Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 23, 2015

H. Wang
V. Nagaraj
X. Chen
Huawei Technologies
May 22, 2015

# Yang Data Model for IKE
draft-wang-ipsecme-ike-yang-00

## Abstract

This document describes a YANG data model for the IKE (Internet Key Exchange) protocol. The model covers the IKE protocol configuration, operational state, remote procedural calls, and event notifications data.

## Requirements Language

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

## 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 November 23, 2015.

## Copyright Notice

Table of Contents



## 1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is a network

management protocol that defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for the NETCONF.





This document introduces a YANG data model for the IKE (Internet Key Exchange) protocol. There are two IKE protocols defined in IETF namely IKEv1(IKE version 1) and IKEv2(IKE version 2). IKEv1 protocol is obsolete now. The model discussed in this document covers IKEv2 [RFC7296] and other generic enhancements that pertain to the base protocol operation.

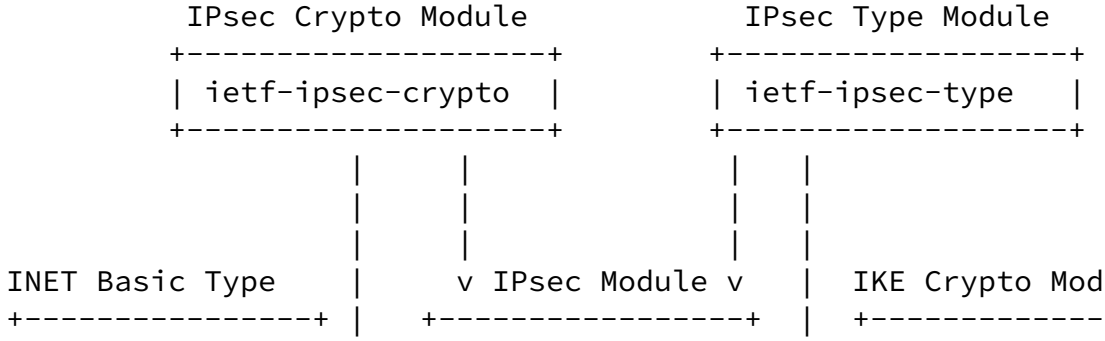
The data model is defined for following constructs that are used for managing the IKE protocol: configuration, operational state, remote procedural calls, and event notifications data.

## 2. IKE YANG Model Organization

### 2.1. Overview

The model discussd in this document covers IKEv2 [RFC7296] and other generic enhancements that pertain to the base protocol operation. The cryptographic algorithms are deliberately separated from ietf-ike model so that these algorithms can be updated or replaced without affecting the standardization progress of the rest of the IKE yang model. IPsec yang model, basic cryptographic algorithms for IPsec and basic IPsec type defines will be left out of this model to support IPsec basic information defined in [RFC4301]. IPsec yang model will be defined in separate document. IKE data model has the following relationship with IPsec module and other modules.

^: import



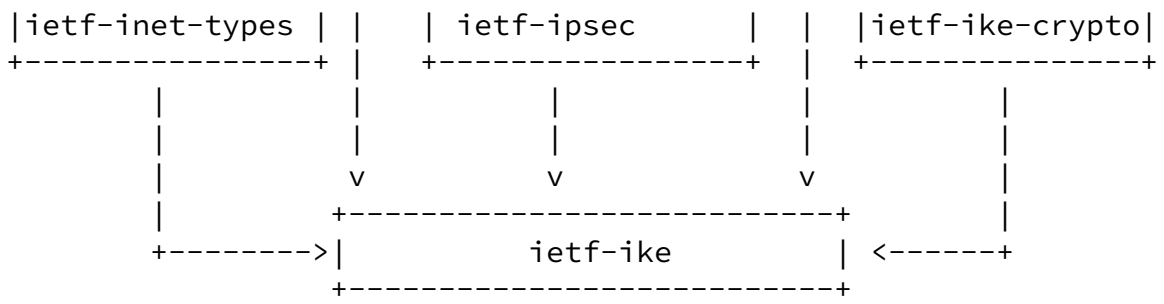



Figure 1: Relationship of IKE with IPsec module and other modules

This model aims to address only the core IKE parameters as per RFC 7296 [RFC7296].





