

Internet
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2018

D. Yeung
Arrcus
Y. Qu
Huawei
J. Zhang
Juniper Networks
I. Chen
Jabil
A. Lindem
Cisco Systems
July 2, 2017

Yang Data Model for OSPF Protocol
draft-ietf-ospf-yang-08

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Overview	2
1.1.	Requirements Language	3
2.	Design of Data Model	3
2.1.	OSPF Operational State	3
2.2.	Overview	3
2.3.	OSPFv2 and OSPFv3	5
2.4.	Optional Features	5
2.5.	OSPF Router Configuration/Operational State	5
2.6.	OSPF Instance Configuration/Operational State	5
2.7.	OSPF Area Configuration/Operational State	8
2.8.	OSPF Interface Configuration/Operational State	13
2.9.	OSPF notification	16
2.10.	OSPF RPC Operations	20
3.	OSPF Yang Module	20
4.	Security Considerations	100
5.	Acknowledgements	101
6.	References	101
6.1.	Normative References	101
6.2.	Informative References	103
Appendix A.	Contributors' Addresses	104
	Authors' Addresses	104

[1.](#) Overview

YANG [[RFC6020](#)] is a data definition language used 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 for implementation of other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage OSPF and it is an augmentation to the core routing data model. A core routing data model is defined in [[RFC8022](#)], and it

provides the basis for the development of data models for routing protocols. The interface data model is defined in [\[RFC7223\]](#) and is used for referencing interfaces from the routing protocol. The key-chain data model used for OSPF authentication is defined in [\[RFC8177\]](#)

and provides both a reference to configured key-chains and an enumeration of cryptographic algorithms.

Both OSPFv2 [\[RFC2328\]](#) and OSPFv3 [\[RFC5340\]](#) are supported. In addition to the core OSPF protocol, features described in other OSPF RFCs are also supported. These includes demand circuit [\[RFC1793\]](#), traffic engineering [\[RFC3630\]](#), multiple address family [\[RFC5838\]](#), graceful restart [\[RFC3623\]](#) [\[RFC5187\]](#), NSSA [\[RFC3101\]](#), and OSPF(v3) as a PE-CE Protocol [\[RFC4577\]](#), [\[RFC6565\]](#). These non-core features are optional in the OSPF data model.

[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 router vendors. Differences are observed in terms of how the protocol engine is tied to the routing domain, how multiple protocol engines are be instantiated among others.

The goal of this document is to define a data model that provides a common user interface to the OSPFv2 and OSPFv3 protocols. There is very little information that is designated as "mandatory", providing freedom for vendors to adapt this data model to their respective product implementations.

[2.1.](#) OSPF Operational State

The OSPF operational state is included in the same tree as OSPF configuration consistent with Network Management Datastore Architecture [\[I-D.ietf-netmod-revised-datastores\]](#). Consequently,

only the routing container in the ietf-routing model [\[RFC8022\]](#) is augmented. The routing-state container is not augmented.

[2.2.](#) Overview

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

The OSPF YANG module augments the /routing/control-plane-protocols/control-plane-protocol path defined in the ietf-routing module.

```
module: ietf-ospf
  augment /rt:routing/rt:control-plane-protocols/
    rt:control-plane-protocol:
      +--rw ospf
        .
        .
        +--rw operation-mode?          identityref
        +--rw instance* [af]
          .
          .
          +--rw areas
            | +--rw area* [area-id]
            |   +--rw area-id          area-id-type
            |   .
            |   .
            |   +--rw virtual-links
            |     | +--rw virtual-link* [transit-area-id router-id]
            |     | .
            |     | .
            |     +--rw sham-links {pe-ce-protocol}?
            |       | +--rw sham-link* [local-id remote-id]
            |       | .
            |       | .
            |       +--rw interfaces
            |         +--rw interface* [name]
            |         .
            |         .
            +--rw topologies {multi-topology}?
              +--rw topology* [name]
```

.

.

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

The ospf container includes one or more OSPF protocol engines, each enclosed in a separate instance entity. Each instance includes information for the routing domain based on the [routing-instance af] specification. There is no default routing domain assumed by the data model. For example, to enable OSPF on a vendor's default IPv4 routing domain, an explicit instance entity with a specification like ["default" "ipv4-unicast"] is required. The instance also contains OSPF router level configuration and operational state.

The instance/area and instance/area/interface containers respectively define the OSPF configuration and operational state for OSPF areas and interfaces.

The instance/topology container defines the OSPF configuration and operational state for OSPF topologies when the multi-topology feature is supported.

[2.3.](#) OSPFv2 and OSPFv3

The data model defined herein supports both OSPFv2 and OSPFv3.

The field 'version' is used to indicate the OSPF version and is mandatory. Based on the configured version, the data model varies to accommodate the differences between OSPFv2 and OSPFv3.

[2.4.](#) Optional Features

Optional features are beyond the basic OSPF configuration and it is the responsibility of each vendor to decide whether to support a given feature on a particular device.

This model defines a number of features, such as NSR, max-LSA, etc. It is expected that vendors will support additional features through vendor-specific augmentations.

[2.5.](#) OSPF Router Configuration/Operational State

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

```
module: ietf-ospf
  augment /rt:routing/rt:control-plane-protocols/
    rt:control-plane-protocol:
      +--rw ospf
        +--rw operation-mode?          identityref
        +--rw instance* [af]
          .
          .
```

[2.6.](#) OSPF Instance Configuration/Operational State

The instance container represents an OSPF protocol engine and contains the router level configuration and operational state. The routing domain for each instance is dictated through the specification of [routing-instance af]. The instance level operational state includes the instance level statistics, IETF SPF delay statistics, AS-Scoped Link State Database, local RIB, SPF Log, and the LSA log.

```
module: ietf-ospf
```

```
augment /rt:routing/rt:control-plane-protocols/
  rt:control-plane-protocol:
    +--rw ospf
      .
      .
      +--rw instance* [af]
        +--rw af          identityref
        +--rw explicit-router-id? rt-types:router-id
          | {explicit-router-id}?
        +--rw preference
          | +--rw (scope)?
          |   +--:(single-value)
          |   | +--rw all?          uint8
          |   +--:(multi-values)
          |   +--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 enable?          boolean {admin-control}?
+--rw auto-cost {auto-cost}?
| +--rw enable?      boolean
| +--rw reference-bandwidth?  uint32
+--rw spf-control
| +--rw paths?          uint16 {max-ecmp}?
| +--rw ietf-spf-delay {ietf-spf-delay}?
|   +--rw initial-delay?  uint16
|   +--rw short-delay?    uint16
|   +--rw long-delay?     uint16
|   +--rw hold-down?      uint16
|   +--rw time-to-learn?  uint16
+--rw database-control
| +--rw max-lsa?  uint32 {max-lsa}?
+--rw stub-router {stub-router}?
| +--rw (trigger)?
|   +--:(always)
|     +--rw always!
+--rw mpls
| +--rw te-rid {te-rid}?

```

```

| | +--rw ipv4-router-id?  inet:ipv4-address
| | +--rw ipv6-router-id?  inet:ipv6-address
| +--rw ldp
|   +--rw igp-sync?  boolean {ldp-igp-sync}?
+--rw fast-reroute {fast-reroute}?
| +--rw lfa {lfa}?
+--rw node-tags {node-tag}?
| +--rw node-tag* [tag]

```

```

|      +---rw tag      uint32
+---ro router-id?
+---ro local-rib
|   +---ro route* [prefix]
|       +---ro prefix      inet:ip-prefix
|       +---ro next-hops
|           |   +---ro next-hop* [next-hop]
|           |       +---ro outgoing-interface?  if:interface-ref
|           |       +---ro next-hop            inet:ip-address
|       +---ro metric?      uint32
|       +---ro route-type?  route-type
|       +---ro route-tag?   uint32
+---ro statistics
|   +---ro originate-new-lsa-count?  yang:counter32
|   +---ro rx-new-lsas-count?        yang:counter32
|   +---ro as-scope-lsa-count?       yang:gauge32
|   +---ro as-scope-lsa-chksum-sum?  uint32
|   +---ro database
|       +---ro as-scope-lsa-type*
|           +---ro lsa-type?      uint16
|           +---ro lsa-count?     yang:gauge32
|           +---ro lsa-cksum-sum? int32
+---ro ietf-spf-delay
|   +---ro initial-delay?          uint16
|   +---ro short-delay?           uint16
|   +---ro long-delay?            uint16
|   +---ro hold-down?             uint16
|   +---ro time-to-learn?         uint16
|   +---ro current-state?         enumeration
|   +---ro remaining-time-to-learn? uint16
|   +---ro remaining-hold-down?    uint16
|   +---ro last-event-received?    yang:timestamp
|   +---ro next-spf-time?          yang:timestamp
|   +---ro last-spf-time?          yang:timestamp
+---ro database
|   +---ro as-scope-lsa-type* [lsa-type]
|       +---ro as-scope-lsas
|           +---ro as-scope-lsa* [lsa-id adv-router]
|               +---ro lsa-id      union
|               +---ro adv-router   inet:ipv4-address

```

```

|      +---ro decoded-completed?  boolean

```



```

|         +---ro raw-data?                yang:hex-string
|         +---ro (version)?
|         +---:(ospfv2)
|         |   +---ro ospfv2
|         .
|         .
|         +---:(ospfv3)
|         +---ro ospfv3
|
|
|
+---ro spf-log
|   +---ro event* [id]
|   |   +---ro id                        uint32
|   |   +---ro spf-type?                 enumeration
|   |   +---ro schedule-timestamp?      yang:timestamp
|   |   +---ro start-timestamp?         yang:timestamp
|   |   +---ro end-timestamp?           yang:timestamp
|   |   +---ro trigger-lsa*
|   |   |   +---ro area-id?              area-id-type
|   |   |   +---ro link-id?              union
|   |   |   +---ro type?                 uint16
|   |   |   +---ro lsa-id?               yang:dotted-quad
|   |   |   +---ro adv-router?           yang:dotted-quad
|   |   |   +---ro seq-num?             uint32
|   +---ro lsa-log
|   |   +---ro event* [id]
|   |   |   +---ro id                    uint32
|   |   |   +---ro lsa
|   |   |   |   +---ro area-id?          area-id-type
|   |   |   |   +---ro link-id?          union
|   |   |   |   +---ro type?             uint16
|   |   |   |   +---ro lsa-id?           yang:dotted-quad
|   |   |   |   +---ro adv-router?       yang:dotted-quad
|   |   |   |   +---ro seq-num?          uint32
|   |   |   +---ro received-timestamp?  yang:timestamp
|   |   |   +---ro reason?               identityref
|   |   .
|   |   .
|   .
|   .

