Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 30, 2015

L. Wang
S. Hares
E. Wu
Huawei

September 26, 2014

Yang Data model I2RS to IS-IS protocol
draft-wang-i2rs-isis-dm-00

Abstract

IS-IS is a widely deployed link-state protocol in routing networks. During the past decades, it has been operated and maintained through typical CLI, SNMP and NETCONF. With the expansion and complication of modern networks, the necessity for rapid and dynamic control has been increased. The I2RS is a standard-based interface which provides a programmatic way to achieve this goal.

This document specifies a data model for the I2RS's interface to the IS-IS protocol based the I2RS IS-IS informational model (draft-ietf-wu-info-model-00). These two models satisfy the requirements suggested by the I2RS use case requirements for the IGPs. This yang data model can be used by I2RS client-agent protocol to program IS-IS routing entities.

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 RFC 2119 [RFC2119].

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 March 30, 2015.

Table of Contents

1. Introduction

IS-IS[ISO.10589.1992] is a widely deployed link-state protocol in routing networks. During the past decades, it has been operated and maintained through typical CLI, SNMP and NETCONF. With the expansion and complication of modern networks, the necessity for rapid and dynamic control has been increased. The I2RS [I-D.ietf-i2rs-architecture] is a standard-based interface which provides a programmatic way to achieve this goal.

The draft-wu-i2rs-info-model specifies an I2RS information model (IM) for IS-IS. This I2RS yang data model (DM) that instantiates the information model in a data model. This pair of ISIS IM and DM provides the functional suggested in the requirements for the IGP defined by by [I-D.hares-i2rs-usecase-reqs-summary].

In order to support large intra-domain, IS-IS has been organized hierarchically into areas. Routing within an area is referred to as Level 1 routing while routing between areas is referred to as Level 2 routing. Each IS-IS routing system advertises and collects link-state information independently then makes decision in the distributed manner. The outcome of this decision is used to populate Routing Information Base (RIB) since IS-IS can be one of the clients of RIB as stated in [I-D.ietf-i2rs-rib-info-model].

1.1. Yang Tree Diagrams

The Yang Tree diagrams used in this draft utilized a simple graphical representation of the data model. The meaning of the symbols are as follows:

- Brackets "[" and "]" enclose list keys
- Abbreviations before data node names: "rw" mean configuration (read-write) and "ro" state diagrams.
- Symbols after data node names: "?" means an optinal node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipis (". . . ") stand for the contents of subtress that are not shown.

Future yang symbols may be added to indicate the object relationship, ephemeral state, and other I2RS specific relationships in yang 1.1

2. IS-IS data Model

This section describes the data involved in the IS-IS information model in detail. IS-IS data includes information related to IS-IS instance, IS-IS level, IS-IS multi-topology, IS-IS circuits, IS-IS adjacencies and IS-IS routes. A high-level architecture of the IS-IS contents is shown as below.

