

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

R. Wilton, Ed.
Cisco Systems
July 3, 2017

Interface Properties for YANG Data Models
draft-wilton-netmod-interface-properties-00

Abstract

This document specifies a better way to restrict interface YANG schema nodes to only those types of interfaces that the nodes are applicable to.

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 4, 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.

Internet-Draft

July 2017

Table of Contents

1.	Introduction	2
2.	Terminology	3
3.	Problem Definition	3
4.	Interface Property Identities	4
5.	Potential issues and considerations:	4
6.	YANG Modules	4
7.	IANA Considerations	7
8.	Security Considerations	7
9.	Acknowledgments	7
10.	References	7
10.1.	Normative References	7
10.2.	Informative References	7
Appendix A.	Examples of possible updates to iana-if-types.yang .	7
Appendix B.	Example of interface properties usage in ietf- interfaces-common.yang	10
Appendix C.	Example of interface properties usage in ietf- interfaces-ethernet-like.yang	15
Appendix D.	Example of interface properties usage in ietf- interfaces.yang	17
Author's Address	21

[1.](#) Introduction

This document introduces the concept of generic interface property identities in YANG [[RFC7950](#)] data models to solve the problem of how to restrict interface configuration and state schema nodes to the appropriate types of interfaces. E.g., it is desirable to restrict MAC address configuration and state schema nodes to only those types of interfaces that use Ethernet framing, and hence for which MAC address configuration may be applicable. This document defines a set of common interface property YANG identities (e.g., Ethernet-like), and proposes that the iana-interface-type identities in iana-if-type.yang are updated to also derive from the interface property identities (a backwards compatible change). Interface based YANG schema nodes can then be made conditional on the interface property identities rather than being conditional on a hardcoded set of IANA interface types. This approach also allows newly defined interface types to reuse published standard interface YANG schema nodes just by deriving from the appropriate interface property identities, and without requiring new revisions of those published standard YANG modules each time a new IANA interface type is defined.

Internet-Draft

July 2017

[2.](#) Terminology

This document defines the following terms:

- o interface property identity: A YANG identity that defines a particular generic interface related property that generally relates to multiple interface type identities defined in `iana-interface-type.yang`. An example of such a property is `'Ethernet-like'`.

[3.](#) Problem Definition

The YANG language offers the "when" statement that can be used to make nodes conditional on some particular X-Path expression.

In the case, of interface related configuration, the when statement can be used to explicitly list all the `iana-interface-type` identities that an interface's 'type' leaf may take.

However, the current solution is not ideal for several reasons:

- o Explicitly listing the interface types is somewhat unwieldy, particularly in the case that a schema node may apply to many interfaces. When faced with this scenario, YANG model authors sometimes decide that it is better to keep the model simple and just allow the schema node to be used regardless of interface type.
- o Explicitly listing the interface types has the further problem that the model cannot easily be extended to handle new interface types. If a new interface type is introduced then to reuse the same interface related schema nodes requires that all of the applicable model when statements must be updated to reference the new IANA interface type identity.
- o The current solution does not really work for bespoke vendor

specific interface types. Even though the proprietary interface types may be defined in the IANA interface type identity list, it is unlikely that it would be acceptable to update a standards based YANG model to reference a proprietary interface type. Requiring bespoke interfaces to use separate similar configuration and state nodes makes the models less generic and more tedious to use by clients.

[4.](#) Interface Property Identities

This draft introduces "Interface Property Identities". These are a set of YANG identities that describe properties that could be associated with multiple different types of interface.

The existing IANA interface type identities are then selectively updated to also derive from one or more of these interface property identities. The YANG module upgrade rules in [\[RFC7950\]](#) allow for the interface type identities to derive from additional identities in a fully backwards compatible way.

YANG modules that define interface specific nodes can then use the YANG 1.1 `derived-from()` macro in the 'when' statement xpath expression to indicate that the node applies to all interfaces that inherit from the specified interface property.

As new IANA interface types are defined they can derive from the interface property identities and hence acquire the interface specific YANG nodes that have been defined in existing YANG modules without requiring any changes beyond the device updating to use the latest iana-if-types YANG file.

Similarly, vendor specific interface types can also inherit from the interface property identities and also acquire access to the appropriate interface configuration nodes.

As required, new interface property identities can also be defined, again in a backwards compatible way.

5. Potential issues and considerations:

- o Would IANA be willing to manage the new interface property types?
- o iana-interface-types.yang would need to be YANG 1.1. Would this be an issue?
- o Need to define a suitable set of base interface properties.
- o Need to then apply the set of base interface properties to appropriate interface types. Coordination between vendors may be required to get a reasonable base coverage.