```

2.7. OSPF Area Configuration/Operational State

The area container contains OSPF area configuration and the list of interface containers representing all the OSPF interfaces in the area. The area operational state includes the area statistics and the area Link State Database (LSDB).

module: ietf-ospf

```

augment /rt:routing/rt:control-plane-protocols/
  rt:control-plane-protocol:
    +--rw ospf
      .
      .
      +--rw instance* [af]
        +--rw areas
          | +--rw area* [area-id]
          |   +--rw area-id                area-id-type
          |   +--rw area-type?             identityref
          |   +--rw summary?               boolean
          |   +--rw default-cost?          uint32
          |   +--rw ranges
          |     | +--rw range* [prefix]
          |     |   +--rw prefix            inet:ip-prefix
          |     |   +--rw advertise?        boolean
          |     |   +--rw cost?             uint24
          |   +--ro statistics
          |     | +--ro spf-runs-count?      yang:counter32
          |     | +--ro abr-count?           yang:gauge32
          |     | +--ro asbr-count?          yang:gauge32
          |     | +--ro ar-nssa-translator-event-count?
          |     |                                     yang:counter32
          |     | +--ro area-scope-lsa-count? yang:gauge32
          |     | +--ro area-scope-lsa-cksum-sum? int32
          |     +--ro database
          |       +--ro area-scope-lsa-type*
          |         +--ro lsa-type?          uint16
          |         +--ro lsa-count?         yang:gauge32
          |         +--ro lsa-cksum-sum?    int32
          +--ro database
            | +--ro area-scope-lsa-type* [lsa-type]
            |   +--ro lsa-type            uint16
            |   +--ro area-scope-lsas
            |     | +--ro area-scope-lsa* [lsa-id adv-router]
            |     |   +--ro lsa-id                union
            .
            .
            .
            +--ro (version)?
            |   +--:(ospfv2)
            |     | +--ro ospfv2
            |     |   +--ro header
            .
            .
            .
            +--ro body
            +--ro router

```

. . . .
. . . .

```
| | | +--ro network
. . .
. . .
| | | +--ro summary
. . .
. . .
| | | +--ro external
. . .
. . .
| | | +--ro opaque
. . .
. . .
| | | +--:(ospfv3)
| | |   +--ro ospfv3
| | |     +--ro header
. . .
. . .
| | |   +--ro body
| | |     +--ro router
. . .
. . .
| | |   +--ro network
. . .
. . .
| | |   +--ro inter-area-prefix
. . .
. . .
| | |   +--ro inter-area-router
. . .
. . .
| | |   +--ro as-external
. . .
. . .
| | |   +--ro nssa
. . .
. . .
| | |   +--ro link
. . .
. . .
```



```

| | | +--ro if-event-count?          yang:counter32
| | | +--ro link-scope-lsa-count?   yang:gauge32
| | | +--ro link-scope-lsa-cksum-sum?
| | |                               uint32
| | | +--ro database
| | |   +--ro link-scope-lsa-type*
| | |     +--ro lsa-type?          uint16
| | |     +--ro lsa-count?         yang:gauge32
| | |     +--ro lsa-cksum-sum?    int32
| | | +--ro neighbors
| | |   +--ro neighbor* [neighbor-router-id]
| | |     +--ro neighbor-router-id
| | |                               rt-types:router-id
| | |   +--ro address?            inet:ip-address
| | |   +--ro dr-router-id?       rt-types:router-id
| | |   +--ro dr-ip-addr?         inet:ip-address

```

```

| | | +--ro bdr-router-id?   rt-types:router-id
| | | +--ro bdr-ip-addr?    inet:ip-address
| | | +--ro state?          nbr-state-type
| | | +--ro dead-timer?     uint32
| | | +--ro statistics
| | |   +--ro nbr-event-count?
| | |                               yang:counter32
| | |   +--ro nbr-retrans-qlen?
| | |                               yang:gauge32
| | | +--ro database
| | |   +--ro link-scope-lsa-type* [lsa-type]
| | |     +--ro lsa-type      uint16
| | |     +--ro link-scope-lsas
| | |
| | | .
| | | .
| | | +--rw sham-links {pe-ce-protocol}?
| | |   +--rw sham-link* [local-id remote-id]
| | |     +--rw local-id      inet:ip-address
| | |     +--rw remote-id     inet:ip-address
| | |     +--rw hello-interval? uint16
| | |     +--rw dead-interval? uint32
| | |     +--rw retransmit-interval? uint16
| | |     +--rw transmit-delay? uint16
| | |     +--rw lls?          boolean {lls}?
| | |     +--rw ttl-security {ttl-security}?

```

```

|   +--rw enable?      boolean
|   +--rw hops?        uint8
+--rw enable?          boolean
                        {admin-control}?
+--rw authentication
|   +--rw (auth-type-selection)?
|       +--:(auth-ipsec)
|           {ospfv3-authentication-ipsec}?
|       |   +--rw sa?          string
|       +--:(auth-trailer-key-chain)
|       |   +--rw key-chain?
|       |       key-chain:key-chain-ref
|       +--:(auth-trailer-key)
|           +--rw key?          string
|           +--rw crypto-algorithm? identityref
+--rw cost?            uint16
+--rw mtu-ignore?      boolean
                        {mtu-ignore}?
+--rw prefix-suppression? boolean
                        {prefix-suppression}?
+--ro state?          if-state-type
+--ro hello-timer?    uint32
+--ro wait-timer?     uint32

```

```

+--ro dr-router-id?    rt-types:router-id
+--ro dr-ip-addr?      inet:ip-address
+--ro bdr-router-id?   rt-types:router-id
+--ro bdr-ip-addr?     inet:ip-address
+--ro statistics
|   +--ro if-event-count?      yang:counter32
|   +--ro link-scope-lsa-count? yang:gauge32
|   +--ro link-scope-lsa-cksum-sum?
|       uint32
|   +--ro database
|       +--ro link-scope-lsa-type*
|           +--ro lsa-type?      uint16
|           +--ro lsa-count?     yang:gauge32
|           +--ro lsa-cksum-sum? int32
+--ro neighbors
|   +--ro neighbor* [neighbor-router-id]
|       +--ro neighbor-router-id
|           rt-types:router-id

```



```

+--rw name if:interface-ref
+--rw interface-type? enumeration
+--rw passive? boolean
+--rw demand-circuit? boolean
                        {demand-circuit}?
+--rw priority? uint8
+--rw multi-areas {multi-area-adj}?
|   +--rw multi-area* [multi-area-id]
|       +--rw multi-area-id area-id-type
|       +--rw cost? uint16
+--rw static-neighbors
|   +--rw neighbor* [identifier]
|       +--rw identifier inet:ip-address
|       +--rw cost? uint16
|       +--rw poll-interval? uint16
|       +--rw priority? uint8
+--rw node-flag? boolean
                        {node-flag}?
+--rw bfd {bfd}?
|   +--rw enable? boolean
+--rw fast-reroute {fast-reroute}?
|   +--rw lfa {lfa}?
|       +--rw candidate-enable? boolean
|       +--rw enable? boolean
|       +--rw remote-lfa {remote-lfa}?
|       +--rw enable? boolean
+--rw hello-interval? uint16
+--rw dead-interval? uint32
+--rw retransmit-interval? uint16
+--rw transmit-delay? uint16
+--rw lls? boolean {lls}?
+--rw ttl-security {ttl-security}?
|   +--rw enable? boolean
|   +--rw hops? uint8
+--rw enable? boolean
                        {admin-control}?
+--rw authentication

```

```

|   +--rw (auth-type-selection)?
|       +--:(auth-ipsec)
|           {ospfv3-authentication-ipsec}?
|           +--rw sa? string

```



```

|         +---:(auth-trailer-key-chain)
|         |   +---rw key-chain?
|         |       key-chain:key-chain-ref
|         +---:(auth-trailer-key)
|             +---rw key?                string
|             +---rw crypto-algorithm?   identityref
+---rw cost?                uint16
+---rw mtu-ignore?          boolean
|                           {mtu-ignore}?
+---rw prefix-suppression? boolean
|                           {prefix-suppression}?
+---ro state?               if-state-type
+---ro hello-timer?         uint32
+---ro wait-timer?          uint32
+---ro dr-router-id?        rt-types:router-id
+---ro dr-ip-addr?          inet:ip-address
+---ro bdr-router-id?       rt-types:router-id
+---ro bdr-ip-addr?         inet:ip-address
+---ro statistics
|   +---ro if-event-count?      yang:counter32
|   +---ro link-scope-lsa-count? yang:gauge32
|   +---ro link-scope-lsa-cksum-sum?
|                                   uint32
|   +---ro database
|       +---ro link-scope-lsa-type*
|           +---ro lsa-type?      uint16
|           +---ro lsa-count?     yang:gauge32
|           +---ro lsa-cksum-sum? int32
+---ro neighbors
|   +---ro neighbor* [neighbor-router-id]
|       +---ro neighbor-router-id
|                                   rt-types:router-id
|       +---ro address?          inet:ip-address
|       +---ro dr-router-id?     rt-types:router-id
|       +---ro dr-ip-addr?       inet:ip-address
|       +---ro bdr-router-id?    rt-types:router-id
|       +---ro bdr-ip-addr?      inet:ip-address
|       +---ro state?            nbr-state-type
|       +---ro dead-timer?       uint32
|       +---ro statistics
|           +---ro nbr-event-count?
|                                   yang:counter32
|           +---ro nbr-retrans-qlen?
|                                   yang:gauge32

```

```
|
|      +---ro database
|      .   +---ro link-scope-lsa-type* [lsa-type]
|      .       +---ro lsa-type                uint16
|      .       +---ro link-scope-lsas
|
|      .
|
|      .
|      +---rw topologies {ospf:multi-topology}?
|      |   +---rw topology* [name]
|      |       +---rw name  -> ../../../../../../../
|      |                                   ../../../../rt:ribs/rib/name
|      |       +---rw cost? uint32
|      +---rw instance-id?                uint8
|
|
|
```

2.9. OSPF notification

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

notifications:

```
+---n if-state-change
|   +--ro routing-protocol-name?
|   +   -> /rt:routing/control-plane-protocols/
|   +       control-plane-protocol/name
|   +--ro af?
|   +   -> /rt:routing/control-plane-protocols/
|   +       control-plane-protocol
|   +       [rt:name=current()/../routing-protocol-name]/
|   +       ospf:ospf/instance/af
|   +--ro (if-link-type-selection)?
|   |   +--:(interface)
|   |   |   +--ro interface
|   |   |   +--ro interface?    if:interface-ref
|   |   +--:(virtual-link)
|   |   |   +--ro virtual-link
|   |   |   +--ro transit-area-id?    area-id-type
|   |   |   +--ro neighbor-router-id? rt-types:router-id
|   |   +--:(sham-link)
|   |   |   +--ro sham-link
|   |   |   +--ro area-id?    area-id-type
|   |   |   +--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-protocol-name?

```
|   +       -> /rt:routing/control-plane-protocols/
|   +       control-plane-protocol/name
|   +--ro af?
|   +       -> /rt:routing/control-plane-protocols/
|   +       control-plane-protocol
|   +       [rt:name=current()/../routing-protocol-name]/
|   +       ospf:ospf/instance/af
|   +--ro (if-link-type-selection)?
|   |   +--:(interface)
|   |   |   +--ro interface
|   |   |   |   +--ro interface?   if:interface-ref
|   |   +--:(virtual-link)
|   |   |   +--ro virtual-link
|   |   |   |   +--ro transit-area-id?       area-id-type
|   |   |   |   +--ro neighbor-router-id?   rt-types:router-id
|   |   +--:(sham-link)
|   |   |   +--ro sham-link
|   |   |   |   +--ro area-id?            area-id-type
|   |   |   |   +--ro local-ip-addr?     inet:ip-address
|   |   |   |   +--ro remote-ip-addr?    inet:ip-address
|   +--ro packet-source?               yang:dotted-quad
|   +--ro packet-type?                 packet-type
|   +--ro error?                        enumeration
+---n nbr-state-change
|   +--ro routing-protocol-name?
|   +       -> /rt:routing/control-plane-protocols/
|   +       control-plane-protocol/name
|   +--ro af?
|   +       -> /rt:routing/control-plane-protocols/
|   +       control-plane-protocol
|   +       [rt:name=current()/../routing-protocol-name]/
|   +       ospf:ospf/instance/af
|   +--ro (if-link-type-selection)?
|   |   +--:(interface)
|   |   |   +--ro interface
|   |   |   |   +--ro interface?   if:interface-ref
|   |   +--:(virtual-link)
|   |   |   +--ro virtual-link
|   |   |   |   +--ro transit-area-id?       area-id-type
|   |   |   |   +--ro neighbor-router-id?   rt-types:router-id
```