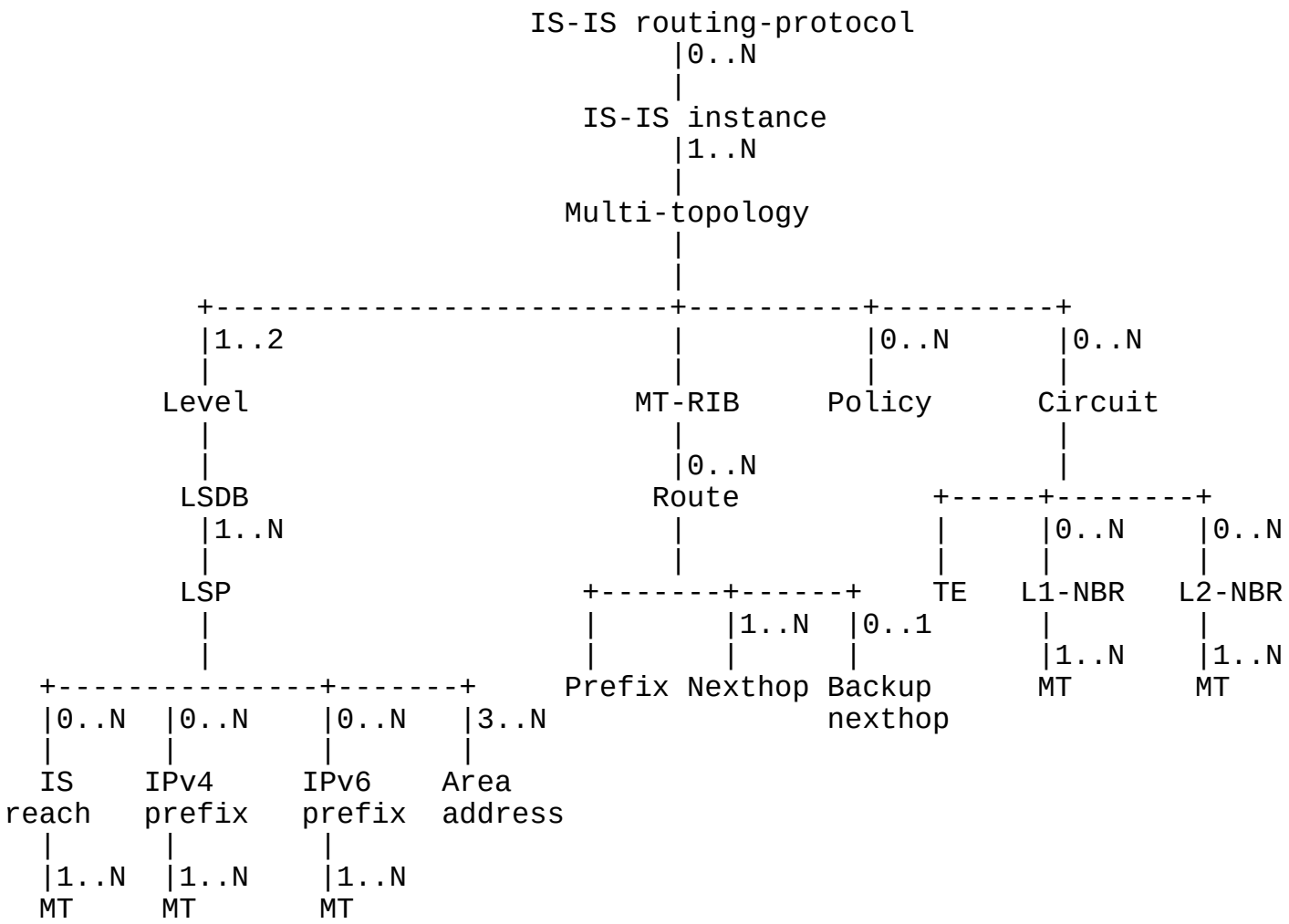

Figure 1: Architecture of IS-IS information model

3. I2RS IS-IS Data Model

```
module: i2rs-isis-protocol
  +--rw isis-routing-protocol
      +--rw isis-instance* [name]
         +--rw name                          string
         +--rw isis-vpn-name?                string
         +--rw address-family                address-family-def
         +--rw net* [area-id]
         |  +--rw area-id      string
         |  +--rw system-id?   isis-system-id-def
         +--ro protocol-status?              protocol-status-def
         +--rw level-type?                   level-type-def
         +--rw metric-style?                 metric-style-def
         +--rw lsp-mtu?                      uint32
```

```
      +--rw lsp-refresh?                uint32
      +--rw lsp-lifetime?               uint32
      +--ro isis-instance-create-mode?  isis-instance-create-mode-def
      +--rw preference?                 uint32
      +--rw hostname?                   string
      +--rw domain-auth-info
      |  +--rw auth-mode
      |  |  +--rw (auth-mode-type)?
      |  |     +--:(mode-simple)
      |  |     |  +--rw simple-password?     string
      |  |     +--:(mode-md5)
      |  |     |  +--rw md5-password?        string
      |  |     +--:(mode-hmac-sha256)
      |  |     |  +--rw hmac-key-id?         uint32
      |  |     |  +--rw hmac-password?       string
      |  |     +--:(mode-keychain)
      |  |        +--rw keychain-key-id?     uint32
      |  |        +--rw keychain-password?   string
      |  |        +--rw keychain-mode?       enumeration
      |  |        +--rw keychain-periodic?   enumeration
      |  |        +--rw send_time?           uint32
      |  |        +--rw receive_time?        uint32
      |  +--rw snp-auth-status?   enumeration
      |  +--rw auth-scope?        enumeration
      +--rw area-auth-info
      |  +--rw auth-mode
      |  |  +--rw (auth-mode-type)?
      |  |     +--:(mode-simple)
      |  |     |  +--rw simple-password?     string
      |  |     +--:(mode-md5)
      |  |     |  +--rw md5-password?        string
      |  |     +--:(mode-hmac-sha256)
      |  |     |  +--rw hmac-key-id?         uint32
      |  |     |  +--rw hmac-password?       string
      |  |     +--:(mode-keychain)
      |  |        +--rw keychain-key-id?     uint32
      |  |        +--rw keychain-password?   string
      |  |        +--rw keychain-mode?       enumeration
      |  |        +--rw keychain-periodic?   enumeration
      |  |        +--rw send_time?           uint32
      |  |        +--rw receive_time?        uint32
      |  +--rw snp-auth-status?   enumeration
      |  +--rw auth-scope?        enumeration
      +--rw multi-topo-list* [mt-id]
         +--rw mt-id             uint16
         +--ro mt-status?        mt-status-def
         +--rw address-family    address-family-def
         +--rw level-list* [level-id]
```

```
             |  +--rw level-id    string
             |  +--rw lsdb
             |     +--rw lsp* [lsp-id sequence-number checksum]
             |     |  +--rw maximum-area-addresses                    uint8
             |     |  +--rw pdu-length                                uint16
             |     |  +--rw remaining-lifetime                        uint16
             |     |  +--rw lsp-id                                    isis-l
sp-id-def
             |     |  +--rw sequence-number                           uint32
             |     |  +--rw checksum                                  uint16
             |     |  +--rw p-bit                                     boolea
n
             |     |  +--rw error-metric-bit                          boolea
n
             |     |  +--rw expense-metric-bit                        boolea
n
             |     |  +--rw delay-metric-bit                          boolea
n
             |     |  +--rw default-metric-bit                        boolea
n
             |     |  +--rw lspdbol-bit?                              enumer
ation
             |     |  +--rw is-type-bit?                              enumer
ation
             |     |  +--rw area-addresse-tlv
             |     |  |  +--rw code      enumeration
             |     |  |  +--rw length    uint8
             |     |  |  +--rw value* [area-addresse]
             |     |  |     +--rw address-length    uint8
             |     |  |     +--rw area-addresse     uint8
             |     |  +--rw intermediate-system-neighbours-tlv
             |     |  |  +--rw code            enumeration
             |     |  |  +--rw length          uint8
             |     |  |  +--rw virtual-flag    boolean
             |     |  |  +--rw value* [neighbor-id]
             |     |  |     +--rw default-metric    uint8
             |     |  |     +--rw delay-metric      uint8
             |     |  |     +--rw expense-metric    uint8
             |     |  |     +--rw error-metric      uint8
             |     |  |     +--rw neighbor-id       string
             |     |  +--rw nlpid-tlv
             |     |  |  +--rw code      enumeration
             |     |  |  +--rw length    uint8
             |     |  |  +--rw value* [nlpid]
             |     |  |     +--rw nlpid    uint8
             |     |  +--rw ip-interface-addresses-tlv
             |     |  |  +--rw code      enumeration
             |     |  |  +--rw length    uint8
             |     |  |  +--rw value* [ip-address]
             |     |  |     +--rw ip-address    inet:ipv4-address
             |     |  +--rw ip-internal-reachability-tlv
             |     |  |  +--rw code      enumeration
             |     |  |  +--rw length    uint8
             |     |  |  +--rw value* [ip-prefix]
```

```
|     |  |     +--rw i-e-bit                boolean
|     |  |     +--rw default-metric         uint8
```

```
|     |  |     +--rw s-bit                    boolean
|     |  |     +--rw delay-metric             uint8
|     |  |     +--rw s-bit-expense-metric     boolean
|     |  |     +--rw expense-metric           uint8
|     |  |     +--rw s-bit-error-metric       boolean
|     |  |     +--rw error-metric             uint8
|     |  |     +--rw ip-prefix                inet:ipv4-prefix
|     |  +--rw ip-external-reachability-tlv
|     |  |  +--rw code      enumeration
|     |  |  +--rw length    uint8
|     |  |  +--rw value* [ip-prefix]
|     |  |     +--rw i-e-bit                  boolean
|     |  |     +--rw default-metric           uint8
|     |  |     +--rw s-bit                    boolean
|     |  |     +--rw delay-metric             uint8
|     |  |     +--rw s-bit-expense-metric     boolean
|     |  |     +--rw expense-metric           uint8
|     |  |     +--rw s-bit-error-metric       boolean
|     |  |     +--rw error-metric             uint8
|     |  |     +--rw ip-prefix                inet:ipv4-prefix
|     |  +--rw inter-domain-information-tlv
|     |  |  +--rw code      enumeration
|     |  |  +--rw length    uint8
|     |  |  +--rw value* [inter-domain-information-type external
-information]
|     |  |     +--rw inter-domain-information-type    uint8
|     |  |     +--rw external-information             uint32
|     |  +--rw extended-is-reachability-tlv
|     |  |  +--rw code      enumeration
|     |  |  +--rw length    uint8
|     |  |  +--rw value* [neighbor-system-id]
|     |  |     +--rw neighbor-system-id
isis-system-id-def
|     |  |     +--rw metric
uint32