This model does not cover any applications running on top of IKE nor does it cover any OAM procedures for IKE. Current revision only describes one address family of type "ipv4". The "ipv6" specific IKE configuration will be covered in later revision.

The figure below describes the overall structure of the IKE Yang model :

```
module: ietf-ike
   +--rw ike-global-configuration
   |  ...
   +--rw ipsec-proposal
   |  ...
   +--rw ike-proposal
   |  ...
   +--rw ike-peer
   |  ...
   +--rw ipsec-policy
   |  +--rw policy-entries* [policy-name sequence-number]
   |  |  ...
   |  +--rw policy-template-entries* [policy-name sequence-number]
   |     ...
   +--rw ipsec-interface-map
   |  ...
   +--ro ike-sa
   |  ...
   +--ro ipsec-sa
      ...
rpcs:
```

```
      +---x reset-ike-sa
      |  ...
      +---x reset-ipsec-sa
         ...
   notifications:
      +---n dpd-failure
      |  ...
      +---n peer-authentication-failure
      |  ...
      +---n ike-reauth-failure
      |  ...
      +---n ike-rekey-failure
      |  ...
      +---n ipsec-rekey-failure
         ...
```





## 2.2. Configuration

This specification defines the configuration parameters for IKE protocols version2 (IKEv2). This specification only supports ipv4 address type for IKE.

### 2.2.1. IPsec Global Configuration

The IKE global configuration includes some configuration that is common and applicable for all the IKE peers. This includes IKE local name, NAT-Keep-Alive interval, DPD Idle timeout, DPD interval, DPD retry count etc.

```
 +--rw ike-global-configuration
      +--rw (df-flag)?
      |  +--:(set)
      |  |  +--rw set?                      empty
      |  +--:(clear)
      |  |  +--rw clear?                    empty
      |  +--:(copy)
      |     +--rw copy?                     empty
      +--rw stateful-frag-check?      boolean
```

```
        +--rw life-time-kb?              uint32
        +--rw life-time-second?          uint32
        +--rw (anti-replay)?
        |  +--:(enable)
        |  |  +--rw enable?                    empty
        |  |  +--rw (anti-replay-windows-size)?
        |  |     +--:(size-32)
        |  |     +--:(size-64)
        |  |     +--:(size-128)
        |  |     +--:(size-256)
        |  |     +--:(size-512)
        |  |     +--:(size-1024)
        |  +--:(disable)
        |     +--rw disable?                   empty
        +--rw inbound-dscp?              uint16
        +--rw outbound-dscp?             uint16
        +--rw local-name?                string
        +--rw nat-keepalive-interval?    uint16
        +--rw dpd-interval?              uint16
```

### 2.2.2. IPsec Proposal Configuration

The IPsec proposal container will be used to include the configuration items related to the IPsec tunnel like tunnel protocol (sp, ah), tunnel encapsulation mode (tunnel/transport),





authentication algorithm for ah/esp and encryption algorithm for esp etc

```
   +--rw ipsec-proposal
      +--rw ipsec-proposal-entries* [proposal-name]
         +--rw proposal-name                   string
         +--rw (protocol)?
            +--:(ah)
            |  +--rw ah                                empty
            |  +--rw ah-authentication-algorithm?    ipsec-crypto:ipsec-authent
            +--:(esp)
               +--rw esp                             empty
               +--rw esp-authentication-algorithm?   ipsec-crypto:ipsec-authent
               +--rw esp-encryption-algorithm?       ipsec-crypto:ipsec-encrypt
```