```

| | +---:(sham-link)
| | |   +---ro sham-link
| | |   |   +---ro area-id?          area-id-type
| | |   |   +---ro local-ip-addr?    inet:ip-address
| | |   |   +---ro remote-ip-addr?   inet:ip-address
| | +---ro neighbor-router-id?       rt-types:router-id
| | +---ro neighbor-ip-addr?         yang:dotted-quad
| | +---ro state?                    nbr-state-type

```

```

+---n nbr-restart-helper-status-change
| +---ro routing-protocol-name?
| +   -> /rt:routing/control-plane-protocols/
| +       control-plane-protocol/name
| +---ro af?
| +   -> /rt:routing/control-plane-protocols/
| +       control-plane-protocol
| +       [rt:name=current()/../routing-protocol-name]/
| +       ospf:ospf/instance/af
| +---ro (if-link-type-selection)?
| | +---:(interface)
| | |   +---ro interface
| | |   |   +---ro interface?    if:interface-ref
| | |   +---:(virtual-link)
| | |   |   +---ro virtual-link
| | |   |   |   +---ro transit-area-id?    area-id-type
| | |   |   |   +---ro neighbor-router-id? rt-types:router-id
| | |   +---:(sham-link)
| | |   |   +---ro sham-link
| | |   |   |   +---ro area-id?          area-id-type
| | |   |   |   +---ro local-ip-addr?    inet:ip-address
| | |   |   |   +---ro remote-ip-addr?   inet:ip-address
| | +---ro neighbor-router-id?       rt-types:router-id
| | +---ro neighbor-ip-addr?         yang:dotted-quad
| | +---ro status?                   restart-helper-status-type
| | +---ro age?                      uint32
| | +---ro exit-reason?              restart-exit-reason-type
+---n if-rx-bad-packet
| +---ro routing-protocol-name?
| +   -> /rt:routing/control-plane-protocols/
| +       control-plane-protocol/name
| +---ro af?
| +   -> /rt:routing/control-plane-protocols/

```

```

| +         control-plane-protocol
| +         [rt:name=current()/../routing-protocol-name]/
| +         ospf:ospf/instance/af
| +--ro (if-link-type-selection)?
| |   +--:(interface)
| | |   +--ro interface
| | | |   +--ro interface?   if:interface-ref
| | | +--:(virtual-link)
| | | |   +--ro virtual-link
| | | | |   +--ro transit-area-id?       area-id-type
| | | | |   +--ro neighbor-router-id?    rt-types:router-id
| | | +--:(sham-link)
| | | |   +--ro sham-link
| | | | |   +--ro area-id?               area-id-type
| | | | |   +--ro local-ip-addr?        inet:ip-address

```

```

| |   +--ro remote-ip-addr?   inet:ip-address
| +--ro packet-source?        yang:dotted-quad
| +--ro packet-type?          packet-type
+---n lsdb-approaching-overflow
| +--ro routing-protocol-name?
| +   -> /rt:routing/control-plane-protocols/
| +       control-plane-protocol/name
| +--ro af?
| +   -> /rt:routing/control-plane-protocols/
| +       control-plane-protocol
| +       [rt:name=current()/../routing-protocol-name]/
| +       ospf:ospf/instance/af
| +--ro ext-lsdb-limit?       uint32
+---n lsdb-overflow
| +--ro routing-protocol-name?
| +   -> /rt:routing/control-plane-protocols/
| +       control-plane-protocol/name
| +--ro af?
| +   -> /rt:routing/control-plane-protocols/
| +       control-plane-protocol
| +       [rt:name=current()/../routing-protocol-name]/
| +       ospf:ospf/instance/af
| +--ro ext-lsdb-limit?       uint32
+---n nssa-translator-status-change
| +--ro routing-protocol-name?
| +   -> /rt:routing/control-plane-protocols/

```

```

| +          control-plane-protocol/name
| +--ro af?
| +      -> /rt:routing/control-plane-protocols/
| +          control-plane-protocol
| +          [rt:name=current()/../routing-protocol-name]/
| +          ospf:ospf/instance/af
| +--ro area-id?          area-id-type
| +--ro status?          nssa-translator-state-type
+---n restart-status-change
    +--ro routing-protocol-name?
    +      -> /rt:routing/control-plane-protocols/
    +          control-plane-protocol/name
    +--ro af?
    +      -> /rt:routing/control-plane-protocols/
    +          control-plane-protocol
    +          [rt:name=current()/../routing-protocol-name]/
    +          ospf:ospf/instance/af
    +--ro status?          restart-status-type
    +--ro restart-interval? uint16
    +--ro exit-reason?     restart-exit-reason-type

```

[2.10.](#) OSPF RPC Operations

The "ietf-ospf" module defines two RPC operations:

- o clear-database: reset the content of a particular OSPF database.
- o clear-neighbor: restart a particular set of OSPF neighbor.

rpcs:

```

+---x clear-neighbor
|   +---w input
|       +---w routing-protocol-name
|           +      -> /rt:routing/control-plane-protocols/
|           +          control-plane-protocol/name
|           +---w interface?          if:interface-ref
+---x clear-database
    +---w input
        +---w routing-protocol-name
            -> /rt:routing/control-plane-protocols/

```

3. OSPF Yang Module

```

<CODE BEGINS> file "ietf-ospf@2017-07-01.yang"
module ietf-ospf {
    namespace "urn:ietf:params:xml:ns:yang:ietf-ospf";

    prefix ospf;

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

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

    import ietf-interfaces {
        prefix "if";
    }

    import ietf-routing-types {
        prefix "rt-types";
    }

    import iana-routing-types {
        prefix "iana-rt-types";
    }

```

```

import ietf-routing {
    prefix "rt";
}

import ietf-key-chain {
    prefix "key-chain";
}

organization
    "IETF OSPF - OSPF Working Group";

contact

```

"WG Web: <<http://datatracker.ietf.org/group/ospf/>>
WG List: <<mailto:ospf@ietf.org>>

Editor: Derek Yeung
<<mailto:derek@arrcus.com>>
Author: Acee Lindem
<<mailto:acee@cisco.com>>
Author: Yingzhen Qu
<<mailto:yingzhen.qu@huawei.com>>
Author: Jeffrey Zhang
<<mailto:zzhang@juniper.net>>
Author: Ing-Wher Chen
<mailto:ing-wher_chen@jabil.com>
Author: Dean Bogdanovic
<<mailto:ivandean@gmail.com>>
Author: Kiran Agrahara Sreenivasa
<<mailto:kkoushik@cisco.com>>";

description

"This YANG module defines the generic configuration and operational state for the OSPF protocol common to all vendor implementations. 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.

Copyright (c) 2017 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in [Section 4.c](#) of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX;
see the RFC itself for full legal notices.";

reference "RFC XXXX";


```

revision 2017-07-01 {
  description
    "* Restructure model to conform to NMDA.
    * Remove features for instance, area and interface
      inheritance.
    * Update static neighbor identifier description to
      allow for router-id, ipv4-address, and ipv6-address.
    * Added spf-log and lsa-log.
    * Use dotted-quad for OSPFv2 LSA ID.
    * Fix virtual-link transit-area-id leafref path and
      must statement.
    ";
  reference
    "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2017-03-12 {
  description
    "* Update authors information.
    * Rename admin distance to preference.
    * Rename network type to interface type.
    * Add ietf-spf-delay as a feature.
    * Add node-tag as a feature and update LSA
      definition accordingly.
    * Remove LDP IGP autoconfig.
    * Add BFD as a feature instead of a separate module.
    * Change TE router ID to support IPv4 and IPv6 router ID.
    * Replace key-chain:crypto-algorithm-types with
      key-chain:crypto-algorithm.
    * Remove type ieee-bandwidth.
    * Import ietf-routing-types and make use of
      router-id, address-family and bandwidth-ieee-float32
      type definitions.
    * Simplify notification header.
    * Fix compilation issue in multiple must statements.
    ";
  reference
    "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2016-10-31 {
  description
    "* Update authors information.

```

```

    * Rename candidate-disabled to candidate-enable
      and set the default value to TRUE.
    * Rename node identifiers that end with
      'enabled' to 'enable'.
    * Set the default value of
      ospf/instance/areas/area/interfaces/interface/
      fast-reroute/lfa/enable (previously named 'enabled')
      to FALSE.
    * Set the default value of
      ospf/instance/areas/area/interfaces/interface/
      fast-reroute/remote-lfa/enable (previously named 'enabled')
      to FALSE.
    * Rename
      ospf/instance/areas/area/interfaces/interface/
      static-neighbors/neighbor/address to 'identifier'
      with type inet:ip-address
    * Add 'dead-timer' to
      ospf-state/instance/areas/area/interfaces/interface/
      neighbors/neighbor.
    * Remove 'mtu-ignore' and 'prefix-suppression' from
      virtual-link configuration.
    * Adjust range specifications from 'transmit-delay',
      'dead-interval', and 'retransmit-interval' in
      ospf/instance/areas/area/interfaces/interface.
    * Change the type of
      ospf/instance/areas/area/interface/interface/dead-interval
      to uint32 to match RFC2328 Appendix A.3.2.
    * Change hello-timer and wait-timer unit to seconds.
    * Update hello-timer, dead-timer and wait-timer descriptions.
    * Add IEEE bandwidth type and update all TE bandwidth fields
      to use it.
    * Add Nt-bit to OSPFv2 router LSA.
    * Remove L-bit from OSPFv2 router LSA.
  ";
  reference
    "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2016-07-07 {
  description
    "* Add ospfv3 AF bit.
    * Add ospfv2 MT, L, O, DN bit.
    * Add interface priority config.
    * Change bdr-ip-address to type ip-address.
    * Rename leaf interface to name.
    * Rename rx-bad-packet to if-rx-bad-packet.
    * Move virtual link placement to backbone area.
    * Remove cost configuration from virtual link.

```

Internet-Draft

OSPF Yang Data Model

July 2017

```
    * Move if-feature multi-area-adj statement.
    * Add type checksum16-type.
    * Change LSA header checksum to use checksum16-type.
    * Change routing-protocol to control-plane-protocol.
    * Change import module name to ietf-key-chain.";
reference
  "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2016-03-20 {
  description
    "* Reorganize *-config and *-operation groupings.
    * Use *-config under state tree for applied config.
    * Rename config router-id to explicit-router-id.
    * Rename feature router-id to explicit-router-id.
    * Add OSPFv3 instance ID.
    * Add OSPFv3 interface ID.
    * Add ip-address for DR and BDR.
    * Remove routing-instance.
    * Change import module name to ietf-routing-key-chain.";
  reference
    "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2015-10-19 {
  description
    "* Remove the abstract identity ospf.
    * Make area-id-type dotted-quad only.
    * Use area-id-type for all area-id leafs.
    * Restructure notifications.
    * Move BFD support to the new ietf-ospf-bfd module.
    * Update author information.
    * Editorial changes.";
  reference
    "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2015-09-02 {
  description
    "* Author information update.
    * Editorial changes";
  reference
```

```

    "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2015-07-06 {
  description
    "* Remove support for protocol-centric config.

```

```

    * Enclose list in container, except for instance.
    * Replace protocol-shutdown with admin-control.
    * Add IP-FRR per-interface config.
    * Reorganize max-path etc node.
    * Add node-flag.
    * Align config/operation hierarchy.
    * Use relative path for reference to rib.
    * Add ability to set single admin distance.
    * Make unreserved bandwidth into list.
    * Add F and T bit to OSPFv3 external LSA.
    * Remove key statement inside LSA body.
    * Add stub router support.
    * Fix usage of af-area-config.
    * Add statistics to operation data.
    * Add local rib.
    * Use dotted-quad for all router-id fields.
    * Support more than one multi-area per interface.
    * Use uint16 for LSA type.
    * Update grouping notification-instance-hdr.
    * Rework condition for opaque type and id in OSPFv2 LSA.
    * Rename local-remote-ipv4-addr with remote-if-ipv4-addr.
    * Add virtual-link/sham-link to operation state.
    * Allow multiple link TLVs in one LSA.
    * Fix bug in as-scope-lsas.
    * Remove OSPFv3 restriction in link-scope-lsas.
    * Editorial changes.";
  reference
    "RFC XXXX: A YANG Data Model for OSPF.";
}

revision 2015-03-09 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for OSPF.";

```