|     |  |     +--rw sub-tlv-length
uint8
|     |  |     +--rw administrative-group-stlv
|     |  |     |  +--rw type      enumeration
|     |  |     |  +--rw length?   uint32
|     |  |     |  +--rw value?    uint32
|     |  |     +--rw ipv4-interface-address--stlv* [ipv4-interfa
ce-address]
|     |  |     |  +--rw type                      enumeration
|     |  |     |  +--rw length?                   uint32
|     |  |     |  +--rw ipv4-interface-address    inet:ipv4-addr
ess
|     |  |     +--rw ipv4-neighbor-address--stlv* [ipv4-neighbor
-address]
|     |  |     |  +--rw type                     enumeration
|     |  |     |  +--rw length?                  uint32
|     |  |     |  +--rw ipv4-neighbor-address    inet:ipv4-addre
ss
|     |  |     +--rw maximum-bandwidth--stlv
```

```
|     |  |     |  +--rw type                enumeration
|     |  |     |  +--rw length?             uint32
```

```
|     |  |     |  +--rw maximum-bandwidth?   uint32
|     |  |     +--rw maximum-reservable-link-bandwidth-stlv
|     |  |     |  +--rw type                 enumeration
|     |  |     |  +--rw length?              uint32
|     |  |     |  +--rw maximum-bandwidth?   uint32
|     |  |     +--rw unreserved-bandwidth-stlv
|     |  |     |  +--rw type                    enumeration
|     |  |     |  +--rw length?                 uint32
|     |  |     |  +--rw unreserved-bandwidth?   uint32
|     |  |     +--rw traffic-engineering-default-metric-stlv
|     |  |     |  +--rw type                                    e
numeration
|     |  |     |  +--rw length?                                 u
int32
|     |  |     |  +--rw utraffic-engineering-default-metric?   u
int32
|     |  |     +--rw ipv6-interface-address--stlv* [ipv6-interfa
ce-address]
|     |  |     |  +--rw type                      enumeration
|     |  |     |  +--rw length?                   uint32
|     |  |     |  +--rw ipv6-interface-address    inet:ipv6-addr
ess
|     |  |     +--rw ipv6-neighbor-address--stlv* [ipv6-neighbor
-address]
|     |  |        +--rw type                     enumeration
|     |  |        +--rw length?                  uint32
|     |  |        +--rw ipv6-neighbor-address    inet:ipv6-addre
ss
|     |  +--rw extended-ip-reachability-tlv
|     |  |  +--rw code      enumeration
|     |  |  +--rw length    uint8
|     |  |  +--rw value* [ip-prefix]
|     |  |     +--rw metric             uint32
|     |  |     +--rw up-down-bit        boolean
|     |  |     +--rw sub-TLV-bit        boolean
|     |  |     +--rw prefix-length      uint8
|     |  |     +--rw ip-prefix          inet:ipv4-prefix
|     |  |     +--rw length-sub-TLVs?   uint8
|     |  |     +--rw sub-tlv* [type value]
|     |  |        +--rw type      uint8
|     |  |        +--rw length?   uint32
|     |  |        +--rw value     string
|     |  +--rw ipv6-reachability-tlv
|     |  |  +--rw code      enumeration
|     |  |  +--rw length    uint8
|     |  |  +--rw value
|     |  |     +--rw metric         uint32
|     |  |     +--rw u-bit          boolean
|     |  |     +--rw x-bit          boolean
|     |  |     +--rw s-bit          boolean
|     |  |     +--rw prefix-len     uint8
|     |  |     +--rw prefix-list* [ipv6-prefix]
|     |  |        +--rw ipv6-prefix    inet:ipv6-prefix
|     |  +--rw ipv6-interface-address-tlv
```

```
|     |  |  +--rw code      enumeration
```

```
            |     |  |  +--rw length    uint8
            |     |  |  +--rw value
            |     |  |     +--rw ipv6-interface-address-list* [ipv6-interfac
e-address]
            |     |  |        +--rw ipv6-interface-address    inet:ipv6-addr
ess
            |     |  +--rw ipv6-te-router-id-tlv
            |     |  |  +--rw code      enumeration
            |     |  |  +--rw length    uint8
            |     |  |  +--rw value
            |     |  |     +--rw ipv6-te-router-id?   inet:ipv6-address
            |     |  +--rw ipv6-srlg-tlv
            |     |  |  +--rw code      enumeration
            |     |  |  +--rw length    uint8
            |     |  |  +--rw value
            |     |  |     +--rw system-id                 isis-system-id-de
f
            |     |  |     +--rw pseudonode-number         uint8
            |     |  |     +--rw flags                     uint8
            |     |  |     +--rw ipv6-interface-address    inet:ipv6-address
            |     |  |     +--rw ipv6-neighbor-address     inet:ipv6-address
            |     |  |     +--rw srlg-list* [srlg]
            |     |  |        +--rw srlg    uint32
            |     |  +--rw multi-topology-tlv
            |     |  |  +--rw code      enumeration
            |     |  |  +--rw length    uint8
            |     |  |  +--rw value
            |     |  |     +--rw o-bit    boolean
            |     |  |     +--rw a-bit    boolean
            |     |  |     +--rw mt-id    uint16
            |     |  +--rw mt-intermediate-systems-tlv
            |     |  |  +--rw code      enumeration
            |     |  |  +--rw length    uint8
            |     |  |  +--rw value
            |     |  |     +--rw mt-id                              uint16
            |     |  |     +--rw mt-intermediate-systems-list* [mt-intermedi
ate-systems-index]
            |     |  |        +--rw mt-intermediate-systems-index    uint32
            |     |  |        +--rw extended-is-reachability-tlv
            |     |  |           +--rw code      enumeration
            |     |  |           +--rw length    uint8
            |     |  |           +--rw value* [neighbor-system-id]
            |     |  |              +--rw neighbor-system-id
        isis-system-id-def
            |     |  |              +--rw metric
        uint32
            |     |  |              +--rw sub-tlv-length
        uint8
            |     |  |              +--rw administrative-group-stlv
            |     |  |              |  +--rw type      enumeration
            |     |  |              |  +--rw length?   uint32
            |     |  |              |  +--rw value?    uint32
            |     |  |              +--rw ipv4-interface-address--stlv* [ipv
4-interface-address]
```

```
            |     |  |              |  +--rw type                    enume
ration
            |     |  |              |  +--rw length?                 uint3
2
```

```
            |     |  |              |  +--rw ipv4-interface-address    inet:
ipv4-address
            |     |  |              +--rw ipv4-neighbor-address--stlv* [ipv4
-neighbor-address]
            |     |  |              |  +--rw type                       enumer
ation
            |     |  |              |  +--rw length?                    uint32
            |     |  |              |  +--rw ipv4-neighbor-address    inet:i
pv4-address
            |     |  |              +--rw maximum-bandwidth--stlv
            |     |  |              |  +--rw type                  enumeratio
n
            |     |  |              |  +--rw length?               uint32
            |     |  |              |  +--rw maximum-bandwidth?    uint32
            |     |  |              +--rw maximum-reservable-link-bandwidth-
stlv
            |     |  |              |  +--rw type                  enumeratio
n
            |     |  |              |  +--rw length?               uint32
            |     |  |              |  +--rw maximum-bandwidth?    uint32
            |     |  |              +--rw unreserved-bandwidth-stlv
            |     |  |              |  +--rw type                     enumera
