaf paths {
            if-feature max-ecmp;
            type uint16 {
              range "1..32";
            }
            description
              "Maximum number of ECMP paths.";
          }
          leaf max-lsa {
            if-feature max-lsa;
            type uint32 {
              range "1..4294967294";
            }
            description
              "Maximum number of LSAs OSPF will receive.";
          }
        }

        container mpls {
          description
            "OSPF MPLS config state.";
          container te-rid {
            if-feature te-rid;
            description
              "Traffic Engineering stable IP address for system.";
            choice source {
              description
                "Different options for specifying TE router ID.";
              case interface {
                leaf interface {
                  type if:interface-ref;
                  description
```

```
                    "Take the interface’s IPv4 address as TE router ID.";
                }
              }
              case explicit {
                leaf router-id {
                  type inet:ipv4-address;
                  description
                    "Explicitly configure the TE router ID.";
                }
              }
            }
          }
          container ldp {
            description
              "OSPF MPLS LDP config state.";
            leaf igp-sync {
              if-feature ldp-igp-sync;
              type boolean;
              description
                "Enable LDP IGP synchronization.";
            }
            leaf autoconfig {
              if-feature ldp-igp-autoconfig;
              type boolean;
              description
                "Enable LDP IGP interface auto-configuration.";
            }
          }
        }
      }


      grouping interface-operation {
        description
          "OSPF interface operation state.";
        reference "RFC2328 Section 9";
        uses interface-config;

        leaf state {
          type if-state-type;
          description "Interface state.";
        }

        leaf hello-timer {
          type uint32;
          units "milliseconds";
          description "Hello timer.";
        }
```

```
       leaf wait-timer {
         type uint32;
         units "milliseconds";
         description "Wait timer.";
       }

       list neighbor {
         description
           "List of neighbors.";
         leaf neighbor-id {
           type leafref {
             path "../../neighbor/neighbor-id";
           }
           description "Neighbor.";
         }
       }

       leaf dr {
         type inet:ipv4-address;
         description "DR.";
       }

       leaf bdr {
         type inet:ipv4-address;
         description "BDR.";
       }
     } // interface-operation

     grouping neighbor-operation {
       description
         "OSPF neighbor operation data.";

       leaf address {
         type inet:ip-address;
         description
           "Neighbor address.";
       }
       leaf dr {
         type inet:ipv4-address;
         description
           "Designated Router.";
       }
       leaf bdr {
         type inet:ipv4-address;
         description
           "Backup Designated Router.";
       }
       leaf state {
         type nbr-state-type;
         description
           "OSPF neighbor state.";
       }
     }

     grouping instance-operation {
       description
         "OSPF Address Family operation state.";
       leaf router-id {
         type yang:dotted-quad;
         description
           "Defined in RFC 2328. A 32-bit number
            that uniquely identifies the router.";
       }
     }

     augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
           + "rt:routing-protocol" {
       when "rt:type = ’ospf:ospfv2’ or rt:type = ’ospf:ospfv3’" {
         description
           "This augment is only valid for a routing protocol instance
            of OSPF (type ’ospfv2’ or ’ospfv3’).";
       }
       description "OSPF augmentation.";

       container ospf {
         description
           "OSPF.";

         container all-instances-inherit {
           if-feature instance-inheritance;
           description
             "Inheritance support to all instances.";
           container area {
             description
               "Area config to be inherited by all areas in
               all instances.";
           }
           container interface {
             description
               "Interface config to be inherited by all interfaces
               in all instances.";
           }
         }

         leaf operation-mode {
           type identityref {
             base operation-mode;
           }
           default ospf:ships-in-the-night;
           description
             "OSPF operation mode.";
         }

         list instance {
           key "routing-instance af";
           description
             "An OSPF routing protocol instance.";
           leaf routing-instance {
             type rt:routing-instance-ref;
             description
               "For protocol centric model, which is supported in
               default-instance only, this could reference any layer 3
               routing-instance.
               For routing-instance centric model, must reference the
               enclosing routing-instance.";
           }