```

}

feature multi-topology {
    description
        "Support Multiple-Topolgy Routing (MTR).";
}

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

feature explicit-router-id {
    description

```

```

        "Set Router-ID per instance explicitly.";
    }

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

feature mtu-ignore {
    description
        "Disable OSPF Database Description packet MTU
        mismatch checking.";
}

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

```

```

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

feature graceful-restart {
  description
    "Graceful OSPF Restart as defined in RFC 3623 and
    RFC 5187.";
}

feature admin-control {
  description
    "Administrative control of the protocol state.";
}

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 the OSPF instance
    will accept.";
}

feature te-rid {
  description
    "TE Router-ID.";
}

feature ldp-igp-sync {
  description

```

```

    "LDP IGP synchronization.";
}

feature ospfv3-authentication-ipsec {
    description
        "Use IPsec for OSPFv3 authentication.";
}

feature fast-reroute {
    description
        "Support of IP Fast Reroute (IP-FRR).";
}

feature node-flag {
    description
        "Support of node flag.";
}

feature node-tag {
    description
        "Support of node tag.";
}

feature lfa {
    description
        "Support of Loop Free Alternates (LFAs).";
}

```

```

feature remote-lfa {
    description
        "Support of Remote Loop Free Alternates (R-LFA).";
}

feature stub-router {
    description
        "Support of RFC 6987 OSPF Stub Router Advertisement.";
}

feature pe-ce-protocol {
    description
        "Support PE-CE protocol";
}

```

```

feature ietf-spf-delay {
    description
        "Support of IETF SPF delay algorithm.";
}

feature bfd {
    description
        "Support of BFD.";
}

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.";
}

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

identity lsa-log-reason {
    description
        "Base identity for an LSA log reason.";
}

identity lsa-refresh {
    base lsa-log-reason;
    description
        "Identity used when the LSA is logged
         as a result of receiving a refresh LSA.";
}

identity lsa-content-change {
    base lsa-log-reason;
    description
        "Identity used when the LSA is logged
         as a result of a change in the content
         of the LSA.";
}

identity lsa-purge {
    base lsa-log-reason;
    description
        "Identity used when the LSA is logged
         as a result of being purged.";
}

typedef uint24 {
    type uint32 {
        range "0 .. 16777215";
    }
}

```

```

    }
    description
        "24-bit unsigned integer.";
}

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

typedef 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 type 1 external route.";
        }
        enum external-2 {
            description "OSPF type 2 external route.";
        }
        enum nssa-1 {
            description "OSPF type 1 NSSA route.";
        }
        enum nssa-2 {
            description "OSPF type 2 NSSA route.";
        }
    }
    description "OSPF route 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 acknowledgement 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.";
}

typedef checksum16-type {
    type string {
        pattern '(0x)?[0-9a-fA-F]{4}';
    }
    description
        "16-bit checksum in hex-string format 0xXXXX.";
}

```

```

}

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 unknown-tlvs {
  description
    "Unknown TLVs grouping - Used for unknown TLVs or
    unknown sub-TLVs.";
  container unknown-tlvs {
    description "All unknown TLVs.";
    list unknown-tlv {
      description "Unknown TLV.";
      uses tlv;
    }
  }
}

grouping node-tag-tlv {

```

```

  description "OSPF Node Admin Tag TLV grouping.";
  list node-tag {
    leaf tag {
      type uint32;
      description
        "Node tag value.";
    }
    description

```



```

        "List of tags.";
    }
}

grouping ospfv2-router-link {
    description "OSPFv2 router 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.";
    }
}

grouping ospfv2-lsa-body {
    description "OSPFv2 LSA body.";
    container router {
        when "../header/type = 1" {
            description
                "Only applies to Router LSAs.";
        }
        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.";
    }
}
description "Flags.";
}
leaf num-of-links {
    type uint16;
    description "Number of links.";
}
container links {
    description "All router Links.";
    list link {
        description "Router LSA link.";
        uses ospfv2-router-link;
        container topologies {
            description "All topologies for the link.";
            list topology {
                description
                    "Topology specific information.";
                leaf mt-id {
                    type uint8;
                    description
                        "The MT-ID for the topology enabled on
                        the link.";
                }
                leaf metric {
                    type uint16;
                    description "Metric for the topology.";
                }
            }
        }
    }
}
}

```

```
    }
  }
  container network {
    when "../..//header/type = 2" {
      description
        "Only applies to Network LSAs.";
    }
    description
      "Network LSA.";
    leaf network-mask {
      type inet:ipv4-address;
      description
        "The IP address mask for the network.";
    }
    container attached-routers {
      description "All attached routers.";
      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 applies to Summary LSAs.";
  }
  description
    "Summary LSA.";
  leaf network-mask {
    type inet:ipv4-address;
    description
      "The IP address mask for the network";
  }
  container topologies {
    description "All topologies for the summary LSA.";
    list topology {
      description
        "Topology specific information.";
      leaf mt-id {
        type uint8;
        description
          "The MT-ID for the topology enabled for
            the summary.";
      }
    }
  }
}
```

```
}
leaf metric {
```

```
        type uint24;
        description "Metric for the topology.";
    }
}
}
}
container external {
    when "../../header/type = 5 or "
        + "../../header/type = 7" {
        description
            "Only applies to AS-external LSAs and NSSA LSAs.";
    }
    description
        "External LSA.";
    leaf network-mask {
        type inet:ipv4-address;
        description
            "The IP address mask for the network";
    }
    container topologies {
        description "All topologies for the external.";
        list topology {
            description
                "Topology specific information.";
            leaf mt-id {
                type uint8;
                description
                    "The MT-ID for the topology enabled for the
                     external or NSSA prefix.";
            }
            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 for the topology.";
    }
}
}
}
}
container opaque {
    when "../..header/type = 9 or "
        + "../..header/type = 10 or "
        + "../..header/type = 11" {
        description
            "Only applies to Opaque LSAs.";
    }
    description
        "Opaque LSA.";

    uses unknown-tlvs;

    container node-tag-tlvs {
        description
            "All node tag tlvs.";
        list node-tag-tlv {
            description
                "Node tag tlv.";
            uses node-tag-tlv;
        }
    }
}

container router-address-tlv {

```

```

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

  container link-tlvs {
    description "All link TLVs in the LSA.";
    list link-tlv {
      description "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.";
  }
  container local-if-ipv4-addrs {
    description "All local interface IPv4 addresses.";
    leaf-list local-if-ipv4-addr {
      type inet:ipv4-address;
      description
        "List of local interface IPv4 addresses.";
    }
  }
  container remote-if-ipv4-addrs {
    description "All remote interface IPv4 addresses.";
    leaf-list remote-if-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 rt-types:bandwidth-ieee-float32;
    description "Maximum bandwidth.";
}
leaf max-reservable-bandwidth {
    type rt-types:bandwidth-ieee-float32;
    description "Maximum reservable bandwidth.";
}
container unreserved-bandwidths {
    description "All unreserved bandwidths.";
    list unreserved-bandwidth {
        leaf priority {
            type uint8 {
                range "0 .. 7";
            }
            description "Priority from 0 to 7.";
        }
        leaf unreserved-bandwidth {

```

```

        type rt-types:bandwidth-ieee-float32;
        description "Unreserved bandwidth.";
    }
    description
        "List of unreserved bandwidths for different
        priorities.";
}
}
leaf admin-group {
    type uint32;
    description
        "Administrative group/Resource Class/Color.";
}
uses unknown-tlvs;
}
}

container extended-prefix-tlvs {

```

```

description "All extended prefix TLVs in the LSA.";
list extended-prefix-tlv {
  description "Extended prefix TLV.";
  leaf route-type {
    type enumeration {
      enum unspecified {
        value "0";
        description "Unspecified.";
      }
      enum intra-area {
        value "1";
        description "OSPF intra-area route.";
      }
      enum inter-area {
        value "3";
        description "OSPF inter-area route.";
      }
      enum external {
        value "5";
        description "OSPF External route.";
      }
      enum nssa {
        value "7";
        description "OSPF NSSA external route.";
      }
    }
    description "Route type.";
  }
  leaf flags {
    type bits {

```

```

    bit A {
      description
        "Attach flag.";
    }
    bit N {
      description
        "Node flag.";
    }
  }
  description "Flags.";
}

```



```

        leaf prefix {
            type inet:ip-prefix;
            description "Address prefix.";
        }
        uses unknown-tlvs;
    }
}

container extended-link-tlvs {
    description "All extended link TLVs in the LSA.";
    list extended-link-tlv {
        description "Extended link TLV.";
        uses ospfv2-router-link;
        uses unknown-tlvs;
    }
}
}

grouping ospfv3-lsa-options {
    description "OSPFv3 LSA options";
    leaf options {
        type bits {
            bit AF {
                description
                    "When set, the router supports OSPFv3 AFs as in RFC5838.";
            }
            bit DC {
                description
                    "When set, the router supports 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
                    translated to an AS External LSA and readvertised
                    by the translating NSSA Border Router.";
            }
            bit DN {
                description
                    "When set, the inter-area-prefix LSA or

```

```
        AS-external LSA prefix has been advertised as an
        L3VPN prefix.";
    }
}
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.";
            }
            bit F {
                description
                    "When set, a Forwarding Address is included
                    in the LSA.";
            }
            bit T {
                description
                    "When set, an External Route Tag is included
                    in the LSA.";
            }
        }
        description "Flags.";
    }

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

    uses ospfv3-lsa-prefix;

    leaf forwarding-address {
        type inet:ipv6-address;
```

```
        "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 applies to Router LSAs.";
        }
        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 "Router LSA flags.";
}

uses ospfv3-lsa-options;

container links {
    description "All router link.";
    list link {
        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 rt-types:router-id;
            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 applies to Network LSA.";
}
description "Network LSA.";

uses ospfv3-lsa-options;

container attached-routers {
    description "All attached routers.";
    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 applies to Inter-Area-Prefix LSAs.";
    }
    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 applies to Inter-Area-Router LSAs.";
    }
    uses ospfv3-lsa-options;
    leaf metric {
        type uint24;
        description "Metric.";
    }
    leaf destination-router-id {
        type rt-types:router-id;
    }
}

```

```

        description
            "The Router ID of the router being described by the LSA.";
    }
    description "Inter-Area-Router LSA.";
}
container as-external {
    when "../..header/type = 16389" { // 0x4005
        description
            "Only applies to AS-external LSAs.";
    }

    uses ospfv3-lsa-external;

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

```

```

    description "NSSA LSA.";
}
container link {
    when "../..header/type = 8" { // 0x0008
        description
            "Only applies to Link LSAs.";
    }
    leaf rtr-priority {
        type uint8;
        description "Router Priority for the interface.";
    }
    uses ospfv3-lsa-options;

    leaf link-local-interface-address {
        type inet:ipv6-address;
        description
            "The originating router's link-local
            interface address for the link.";
    }
}

```

```

leaf num-of-prefixes {
    type uint32;
    description "Number of prefixes.";
}

container prefixes {
    description "All prefixes for the link.";
    list 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 applies 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.";

```