tion
            |     |  |              |  +--rw length?                  uint32
            |     |  |              |  +--rw unreserved-bandwidth?    uint32
            |     |  |              +--rw traffic-engineering-default-metric
-stlv
            |     |  |              |  +--rw type
        enumeration
            |     |  |              |  +--rw length?
        uint32
            |     |  |              |  +--rw utraffic-engineering-default-me
tric?   uint32
            |     |  |              +--rw ipv6-interface-address--stlv* [ipv
6-interface-address]
            |     |  |              |  +--rw type                       enume
ration
            |     |  |              |  +--rw length?                    uint3
2
            |     |  |              |  +--rw ipv6-interface-address    inet:
ipv6-address
            |     |  |              +--rw ipv6-neighbor-address--stlv* [ipv6
-neighbor-address]
            |     |  |                 +--rw type                       enumer
ation
            |     |  |                 +--rw length?                    uint32
            |     |  |                 +--rw ipv6-neighbor-address    inet:i
pv6-address
            |     |  +--rw mt-topology-reachable-ipv4-prefixes-tlv
            |     |  |  +--rw code      enumeration
            |     |  |  +--rw length    uint8
            |     |  |  +--rw value
            |     |  |     +--rw mt-id
 uint16
```

```
            |     |  |     +--rw mt-topology-reachable-ipv4-prefixes-list* [
mt-topology-reachable-ipv4-prefixes-index]
            |     |  |        +--rw mt-topology-reachable-ipv4-prefixes-inde
x    uint32
            |     |  |        +--rw extended-ip-reachability-tlv
            |     |  |           +--rw code      enumeration
            |     |  |           +--rw length    uint8
            |     |  |           +--rw value* [ip-prefix]
            |     |  |              +--rw metric             uint32
            |     |  |              +--rw up-down-bit        boolean
            |     |  |              +--rw sub-TLV-bit        boolean
            |     |  |              +--rw prefix-length      uint8
            |     |  |              +--rw ip-prefix          inet:ipv4-prefi
x
            |     |  |              +--rw length-sub-TLVs?   uint8
            |     |  |              +--rw sub-tlv* [type value]
            |     |  |                 +--rw type      uint8
```

```
            |     |  |                +--rw length?   uint32
            |     |  |                +--rw value     string
            |     |  +--rw mt-topology-reachable-ipv6-prefixes-tlv
            |     |     +--rw code      enumeration
            |     |     +--rw length    uint8
            |     |     +--rw value
            |     |        +--rw mt-id
 uint16
            |     |        +--rw mt-topology-reachable-ipv6-prefixes-list* [
mt-topology-reachable-ipv6-prefixes-index]
            |     |           +--rw mt-topology-reachable-ipv6-prefixes-inde
x    uint32
            |     |           +--rw ipv6-reachability-tlv
            |     |              +--rw code      enumeration
            |     |              +--rw length    uint8
            |     |              +--rw value
            |     |                 +--rw metric         uint32
            |     |                 +--rw u-bit          boolean
            |     |                 +--rw x-bit          boolean
            |     |                 +--rw s-bit          boolean
            |     |                 +--rw prefix-len     uint8
            |     |                 +--rw prefix-list* [ipv6-prefix]
            |     |                    +--rw ipv6-prefix    inet:ipv6-prefix
            |     +--rw lsdb-status?   lsdb-status-def
            |     +--rw lsdb-size?     uint32
            |     +--ro is-number?     uint32
            +--rw mt-ipv4-rib* [ipv4-prefix]
            |  +--rw ipv4-prefix             inet:ipv4-prefix
            |  +--rw nexthop-list* [nexthop]
            |  |  +--rw nexthop    inet:ipv4-prefix
            |  +--rw back-nexthop?           inet:ipv4-prefix
            |  +--rw ipv4-isis-route-para
            |     +--rw metric?                 uint32
            |     +--ro type?                   enumeration
            |     +--ro route-current-state?    route-state-def
            |     +--ro route-previous-state?   route-state-def
            |     +--rw lsp-id?                 isis-lsp-id-def
            |     +--ro route-chg-reason?       enumeration
            +--rw mt-ipv6-rib* [ipv6-prefix]
            |  +--rw ipv6-prefix             inet:ipv6-prefix
            |  +--rw nexthop-list* [nexthop]
            |  |  +--rw nexthop    inet:ipv6-prefix
            |  +--rw back-nexthop?           inet:ipv4-prefix
            |  +--rw ipv6-isis-route-para
            |     +--rw metric?                 uint32
            |     +--ro type?                   enumeration
            |     +--ro route-current-state?    route-state-def
            |     +--ro route-previous-state?   route-state-def
            |     +--rw lsp-id?                 isis-lsp-id-def
            |     +--ro route-chg-reason?       enumeration
            +--rw circuit-list* [circuit-index]
```

```
|  +--rw circuit-index          uint32
|  +--rw circuit-name?          string
|  +--rw ip-address?            inet:ip-address
|  +--ro circuit-status?        circuit-status-def
|  +--ro circuit-down-reason?   circuit-down-reason-def
|  +--rw circuit-net-type?      circuit-net-type-def
|  +--rw circuit-level-type?    circuit-level-type-def
|  +--rw circuit-auth-info
|  |  +--rw auth-mode
|  |  |  +--rw (auth-mode-type)?
|  |  |     +--:(mode-simple)
|  |  |     |  +--rw simple-password?     string
|  |  |     +--:(mode-md5)
|  |  |     |  +--rw md5-password?        string
|  |  |     +--:(mode-hmac-sha256)
|  |  |     |  +--rw hmac-key-id?         uint32
|  |  |     |  +--rw hmac-password?       string
|  |  |     +--:(mode-keychain)
|  |  |        +--rw keychain-key-id?     uint32
|  |  |        +--rw keychain-password?   string
|  |  |        +--rw keychain-mode?       enumeration
|  |  |        +--rw keychain-periodic?   enumeration
|  |  |        +--rw send_time?           uint32
|  |  |        +--rw receive_time?        uint32
|  |  +--rw snp-auth-status?   enumeration
|  |  +--rw auth-scope?        enumeration
|  +--rw l1-dis-id?             inet:ipv4-prefix
|  +--rw l2-dis-id?             inet:ipv4-prefix
|  +--rw l1-nbr-list* [l1nbr-system-id]
|  |  +--rw l1nbr-system-id    isis-system-id-def
|  |  +--rw snpa?              uint32
|  |  +--ro nbr-status?        nbr-status-def
|  |  +--ro nbr-down-reason?   nbr-down-reason-def
|  |  +--ro nbr-type?          nbr-type-def
|  +--rw l2-nbr-list* [l2nbr-system-id]
|  |  +--rw l2nbr-system-id    isis-system-id-def
|  |  +--rw snpa?              uint32
|  |  +--ro nbr-status?        nbr-status-def
|  |  +--ro nbr-down-reason?   nbr-down-reason-def
|  |  +--ro nbr-type?          nbr-type-def
|  +--rw circuit-te
|     +--rw admin-group?         uint32
|     +--rw ipv4-addr?           inet:ipv4-prefix
|     +--rw nbr-ipv4-addr?       inet:ipv4-prefix
|     +--rw max-bandwidth?       uint32
|     +--rw max-rsv-bandwidth?   uint32
|     +--rw unrsv-bandwidth?     uint32
+--rw policy-list* [policy-id]
```

```
               +--rw policy-id    string
```