6. YANG Modules

<CODE BEGINS> file "iana-if-property-type@2017-06-27.yang"

```
module iana-if-property-type {
```

Wilton

Expires January 4, 2018

[Page 4]

Internet-Draft

July 2017

```
namespace "urn:ietf:params:xml:ns:yang:iana-if-property-type";
prefix ianaifp;
```

```
organization "IANA";
```

```
contact "";
```

```
description
```

```
"This YANG module defines YANG identities for IANA-registered
interface property types.
```

```
This YANG module is maintained by IANA.
```

```
The latest revision of this YANG module can be obtained from
the IANA web site.
```

```
Requests for new values should be made to IANA via
email (iana@iana.org).
```

```
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>).

The initial version of this YANG module is part of XXX; see the RFC itself for full legal notices.";

reference

"IANA 'interface property definitions' registry.";

revision 2017-06-27 {

description

"Initial revision";

reference

"RFC XXX: IANA Interface Property Type YANG Module";

}

identity iana-if-property-type {

description

"Base identity from which specific interface property types are derived.";

}

identity physical {

base iana-if-property-type;

description

"This property represents an interface that has a physical hardware representation, or is modelled as such.";

}

identity virtual {

base iana-if-property-type;

description

"This property represents an interface that has no external physical hardware representation, and is used to represent interfaces that are created via configuration.";

}

identity sub-interface {

base iana-if-property-type;

description

```

        "This property represents an interface that is a child
        interface that is parented to another interface.";
    }

    identity point-to-point {
        base iana-if-property-type;
        description
            "This property represents an interface that is always
            point-to-point, i.e. the interface can only ever be connected
            to a single other interface, and hence broadcast and multicast
            packet statistics do not make sense.";
    }

    identity multicast {
        base iana-if-property-type;
        description
            "This property represents an interface that could have multiple
            end points, e.g. like an Ethernet interface. Such an
            interface could have separate broadcast and multicast packet
            counters.";
    }

    identity ethernet-like {
        base iana-if-property-type;
        description
            "This property represents an interface that is similar in
            behaviour to an Ethernet interface, and uses Ethernet
            framing.";
    }
}

<CODE ENDS>

```

[7.](#) IANA Considerations

This draft proposes that IANA also manage a new registry of "interface properties" alongside the existing "interface type" registry, and to

extend the "interface type" registry to also derive from interface properties identities.

8. Security Considerations

This document discusses an approach how to structure interface related YANG schema. It has no security impact on the Internet.

9. Acknowledgments

This document arose from discussions with Martin Bjorklund, Ladislav Lhotka, and Vladimir Vassilev on the Netmod WG alias.

10. References

10.1. Normative References

- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", [RFC 7224](#), DOI 10.17487/RFC7224, May 2014, <<http://www.rfc-editor.org/info/rfc7224>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", [RFC 7950](#), DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.

10.2. Informative References

- [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>>.

Appendix A. Examples of possible updates to iana-if-types.yang

The example-iana-if-type.yang module illustrates the type of updates that would be made to iana-if-types.yang to make use of interface properties.

```
<CODE BEGINS> file "example-iana-if-type@2017-06-27.yang"

module example-iana-if-type {
  yang-version "1.1";
  namespace "urn:ietf:params:xml:ns:yang:example-iana-if-type";
```



```

import ietf-interfaces {
    prefix if;
}

import iana-if-property-type {
    prefix ianaifp;
}

// Full description, etc elided for clarity.
organization "IANA";
contact "";
description
    "This example module illustrates how iana-if-type.yang could
    be extended to use interface properties.";
reference
    "IANA 'ifType definitions' registry.
    <http://www.iana.org/assignments/smi-numbers>";

revision 2017-06-27 {
    description
        "Initial example of how IANA if type could be extended";
    reference
        "Taken from draft-rwilton-netmod-interface-properties-00";
}

// Previous revision statements elided for clarity.

identity iana-interface-type {
    base if:interface-type;
    description
        "This identity is used as a base for all interface types
        defined in the 'ifType definitions' registry.";
}

// Most identities elided for clarity, some are retained to
// illustrate how the interface properties are defined.
identity other {
    base iana-interface-type;
    description
        "None of the following";
}

identity ethernetCsmacd {
    base iana-interface-type;
    base ianaifp:physical;
    base ianaifp:multicast;
}

```