```

    }
    container prefixes {
        description "All prefixes in this LSA.";
        list prefix {
            description "List of prefixes in this LSA.";
            uses ospfv3-lsa-prefix;
            leaf metric {
                type uint24;
                description "Prefix Metric.";
            }
        }
    }
}
container router-information {
    when "../../header/type = 32780 or " // 0x800C
    + "../../header/type = 40972 or " // 0xA00C
    + "../../header/type = 49164 or " // 0xC00C
    + "../../header/type = 57356" { // 0xE00C
        description
            "Only applies to Router Information LSAs (RFC7770).";
    }
    container node-tag-tlvs {
        description
            "All node tag tlvs.";
        list node-tag-tlv {
            description
                "Node tag tlv.";
            uses node-tag-tlv;
        }
    }
    description "Router Information LSA.";
}
}

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 checksum16-type;
    mandatory true;
    description "LSA checksum.";
}
leaf length {
    type uint16;
    mandatory true;
    description "LSA length.";
}
}

grouping ospfv2-lsa {
    description
        "OSPFv2 LSA.";
    container header {
        must "(type = 9 or type = 10 or type = 11) and "
            + "opaque-id and opaque-type "
            + "or (type != 9 and type != 10 and type != 11) "
            + "and not(opaque-id) and "
            + "not(opaque-type)" {
            description
                "Opaque type and ID only apply to Opaque LSAs.";
        }
        description
            "Decoded OSPFv2 LSA header data.";
        leaf option {

```

```

type bits {
  bit MT {
    description
      "When set, the router supports multi-topology as
        in RFC 4915.";
  }
  bit DC {
    description
      "When set, the router supports demand circuits.";
  }
  bit P {
    description
      "Only used in type-7 LSA. When set, an NSSA
        border router should translate the type-7 LSA
        to a type-5 LSA.";
  }
  bit MC {
    description
      "When set, the router supports MOSPF.";
  }
  bit E {
    description
      "This bit describes the way AS-external LSAs
        are flooded.";
  }
  bit O {
    description
      "When set, the router is opaque-capable as in
        RFC 5250.";
  }
  bit DN {
    description
      "When a type 3, 5 or 7 LSA is sent from a PE to a CE,
        the DN bit must be set. See RFC 4576.";
  }
}
mandatory true;
description "LSA options.";
}
leaf lsa-id {
  type yang:dotted-quad;
  mandatory true;
  description "LSA ID.";
}

leaf opaque-type {
  type uint8;
  description "Opaque type.";
}

```

Internet-Draft

OSPF Yang Data Model

July 2017

```
    }

    leaf opaque-id {
        type uint24;
        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-common {
    description
        "Common field for OSPF LSA represenation.";
    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 hexadecimal as received or originated.";
}

```

```

}
}

grouping lsa {
    description
        "OSPF LSA.";
    uses lsa-common;
    choice version {
        description
            "OSPFv2 or OSPFv3 LSA body.";
        container ospfv2 {
            description "OSPFv2 LSA";
            uses ospfv2-lsa;
        }
        container ospfv3 {
            description "OSPFv3 LSA";
            uses ospfv3-lsa;
        }
    }
}

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

```

```

        "Advertising router.";
    }
}

grouping instance-stat {
    description "Per-instance statistics";
    leaf originate-new-lsa-count {
        type yang:counter32;
        description "The number of new LSAs originated.";
    }
    leaf rx-new-lsas-count {
        type yang:counter32;
        description "The number of LSAs received.";
    }
}

```

```

    }
    leaf as-scope-lsa-count {
        type yang:gauge32;
        description "The number of AS Scope LSAs.";
    }
    leaf as-scope-lsa-chksum-sum {
        type uint32;
        description
            "The sum of the LSA checksums for AS Scope LSAs.";
    }
    container database {
        description "Container for per AS-scope LSA statistics.";
        list as-scope-lsa-type {
            description "List of AS-scope LSA statistics";
            leaf lsa-type {
                type uint16;
                description "AS-scope LSA type.";
            }
            leaf lsa-count {
                type yang:gauge32;
                description "The number of LSAs of the LSA type.";
            }
            leaf lsa-cksum-sum {
                type int32;
                description
                    "The sum of the LSA checksums of the LSA type.";
            }
        }
    }
}

```

```

    }
}

grouping area-stat {
    description "Per-area statistics.";
    leaf spf-runs-count {
        type yang:counter32;
        description
            "The number of times the intra-area SPF has run.";
    }
    leaf abr-count {
        type yang:gauge32;
        description
            "The total number of Area Border Routers (ABRs)
            reachable within this area.";
    }
    leaf asbr-count {
        type yang:gauge32;
        description
            "The total number of AS Border Routers (ASBRs).";
    }
}

```

```

}
leaf ar-nssa-translator-event-count {
    type yang:counter32;
    description
        "The number of NSSA translator-state changes.";
}
leaf area-scope-lsa-count {
    type yang:gauge32;
    description
        "The number of area scope LSAs in the area.";
}
leaf area-scope-lsa-cksum-sum {
    type int32;
    description "The sum of the area scope LSAs checksums.";
}
container database {
    description "Container for area scope LSA type statistics.";
    list area-scope-lsa-type {
        description "List of area scope LSA statistics";
        leaf lsa-type {
            type uint16;
        }
    }
}

```

```

        description "Area scope LSA type.";
    }
    leaf lsa-count {
        type yang:gauge32;
        description "The number of LSAs of the LSA type.";
    }
    leaf lsa-cksum-sum {
        type int32;
        description
            "The sum of the LSA checksums of the LSA type.";
    }
}
}
}

grouping interface-stat {
    description "Per-interface statistics";
    leaf if-event-count {
        type yang:counter32;
        description
            "The number of times this interface has changed its
            state or an error has occurred.";
    }
    leaf link-scope-lsa-count {
        type yang:gauge32;
        description "The number of link scope LSAs.";
    }
}

```

```

leaf link-scope-lsa-cksum-sum {
    type uint32;
    description "The sum of link scope LSA checksums.";
}
container database {
    description "Container for link scope LSA type statistics.";
    list link-scope-lsa-type {
        description "List of link scope LSA statistics";
        leaf lsa-type {
            type uint16;
            description "Link scope LSA type.";
        }
    }
    leaf lsa-count {
        type yang:gauge32;
    }
}

```



```

        description "The number of LSAs of the LSA type.";
    }
    leaf lsa-cksum-sum {
        type int32;
        description
            "The sum of the LSA checksums of the LSA type.";
    }
}

grouping neighbor-stat {
    description "Per-neighbor statistics.";
    leaf nbr-event-count {
        type yang:counter32;
        description
            "The number of times this neighbor has changed
            state or an error has occurred.";
    }
    leaf nbr-retrans-qlen {
        type yang:gauge32;
        description
            "The current length of the retransmission queue.";
    }
}

grouping instance-fast-reroute-config {
    description
        "This group defines global configuration of IP-FRR.";
    container fast-reroute {
        if-feature fast-reroute;
        description
            "This container may be augmented with global
            parameters for IP-FRR.";
    }
}

```

```

    container lfa {
        if-feature lfa;
        description
            "This container may be augmented with
            global parameters for LFA. Container creation
            has no effect on LFA activation.";
    }
}

```

```

    }
}

grouping interface-fast-reroute-config {
    description
        "This group defines interface configuration of IP-FRR.";
    container fast-reroute {
        if-feature fast-reroute;
        container lfa {
            if-feature lfa;
            leaf candidate-enable {
                type boolean;
                default true;
                description
                    "Enable the interface to be used as backup.";
            }
            leaf enable {
                type boolean;
                default false;
                description
                    "Activates LFA - Per-prefix LFA computation
                     is assumed.";
            }
            container remote-lfa {
                if-feature remote-lfa;
                leaf enable {
                    type boolean;
                    default false;
                    description
                        "Activates Remote LFA (R-LFA).";
                }
                description
                    "Remote LFA configuration.";
            }
            description
                "LFA configuration.";
        }
        description
            "IP Fast-reroute configuration.";
    }
}

```

```

grouping interface-physical-link-config {
  description
    "Interface cost configuration that only applies to
    physical interfaces and sham links.";
  leaf cost {
    type uint16 {
      range "1..65535";
    }
    description
      "Interface cost.";
  }
  leaf mtu-ignore {
    if-feature mtu-ignore;
    type boolean;
    description
      "Enable/Disable bypassing the MTU mismatch check in
      Database Description packets.";
  }
  leaf prefix-suppression {
    if-feature prefix-suppression;
    type boolean;
    description
      "Suppress advertisement of the prefixes associated
      with the interface.";
  }
}

```

```

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

  leaf hello-interval {
    type uint16 {
      range "1..65535";
    }
    units seconds;
    description
      "Interval between hello packets in seconds.";
  }

  leaf dead-interval {
    type uint32 {
      range "1..2147483647";
    }
    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-interval'.";
}
description
    "Interval after which a neighbor is
    declared down in seconds.";
}

leaf retransmit-interval {
    type uint16 {
        range "1..3600";
    }
    units seconds;
    description
        "Interval between retransmitting unacknowledged Link
        State Advertisements (LSAs) in seconds.";
}

leaf transmit-delay {
    type uint16 {
        range "1..3600";
    }
    units seconds;
    description
        "Estimated time needed to transmit Link State Update
        packets on the interface in seconds.";
}

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

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 an OSPF packet may
            have traversed before reception.";
    }
}
leaf enable {
    if-feature admin-control;
    type boolean;
    default true;
    description
        "Enable/disable protocol on the interface.";
}

container authentication {
    description "Authentication configuration.";
    choice auth-type-selection {
        description
            "Options for expressing authentication setting.";
        case auth-ipsec {
            when "../..../..../rt:type = 'ospf:ospfv3'" {
                description "Applied to OSPFv3 only.";
            }
            if-feature ospfv3-authentication-ipsec;
            leaf sa {
                type string;
                description
                    "Security Association name.";
            }
        }
        case auth-trailer-key-chain {
            leaf key-chain {
                type key-chain:key-chain-ref;
                description
                    "key-chain name.";
            }
        }
    }
}

```

```

    case auth-trailer-key {
      leaf key {
        type string;
        description
          "Key string in ASCII format.";
      }
      leaf crypto-algorithm {
        type identityref {
          base key-chain:crypto-algorithm;
        }
        description
          "Cryptographic algorithm associated with key.";
      }
    }
  }
}

```

```

    }
  }
}
} // interface-common-config

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

  leaf interface-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
            "Interface type.";
    }

    leaf passive {
        type boolean;
        description
            "Enable/Disable passive interface - a passive interface's
            prefix will be advertised but no neighbor adjacencies
            will be formed on the interface.";
    }

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

```

```

    leaf priority {
        type uint8;
        description
            "Configure OSPF router priority.";
    }

    container multi-areas {
        if-feature multi-area-adj;
        description "Container for multi-area config.";
        list multi-area {
            key multi-area-id;
            description
                "Configure OSPF multi-area adjacency.";
            leaf multi-area-id {
                type area-id-type;
                description
                    "Multi-area adjacency area ID.";
            }
            leaf cost {
                type uint16;
                description
                    "Interface cost for multi-area adjacency.";
            }
        }
    }

```

```

    }
  }
}

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

  list neighbor {
    key "identifier";
    description
      "Specify a static OSPF neighbor.";

    leaf identifier {
      type inet:ip-address;
      description
        "Neighbor router ID, IPv4 address, or IPv6 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 in seconds.";
  }
  leaf priority {
    type uint8 {
      range "1..255";
    }
    description "Neighbor priority for DR election.";
  }
}
}

leaf node-flag {

```



```

    if-feature node-flag;
    type boolean;
    default false;
    description
        "Set prefix as a node representative prefix.";
}

container bfd {
    if-feature bfd;
    description "BFD configuration.";
    leaf enable {
        type boolean;
        default false;
        description
            "True if BFD is enabled for the OSPF interface.";
    }
}

uses interface-fast-reroute-config;
uses interface-common-config;
uses interface-physical-link-config;
} // grouping interface-config

grouping neighbor-state {
    description
        "OSPF neighbor operational state.";