           leaf af {
             type identityref {
               base rt:address-family;
             }
             description
               "Address-family of the instance.";
           }

           uses instance-config;

           container all-areas-inherit {
             if-feature area-inheritance;
             description
               "Inheritance for all areas.";
             container area {
               description
                 "Area config to be inherited by all areas.";
             }
             container interface {
               description
                 "Interface config to be inherited by all interfaces
                 in all areas.";
             }
           }

           list area {
             key "area-id";
```

```
             description
               "List of ospf areas";
             leaf area-id {
               type area-id-type;
               description
                 "Area ID.";
             }

             uses area-config;

             container all-interfaces-inherit {
               if-feature interface-inheritance;
               description
                 "Inheritance for all interfaces";
               container interface {
                 description
                   "Interface config to be inherited by all
                   interfaces.";
               }
             }

             list interface {
               key "interface";
               description
                 "List of OSPF interfaces.";
               leaf interface {
                 type if:interface-ref;
                 description
                   "Interface.";
               }
               uses interface-config;
             } // list of interfaces
           } // list of areas
         } // list of instance
       } // container ospf
     }


     augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
           + "rt:routing-protocol/ospf:ospf/ospf:instance" {
       when "../rt:type = 'ospf:ospfv2' or ../rt:type = 'ospf:ospfv3'" {
         description
           "This augment is only valid for OSPF
           (type 'ospfv2' or 'ospfv3').";
       }
       if-feature multi-topology;
       description
         "OSPF multi-topology routing-protocol augmentation.";
       list topology {
         // Topology must be in the same routing-instance
         //  and of same AF as the container.
         key "name";
         description "OSPF topology.";
         leaf name {
           type rt:rib-ref;
           description "RIB";
         }
         list area {
           key "area-id";
           description
             "List of ospf areas";
           leaf area-id {
             type area-id-type;
             description
               "Area ID.";
           }
           uses area-config;
         }
       }
     }

     augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
           + "rt:routing-protocol/ospf:ospf/ospf:instance/"
           + "ospf:area/ospf:interface" {
       when "../../rt:type = 'ospf:ospfv2'" {
         description
           "This augment is only valid for OSPFv2.";
       }
       if-feature ospf:multi-topology;
       description "OSPF multi-topology interface augmentation.";
       list topology {
         key "name";
         description "OSPF interface topology.";
         leaf name {
           type rt:rib-ref;
           description
             "One of the topology enabled on this interface";
         }
         leaf cost {
            type uint32;
            description
              "Interface cost for this topology";
         }
       }
     }
     augment "/rt:routing-state/rt:routing-instance/"
           + "rt:routing-protocols/rt:routing-protocol" {
       when "rt:type = 'ospf:ospfv2' or rt:type = 'ospf:ospfv3'"  {
         description
           "This augment is only valid for a routing protocol instance
            of type 'ospfv2' or 'ospfv3'.";
       }
       description
            "OSPF configuration.";
       container ospf {
         description "OSPF";

         leaf operation-mode {
           type identityref {
             base operation-mode;
           }
           description
             "OSPF operation mode.";
         }

         list instance {
           key "routing-instance af";
           description
             "An OSPF routing protocol instance.";
           leaf routing-instance {
             type rt:routing-instance-ref;
             description
               "For protocol centric model, which is supported in
               default-instance only, this could reference any layer 3
               routing-instance.
               For routing-instance centric model, must reference the
               enclosing routing-instance.";
           }

           leaf af {
             type identityref {
               base rt:address-family;
             }
             description
               "Address-family of the instance.";
           }

           uses instance-operation;

           list neighbor {
             key "area-id interface neighbor-id";
             description
               "List of OSPF neighbors.";
             leaf area-id {
               type area-id-type;
               description
                 "Area ID.";
             }
             leaf interface {
               // Should it refer to config state leaf?
               type if:interface-ref;
               description
                 "Interface.";
             }
             leaf neighbor-id {
               type inet:ipv4-address;
               description
                 "Neighbor ID.";
             }