```
    base ianaifp:ethernet-like;
    description
        "For all Ethernet-like interfaces, regardless of speed,
        as per RFC 3635.";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types";
}

identity ieee8023adLag {
    base iana-interface-type;
    base ianaifp:virtual;
    base ianaifp:multicast;
    base ianaifp:ethernet-like;
    description
        "IEEE 802.3ad Link Aggregate.";
}

identity atm {
    base iana-interface-type;
    base ianaifp:physical;
    base ianaifp:multicast;
    description
        "ATM cells.";
}

identity atmSubInterface {
    base iana-interface-type;
    base ianaifp:virtual;
    base ianaifp:sub-interface;
    base ianaifp:multicast;
    description
        "ATM Sub Interface.";
}

identity l2vlan {
    base iana-interface-type;
    base ianaifp:virtual;
    base ianaifp:sub-interface;
    base ianaifp:multicast;
    description
        "Layer 2 Virtual LAN using 802.1Q.";
}

identity pos {
    base iana-interface-type;
```

```
base ianaifp:point-to-point;
description
```

```
    "Packet over SONET/SDH Interface.";
}

identity irb {
  base iana-interface-type;
  base ianaifp:virtual;
  base ianaifp:multicast;
  base ianaifp:ethernet-like;

  description
    "Integrated Routing and Bridging interface, also known as an
    SVI or BVI interface.

    Note, not currently defined in if-type.yang";
}
}

<CODE ENDS>
```

[Appendix B](#). Example of interface properties usage in ietf-interfaces-common.yang

The ietf-interfaces-common module defines various interface configuration nodes that are applicable to different types of interfaces and hence would benefit from interface properties.

<CODE BEGINS> file "example-ietf-interfaces-common@2017-06-27.yang"

```
module example-ietf-interfaces-common {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:example-ietf-interfaces-common";

  prefix if-cmn;

  import ietf-interfaces {
    prefix if;
  }
```

```
import iana-if-type {
  prefix ianaift;
}

import iana-if-property-type {
  prefix ianaifp;
}
```

Wilton

Expires January 4, 2018

[Page 10]

Internet-Draft

July 2017

```
organization
  "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

contact
  "WG Web:  <http://tools.ietf.org/wg/netmod/>
  WG List:  <mailto:netmod@ietf.org>

  WG Chair: Lou Berger
             <mailto:lberger@labn.net>

  WG Chair: Kent Watsen
             <mailto:kwatsen@juniper.net>

  Editor:   Robert Wilton
             <mailto:rwilton@cisco.com>";

description
  "Example of using when statements with interface properties";

revision 2017-06-27 {
  description
    "Examples of using when statements with interface properties";

  reference "Internet draft: draft-ietf-netmod-intf-ext-yang-04";
}

feature bandwidth {
  description "<description elided>";
  reference "Section 3.1 Bandwidth";
}

feature carrier-delay {
```

```
description "<description elided>";
reference "Section 3.2 Carrier Delay";
}
```

```
feature dampening {
  description "<description elided>";
  reference "Section 3.3 Dampening";
}
```

```
feature loopback {
  description "<description elided>";
  reference "Section 3.5 Loopback";
}
```

```
feature configurable-l2-mtu {
  description "<description elided>";
}
```

```
reference "Section 3.6 MTU";
}
```

```
feature sub-interfaces {
  description
    "This feature indicates that the device supports the
    instantiation of sub-interfaces. Sub-interfaces are defined
    as logical child interfaces that allow features and forwarding
    decisions to be applied to a subset of the traffic processed
    on the specified parent interface.";
  reference "Section 3.7 Sub-interface";
}
```

```
feature forwarding-mode {
  description "<description elided>";
  reference "Section 3.8 Forwarding Mode";
}
```

```
/*
 * Define common identities to help allow interface types to be
 * assigned properties.
 */
```

```
identity loopback {
  description "Base identity for interface loopback options";
}
```

```

}
identity loopback-internal {
    base loopback;
    description "<description elided>";
}
identity loopback-line {
    base loopback;
    description "<description elided>";
}
identity loopback-connector {
    base loopback;
    description "<description elided>";
}

/*
 * Augments the IETF interfaces model with a leaf to explicitly
 * specify the bandwidth available on an interface.
 */
augment "/if:interfaces/if:interface" {
    description
        "Augments the IETF interface model with optional common
        interface level commands that are not formally covered by any

```

```

        specific standard";

// Various features/nodes elided.