### 2.2.3. IKE Proposal Configuration

The IKE proposal container is mainly use to hold information related to the IKE SA establishment parameters. These parameters are mainly negotiated between the IKE peers at the time of SA establishment. The various parameters in this container are proposal number, authentication method, integrity algorithm, encryption algorithm, Psuedo-Random function (prf), dh group, reauth , rekey lifetime etc

```
 +--rw ike-proposal
   +--rw ike-proposal-entries* [proposal-number]
      +--rw proposal-number        uint32
      +--rw auth-method?           ike-auth-method
      +--rw integrity-algorithm?   ike-crypto:ike-integrity-algorithm
      +--rw encrypt-algorithm?     ike-crypto:ike-encryption-algorithm
      +--rw prf-algorithm?         ike-crypto:ike-prf-algorithm
      +--rw dh-group?              ike-crypto:ike-dh-group
      +--rw reauth-interval?       uint32
      +--rw life-time?             uint32
```

### 2.2.4. IKE Peer Configuration

The IKE peer container will hold information about peer. The IKE peer is an entity that is going to establish security association with the remote peer. The main configuration parameters related to the IKE peer are: Key information, Name, proposal number, ID type, remote address, local address, certificate information etc





```
   +--rw ike-peer
      +--rw ike-peer-entries* [peer-name]
         +--rw peer-name              string
         +--rw ike-proposal-number?   ike-proposal-number-ref
         +--rw PresharedKey?          string
         +--rw nat-traversal?         boolean
         +--rw (local-id-type)?
         |  +--:(ip)
```

```
|  |  +--rw ip?                    empty
|  +--:(fqdn)
|  |  +--rw fqdn?                  empty
|  +--:(dn)
|  |  +--rw dn?                    empty
|  +--:(user_fqdn)
|     +--rw user_fqdn?             empty
+--rw local-id?               string
+--rw remote-id?              string
+--rw low-remote-address?     inet:ip-address
+--rw high-remote-address?    inet:ip-address
+--rw certificate?            string
+--rw auth-address-begin?     inet:ip-address
+--rw auth-address-end?       inet:ip-address
```