Figure 2 The I2RS YANG model of IS-IS

4. Relationship to other I2RS Data Models

(TBD)

5. I2RS IS-IS Yang Module

```
// <code begins> file "i2rs isis@2014-08-31.yang"

module i2rs-isis-protocol  {

  namespace "urn:huawei:params:xml:ns:yang:rt:i2rs:isis";
    // replace with iana namespace when assigned
    prefix "isis";

  import ietf-inet-types {
    prefix inet;
    //rfc6991
  }


  organization
    "Huawei Technologies Co., Ltd. ";
  contact
     "Email: wanglixing@huawei.com
      Email: shares@ndzh.com
      Email: eric.wu@huawei.com";

    description
      "
        Terms and acronyms

        isis (isis):Intermediate System to Intermediate System

        ip (ip): internet protocol

        ipv4 (ipv4):internet protocol version 4

        ipv6 (ipv6): internet protocol version 6

        metric(metric): multi exit discriminator

        igp (igp): interior gateway protocol
```

```
       mtu (mtu) maximum transmission unit
      ";


  revision "2014-08-22" {
    description "initial revision";
    reference "draft-wu-i2rs-isis-info-model-00";
  }

  typedef address-family-def {
    description
      "The definition of address family.";
    type enumeration {
      enum "ipv4";
      enum "ipv6";
      enum "ipv4ipv6";
    }
  }

  typedef route-state-def {
    description
      "The definition of the state of the route.";
    type enumeration {
      enum "active";
      enum "inactive";
      enum "primary";
      enum "backup";
    }
  }
  typedef isis-lsp-id-def {
    description
      "This type defines isis LSP ID using pattern,
       system id looks like : f000 1234 3210 251 254. ";
    type string {
      pattern
        '[0-9A-Fa-f]{4} [0-9A-Fa-f]{4} [0-9A-Fa-f]{4}'
       +'([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
        +'([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
    }
  }

  typedef isis-system-id-def {
    description
      "This type defines isis system id using pattern,
       system id looks like : f000 1234 3210 00. ";
    type string {
      pattern
        '[0-9A-Fa-f]{4}[0-9A-Fa-f]{4}[0-9A-Fa-f]{4}00';
```

```
    }
  }

  typedef nbr-status-def {
    type enumeration {
      enum "init";
      enum "up";
      enum "down";
        }
  }

  typedef mt-status-def {
    type enumeration {
      enum "active";
      enum "inactive";
     }
  }

  typedef nbr-down-reason-def {
    type enumeration {
      enum "circ-down";
      enum "bfd-down";
      enum "expiration";
      enum "cfg-chg";
      enum "i2rs-down";
   }
  }
  typedef nbr-type-def {
    type enumeration {
      enum "level-1-only";
      enum "level-2-only";
      enum "level-1-2-both";
    }
  }

  typedef circuit-status-def {
    type enumeration {
      enum "circuit-up";
      enum "circuit-down";
    }
  }

  typedef circuit-down-reason-def {
    type enumeration {
      enum "phy-down";
      enum "ip-down";
      enum "i2rs-down";
    }
```

```
  }
  typedef circuit-net-type-def {
    type enumeration {
      enum "p2p";
      enum "brodcast";
      enum "nbma";
    }
  }

  typedef circuit-level-type-def {
    type enumeration {
      enum "level-1-only";
      enum "level-2-only";
      enum "level-1-2-both";
    }
  }

  typedef protocol-status-def {
    description
      "The current status of this is-is instance.";
    type enumeration {
      enum "active";
      enum "reset";
      enum "shutdown";
      enum "overload";

    }
  }

  typedef level-type-def {
    description     "The level-type of this is-is instance.";
      type enumeration {
        enum "level-1-only";
        enum "level-2-only";
        enum "level-1-2-both";
    }
  }

  typedef metric-style-def {
    description     "The metric style of this is-is instance.";
      type enumeration {
        enum "narrow";
        enum "wide";
        enum "compatible";
    }
  }

  typedef isis-instance-create-mode-def {
```

```
    type enumeration {
      enum "not-i2rs";
      enum "i2rsclient-create-isis-instance";
      enum "i2rsagent-fails-isis-instance-create";
      enum "i2rsagent-created-isis-instance";
      enum "i2rsagent-isis-instance-create";
      enum "i2rsagent-rejects-isis-instance-create";
      enum "i2rsagent-attempts-isis-instance-create";
     }
  }

  typedef lsdb-status-def {
    type enumeration {
      enum "normal";
      enum "overflow";
    }
  }


  grouping isis-route-common{
    description
      "The common attributes used for all routes. ";
    leaf metric {
      type uint32;
      mandatory false;
    }
    leaf type {
      type enumeration {
        enum "isis-internal-route";
        enum "isis-external-route";
      }
      config "false";
    }
    leaf route-current-state {
      type route-state-def ;
      mandatory false;
      config false;
    }
    leaf route-previous-state {
      type route-state-def ;
      mandatory false;
      config false;
    }
    leaf lsp-id {
      type isis-lsp-id-def;
      mandatory false;
    }
    leaf route-chg-reason {
```

```
      type enumeration {
        enum "orig-adv";
        enum "orig-withdraw";
        enum "adj-down";
        enum "policy-deny";

      }
      config "false";
    }
  }

  grouping extended-is-reachability-tlv {
    container extended-is-reachability-tlv{
      description "The proposed extended is reachability tlv contains a new
data
        structure, consisting of:7 octets of system id and pseudonode number
        3 octets of default metric; 1 octet of length of sub-tlvs
        0-244 octets of sub-tlvs .";

      leaf code {
        description "Code - 22.";
        mandatory true;
        type enumeration {
          enum type {
            value "22";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      list value {
        key "neighbor-system-id";
        leaf neighbor-system-id  {
          type isis-system-id-def;
          mandatory true;
        }
        leaf metric  {
          type uint32 {
            range "1 .. 16777215";
          }
          mandatory true;
        }
        leaf sub-tlv-length  {
          type uint8;
          mandatory true;
        }
        container administrative-group-stlv {
```

```
          leaf type {
            description " The administrative group sub-tlv is tlv type 3.";
            mandatory true;
            type enumeration {
              enum type {
                value "3";
              }
            }
          }
          leaf length {
            type uint32;
          }
          leaf value {
            type uint32;
          }
        }

        list ipv4-interface-address--stlv {
          description  "This sub-tlv contains a 4-octet ipv4 address for the
 interface
            described by the (main) tlv.  this sub-tlv can occur multiple ti
mes.";
          key "ipv4-interface-address";
          leaf type {
            description " The ipv4 interface address sub-tlv is tlv type 6."
;
            mandatory true;
            type enumeration {
              enum type {
                value "6";
              }
            }
          }
          leaf length {
            type uint32;
          }
          leaf ipv4-interface-address {
            type inet:ipv4-address;
          }
        }

        list ipv4-neighbor-address--stlv {
          description  "This sub-tlv contains a single ipv4 address for a ne
ighboring router
            on this link.  this sub-tlv can occur multiple times. ";
          key "ipv4-neighbor-address";
          leaf type {
            description "The ipv4 neighbor address sub-tlv is tlv type 8.";
            mandatory true;
            type enumeration {
              enum type {
                value "8";
```

```
            }
          }
        }
        leaf length {
          type uint32;
        }
        leaf ipv4-neighbor-address {
          type inet:ipv4-address;
        }
      }

      container maximum-bandwidth--stlv {
        description  "This sub-tlv contains the maximum bandwidth that can
 be used on this
          link in this direction (from the system originating the lsp to i
ts neighbors).";
        leaf type {
          description " Type of maximum link bandwidth of sub tlv  is 9.";
          mandatory true;
          type enumeration {
            enum type {
              value "9";
            }
          }
        }
        leaf length {
          type uint32;
        }
        leaf maximum-bandwidth {
          type uint32;
        }
      }

      container maximum-reservable-link-bandwidth-stlv {
        description  "This sub-tlv contains the maximum amount of bandwidt
h that can be
          reserved in this direction on this link.  note that for
          oversubscription purposes, this can be greater than the bandwidt
h of the link. ";
        leaf type {
          description "Type of maximum reservable link bandwidth of sub tl
v  is 10.";
          mandatory true;
          type enumeration {
            enum type {
              value "10";
            }
          }
        }
        leaf length {
          type uint32;
        }
        leaf maximum-bandwidth {
```

```
            type uint32;
          }
        }
        container unreserved-bandwidth-stlv {
          description  "This sub-tlv contains the amount of bandwidth reserv
able in this
            direction on this link.  note that for oversubscription purposes
,
            this can be greater than the bandwidth of the link. ";
          leaf type {
            description "type of unreserved bandwidth bandwidth of sub tlv
is 11.";
            mandatory true;
            type enumeration {
              enum type {
                value "11";
              }
            }
          }
          leaf length {
            type uint32;
          }
          leaf unreserved-bandwidth {
            type uint32;
          }
        }

        container traffic-engineering-default-metric-stlv {
          description  "This sub-tlv contains a 24-bit unsigned integer.  th
is metric is
            administratively assigned and can be used to present a different
ly
            weighted topology to traffic engineering spf calculations.";
          leaf type {
            description "Type of te default metric of sub tlv  is 18.";
            mandatory true;
            type enumeration {
              enum type {
                value "18";
              }
            }
          }
          leaf length {
            type uint32;
          }
          leaf utraffic-engineering-default-metric {
            type uint32;
          }
        }

        list ipv6-interface-address--stlv {
          description  "Ipv6 interface address sub-tlv of the extended is re
achability
            tlv has sub-tlv type 12.  it contains a 16-octet ipv6 address fo
```