             uses neighbor-operation;
           } // list of OSPF neighbors

           list interface {
             key "area-id interface";
             description
               "List of OSPF interfaces.";

             leaf area-id {
               type area-id-type;
               description "Area ID.";
             }
             leaf interface {
               // Should it refer to config state leaf?
               type if:interface-ref;
               description "Interface.";
             }

             uses interface-operation;
           } // list of OSPF interfaces

           list area {
             key "area-id";
             description "List of OSPF areas";
             leaf area-id {
               type area-id-type;
               description "Area ID.";
             }
           } // list of OSPF areas

           container databases {
             description
               "OSPF databases.";
             list link-scope-lsas {
               when "../../../rt:type = 'ospfv3'" {
                 description
                   "Link scope LSA only exists in OSPFv3.";
               }
               key "area-id interface lsa-type";
               description "List OSPF link scope LSA databases";
               leaf area-id {
                 type uint32; // Should it refer to config state leaf?
                 description "Area ID.";
               }
               leaf interface {
                 // Should it refer to config state leaf?
                 type if:interface-ref;
                 description "Interface.";
               }
               leaf lsa-type {
                 type uint8;
                 description "OSPF link scope LSA type.";
               }
               list link-scope-lsa {
                 key "lsa-id adv-router";
                 description "List of OSPF link scope LSAs";
                 uses lsa-key;
                 uses lsa;
               }
             } // list link-scope-lsas

             list area-scope-lsas {
               key "area-id lsa-type";
               description "List OSPF area scope LSA databases";
               leaf lsa-type {
                 type uint8;
                 description "OSPF area scope LSA type.";
               }
               leaf area-id {
                 type uint32; // Should it refer to config state leaf?
                 description "Area ID.";
               }
               list area-scope-lsa {
                 key "lsa-id adv-router";
                 description "List of OSPF area scope LSAs";
                 uses lsa-key;
                 uses lsa;
               }
             } // list area-scope-lsas
             list as-scope-lsas {
               key "lsa-type";
               description "List OSPF AS scope LSA databases";
               leaf lsa-type {
                 type uint8;
                 description "OSPF AS scope LSA type.";
               }
               list as-scope-lsa {
                 key "lsa-id adv-router";
                 description "List of OSPF AS scope LSAs";
                 uses lsa-key;
                 uses lsa;
               }
             } // list as-scope-lsas
           } // container databases
         }
       } // container ospf
     }

     augment "/rt:routing-state/rt:routing-instance/"
           + "rt:routing-protocols/rt:routing-protocol/"
           + "ospf:ospf/ospf:instance" {
       when "../rt:type = ’ospf:ospfv2’" {
         description
           "This augment is only valid for OSPFv2.";
       }
       if-feature multi-topology;
       description
         "OSPF multi-topology routing-protocol augmentation.";
       list topology {
         // Topology must be in the same routing-instance
         // and of same AF as the container.
         key "name";
         description "OSPF topology.";
         leaf name {
           type rt:rib-ref;
           description "RIB";
         }
         list area {
           key "area-id";
           description
             "List of ospf areas";
           leaf area-id {
             type area-id-type;
             description
               "Area ID.";
           }
         }
       }
     }

     augment "/rt:routing-state/rt:routing-instance/"
           + "rt:routing-protocols/rt:routing-protocol/"
           + "ospf:ospf/ospf:instance/ospf:interface" {
       when "../../rt:type = 'ospf:ospfv2'" {
         description
           "This augment is only valid for OSPFv2.";
       }
       if-feature ospf:multi-topology;
       description "OSPF multi-topology interface augmentation.";
       list topology {
         key "name";
         description "OSPF interface topology.";
         leaf name {
           type rt:rib-ref;
           description
             "One of the topology enabled on this interface";
         }
       }
     }