### 2.2.5. IPsec Policy Configuration

The IPsec policy container will hold values related to the IPsec policy that is bound to an interface (tunnel or physical interface). The information contained in the IPsec policy will determine the characteristics of the tunnel that is going to be establishment. The main attributes related to IPsec policy are: ACL, PFS (to do an additional DH exchange), peer name, IPsec proposal number, policy name, sequence number, policy-mode (ISAKMP, Template etc)

```
 +--rw ipsec-policy
   +--rw policy-entries* [policy-name sequence-number]
   |  +--rw policy-name               string
   |  +--rw sequence-number           uint32
   |  +--rw (policy-mode)?
   |     +--:(isakmp)
   |     |  +--rw isakmp?                    empty
   |     |  +--rw local-address?             inet:ip-address
   |     |  +--rw binding-interface-name?    string
   |     |  +--rw (acl)?
   |     |  |  +--:(acl-number)
   |     |  |  |  +--rw acl-number?               uint32
   |     |  |  +--:(advance-acl)
   |     |  |     +--rw advance-acl?              string
   |     |  +--rw pfs?                       ike-crypto:ike-dh-group
```





```
|     |  +--rw peer-name?                 ike-peer-name-ref
```

```
|      |  +--rw (df-flag)?
|      |  |  +--:(set)
|      |  |  |  +--rw set?                      empty
|      |  |  +--:(clear)
|      |  |  |  +--rw clear?                    empty
|      |  |  +--:(copy)
|      |  |     +--rw copy?                     empty
|      |  +--rw stateful-frag-check?      boolean
|      |  +--rw life-time-kb?             uint32
|      |  +--rw life-time-second?         uint32
|      |  +--rw (anti-replay)?
|      |  |  +--:(enable)
|      |  |  |  +--rw enable?                   empty
|      |  |  |  +--rw (anti-replay-windows-size)?
|      |  |  |     +--:(size-32)
|      |  |  |     +--:(size-64)
|      |  |  |     +--:(size-128)
|      |  |  |     +--:(size-256)
|      |  |  |     +--:(size-512)
|      |  |  |     +--:(size-1024)
|      |  |  +--:(disable)
|      |  |     +--rw disable?                  empty
|      |  +--rw inbound-dscp?             uint16
|      |  +--rw outbound-dscp?            uint16
|      |  +--rw ipsec-proposal* [proposal-name]
|      |     +--rw proposal-name    ipsec-proposal-name-ref
|      +--:(template)
|         +--rw template?                  empty
|         +--rw template-name              ipsec-policy-template-name-ref
+--rw policy-template-entries* [policy-name sequence-number]
   +--rw policy-name                string
   +--rw sequence-number            uint32
   +--rw local-address?             inet:ip-address
   +--rw binding-interface-name?    string
   +--rw (acl)?
   |  +--:(acl-number)
   |  |  +--rw acl-number?                    uint32
   |  +--:(advance-acl)
   |     +--rw advance-acl?                   string
   +--rw pfs?                       ike-crypto:ike-dh-group
   +--rw peer-name?                 ike-peer-name-ref
   +--rw (df-flag)?
   |  +--:(set)
   |  |  +--rw set?                           empty
   |  +--:(clear)
   |  |  +--rw clear?                         empty
   |  +--:(copy)
```

```
|     +--rw copy?                        empty
+--rw stateful-frag-check?      boolean
+--rw life-time-kb?             uint32
+--rw life-time-second?         uint32
+--rw (anti-replay)?
|  +--:(enable)
|  |  +--rw enable?                      empty
|  |  +--rw (anti-replay-windows-size)?
|  |     +--:(size-32)
|  |     +--:(size-64)
|  |     +--:(size-128)
|  |     +--:(size-256)
|  |     +--:(size-512)
|  |     +--:(size-1024)
|  +--:(disable)
|     +--rw disable?                     empty
+--rw inbound-dscp?             uint16
+--rw outbound-dscp?            uint16
+--rw ipsec-proposal* [proposal-name]
   +--rw proposal-name    ipsec-proposal-name-ref
```

### 2.2.6. IPsec Interface Map Configuration

The IPsec interface map container will have information related to the interface on which IPsec policy will be applied. It will have information like IPsec policy name, tunnel protocol, tunnel name , enable-disable UNR route generation etc

```
    +--rw ipsec-interface-map
      +--rw policy-interface* [interface-name]
         +--rw interface-name        string
         +--rw policy-name           ipsec-policy-name-ref
         +--rw generate-unr-route?   boolean
```

## 2.3. Operational State

The Operational state of the IKE SA or IPsec SA can be queried and obtained from the respective container. All the attributes/items in this container are read-only attributes and they reflect the run-time information of any established IKE SA.

### 2.3.1. IKE SA Container State

The IKE SA container is used to maintain information related to the IKE SA established. This SA is a run-time data structure that is created and has information about established SA like SPI, local and

remote address, established time, remaining life time, dh group, auth method, prf, encryption algorithm , integrity algorithm etc





```
+--ro ike-sa
   +--ro ike-sa-entries* [initiator-spi responder-spi]
      +--ro remote-address?          inet:ip-address
      +--ro local-address?           inet:ip-address
      +--ro initiator-spi            uint64
      +--ro responder-spi            uint64
      +--ro remote-id?               string
      +--ro local-id?                string
      +--ro auth-method?             ike-auth-method
      +--ro integrity-algorithm?     ike-crypto:ike-integrity-algorithm
      +--ro encryption-algorithm?    ike-crypto:ike-encryption-algorithm
      +--ro prf-algorithm?           ike-crypto:ike-prf-algorithm
      +--ro dh-group?                ike-crypto:ike-dh-group
      +--ro sa-established-time?     string
      +--ro remaining-time?          uint32
```