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

```

```

leaf dr-router-id {
    type rt-types:router-id;
    config false;
    description "Neighbor's Designated Router (DR) router ID.";
}

leaf dr-ip-addr {
    type inet:ip-address;
}

```

```

        config false;
        description "Neighbor's Designated Router (DR) IP address.";
    }

    leaf bdr-router-id {
        type rt-types:router-id;
        config false;
        description
            "Neighbor's Backup Designated Router (BDR) router ID.";
    }

    leaf bdr-ip-addr {
        type inet:ip-address;
        config false;
        description
            "Neighbor's Backup Designated Router (BDR) IP Address.";
    }

    leaf state {
        type nbr-state-type;
        config false;
        description
            "OSPF neighbor state.";
    }

    leaf dead-timer {
        type uint32;
        units "seconds";
        config false;
        description "This timer tracks the remaining time before
                    the neighbor is declared dead.";
    }

    container statistics {
        config false;
        description "Per neighbor statistics";
        uses neighbor-stat;
    }
}

grouping interface-common-state {
    description
        "OSPF interface common operational state.";
}

```

```

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

leaf hello-timer {
    type uint32;
    units "seconds";
    config false;
    description "This timer tracks the remaining time before
                the next hello packet is sent.";
}

leaf wait-timer {
    type uint32;
    units "seconds";
    config false;
    description "This timer tracks the remaining time before
                the interface exits the Waiting state.";
}

leaf dr-router-id {
    type rt-types:router-id;
    config false;
    description "Designated Router (DR) router ID.";
}

leaf dr-ip-addr {
    type inet:ip-address;
    config false;
    description "Designated Router (DR) IP address.";
}

leaf bdr-router-id {
    type rt-types:router-id;
    config false;
    description "Backup Designated Router (BDR) router ID.";
}

leaf bdr-ip-addr {
    type inet:ip-address;
    config false;
    description "Backup Designated Router (BDR) IP Address.";
}

```

```
container statistics {
  config false;
  description "Per interface statistics";
  uses interface-stat;
}

container neighbors {
  config false;
  description "All neighbors for the interface.";
  list neighbor {
    key "neighbor-router-id";
    description
      "List of OSPF neighbors.";
    leaf neighbor-router-id {
      type rt-types:router-id;
      description
        "Neighbor router ID.";
    }
    uses neighbor-state;
  } // list of OSPF neighbors
}

container database {
  config false;
  description "Link scope LSA database.";
  list link-scope-lsa-type {
    key "lsa-type";
    description
      "List OSPF link scope LSA databases.";
    leaf lsa-type {
      type uint16;
      description "OSPF link scope LSA type.";
    }
  }
  container link-scope-lsas {
    description
      "All link scope LSAs of this LSA type.";
    list link-scope-lsa {
      key "lsa-id adv-router";
      description "List of OSPF link scope LSAs";
      uses lsa-key;
      uses lsa {
        refine "version/ospfv2/ospfv2" {
          must "../..../..../..../..../..../..../..../..../.."
            + "rt:type = 'ospf:ospfv2'" {
            description "OSPFv2 LSA.";
          }
        }
      }
    }
  }
}
```

```
refine "version/ospfv3/ospfv3" {  
    must "../../../../../../../../../../../../../../../"
```

```
        + "rt:type = 'ospf:ospfv3'" {  
            description "OSPFv3 LSA.";  
        }  
    }  
} // list link-scope-lsas  
}  
} // interface-common-state  
  
grouping interface-state {  
    description  
        "OSPF interface operational state.";  
    reference "RFC2328 Section 9";  
  
    uses interface-common-state;  
}  
  
grouping virtual-link-config {  
    description  
        "OSPF virtual link configuration state.";  
  
    uses interface-common-config;  
}  
  
grouping virtual-link-state {  
    description  
        "OSPF virtual link operational state.";  
  
    leaf cost {  
        type uint16 {  
            range "1..65535";  
        }  
        config false;  
        description  
            "Virtual link interface cost.";  
    }  
    uses interface-common-state;
```

```

}

grouping sham-link-config {
  description
    "OSPF sham link configuration state.";

  uses interface-common-config;
  uses interface-physical-link-config;
}

```

```

grouping sham-link-state {
  description
    "OSPF sham link operational state.";
  /* All container/leaf should be config false. */
  uses interface-common-state;
}

grouping af-area-config {
  description
    "OSPF address-family specific area config state.";

  container ranges {
    description "Container for summary ranges";

    list range {
      key "prefix";
      description
        "Summarize routes matching address/mask
        (Area Border Routers (ABRs) 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
        "Advertised cost of summary route.";
    }
}
}
}

```

```

grouping area-common-config {
    description
        "OSPF area common configuration state.";

    leaf summary {
        when "../area-type = 'ospf:stub' or "
            + "../area-type = 'ospf:nssa'" {
            description

```

```

        "Summary advertisement into the stub/NSSA area.";
    }
    type boolean;
    description
        "Enable/Disable summary advertisement into the stub or
        NSSA area.";
}
leaf default-cost {
    when "../area-type = 'ospf:stub' or "
        + "../area-type = 'ospf:nssa'" {
        description
            "Cost for LSA default route advertised into the
            stub or NSSA area.";
    }
    type uint32 {
        range "1..16777215";
    }
    description
        "Set the summary default route cost for a
        stub or NSSA area.";
}
}
}

```

```

grouping area-config {
    description

```

```

    "OSPF area configuration state.";

leaf area-type {
    type identityref {
        base area-type;
    }
    default normal;
    description
        "Area type.";
}

uses area-common-config;

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

grouping area-state {

```

```

description
    "OSPF area operational state.";

container statistics {
    config false;
    description "Per area statistics";
    uses area-stat;
}

container database {
    config false;
    description "Area scope LSA database.";
    list area-scope-lsa-type {
        key "lsa-type";
        description "List OSPF area scope LSA databases.";
        leaf lsa-type {
            type uint16;
            description "OSPF area scope LSA type.";
        }
    }
}

```



```

    }
    container area-scope-lsas {
        description
            "All area scope LSAs of an area scope
            LSA type.";
        list area-scope-lsa {
            key "lsa-id adv-router";
            description "List of OSPF area scope LSAs";
            uses lsa-key;
            uses lsa {
                refine "version/ospfv2/ospfv2" {
                    must "../.../.../.../.../.../.../.../.../"
                        + "rt:type = 'ospf:ospfv2'" {
                        description "OSPFv2 LSA.";
                    }
                }
                refine "version/ospfv3/ospfv3" {
                    must "../.../.../.../.../.../.../.../.../"
                        + "rt:type = 'ospf:ospfv3'" {
                        description "OSPFv3 LSA.";
                    }
                }
            }
        }
    } // list area-scope-lsas
}

grouping local-rib {

```

```

description "Local-rib grouping.";
container local-rib {
    config false;
    description "Local-rib.";
    list route {
        key "prefix";
        description "Routes";
        leaf prefix {
            type inet:ip-prefix;
            description "Destination prefix.";
        }
    }
}

```

```

    container next-hops {
      description "All next hops for the route.";
      list next-hop {
        key "next-hop";
        description "List of next hop for the route";
        leaf outgoing-interface {
          type if:interface-ref;
          description
            "Name of the outgoing interface.";
        }
        leaf next-hop {
          type inet:ip-address;
          description "Nexthop address.";
        }
      }
    }
    leaf metric {
      type uint32;
      description "Metric for this route.";
    }
    leaf route-type {
      type route-type;
      description "Route type for this route.";
    }
    leaf route-tag {
      type uint32;
      description "Route tag for this route.";
    }
  }
}

```

```

grouping ietf-spf-delay-config {
  leaf initial-delay {
    type uint16;
    units msec;
    description

```

```

      "Delay used while in QUIET state.";
    }
    leaf short-delay {
      type uint16;

```

```

        units msec;
        description
            "Delay used while in SHORT_WAIT state.";
    }
    leaf long-delay {
        type uint16;
        units msec;
        description
            "Delay used while in LONG_WAIT state.";
    }
    leaf hold-down {
        type uint16;
        units msec;
        description
            "Timer used to consider an IGP stability period.";
    }
    leaf time-to-learn {
        type uint16;
        units msec;
        description
            "Duration used to learn all the IGP events
             related to a single component failure.";
    }
    description
        "Grouping for IETF SPF delay configuration.";
}

grouping ietf-spf-delay-state {
    leaf current-state {
        type enumeration {
            enum "QUIET" {
                description "QUIET state";
            }
            enum "SHORT_WAIT" {
                description "SHORT_WAIT state";
            }
            enum "LONG_WAIT" {
                description "LONG_WAIT state";
            }
        }
        config false;
        description
            "Current state of the algorithm.";
    }
}

```

```
    leaf remaining-time-to-learn {
      type uint16;
      units "seconds";
      config false;
      description
        "Remaining time until time-to-learn timer fires.";
    }
    leaf remaining-hold-down {
      type uint16;
      units "seconds";
      config false;
      description
        "Remaining time until hold-down timer fires.";
    }
    leaf last-event-received {
      type yang:timestamp;
      config false;
      description
        "Time of last IGP event received";
    }
    leaf next-spf-time {
      type yang:timestamp;
      config false;
      description
        "Time when next SPF has been scheduled.";
    }
    leaf last-spf-time {
      type yang:timestamp;
      config false;
      description
        "Time of last SPF computation.";
    }
    description
      "Grouping for IETF SPF delay operational states.";
  }

  grouping node-tag-config {
    description
      "OSPF node tag config state.";
    container node-tags {
      if-feature node-tag;
      list node-tag {
        key tag;
        leaf tag {
          type uint32;
          description
            "Node tag value.";
        }
      }
    }
  }
```

```
}
```

```
        description
            "List of tags.";
    }
    description
        "Container for node tags.";
}

grouping instance-config {
    description
        "OSPF instance config state.";

    leaf explicit-router-id {
        if-feature explicit-router-id;
        type rt-types:router-id;
        description
            "Defined in RFC 2328. A 32-bit number
            that uniquely identifies the router.";
    }

    container preference {
        description "Route preference config state.";
        choice scope {
            description
                "Options for expressing preference
                as single or multiple values.";
            case single-value {
                leaf all {
                    type uint8;
                    description
                        "Preference for intra-area, inter-area and
                        external routes.";
                }
            }
            case multi-values {
                choice granularity {
                    description
                        "Options for expressing preference
                        for intra-area and inter-area routes.";
                    case detail {
```

```

    leaf intra-area {
        type uint8;
        description
            "Preference for intra-area routes.";
    }
    leaf inter-area {
        type uint8;
        description

```

```

        "Preference for inter-area routes.";
    }
}
case coarse {
    leaf internal {
        type uint8;
        description
            "Preference for both intra-area and
            inter-area routes.";
    }
}
}
leaf external {
    type uint8;
    description
        "Preference for external routes.";
}
}
}

container nsr {
    if-feature nsr;
    description
        "Non-Stop Routing (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
        for OSPFv2 and RFC 5187 for OSPFv3.";
}
leaf helper-enable {
    type boolean;
    description
        "Enable graceful restart helper support for restarting
        routers (RFC 3623 Section 3).";
}

```

```

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
        "Interval in seconds to attempt graceful restart prior
        to failing (RFC 3623 Section B.1)";
}
leaf helper-strict-lsa-checking {
    type boolean;
    description
        "Terminate graceful restart when an LSA topology change
        is detected (RFC 3623 Section B.2).";
}
}

leaf enable {
    if-feature admin-control;
    type boolean;
    default true;
    description
        "Enable/Disable the protocol.";
}

```