r the

```
            interface described by the containing extended is reachability t
lv.
            this sub-tlv can occur multiple times.";
          key "ipv6-interface-address";
          leaf type {
            description "The ipv6 interface address sub-tlv is tlv type 12."
;
            mandatory true;
            type enumeration {
              enum type {
                value "12";
              }
            }
          }
          leaf length {
            type uint32;
          }
          leaf ipv6-interface-address {
            type inet:ipv6-address;
          }
        }

        list ipv6-neighbor-address--stlv {
          description  "The ipv6 neighbor address sub-tlv of the extended is
 reachability tlv
            has sub-tlv type 13.  it contains a 16-octet ipv6 address for a
            neighboring router on the link described by the (main) tlv.  thi
s
            sub-tlv can occur multiple times. ";
          key "ipv6-neighbor-address";
          leaf type {
            description "The ipv6 neighbor address sub-tlv is tlv type 13.";
            mandatory true;
            type enumeration {
              enum type {
                value "13";
              }
            }
          }
          leaf length {
            type uint32;
          }
          leaf ipv6-neighbor-address {
            type inet:ipv6-address;
          }
        }
      }
    }
  }

  grouping extended-ip-reachability-tlv {
    container extended-ip-reachability-tlv{
```

```
      description "To address these two issues, the proposed extended ip rea
chability
        tlv provides for a 32-bit metric and adds one bit to indicate that a
        prefix has been redistributed ’down’ in the hierarchy.";
      leaf code {
        description "Code - 135.";
        mandatory true;
        type enumeration {
          enum type {
            value "135";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      list value {
        key "ip-prefix";
        leaf metric  {
          type uint32;
          mandatory true;
        }
        leaf up-down-bit  {
          type boolean;
          mandatory true;
        }
        leaf sub-TLV-bit  {
          type boolean;
          mandatory true;
        }
        leaf prefix-length  {
          type uint8{
            range "0..63";
          }
          mandatory true;
        }
        leaf ip-prefix  {
          type inet:ipv4-prefix;
          mandatory true;
        }
        leaf length-sub-TLVs  {
          type uint8;
          mandatory false;
        }
        list sub-tlv {
          key "type value";
          leaf type {
            type uint8;
```

```
          }
          leaf length {
            type uint32;
          }
          leaf value {
            type string;
          }
        }
      }
    }
  }

  grouping ipv6-reachability-tlv {
    container ipv6-reachability-tlv {
      description "The ipv6 reachability tlv describes network reachability
through
        the specification of a routing prefix, metric information, a bit to
        indicate if the prefix is being advertised down from a higher level,
        a bit to indicate if the prefix is being distributed from another
        routing protocol, and optionally the existence of sub-tlvs to allow
        for later extension. ";
      leaf code {
        description "Code - 236.";
        mandatory true;
        type enumeration {
          enum type {
            value "236";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        leaf metric{
          type uint32;
          mandatory true;
        }
        leaf u-bit{
          type boolean;
          mandatory true;
        }
        leaf x-bit{
          type boolean;
          mandatory true;
        }
        leaf s-bit{
          type boolean;
```

```
          mandatory true;
        }
        leaf prefix-len{
          type uint8;
          mandatory true;
        }
        list prefix-list{
          key ipv6-prefix;
          leaf ipv6-prefix{
            type inet:ipv6-prefix;
          }
        }
      }
    }
  }

  grouping area-addresse-tlv {
    container area-addresse-tlv {
      description "The set of manualareaaddresses of this intermediate syste
m.";

      leaf code {
        description "Code - 1.";
        mandatory true;
        type enumeration {
          enum type {
            value "1";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      list value {
        key "area-addresse";
        leaf address-length  {
          type uint8;
          mandatory true;
        }
        leaf area-addresse  {
          type uint8;
          mandatory true;
        }
      }
    }
  }

  grouping intermediate-system-neighbours-tlv {
```

```
    container intermediate-system-neighbours-tlv {
      description "Ip internal reachability information -- ip addresses with
in the
      routing domain reachable directly via one or more interfaces on
      this intermediate system.";

      leaf code {
        description "Code - 2.";
        mandatory true;
        type enumeration {
          enum type {
            value "2";
          }
        }
      }

      leaf length {
        type uint8;
        mandatory true;
      }
      leaf virtual-flag {
        type boolean;
        mandatory true;
      }
      list value {
        key "neighbor-id";
        leaf default-metric  {
          type uint8 {
            range "0 .. 63";
          }
          mandatory true;
        }
        leaf delay-metric  {
          type uint8 {
            range "0 .. 63";
          }
          mandatory true;
        }
        leaf expense-metric  {
          type uint8 {
            range "0 .. 63";
          }
          mandatory true;
        }
        leaf error-metric  {
          type uint8 {
            range "0 .. 63";
          }
          mandatory true;
```

```
        }
        leaf neighbor-id  {
          type string;
          mandatory true;
        }
      }
    }
  }

  grouping nlpid-tlv {
    container nlpid-tlv {
      description "The set network layer protocol identifiers
      for network layer protocols that this intermediate system is
      capable of relaying.this must appear once in lsp number 0.";

      leaf code {
        description "Code - 129.";
        mandatory true;
        type enumeration {
          enum type {
            value "129";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      list value {
        key "nlpid";
        leaf nlpid  {
          type uint8;
          mandatory true;
        }
      }
    }
  }
  grouping ip-interface-addresses-tlv {
    container ip-interface-addresses-tlv {
      description "The ip addresss of one or more interfaces
      corresponding to the snpas enabled on this intermediate system.";

      leaf code {
        description "Code - 132.";
        mandatory true;
        type enumeration {
          enum type {
            value "132";
```

```
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      list value {
        key "ip-address";
        leaf ip-address  {
          type inet:ipv4-address;
          mandatory true;
        }
      }
    }
  }

  grouping ip-internal-reachability-tlv {
    container ip-internal-reachability-tlv {
      description "Ip internal reachability information -- ip addresses with
in the
      routing domain reachable directly via one or more interfaces on
      this intermediate system.";

      leaf code {
        description "Code - 128.";
        mandatory true;
        type enumeration {
          enum type {
            value "128";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      list value {
        key "ip-prefix";
        leaf i-e-bit  {
          description "Bit 7 of this field (marked I/E) indicates the metric
 type
            (internal or external) for all four TOS metrics, and must be
             set to zero indicating internal metrics.";
          type boolean ;
          mandatory true;
        }
        leaf default-metric  {
          type uint8 {
            range "0 .. 64";
```

```
          }
          mandatory true;
        }
        leaf s-bit  {
          description "If this IS does not support this metric it shall
            set the bit S to 1 to indicate that the metric is unsupported.";
          type boolean ;
          mandatory true;
        }
        leaf delay-metric  {
          type uint8 {
            range "0 .. 64";
          }
          mandatory true;
        }
        leaf s-bit-expense-metric  {
          description "If this IS does not support this metric it shall
            set the bit S to 1 to indicate that the metric is unsupported.";
          type boolean ;
          mandatory true;
        }
        leaf expense-metric  {
          type uint8 {
            range "0 .. 64";
          }
          mandatory true;
        }
        leaf s-bit-error-metric  {
          description "If this IS does not support this metric it shall
            set the bit S to 1 to indicate that the metric is unsupported.";
          type boolean ;
          mandatory true;
        }
        leaf error-metric  {
          type uint8 {
            range "0 .. 64";
          }
          mandatory true;
        }
        leaf ip-prefix  {
          type inet:ipv4-prefix;
          mandatory true;
        }
      }
    }
  }

  grouping ip-external-reachability-tlv {
```

```
    container ip-external-reachability-tlv {
      description "Ip external reachability information -- ip addresses outs
ide the
      routing domain reachable via interfaces on this intermediate  system."
;

      leaf code {
        description "Code - 130.";
        mandatory true;
        type enumeration {
          enum type {
            value "130";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      list value {
        key "ip-prefix";
        leaf i-e-bit  {
          description "Bit 7 of this field (marked I/E) indicates the metric
 type
            (internal or external) for all four TOS metrics, and must be
             set to zero indicating internal metrics.";
          type boolean ;
          mandatory true;
        }
        leaf default-metric  {
          type uint8 {
            range "0 .. 64";
          }
          mandatory true;
        }
        leaf s-bit  {
          description "If this IS does not support this metric it shall
            set the bit S to 1 to indicate that the metric is unsupported.";
          type boolean ;
          mandatory true;
        }
        leaf delay-metric  {
          type uint8 {
            range "0 .. 64";
          }
          mandatory true;
        }
        leaf s-bit-expense-metric  {
          description "If this IS does not support this metric it shall
            set the bit S to 1 to indicate that the metric is unsupported.";
          type boolean ;
```

```
          mandatory true;
        }
        leaf expense-metric  {
          type uint8 {
            range "0 .. 64";
          }
          mandatory true;
        }
        leaf s-bit-error-metric  {
          description "If this IS does not support this metric it shall
            set the bit S to 1 to indicate that the metric is unsupported.";
          type boolean ;
          mandatory true;
        }
        leaf error-metric  {
          type uint8 {
            range "0 .. 64";
          }
          mandatory true;
        }
        leaf ip-prefix  {
          type inet:ipv4-prefix;
          mandatory true;
        }
      }
    }
  }

  grouping inter-domain-information-tlv {
    container inter-domain-information-tlv {
      description "Inter-domain routing protocol information -- inter-domain
routing
      protocol information carried transparently through level 2 for
      the convenience of any inter-domain protocol that may be running
      in the boundary iss.";

      leaf code {
        description "Code - 131.";
        mandatory true;
        type enumeration {
          enum type {
            value "131";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
```

```
      list value {
        key "inter-domain-information-type external-information";
        leaf inter-domain-information-type  {
          type uint8;
          mandatory true;
        }
        leaf external-information  {
          type uint32;
          mandatory true;
        }
      }
    }
  }

  grouping ipv6-interface-address-tlv {
    container ipv6-interface-address-tlv {
      description "Ipv6 interface address tlv is tlv type 232 (0xe8).
        tlv 232 maps directly to ip interface address tlv in [rfc1195] .
        we necessarily modify the contents to be 0-15 16-octet ipv6 interfac
e
        addresses instead of 0-63 4-octet ipv4 interface addresses. ";

      leaf code {
        description "Code - 232.";
        mandatory true;
        type enumeration {
          enum type {
            value "232";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        list ipv6-interface-address-list{
        description "The interface address tlvs
          must contain only the non-link-local ipv6 addresses assigned to th
e is. ";
          key ipv6-interface-address;
          leaf ipv6-interface-address{
            type inet:ipv6-address;
          }
        }
      }
    }
  }

  grouping ipv6-te-router-id-tlv {
```

```
    container ipv6-te-router-id-tlv {
      description "The ipv6 te router id tlv contains a 16-octet ipv6 addres
s.  a stable
        global ipv6 address must be used, so that the router id provides a
        routable address, regardless of the state of a node's interfaces. ";
      leaf code {
        description "Code - 140.";
        mandatory true;
        type enumeration {
          enum type {
            value "140";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        leaf ipv6-te-router-id{
          type inet:ipv6-address;
        }
      }
    }
  }

  grouping ipv6-srlg-tlv {
    container ipv6-srlg-tlv {
      description "The ipv6 srlg tlv has type 139.  the tlv carries the shar
ed risk link
        group information (see the shared risk link group information
        section of [gmpls-routing]).";

      leaf code {
        description "Code - 139.";
        mandatory true;
        type enumeration {
          enum type {
            value "139";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        leaf system-id {
          type isis-system-id-def;
          mandatory true;
```

```
        }
        leaf pseudonode-number {
          type uint8;
          mandatory true;
        }
        leaf flags {
          type uint8;
          mandatory true;
        }
        leaf ipv6-interface-address {
          type inet:ipv6-address;
          mandatory true;
        }
        leaf ipv6-neighbor-address {
          type inet:ipv6-address;
          mandatory true;
        }
        list srlg-list{
          key srlg;
          leaf srlg{
            type uint32;
          }
        }
      }
    }
  }

  grouping multi-topology-tlv {
    container multi-topology-tlv {
      description "The multi-topology number of this tlv is 229.