### 2.3.2. IPsec SA State

The IPsec SA container is used to maintain information related to the IPsec SA established. This is a run-time data structure that is created and has information about established SA like SPI, local and remote address, remaining life time, protocol etc

```
+--ro ipsec-sa
      +--ro ipsec-sa-entries*
         +--ro remote-address?   inet:ip-address
         +--ro local-address?    inet:ip-address
         +--ro responder-spi?    uint32
         +--ro remaining-time?   uint32
```

## 2.4. Actions

This model defines a list of RPCs that allow performing an action or executing a command on the protocol. For example, it allows clearing (reset) IKE SAs, IPsec SAs, statistics etc. The model makes an effort to provide different level of control so that a user is able to either clear all, or clear all of a given type, or clear a specific entity.

### 2.4.1. IKE SA reset action

This operation type is executed when the user wants to delete IKE SAs. The command gives a flexibility to delete all SAs or a particular SA only based on remote address, connection-id etc.





```
      +---x reset-ike-sa
         +---w input
         |  +---w (peer-info)?
         |     +--:(peer-id)
         |     |  +---w peer-id?         string
         |     +--:(peer-address)
         |        +---w peer-address?   inet:ip-address
         +--ro output
            +--ro status?   string
```

### 2.4.2. IPsec SA reset action

This operation type is executed when the user wants to delete IPsec SAs. The command gives a flexibility to delete all SAs or a particular SA only based on remote address, policy sequence number, remote peer address etc.

```
   +---x reset-ipsec-sa
         +---w input
         |  +---w (sa-info)?
         |     +--:(parameters)
         |     |  +---w (peer-info)
         |     |  |  +--:(peer-id)
         |     |  |  |  +---w peer-id?         string
         |     |  |  +--:(peer-address)
         |     |  |     +---w peer-address?   inet:ip-address
         |     |  +---w protocol         ipsec-type:ipsec-protocol
         |     |  +---w spi              ipsec-type:ipsec-spi
         |     +--:(remote-peer)
         |     |  +---w remote-peer
         |     |     +---w (peer-info)?
         |     |        +--:(peer-id)
```

```
|     |        |  +---w peer-id?        string
|     |        +--:(peer-address)
|     |           +---w peer-address?   inet:ip-address
|     +--:(policy)
|        +---w policy?         ipsec-policy-name-ref
+--ro output
   +--ro status?   string
```

## 2.5. Notifications

This model defines a list of notifications to inform client of important events detected during the protocol operation. These events include events related to changes in the operational state of an IKE SA, IPsec SA, Statistics etc.





### 2.5.1. DPD failure

This notification type is reported to the NETCONF client when there is a peer that is not responding to the DPD Keep Alive messages.

```
    +---n dpd-failure
         +--ro peer-id?   string
```

### 2.5.2. Peer Authentication failure

This notification type is reported to the NETCONF client when the peer authentication has failed due to either invalid key, certificate, invalid id etc

```
+---n peer-authentication-failure
    +--ro peer-id?   string
```

### 2.5.3. IKE Reauth failure

This notification type is reported to the NETCONF client when the re-authentication of the peer has failed. Reauth can fail due to many reasons like proposal mismatch during re-auth procedure, packet drop, dead peer etc

```
+---n ike-reauth-failure
```

```
     +--ro peer-id?   string
```

### 2.5.4. IKE Rekey failure

This notification type is reported to the NETCONF client when the IKE SA rekey has failed. The rekey is an operation used to refresh the IKE SA keys. It can fail due to proposal mismatch during rekey procedure, packet drop, dead peer etc

```
+---n ike-rekey-failure
    +--ro peer-id?     string
    +--ro old-i-spi?   uint64
  +--ro old-r-spi?   uint64
```

### 2.5.5. IPsec Rekey failure

This notification type is reported to the NETCONF client when the IKE SA rekey has failed. The rekey is an operation used to refresh the IPsec SA keys. It can fail due to proposal mismatch during rekey procedure, packet drop, dead peer etc





```
    +---n ipsec-rekey-failure
         +--ro peer-id?            string
         +--ro old-inbound-spi?    ipsec-type:ipsec-spi
         +--ro old-outbound-spi?   ipsec-type:ipsec-spi
```

## 3. IKE Yang Module

To support separately upgrade the algorithm part, the base data model and the algorithm part are defined as two separately parts.