     grouping route-content {
       description
         "This grouping defines OSPF-specific route attributes.";
       leaf metric {
         type uint32;
         description "OSPF route metric.";
       }
       leaf tag {
         type uint32;
         default "0";
         description "OSPF route tag.";
       }
       leaf route-type {
         type enumeration {
           enum intra-area {
             description "OSPF intra-area route";
           }
           enum inter-area {
             description "OSPF inter-area route";
           }
           enum external-1 {
             description "OSPF external route type 1";
           }
           enum external-2 {
             description "OSPF External route type 2";
           }
           enum nssa-1 {
             description "OSPF NSSA external route type 1";
           }
           enum nssa-2 {
             description "OSPF NSSA external route type 2";
           }
         }
         description "OSPF route type";
       }
     }

     augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
       when "rt:source-protocol = 'ospf:ospfv2' or "
          + "rt:source-protocol = 'ospf:ospfv3'" {
         description
           "This augment is only valid for a routes whose source
            protocol is OSPF.";
         }
        description
          "OSPF-specific route attributes.";
        uses route-content;
     }

     augment "/rt:active-route/rt:output/rt:route" {
       description
         "OSPF-specific route attributes in the output of 'active-route'
         RPC.";
       uses route-content;
     }

     identity if-link-type {
       description "Base identity for OSPF interface link type.";
     }

     identity if-link-type-normal {
       base if-link-type;
       description "OSPF interface link type normal.";
     }

     identity if-link-type-virtual-link {
       base if-link-type;
       description "OSPF interface link type virtual link.";
     }

     identity if-link-type-sham-link {
       base if-link-type;
       description "OSPF interface link type sham link.";
     }

     grouping notification-instance-hdr {
       description
        "This group describes common instance specific
         data for notifications.";

       leaf routing-instance {
         type rt:routing-instance-ref;
         description
           "Describe the routing instance.";
       }

       leaf routing-protocol-name {
         type string;
         description
          "Describes the name of the OSPF routing protocol.";
       }

       container instance-af {
         leaf af {
           type identityref {
             base rt:address-family;
           }
           description
             "Address-family of the instance.";
         }
         description
          "Describes the address family of the OSPF instance.";
       }
     }

     notification if-state-change {
       uses notification-instance-hdr;

       leaf link-type {
         type identityref {
           base if-link-type;
         }
         description "Type of OSPF interface.";
       }

       container interface {
         description "Normal interface.";
         leaf interface {
           type if:interface-ref;
           description "Interface.";
         }
       }
       container virtual-link {
         description "virtual-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
           description "Neighbor router id.";
         }
       }
       container sham-link {
         description "sham-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf local-ip-addr {
           type inet:ip-address;
           description "Sham link local address.";
         }

         leaf remote-ip-addr {
           type inet:ip-address;
           description "Sham link remote address.";
         }
       }

       leaf state {
         type if-state-type;
         description "Interface state.";
       }

       description
         "This notification is sent when interface
         state change is detected.";
     }

     notification if-config-error {
       uses notification-instance-hdr;

       leaf link-type {
         type identityref {
           base if-link-type;
         }
         description "Type of OSPF interface.";
       }
       container interface {
         description "Normal interface.";
         leaf interface {
           type if:interface-ref;
           description "Interface.";
         }
         leaf packet-source {
           type yang:dotted-quad;
           description "Source address.";
         }
       }
       container virtual-link {
         description "virtual-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
           description "Neighbor router id.";
         }
       }

       container sham-link {
         description "sham-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf local-ip-addr {
           type inet:ip-address;
           description "Sham link local address.";
         }

         leaf remote-ip-addr {
           type inet:ip-address;
           description "Sham link remote address.";
         }
       }

       leaf packet-type {
         type packet-type;
         description "OSPF packet type.";
       }