      it contains one or more mts; the router is participating in the follow
ing structure.";

      leaf code {
        description "Code - 229.";
        mandatory true;
        type enumeration {
          enum type {
            value "229";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        leaf o-bit{
          description "Bit O represents the OVERLOAD bit for the MT (only va
lid in LSP
```

```
            fragment zero for MTs other than ID #0, otherwise SHOULD be set
to
            0 on transmission and ignored on receipt).";
          type boolean;
          mandatory true;
        }
        leaf a-bit{
          description "Bit A represents the ATTACH bit for the MT (only vali
d in LSP
            fragment zero for MTs other than ID #0, otherwise SHOULD be set
to
            0 on transmission and ignored on receipt).";
          type boolean;
          mandatory true;
        }
        leaf mt-id{
          type uint16 {
            range "0 .. 4095";
          }
          mandatory true;
        }
      }
    }
  }

  grouping mt-intermediate-systems-tlv {
    container mt-intermediate-systems-tlv {
      description "Mt intermediate systems tlv is 222.  it is aligned with e
xtended is
        reachability tlv type 22 beside an additional two bytes in front at
the beginning of the tlv.";

      leaf code {
        description "Code - 222.";
        mandatory true;
        type enumeration {
          enum type {
            value "222";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        leaf mt-id{
          type uint16 {
            range "0 .. 4095";
          }
          mandatory true;
        }
        list mt-intermediate-systems-list{
```

```
          key mt-intermediate-systems-index;
          leaf mt-intermediate-systems-index{
            type uint32;
          }
          uses extended-is-reachability-tlv;
        }
      }
    }
  }

  grouping mt-topology-reachable-ipv4-prefixes-tlv {
    container mt-topology-reachable-ipv4-prefixes-tlv {
      description "Multi-topology reachable ipv4 prefixes tlv is 235.  it is
 aligned with extended ip
        reachability tlv type 135 beside an additional two bytes in front.";
      leaf code {
        description "Code - 235.";
        mandatory true;
        type enumeration {
          enum type {
            value "235";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        leaf mt-id{
          type uint16 {
            range "0 .. 4095";
          }
          mandatory true;
        }
        list mt-topology-reachable-ipv4-prefixes-list{
          key mt-topology-reachable-ipv4-prefixes-index;
          leaf mt-topology-reachable-ipv4-prefixes-index{
            type uint32;
          }
          uses extended-ip-reachability-tlv;
        }
      }
    }
  }

  grouping mt-topology-reachable-ipv6-prefixes-tlv {
    container mt-topology-reachable-ipv6-prefixes-tlv {
      description "Multi-topology reachable ipv4 prefixes tlv is 237.  it is
 aligned with extended ip
```

```
        reachability tlv type 137 beside an additional two bytes in front.";

      leaf code {
        description "Code - 237.";
        mandatory true;
        type enumeration {
          enum type {
            value "237";
          }
        }
      }
      leaf length {
        type uint8;
        mandatory true;
      }
      container value {
        leaf mt-id{
          type uint16 {
            range "0 .. 4095";
          }
          mandatory true;
        }
        list mt-topology-reachable-ipv6-prefixes-list{
          key mt-topology-reachable-ipv6-prefixes-index;
          leaf mt-topology-reachable-ipv6-prefixes-index{
            type uint32;
          }
          uses ipv6-reachability-tlv ;
        }
      }
    }
  }

  grouping auth-info {
    container auth-mode{
      choice auth-mode-type {
        case mode-simple  {
          leaf simple-password {
            type string;
          }
        }
        case mode-md5  {
          leaf md5-password {
            type string;
          }
        }
        case mode-hmac-sha256  {
          leaf hmac-key-id {
```

```
            type uint32;
          }
          leaf hmac-password {
            type string;
          }
        }
        case mode-keychain  {
          leaf keychain-key-id {
            type uint32;
          }
          leaf keychain-password {
             type string;
          }
          leaf keychain-mode {
            type enumeration {
              enum "absolute";
              enum "periodic";
            }
          }
          leaf keychain-periodic {
            type enumeration {
              enum "daily";
              enum "weekly";
              enum "monthly";
              enum "yearly";
            }
          }
          leaf send_time {
            type uint32;
          }
          leaf receive_time {
            type uint32;
          }
        }
      }
    }
    leaf snp-auth-status {
      type enumeration {
        enum "auth-and-validate";
        enum "non-auth-and-validate";
        enum "auth-and-non-validate";
      }
    }
    leaf auth-scope {
      type enumeration {
        enum "lsp-packet";
        enum "lsp-snp-hello-packet";
        enum "hello-packet";
```

```
        enum "lsp-snp-packet";
      }
    }
  }

  container isis-routing-protocol {

    list isis-instance {
      description  "The common structure of is-is protocol.";
      key "name";
      leaf name {
        type string;
        mandatory true;
      }

      leaf isis-vpn-name {
        type string;
        mandatory false;
      }

      leaf address-family {
        description
          "The address family of this is-is instance.";
        type address-family-def;
        mandatory true;
      }
      list net {
        description
          "The network-entity of this is-is instance.";
        key "area-id";
        leaf area-id {
          type string;
        }

        leaf system-id {
          type isis-system-id-def;
        }
      }

      leaf protocol-status {
        description
          "The current status of this is-is instance.";
        type protocol-status-def ;
        config "false";
      }

      leaf level-type {
        description     "The level-type of this is-is instance.";
```

```
        type level-type-def ;
      }

      leaf metric-style {
        description     "The metric style of this is-is instance.";
        type metric-style-def ;
      }

      leaf lsp-mtu {
        type uint32 ;
        mandatory false;
      }


      leaf lsp-refresh {
        type uint32 ;
        mandatory false;
      }

      leaf lsp-lifetime {
        type uint32 ;
        mandatory false;
      }


      leaf isis-instance-create-mode {
        type isis-instance-create-mode-def ;
         config "false";
      }

      leaf preference {
         type uint32 {
           range "1..4294967295";
         }
      }

      leaf hostname {
        type string;
        mandatory false;
      }
      container domain-auth-info {
        uses auth-info;
      }
      container area-auth-info {
        uses auth-info;
      }
      list multi-topo-list {
        description
```

```
         "A list of multi-tology supported in this instance.";
        key "mt-id";
        leaf mt-id {
          type uint16 {
            range "0 .. 4095";
          }
        }
        leaf mt-status {
          type mt-status-def;
           config "false";
        }

        leaf address-family {
          description
          "The address family supported in this topology.";
          type address-family-def;
          mandatory true;
        }

        list level-list {
          description
           "The list of level supported in this topology.";
          key "level-id";
          leaf level-id {
            type string;
            mandatory true;
          }

          container lsdb {
            list lsp {
              description
               "The definition of link state pdu.";
              key "lsp-id sequence-number checksum";

              leaf maximum-area-addresses {
                type uint8;
                mandatory true;
               }
              leaf pdu-length {
                type uint16;
                mandatory true;
               }
              leaf remaining-lifetime {
                type uint16;
                mandatory true;
              }
              leaf lsp-id {
                type isis-lsp-id-def;
```

```
            mandatory true;
           }
          leaf sequence-number {
            type uint32;
            mandatory true;
          }

          leaf checksum {
            type uint16;
            mandatory true;
          }
          leaf p-bit {
            type boolean;
            mandatory true;
          }
          leaf error-metric-bit {
            type boolean;
            mandatory true;
          }
          leaf expense-metric-bit {
            type boolean;
            mandatory true;
          }
          leaf delay-metric-bit {
            type boolean;
            mandatory true;
          }
          leaf default-metric-bit {
            type boolean;
            mandatory true;
          }
          leaf lspdbol-bit {
            type enumeration {
              enum no-overload {
                value "0";
              }
              enum overload {
                value "1";
              }
            }
          }
          leaf is-type-bit {
            type enumeration {
              enum level-1 {
                value "1";
              }
              enum level-2 {
                value "3";
```

```
            }
          }
        }

        uses area-addresse-tlv;

        uses intermediate-system-neighbours-tlv;

        uses nlpid-tlv;

        uses ip-interface-addresses-tlv;

        uses ip-internal-reachability-tlv ;

        uses ip-external-reachability-tlv ;

        uses inter-domain-information-tlv ;

        uses extended-is-reachability-tlv;

        uses extended-ip-reachability-tlv;

        uses ipv6-reachability-tlv;

        uses ipv6-interface-address-tlv ;

        uses ipv6-te-router-id-tlv ;

        uses ipv6-srlg-tlv ;

        uses multi-topology-tlv ;

        uses mt-intermediate-systems-tlv ;

        uses mt-topology-reachable-ipv4-prefixes-tlv ;

        uses mt-topology-reachable-ipv6-prefixes-tlv ;
      }

      leaf lsdb-status {
        type lsdb-status-def ;
      }
      leaf lsdb-size {
        type uint32 ;
        mandatory false;
      }
      leaf is-number {
        type uint32 ;
```

```
            mandatory false;
            config "false";
          }
        }
      }

      list mt-ipv4-rib {
        description
         "The route information base for ipv4 prefixes of this topology.";
        key "ipv4-prefix";
        leaf ipv4-prefix {
          type inet:ipv4-prefix;
          mandatory true;
        }
        list nexthop-list{
          key "nexthop";
          leaf nexthop {
            type inet:ipv4-prefix;
            mandatory true;
          }
        }
        leaf back-nexthop {
          type inet:ipv4-prefix;
         mandatory false;
        }
        container ipv4-isis-route-para {
          uses isis-route-common;
        }
      }

      list mt-ipv6-rib {
        description
          "The route information base for ipv6 prefixes of this topology."
;
        key "ipv6-prefix";
        leaf ipv6-prefix {
          type inet:ipv6-prefix;
          mandatory true;
        }
        list nexthop-list{
          key "nexthop";
          leaf nexthop {
            type inet:ipv6-prefix;
            mandatory true;
          }
        }
        leaf back-nexthop {
          type inet:ipv4-prefix;
          mandatory false;
```

```
        }
        container ipv6-isis-route-para {
          uses isis-route-common;
        }
      }

      list circuit-list {
        description
         "The circuits supported in this topology.";
        key "circuit-index";
        leaf circuit-index {
          type uint32 ;
        }

        leaf circuit-name {
          description  "The name of this circuit.";
          type string;
        }

        leaf ip-address {
          description
            "Ip address for this circuit.";
          type inet:ip-address;
        }

        leaf circuit-status {
          description     "The status of this circuit.";
          type circuit-status-def ;
          config "false";
        }

        leaf circuit-down-reason {
          type circuit-down-reason-def ;
          config "false";
        }
        leaf circuit-net-type {
          type circuit-net-type-def ;
        }

        leaf circuit-level-type {
          type circuit-level-type-def ;
        }
        container circuit-auth-info {
          uses auth-info;
        }

        leaf l1-dis-id {
          type inet:ipv4-prefix;
```

```
        }

        leaf l2-dis-id {
          type inet:ipv4-prefix;
        }

        list l1-nbr-list {
          description
            "The level-1 neighbors supported under this circuit.";
          key "l1nbr-system-id";
          leaf l1nbr-system-id {
            type isis-system-id-def;
          }
          leaf snpa {
             type uint32;
          }
          leaf nbr-status {
            description     "The state of this neighbor.";
            type nbr-status-def ;
            config "false";
          }

          leaf nbr-down-reason {
            type nbr-down-reason-def ;
        config "false";
          }
          leaf nbr-type {
            type nbr-type-def ;
            config "false";
          }
        }

        list l2-nbr-list {
          description
          "The level-2 neighbors supported under this circuit.";
          key "l2nbr-system-id";
          leaf l2nbr-system-id {
            type isis-system-id-def;
          }
          leaf snpa {
            type uint32;
          }
          leaf nbr-status {
            description     "The state of this neighbor.";
            type nbr-status-def ;
            config "false";
          }
```

```
            leaf nbr-down-reason {
              type nbr-down-reason-def ;
          config "false";
             }
            leaf nbr-type {
              type nbr-type-def ;
              config "false";
            }
          }

          container circuit-te {
            leaf admin-group {
              type uint32;
             }

            leaf ipv4-addr {
               type inet:ipv4-prefix;
            }
            leaf nbr-ipv4-addr {
              type inet:ipv4-prefix;
            }

            leaf max-bandwidth {
              type uint32;
            }

            leaf max-rsv-bandwidth {
              type uint32;
            }

            leaf unrsv-bandwidth {
              type uint32;
            }
          }
        }

        list policy-list {
          description
          "The policies supported in this toplogy.";
          key "policy-id";
          leaf policy-id {
            type string;
          }
        }
      }
    }
  }
}
```

```
  //    </code ends>
```