### 3.1. IKE Basic Yang Module

```
module ietf-ike {
    namespace "urn:ietf:params:xml:ns:yang:ietf-ike";
    // replace with IANA namespace when assigned
    prefix "ike";

    import "ietf-ipsec-crypto" {
```

```
        prefix "ipsec-crypto";
    }

    import "ietf-ike-crypto" {
        prefix "ike-crypto";
    }

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

    import "ietf-ipsec-type" {
        prefix "ipsec-type";
    }

    import "ietf-ipsec" {
        prefix "ipsec";
    }

    organization "Huawei Technologies India Pvt Ltd";
    contact "stonewater.wang@huawei.com";
    description "IKE Yang module define";

    revision 2015-04-18 {
        description "Initial revision.";
        reference "RFC XXXX: IKE Yang Modules";
    }
```





```
    grouping ipsec-common-configuration {
        choice df-flag {
            default copy;
            case set {
                leaf set {
                    type empty;
                    description
                        "Set the df bit when encapsulate IPsec tunnel.";
                }
            }
            case clear {
```

```
                leaf clear {
                    type empty;
                    description
                        "Clear the df bit when encapsulate IPsec tunnel.";
                }
            }
            case copy {
                leaf copy {
                    type empty;
                    description
                        "Copy the inner IP header df bit.";
                }
            }

            description
                "It indicates how to process the df bit when encapsulate IPsec
        }
        leaf stateful-frag-check {
            type boolean;
            default false;
            description "Whether stateful fragment checking applies.";
        }
        leaf life-time-kb {
            type uint32;
            units "KB";
            default 2000000;

            description "IPsec SA Life time in KB.";
        }
        leaf life-time-second {
            type uint32;
            units "Second";
            default 18400;
            description "IPsec SA Life time in Seconds";
        }
        choice anti-replay {
            default enable;
```





```
            case enable {
               leaf enable {
                    type empty;
                    description "Enable Anti-replay";
```

```
                }
                choice anti-replay-windows-size {

                     case size-32;
                     case size-64;
                     case size-128;
                     case size-256;
                     case size-512;
                     case size-1024;
                     default size-1024;
                     description "It indicate the size of anti-replay window";
                }
            }
            case disable {
                leaf disable {
                    type empty;
                    description "Disable Anti-replay";
                }
            }
            description "Whether enable or disable anti-replay";
        }

        leaf inbound-dscp {
            type uint16 {
                range "0..63";
            }
            default 0;
            description "Inbound DSCP value";
        }
        leaf outbound-dscp {
            type uint16 {
                range "0..63";
            }
            default 0;
            description "Outbound DSCP value";
        }
        description "Common IPsec configurations";
    }

    grouping choose-ipsec-peer {
        choice peer-info {
            case peer-id {
                leaf peer-id {
                    type string;
```

```
                    description "Peer ID";
                }
            }
            case peer-address {
                leaf peer-address {
                    type inet:ip-address;
                    description "Peer IP Address";
                }
            }
            description "Reset according to peer information";
        }
        description "IKE peer information when do reset operation";
    }

    typedef ike-peer-name-ref {
        type leafref {
            path "/ike-peer/ike-peer-entries/peer-name";
        }
        description "reference to ike peer name";
    }

    typedef ike-proposal-number-ref {
        type leafref {
            path "/ike-proposal/ike-proposal-entries/proposal-number";
        }
        description "reference to ike proposal number";
    }

    typedef ipsec-proposal-name-ref{
        type leafref {
            path "/ipsec-proposal/ipsec-proposal-entries/proposal-name";
        }
        description "reference to ike proposal name";
    }

    typedef ipsec-policy-template-name-ref {
        type leafref {
            path "/ipsec-policy/policy-template-entries/policy-name";
        }
        description "reference to ipsec policy template name";
    }

    typedef ike-auth-method {
        type enumeration {
            enum pre-share {