```

container auto-cost {
  if-feature auto-cost;
  description
    "Interface Auto-cost configuration state.";
  leaf enable {
    type boolean;
    description
      "Enable/Disable interface auto-cost.";
  }
  leaf reference-bandwidth {
    when "../enable = 'true'" {
      description "Only when auto cost is enabled";
    }
    type uint32 {
      range "1..4294967";
    }
    units Mbits;
    description
      "Configure reference bandwidth used to automatically
       determine interface cost (Mbits). The cost is the
       reference bandwidth divided by the interface speed
       with 1 being the minimum cost.";
  }
}

```

```

}
}

container spf-control {
  leaf paths {
    if-feature max-ecmp;
    type uint16 {
      range "1..32";
    }
    description
      "Maximum number of Equal-Cost Multi-Path (ECMP) paths.";
  }
  container ietf-spf-delay {
    if-feature ietf-spf-delay;
    uses ietf-spf-delay-config;
    description
      "IETF spf delay algorithm configuration.";
  }
  description "SPF calculation control.";
}

```



```

}

container database-control {
  leaf max-lsa {
    if-feature max-lsa;
    type uint32 {
      range "1..4294967294";
    }
    description
      "Maximum number of LSAs OSPF the router will accept.";
  }
  description "Database maintenance control.";
}

container stub-router {
  if-feature stub-router;
  description "Set maximum metric configuration";

  choice trigger {
    description
      "Specific triggers which will enable stub
      router state.";
    container always {
      presence
        "Enables unconditional stub router support";
      description
        "Unconditional stub router state (advertise
        transit links with max metric";
    }
  }
}

```

```

}
}

container mpls {
  description
    "OSPF MPLS config state.";
  container te-rid {
    if-feature te-rid;
    description
      "Stable OSPF Router IP Address used for Traffic
      Engineering (TE)";
    leaf ipv4-router-id {

```

```

        type inet:ipv4-address;
        description
            "Explicitly configure the TE IPv4 router ID.";
    }
    leaf ipv6-router-id {
        type inet:ipv6-address;
        description
            "Explicitly configure the TE IPv6 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.";
    }
}
}
uses instance-fast-reroute-config;
uses node-tag-config;
}

grouping instance-state {
    description
        "OSPF instance operational state.";

    leaf router-id {
        type rt-types:router-id;
        config false;
        description
            "Defined in RFC 2328. A 32-bit number
            that uniquely identifies the router.";
    }
}

```

```

uses local-rib;

container statistics {
    config false;
    description "Per instance statistics";
}

```

```

    uses instance-stat;
}

container ietf-spf-delay {
    if-feature ietf-spf-delay;
    config false;
    uses ietf-spf-delay-config;
    uses ietf-spf-delay-state;
    description
        "IETF SPF delay operational states.";
}

container database {
    config false;
    description "AS scope LSA database.";
    list as-scope-lsa-type {
        key "lsa-type";
        description "List OSPF AS scope LSA databases.";
        leaf lsa-type {
            type uint16;
            description "OSPF AS scope LSA type.";
        }
        container as-scope-lsas {
            description "All AS scope of LSA of this LSA type.";
            list as-scope-lsa {
                key "lsa-id adv-router";
                description "List of OSPF area scope LSAs";
                uses lsa-key;
                uses lsa {
                    refine "version/ospfv2/ospfv2" {
                        must "../.../rt:type = "
                            + "'ospf:ospfv2'" {
                            description "OSPFv2 LSA.";
                        }
                    }
                    refine "version/ospfv3/ospfv3" {
                        must "../.../rt:type = "
                            + "'ospf:ospfv3'" {
                            description "OSPFv3 LSA.";
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    } // list as-scope-lsas
}
uses spf-log;
uses lsa-log;
}

grouping ospf-config {
    description
        "OSPF top configuration state.";

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

grouping ospf-state {
    /* All leaf/container must be config false. */
    description
        "OSPF top operational state.";
}

grouping multi-topology-area-common-config {
    description
        "OSPF multi-topology area common configuration state.";
    leaf summary {
        when "../..../..../areas/area[area-id=current()/../area-id]/"
            + "area-type = 'ospf:stub' or "
            + "../..../..../areas/area[area-id=current()/../area-id]/"
            + "area-type = 'ospf:nssa'" {
            description
                "Summary advertisement into the stub/NSSA area.";
        }
        type boolean;
        description
            "Enable/Disable summary advertisement into the
            topology in the stub or NSSA area.";
    }
    leaf default-cost {
        when "../..../..../areas/area[area-id=current()/../area-id]/"
            + "area-type = 'ospf:stub' or "
            + "../..../..../areas/area[area-id=current()/../area-id]/"
            + "area-type = 'ospf:nssa'" {

```

```
        description
            "Cost for LSA default route advertised into the
            topology into the stub or NSSA area.";
    }
    type uint32 {
        range "1..16777215";
    }
    description
        "Set the summary default route cost for a
        stub or NSSA area.";
}
}

grouping multi-topology-area-config {
    description
        "OSPF multi-topology area configuration state.";

    uses multi-topology-area-common-config;

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

grouping multi-topology-area-state {
    /* All leaf/container must be config false. */
    description
        "OSPF multi-topology area operational state.";
}

grouping multi-topology-config {
    description
        "OSPF multi-topology configuration state.";
}

grouping multi-topology-state {
    /* All leaf/container must be config false. */
    description
```

```

        "OSPF multi-topology operational state.";

    uses local-rib;
}

grouping multi-topology-interface-config {

```

```

    description
        "OSPF multi-topology configuration state.";

    leaf cost {
        type uint32;
        description
            "Interface cost for this topology.";
    }
}

grouping multi-topology-interface-state {
    /* All leaf/container must be config false. */
    description
        "OSPF multi-topology operational state.";
}

grouping ospfv3-interface-config {
    description
        "OSPFv3 interface specific configuration state.";

    leaf instance-id {
        type uint8 {
            range "0 .. 31";
        }
        description
            "OSPFv3 instance ID.";
    }
}

grouping ospfv3-interface-state {
    description
        "OSPFv3 interface specific operational state.";

    leaf interface-id {
        type uint16;
    }
}

```

```

        config false;
        description
            "OSPFv3 interface ID.";
    }
}

grouping lsa-identifiers {
    description
        "The parameters that uniquely identify an LSA.";
    leaf area-id {
        type area-id-type;
        description
            "Area ID";
    }
}

```

```

    }
    leaf link-id {
        type union {
            type inet:ipv4-address;
            type yang:dotted-quad;
        }
        description "Link ID.";
    }
    leaf type {
        type uint16;
        description
            "LSA type.";
    }
    leaf lsa-id {
        type yang:dotted-quad;
        description "LSA ID.";
    }
    leaf adv-router {
        type yang:dotted-quad;
        description
            "LSA advertising router.";
    }
    leaf seq-num {
        type uint32;
        description
            "LSA sequence number.";
    }
}

```

```

grouping spf-log {
  description
    "Grouping for SPF log.";
  container spf-log {
    config false;
    description
      "This container lists the SPF log.";
    list event {
      key id;
      description
        "List of SPF logs.
        It is used as a wrapping buffer.";
      leaf id {
        type uint32;
        description
          "This leaf defines the event identifier.
          This is a purely internal value.";
      }
      leaf spf-type {

```

```

  type enumeration {
    enum full {
      description
        "Computation done is a Full SPF.";
    }
    enum intra {
      description
        "Computation done is only for intra-area routes.";
    }
    enum inter {
      description
        "Computation done is only for inter-area
        summary routes.";
    }
    enum external {
      description
        "Computation done is only for AS external routes.";
    }
  }
  description
    "The SPF computation type.";

```



```

    }
    leaf schedule-timestamp {
        type yang:timestamp;
        description
            "This leaf describes the timestamp
            when the computation was scheduled.";
    }
    leaf start-timestamp {
        type yang:timestamp;
        description
            "This leaf describes the timestamp
            when the computation was started.";
    }
    leaf end-timestamp {
        type yang:timestamp;
        description
            "This leaf describes the timestamp
            when the computation was completed.";
    }
    list trigger-lsa {
        description
            "The list of LSAs that triggered the computation.";
        uses lsa-identifiers;
    }
}
}
}

```

```

grouping lsa-log {
    description
        "Grouping for LSA log.";
    container lsa-log {
        config false;
        description
            "This conatiner lists the LSA log.
            Local LSA modifications are also included
            in the list.";
        list event {
            key id;
            description
                "List of LSA logs.
                It is used as a wrapping buffer.";
        }
    }
}

```

```

    leaf id {
        type uint32;
        description
            "This leaf defines the event identifier.
            This is a purely internal value.";
    }
    container lsa {
        description
            "This container describes the logged LSA.";
        uses lsa-identifiers;
    }
    leaf received-timestamp {
        type yang:timestamp;
        description
            "This leaf describes the timestamp
            when the LSA was received. In case of
            local LSA update, the timestamp refers
            to the local LSA update time.";
    }
    leaf reason {
        type identityref {
            base lsa-log-reason;
        }
        description
            "This leaf describes the reason
            that resulted in this LSA log.";
    }
}
}
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol" {
    when "rt:type = 'ospf:ospfv2' or rt:type = 'ospf:ospfv3'" {

```

```

    description
        "This augmentation is only valid for a routing protocol
        instance of OSPF (type 'ospfv2' or 'ospfv3').";
    }
    description "OSPF augmentation.";

    container ospf {

```

```

description
  "OSPF.";

uses ospf-config;
uses ospf-state;

list instance {
  key "af";
  description
    "An OSPF routing protocol instance.";

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

  uses instance-config;
  uses instance-state;

  container areas {
    description "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;
      uses area-state;

      container virtual-links {
        when "../area-id = '0.0.0.0' and "
          + "../area-type = 'ospf:normal'" {
          description

```

```

        "Virtual links must be in backbone area.";
    }
description "All virtual links.";
list virtual-link {
    key "transit-area-id router-id";
    description
        "OSPF virtual link";
    leaf transit-area-id {
        type leafref {
            path "../..../..../area/area-id";
        }
        must "../..../..../area[area-id=current()]/"
            + "area-id != '0.0.0.0' and "
            + "../..../..../area[area-id=current()]/"
            + "area-type = 'ospf:normal'" {
            error-message "Virtual link transit area must "
                + "be non-zero.";
            description
                "Virtual-link trasit area must be
                non-zero area.";
        }
    }
    description
        "Virtual link tranist area ID.";
}
leaf router-id {
    type rt-types:router-id;
    description
        "Virtual Link remote endpoint router ID.";
}

uses virtual-link-config;
uses virtual-link-state;
}
}
container sham-links {
    if-feature pe-ce-protocol;
    description "All sham links.";
    list sham-link {
        key "local-id remote-id";
        description
            "OSPF sham link";
        leaf local-id {
            type inet:ip-address;
            description
                "Address of the local Sham Link endpoint.";
        }
        leaf remote-id {
            type inet:ip-address;

```