       leaf error {
         type enumeration {
           enum "badVersion" {
             description "Bad version";
           }
           enum "areaMismatch" {
             description "Area mistmatch";
           }
           enum "unknownNbmaNbr" {
             description "Unknown NBMA neighbor";
           }
           enum "unknownVirtualNbr" {
             description "Unknown virtual link neighbor";
           }
           enum "authTypeMismatch" {
             description "Auth type mismatch";
           }
           enum "authFailure" {
             description "Auth failure";
           }
           enum "netMaskMismatch" {
             description "Network mask mismatch";
           }
           enum "helloIntervalMismatch" {
             description "Hello interval mismatch";
           }
           enum "deadIntervalMismatch" {
             description "Dead interval mismatch";
           }
           enum "optionMismatch" {
             description "Option mismatch";
           }
           enum "mtuMismatch" {
             description "MTU mismatch";
           }
           enum "duplicateRouterId" {
             description "Duplicate router ID";
           }
           enum "noError" {
             description "No error";
           }
         }
         description "Error code.";
       }
       description
         "This notification is sent when interface
         config error is detected.";
     }

     notification nbr-state-change {
       uses notification-instance-hdr;
       leaf link-type {
         type identityref {
           base if-link-type;
         }
         description "Type of OSPF interface.";
       }

       container interface {
         description "Normal interface.";
         leaf interface {
           type if:interface-ref;
           description "Interface.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
           description "Neighbor router id.";
         }
         leaf neighbor-ip-addr {
           type yang:dotted-quad;
           description "Neighbor address.";
         }
       }
       container virtual-link {
         description "virtual-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
           description "Neighbor router id.";
         }
       }
       container sham-link {
         description "sham-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf local-ip-addr {
           type inet:ip-address;
           description "Sham link local address.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
           description "Neighbor router id.";
         }
         leaf neighbor-ip-addr {
           type yang:dotted-quad;
           description "Neighbor address.";
         }
       }

       leaf state {
           type nbr-state-type;
         description "Neighbor state.";
       }

       description
         "This notification is sent when neighbor
         state change is detected.";
     }

     notification nbr-restart-helper-status-change {
       uses notification-instance-hdr;

       leaf link-type {
         type identityref {
           base if-link-type;
         }
         description "Type of OSPF interface.";
       }

       container interface {
         description "Normal interface.";
         leaf interface {
           type if:interface-ref;
           description "Interface.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
           description "Neighbor router id.";
         }
         leaf neighbor-ip-addr {
           type yang:dotted-quad;
           description "Neighbor address.";
         }
       }
       container virtual-link {
         description "virtual-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
```

```
          description "Neighbor router id.";
        }
      }

      leaf status {
        type restart-helper-status-type;
        description "Restart helper status.";
      }

      leaf age {
        type uint32;
        units seconds;
        description
          "Remaining time in current OSPF graceful restart
          interval, if the router is acting as a restart
          helper for the neighbor.";
      }

      leaf exit-reason {
        type restart-exit-reason-type;
        description
          "Restart helper exit reason.";
      }
      description
        "This notification is sent when neighbor restart
        helper status change is detected.";
    }

    notification rx-bad-packet {
      uses notification-instance-hdr;

      leaf link-type {
        type identityref {
          base if-link-type;
        }
        description "Type of OSPF interface.";
      }

      container interface {
        description "Normal interface.";
        leaf interface {
          type if:interface-ref;
          description "Interface.";
        }
        leaf packet-source {
          type yang:dotted-quad;
          description "Source address.";
        }
```