/*
 * Various types of interfaces support a configurable layer 2
 * encapsulation, any that are supported by YANG should be
 * listed here.
 *
 * Different encapsulations can hook into the common encaps-type
 * choice statement.
 */
container encapsulation {
    when
        "derived-from-or-self(..if:type, 'ianaifp:ethernet-like') or
        derived-from-or-self(..if:type, 'ianaifp:sub-interface')" {

```

```

    description
        "All interface types that can have a configurable L2
        encapsulation";
    /*
    * TODO - Should we introduce an abstract type to make this
    *         extensible to new interface types, or vendor
    *         specific interface types?
    */
}

description
    "Holds the OSI layer 2 encapsulation associated with an
    interface";
choice encaps-type {
    description "Extensible choice of L2 encapsulations";
}
}

/*
* Various types of interfaces support loopback configuration,
* any that are supported by YANG should be listed here.
*/
leaf loopback {
    when "derived-from(if:type, 'ianaifp:physical')" {
        description
            "Applies to all interfaces that derive from the physical
            interface property.";
    }

    if-feature "loopback";
}

```

```

type identityref {
    base loopback;
}
description "Enables traffic loopback.";
}

/*
* Many types of interfaces support a configurable layer 2 MTU.
*/
leaf l2-mtu {
    if-feature "configurable-l2-mtu";
}

```

```

    type uint16 {
        range "64 .. 65535";
    }
    description
        "The maximum size of layer 2 frames that may be transmitted
        or received on the interface (excluding any FCS overhead).
        In the case of Ethernet interfaces it also excludes the
        4-8 byte overhead of any known (i.e. explicitly matched by
        a child sub-interface) 801.1Q VLAN tags.";
    }
}

/*
 * Add generic support for sub-interfaces.
 *
 * This should be extended to cover all interface types that are
 * child interfaces of other interfaces.
 */
augment "/if:interfaces/if:interface" {
    when "derived-from(if:type, 'ianaifp:sub-interface')" {
        description
            "Applies to all interfaces that derive from the Ethernet-like
            interface property.";
    }
}

if-feature "sub-interfaces";

description
    "Add a parent interface field to interfaces that model
    sub-interfaces";
leaf parent-interface {

    type if:interface-ref;

    mandatory true;
    description
        "This is the reference to the parent interface of this

```

```

        sub-interface.";
    }
}
}

```


<CODE ENDS>

[Appendix C](#). Example of interface properties usage in ietf-interfaces-ethernet-like.yang

The ietf-interfaces-ethernet-like module defines various interface configuration nodes that are applicable to any interfaces that have "Ethernet-like" semantics, and hence would benefit from interface properties.

<CODE BEGINS> file "example-ietf-interfaces-ethernet-like@2017-06-27.yang"

```
module example-ietf-interfaces-ethernet-like {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:" +
    "example-ietf-interfaces-ethernet-like";

  prefix ethlike;

  import ietf-interfaces {
    prefix if;
  }

  import ietf-yang-types {
    prefix yang;
  }

  import iana-if-property-type {
    prefix ianaifp;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web:  <http://tools.ietf.org/wg/netmod/>
    WG List:  <mailto:netmod@ietf.org>

    WG Chair: Lou Berger
              <mailto:lberger@labn.net>
```

WG Chair: Kent Watsen
<mailto:kwatsen@juniper.net>

Editor: Robert Wilton
<mailto:rwilton@cisco.com>;

```
description
  "Example for when statements using interface properties.";

revision 2017-06-27 {
  description
    "Examples of using when statements with interface properties";

  reference "Internet draft: draft-ietf-netmod-intf-ext-yang-04";
}

/*
 * Configuration parameters for Ethernet-like interfaces.
 */
augment "/if:interfaces/if:interface" {
  when "derived-from(if:type, 'ianaifp:ethernet-like')" {
    description
      "Applies to all interfaces that derive from the Ethernet-like
      interface property.";
  }
  description
    "Augment the interface model with configuration parameters for
    all Ethernet-like interfaces";

  container ethernet-like {
    description "Contains configuration parameters for interfaces
      that use Ethernet framing and expose an Ethernet
      MAC layer.";
    leaf mac-address {
      type yang:mac-address;
      description
        "The configured MAC address of the interface.";
    }
  }
}

/*
 * Operational state for Ethernet-like interfaces.
 */
augment "/if:interfaces-state/if:interface" {
  when "derived-from(if:type, 'ianaifp:ethernet-like')" {
    description
```

"Applies to all interfaces that derive from the Ethernet-like