```
        description
            "Address of the remote Sham Link endpoint.";
    }
    uses sham-link-config;
    uses sham-link-state;
}
}
container interfaces {
    description "All interfaces.";
    list interface {
        key "name";
        description
            "List of OSPF interfaces.";
        leaf name {
            type if:interface-ref;
            description
                "Interface name.";
        }
        uses interface-config;
        uses interface-state;
    } // list of interfaces
}
} // list of areas
}
} // list of instance
} // container ospf
}

augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/ospf:ospf/ospf:instance" {
    when "../..//rt:type = 'ospf:ospfv2' or
        ../..//rt:type = 'ospf:ospfv3'" {
        description
            "This augmentation is only valid for OSPF
            (type 'ospfv2' or 'ospfv3').";
    }
    if-feature multi-topology;
    description
        "OSPF multi-topology instance configuration
        state augmentation.";
    container topologies {
        description "All topologies.";
        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 leafref {
      path "../..../..../rt:ribs/rt:rib/rt:name";
    }
    description "RIB";
  }

  uses multi-topology-config;
  uses multi-topology-state;

  container areas {
    description "All areas in the topology.";
    list area {
      key "area-id";
      description
        "List of OSPF areas";
      leaf area-id {
        type area-id-type;
        description
          "Area ID.";
      }
      uses multi-topology-area-config;
      uses multi-topology-area-state;
    }
  }
}

augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/ospf:ospf/ospf:instance/"
  + "ospf:areas/ospf:area/ospf:interfaces/ospf:interface" {
  when "../..../..../rt:type = 'ospf:ospfv2'" {
    description
      "This augmentation is only valid for OSPFv2.";
  }
  if-feature ospf:multi-topology;
```

```

description
  "OSPF multi-topology interface configuration state
  augmentation.";
container topologies {
  description "All topologies for the interface.";
  list topology {
    key "name";
    description "OSPF interface topology.";
    leaf name {
      type leafref {
        path "../..../..../..../..../..../..../..../"
          + "rt:ribs/rt:rib/rt:name";
      }
    }
  }
}

```

```

    }
    description
      "One of the topologies enabled on this interface.";
  }

  uses multi-topology-interface-config;
  uses multi-topology-interface-state;
}
}

augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/ospf:ospf/ospf:instance/"
  + "ospf:areas/ospf:area/ospf:interfaces/ospf:interface" {
  when "../..../..../..../..../rt:type = 'ospf:ospfv3'" {
    description
      "This augmentation is only valid for OSPFv3.";
  }
  description
    "OSPFv3 interface specific configuration state
    augmentation.";
  uses ospfv3-interface-config;
  uses ospfv3-interface-state;
}

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 route-type;
        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 augmentation is only valid for a routes whose
        source protocol is OSPF.";
    }
    description
        "OSPF-specific route attributes.";
    uses route-content;
}

/*
 * RPCs
 */

rpc clear-neighbor {
    description
        "This RPC request clears a particular
        set of OSPF neighbors. If the operation
        fails for OSPF internal reason, then
        error-tag and error-app-tag should be set
        to a meaningful value.";
    input {
        leaf routing-protocol-name {
            type leafref {

```



```

        path "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol/rt:name";
    }
    mandatory "true";
    description
        "Name of the OSPF protocol instance which information
        is being queried.

        If the OSPF instance with name equal to the
        value of this parameter doesn't exist, then this
        operation SHALL fail with error-tag 'data-missing'
        and error-app-tag
        'routing-protocol-instance-not-found'.";
    }

    leaf interface {
        type if:interface-ref;
        description
            "Name of the OSPF interface.

            If the OSPF interface with name equal to the
            value of this parameter doesn't exist, then this
            operation SHALL fail with error-tag 'data-missing'
            and error-app-tag
            'ospf-interface-not-found'.";
    }

```

```

    }
}

rpc clear-database {
    description
        "This RPC request clears a particular
        OSPF database. If the operation
        fails for OSPF internal reason, then
        error-tag and error-app-tag should be set
        to a meaningful value.";
    input {
        leaf routing-protocol-name {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols/"
                    + "rt:control-plane-protocol/rt:name";
            }
        }
    }
}

```

```

    }
    mandatory "true";
    description
        "Name of the OSPF protocol instance whose
        information is being queried.

        If the OSPF instance with name equal to the
        value of this parameter doesn't exist, then this
        operation SHALL fail with error-tag 'data-missing'
        and error-app-tag
        'routing-protocol-instance-not-found'.";
    }
}

/*
 * Notifications
 */

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

    leaf routing-protocol-name {
        type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
                + "rt:control-plane-protocol/rt:name";
        }
        must "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol[rt:name=current()]/"
            + "rt:type = 'ospf:ospfv2' or "
            + "/rt:routing/rt:control-plane-protocols/"

```

```

        + "rt:control-plane-protocol[rt:name=current()]/"
        + "rt:type = 'ospf:ospfv3'";
    description
        "OSPF routing protocol instance name.";
}

leaf af {
    type leafref {

```

```

    path "/rt:routing/"
      + "rt:control-plane-protocols/rt:control-plane-protocol"
      + "[rt:name=current()/../routing-protocol-name]/"
      + "ospf:ospf/ospf:instance/af";
  }
  description
    "Address family of the OSPF instance.";
}
}

```

```

grouping notification-interface {
  description
    "This grouping provides the interface information
    for the notifications.";

```

```

  choice if-link-type-selection {
    description
      "Options for link type.";
    container interface {
      description "Normal interface.";
      leaf interface {
        type if:interface-ref;
        description "Interface.";
      }
    }
    container virtual-link {
      description "virtual-link.";
      leaf transit-area-id {
        type area-id-type;
        description "Area ID.";
      }
      leaf neighbor-router-id {
        type rt-types:router-id;
        description "Neighbor Router ID.";
      }
    }
    container sham-link {
      description "sham-link.";
      leaf area-id {
        type area-id-type;

```

```

    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.";
    }
}
}

grouping notification-neighbor {
    description
        "This grouping provides the neighbor information
        for the notifications.";

    leaf neighbor-router-id {
        type rt-types:router-id;
        description "Neighbor Router ID.";
    }

    leaf neighbor-ip-addr {
        type yang:dotted-quad;
        description "Neighbor address.";
    }
}

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

    leaf state {
        type if-state-type;
        description "Interface state.";
    }
    description
        "This notification is sent when an interface
        state change is detected.";
}

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

    leaf packet-source {

```

```
    type yang:dotted-quad;
    description "Source address.";
}

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

leaf error {
    type enumeration {
        enum "badVersion" {
            description "Bad version.";
        }
        enum "areaMismatch" {
            description "Area mismatch.";
        }
        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;
    uses notification-interface;
    uses notification-neighbor;

    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;
    uses notification-interface;
    uses notification-neighbor;

    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 a neighbor restart
    helper status change is detected.";
}

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

    leaf packet-source {
        type yang:dotted-quad;
        description "Source address.";
    }

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

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

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 area-id-type;
            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 role in translating OSPF NSSA LSAs
        to 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 YANG module defined in this document is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport

layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [[RFC6242](#)]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [[RFC5246](#)].

The NETCONF access control model [[RFC6536](#)] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. For OSPF, the ability to modify OSPF configuration will allow the entire OSPF domain to be compromised including peering with unauthorized routers to misroute traffic or mount a massive Denial-of-Service (DoS) attack. The security considerations of OSPFv2 [[RFC2328](#)] and [[RFC5340](#)].

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. The exposure of the Link State Database (LSDB) will expose the detailed topology of the network. This may be undesirable since both due to the fact that exposure may facilitate other attacks. Additionally, network operators may consider their topologies to be proprietary.

For OSPF authentication, configuration is supported via the specification of key-chains [[RFC8177](#)] or the direct specification of key and authentication algorithm. Hence, authentication configuration using the "auth-table-trailer" case in the "authentication" container inherits the security considerations of [[RFC8177](#)]. This includes the considerations with respect to the local storage and handling of authentication keys.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. The OSPF Yang

module support the "clear-neighbor" and "clear-database" RPCs. If access to either of these is compromised, they can result in temporary network outages being employed to mount DoS attacks.

[5.](#) Acknowledgements

The authors wish to thank Yi Yang, Alexander Clemm, Gaurav Gupta, Ladislav Lhotka, Stephane Litkowski, Greg Hankins, Manish Gupta and Alan Davey 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](#), DOI 10.17487/RFC1793, April 1995, <<http://www.rfc-editor.org/info/rfc1793>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, [RFC 2328](#), DOI 10.17487/RFC2328, April 1998, <<http://www.rfc-editor.org/info/rfc2328>>.
- [RFC3101] Murphy, P., "The OSPF Not-So-Stubby Area (NSSA) Option", [RFC 3101](#), DOI 10.17487/RFC3101, January 2003, <<http://www.rfc-editor.org/info/rfc3101>>.
- [RFC3623] Moy, J., Pillay-Esnault, P., and A. Lindem, "Graceful OSPF Restart", [RFC 3623](#), DOI 10.17487/RFC3623, November 2003, <<http://www.rfc-editor.org/info/rfc3623>>.

- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", [RFC 3630](#), DOI 10.17487/RFC3630, September 2003, <<http://www.rfc-editor.org/info/rfc3630>>.

- [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](#), DOI 10.17487/RFC4577, June 2006, <<http://www.rfc-editor.org/info/rfc4577>>.
- [RFC4750] Joyal, D., Ed., Galecki, P., Ed., Giacalone, S., Ed., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", [RFC 4750](#), DOI 10.17487/RFC4750, December 2006, <<http://www.rfc-editor.org/info/rfc4750>>.
- [RFC5187] Pillay-Esnault, P. and A. Lindem, "OSPFv3 Graceful Restart", [RFC 5187](#), DOI 10.17487/RFC5187, June 2008, <<http://www.rfc-editor.org/info/rfc5187>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", [RFC 5340](#), DOI 10.17487/RFC5340, July 2008, <<http://www.rfc-editor.org/info/rfc5340>>.
- [RFC5643] Joyal, D., Ed. and V. Manral, Ed., "Management Information Base for OSPFv3", [RFC 5643](#), DOI 10.17487/RFC5643, August 2009, <<http://www.rfc-editor.org/info/rfc5643>>.
- [RFC5838] Lindem, A., Ed., Mirtorabi, S., Roy, A., Barnes, M., and R. Aggarwal, "Support of Address Families in OSPFv3", [RFC 5838](#), DOI 10.17487/RFC5838, April 2010, <<http://www.rfc-editor.org/info/rfc5838>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.

- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), DOI 10.17487/RFC6242, June 2011, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6565] Pillay-Esnault, P., Moyer, P., Doyle, J., Ertekin, E., and M. Lundberg, "OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol", [RFC 6565](#), DOI 10.17487/RFC6565, June 2012, <<http://www.rfc-editor.org/info/rfc6565>>.
- [RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", [RFC 7223](#), DOI 10.17487/RFC7223, May 2014, <<http://www.rfc-editor.org/info/rfc7223>>.
- [RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", [RFC 8022](#), DOI 10.17487/RFC8022, November 2016, <<http://www.rfc-editor.org/info/rfc8022>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [RFC 8040](#), DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", [RFC 8177](#), DOI 10.17487/RFC8177, June 2017, <<http://www.rfc-editor.org/info/rfc8177>>.

[6.2](#). Informative References

- [I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", [draft-ietf-netmod-revised-datastores-02](#) (work in progress), May 2017.

Internet-Draft

OSPF Yang Data Model

July 2017

[Appendix A](#). Contributors' Addreses

Dean Bogdanovic
Volta Networks, Inc.

EMail: dean@voltanet.io

Kiran Koushik Agrahara Sreenivasa
Cisco Systems
12515 Research Blvd, Bldg 4
Austin, TX 78681
USA

EMail: kkoushik@cisco.com

Authors' Addresses

Derek Yeung
Arrcus

EMail: derek@arrcus.com

Yingzhen Qu
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

EMail: yingzhen.qu@huawei.com

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

EMail: zzhang@juniper.net

Ing-Wher Chen
Jabil

EMail: ing-wher_chen@jabil.com

Yeung, et al.

Expires January 3, 2018

[Page 104]

Internet-Draft

OSPF Yang Data Model

July 2017

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513

EMail: acee@cisco.com