```
       }
       container virtual-link {
         description "virtual-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf neighbor-router-id {
           type yang:dotted-quad;
           description "Neighbor router id.";
         }
       }

       container sham-link {
         description "sham-link.";
         leaf area-id {
           type uint32;
           description "Area ID.";
         }
         leaf local-ip-addr {
           type inet:ip-address;
           description "Sham link local address.";
         }

         leaf remote-ip-addr {
           type inet:ip-address;
           description "Sham link remote address.";
         }
       }

       leaf packet-type {
         type packet-type;
         description "OSPF packet type.";
       }

       description
         "This notification is sent when an OSPF packet
         has been received on a interface that cannot be parsed.";
     }

     notification lsdb-approaching-overflow {
       uses notification-instance-hdr;

       leaf ext-lsdb-limit {
         type uint32;
         description
           "The maximum number of non-default AS-external LSAs
           entries that can be stored in the link state database.";
       }

       description
         "This notification is sent when the number of LSAs
         in the router's link state database has exceeded
         ninety percent of the ext-lsdb-limit.";
     }

     notification lsdb-overflow {
       uses notification-instance-hdr;

       leaf ext-lsdb-limit {
         type uint32;
         description
           "The maximum number of non-default AS-external LSAs
           entries that can be stored in the link state database.";
       }

       description
         "This notification is sent when the number of LSAs
         in the router's link state database has exceeded
         ext-lsdb-limit.";
     }

     notification nssa-translator-status-change {
       uses notification-instance-hdr;

       leaf area-id {
         type uint32;
         description "Area ID.";
       }

       leaf status {
         type nssa-translator-state-type;
         description
           "NSSA translator status.";
       }

       description
         "This notification is sent when there is a change
         in the router's ability to translate OSPF NSSA LSAs
         OSPF AS-External LSAs.";
     }

     notification restart-status-change {
       uses notification-instance-hdr;

       leaf status {
         type restart-status-type;
         description
           "Restart status.";
       }

       leaf restart-interval {
         type uint16 {
           range "1..1800";
         }
         units seconds;
         default "120";
         description
           "Restart interval.";
       }

       leaf exit-reason {
         type restart-exit-reason-type;
         description
           "Restart exit reason.";
       }

       description
         "This notification is sent when the graceful restart
         state for the router has changed.";
     }
   }
   <CODE ENDS>
```

4. Security Considerations

The data model defined does not create any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

5. Acknowledgements

The authors wish to thank Acee Lindem, Yi Yang, Alexander Clemm, Gaurav Gupta, Ing-Wher Chen, Ladislav Lhotka and Stephane Litkowski for their thorough reviews and helpful comments.

This document was produced using Marshall Rose’s xml2rfc tool.

6. References

6.1. Normative References

[RFC1793] Moy, J., "Extending OSPF to Support Demand Circuits", RFC 1793, April 1995.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.

[RFC3101] Murphy, P., "The OSPF Not-So-Stubby Area (NSSA) Option", RFC 3101, January 2003.

[RFC3623] Moy, J., Pillay-Esnault, P., and A. Lindem, "Graceful OSPF Restart", RFC 3623, November 2003.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.

[RFC4577] Rosen, E., Psenak, P., and P. Pillay-Esnault, "OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4577, June 2006.

[RFC4750] Joyal, D., Galecki, P., Giacalone, S., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", RFC 4750, December 2006.

[RFC5187] Pillay-Esnault, P. and A. Lindem, "OSPFv3 Graceful Restart", RFC 5187, June 2008.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.

[RFC5643] Joyal, D. and V. Manral, "Management Information Base for OSPFv3", RFC 5643, August 2009.

[RFC5838] Lindem, A., Mirtorabi, S., Roy, A., Barnes, M., and R. Aggarwal, "Support of Address Families in OSPFv3", RFC 5838, April 2010.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

6.2. Informative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-15 (work in progress), May 2014.

Authors’ Addresses

Derek Yeung
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

EMail: myeung@cisco.com

Yingzhen Qu
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

EMail: yiqu@cisco.com

Jeffrey Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

EMail: zzhang@juniper.net

Dean Bogdanovic
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

EMail: deanb@juniper.net

Kiran Agrahara Sreenivasa
Brocade Communications System
9442 Capital of Texas Hwy North
Arboretum Plaza One, Suite 500
Austin, TX 78759
USA

EMail: kkoushik@brocade.com