```
        interface property.";
    }
    description
        "Augments the interface model with operational state parameters
        for all Ethernet-like interfaces.";
    container ethernet-like {
        description
            "Contains operational state parameters for interfaces that use
            Ethernet framing and expose an Ethernet MAC layer.";
        leaf mac-address {
            type yang:mac-address;
            description
                "The operational MAC address of the interface, if
                applicable";
        }

        leaf bia-mac-address {
            type yang:mac-address;
            description
                "The 'burnt-in' MAC address. I.e the default MAC address
                assigned to the interface if none is explicitly
                configured.";
        }

        container statistics {
            description
                "Packet statistics that apply to all Ethernet-like
                interfaces";
            leaf in-drop-unknown-dest-mac-pkts {
                type yang:counter64;
                units frames;
                description "<long description elided>";
            }
        }
    }
}
}
```

<CODE ENDS>

[Appendix D](#). Example of interface properties usage in ietf-interfaces.yang

Here is an example of how the ietf-interfaces.yang module could have used interface properties to restrict multicast packet statistics to only those interfaces that support it.

<CODE BEGINS> file "example-ietf-interfaces@2017-06-27.yang"

Wilton

Expires January 4, 2018

[Page 17]

Internet-Draft

July 2017

```
module example-ietf-interfaces {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:example-ietf-interfaces";

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

  import "iana-if-property-type" {
    prefix ianaifp;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact "<elided>";
  description
    "Example of when statements for refining interface statistics";

  revision 2017-06-27 {
    description
      "Example of how some interface statistics could make use of
      interface properties";
    reference
      "RFC 7223: A YANG Data Model for Interface Management";
  }
  /*
  * Typedefs
  */
  // interface-ref typedef elided.
  typedef interface-state-ref {
    type leafref {
```

```

    path "/if:interfaces-state/if:interface/if:name";
  }
  description
    "This type is used by data models that need to reference
    the operationally present interfaces.";
}
/*
 * Identities
 */
identity interface-type {
  description
    "Base identity from which specific interface types are
    derived.";
}

```

Wilton

Expires January 4, 2018

[Page 18]

Internet-Draft

July 2017

```

// Features elided.
// Configuration tree elided.

/*
 * Operational state data nodes
 */
container interfaces-state {
  config false;
  description
    "Data nodes for the operational state of interfaces.";
  list interface {
    key "name";
    description
      "The list of interfaces on the device.
      System-controlled interfaces created by the system are
      always present in this list, whether they are configured or
      not.";
    leaf name {
      type string;
      description
        "The name of the interface.
        A server implementation MAY map this leaf to the ifName
        MIB object. Such an implementation needs to use some
        mechanism to handle the differences in size and characters
        allowed between this leaf and ifName. The definition of
        such a mechanism is outside the scope of this document.";
    }
  }
}

```

```

        reference
            "RFC 2863: The Interfaces Group MIB - ifName";
    }
    leaf type {
        type identityref {
            base interface-type;
        }
        mandatory true;
        description
            "The type of the interface.";
        reference
            "RFC 2863: The Interfaces Group MIB - ifType";
    }

    // Various leaves elided.

    container statistics {
        description
            "A collection of interface-related statistics objects.";
        leaf discontinuity-time {
            type yang:date-and-time;
            mandatory true;

```

```

        description "<description elided>";
    }
    leaf in-octets {
        type yang:counter64;
        description "<description elided>";
        reference
            "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
    }
    leaf in-unicast-pkts {
        type yang:counter64;
        description
            "The number of packets, delivered by this sub-layer to a
            higher (sub-)layer, that were not addressed to a
            multicast or broadcast address at this sub-layer.
            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            'discontinuity-time'.";
        reference

```

```

    "RFC 2863: The Interfaces Group MIB - ifHCInUcastPkts";
}
leaf in-broadcast-pkts {
    when "derived-from(if:type, 'ianaifp:multicast')" {
        description
            "Applies to interfaces that inherit the multicast
            interface property.";
    }

    type yang:counter64;
    description
        "The number of packets, delivered by this sub-layer to a
        higher (sub-)layer, that were addressed to a broadcast
        address at this sub-layer.
        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB -
        ifHCInBroadcastPkts";
}
leaf in-multicast-pkts {
    when "derived-from(if:type, 'ianaifp:multicast')" {
        description
            "Applies to interfaces that inherit the multicast
            interface property.";
    }
}

```

```

type yang:counter64;
description
    "The number of packets, delivered by this sub-layer to a
    higher (sub-)layer, that were addressed to a multicast
    address at this sub-layer. For a MAC-layer protocol,
    this includes both Group and Functional addresses.
    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
reference
    "RFC 2863: The Interfaces Group MIB -

```

```
        ifHCInMulticastPkts";
    }
    // Remaining counters elided.
}
}
}
}
```

<CODE ENDS>

Author's Address

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com