6. IANA Considerations

This draft registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC3688, the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:rt:i2rs:isis

Registrant Contact: The I2RS WG of IETF

XML: N/A, the request URI is in the XML namespace.

This document registres a Yang module in the Yang Module Names registry [RFC6020] with the following information:

name: IETF-i2rs-isis-protocol

namespace: urn:ietf.params:xml:ns:yang:rt:i2rs:isis

prefix:isis-protocol

reference: RFC XXXX

7. Security Considerations

This document introduces no new security threat over the security threats posed by security requirements as stated in [I-D.ietf-i2rs-architecture]. (The authors would like feedback on the security issues.)

8. Acknowledgements

TBD

9. References

9.1. Normative References

[I-D.hares-i2rs-usecase-reqs-summary]
Hares, S., "Summary of I2RS Use Case Requirements", draft-hares-i2rs-usecase-reqs-summary-00 (work in progress), July 2014.

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-05 (work in progress), July 2014.

[I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-03 (work in progress), May 2014.

9.2. Informative References

[ISO.10589.1992]
International Organization for Standardization, "Intermediate system to intermediate system intra-domain-routing routine information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", ISO Standard 10589, 1992.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

[RFC6976] Shand, M., Bryant, S., Previdi, S., Filsfils, C., Francois, P., and O. Bonaventure, "Framework for Loop-Free Convergence Using the Ordered Forwarding Information Base (oFIB) Approach", RFC 6976, July 2013.

Authors' Addresses

Lixing Wang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 10095
China

Email: wanglixing@huawei.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Eric Wu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: eric.wu@huawei.com