                description "Select pre-shared key message as the authenticatio
            }
```

```
            enum rsa-digital-signature {
```





```
                description "Select rsa digital signature as the authentication
            }
            enum dss-digital-signature {
                description "Select dss digital signature as the authentication
            }
        }
        description "IKE authentication methods";
    }

    typedef ipsec-policy-name-ref {
        type leafref {
            path "/ipsec-policy/policy-entries/policy-name";
        }
        description "reference to ipsec policy name";
    }

    container ike-global-configuration {
        description "Global IKE configurations";

        uses ipsec-common-configuration;

        leaf local-name {
            type string;
            description "Global local name configuration, if it is not configed
                         ip address will be used as default. If configing specia
                         local name for special peer, it will overwrite the glob
                         name configuration when negotion with that peer.";
        }
        leaf nat-keepalive-interval {

            type uint16 {
                range "5..300";
            }
            units "Seconds";
            default 20;
            description "Global nat keepalive interval";
        }
        leaf dpd-interval {
            type uint16 {
                range "10..3600";
```

```
            }

            units "Seconds";
            default 30;
            description "Global DPD interval";
        }
    }
```





```
    container ipsec-proposal {
        description "IPsec proposal information";

        list ipsec-proposal-entries {
            key "proposal-name";
            description "IPsec proposal information";

            leaf proposal-name {
                type string;
                mandatory true;
                description "Name of IPsec proposal.";
            }
            choice protocol {
                default esp;
                case ah {
                    leaf ah {
                        type empty;
                        mandatory true;
                        description "Choose AH as IPsec protocol";
                    }
                    leaf ah-authentication-algorithm {
                        type ipsec-crypto:ipsec-authentication-algorithm;
                        must "ah-authentication-algorithm != 'null'" {
                            error-message "AH authentication algorithm MUST not
                            description "AH authentication algorithm MUST not b

                        }
                        default sha2-256;
                        description "IPsec authentication algorithm for AH";
                    }
                    description "Choose AH as IPsec protocol";
                }
                case esp {
```

```
                    leaf esp {
                        type empty;
                        description "Choose ESP as IPsec protocol";
                    }
                    leaf esp-authentication-algorithm {
                        type ipsec-crypto:ipsec-authentication-algorithm;
                        default sha2-256;
                        description "IPsec authentication algorithm for ESP";
                    }

                    leaf esp-encryption-algorithm {
                        type ipsec-crypto:ipsec-encryption-algorithm;
                        default aes-256;
                        description "IPsec encryption algorithm for ESP";
```





```
                    }
                    must "esp-authentication-algorithm != 'null' or esp-encrypt

                        error-message "ESP authentication algorithm and encrypt
                        description "ESP authentication algorithm and encryptio
                    }
                    description "Choose ESP as IPsec protocol";
                }
                description "Choose IPsec protocol";
            }

        } //End of IPsecProposalEntries
    }//End of IPsec Proposal

    container ike-proposal {
        description "IKE proposal information";

        list ike-proposal-entries {

            key "proposal-number";
            description "IKE proposal information";

            leaf proposal-number {
                type uint32;
                mandatory true;
                description "Proposal seq-number of ike proposal";
```

```
            }
            leaf auth-method {

                type ike-auth-method;
                default pre-share;
                description "authentication method of ike peer";
            }
            leaf integrity-algorithm {

                type ike-crypto:ike-integrity-algorithm;
                default hmac-sha2-256;
                description "integrity algorithm of ike protocol";
            }
            leaf encrypt-algorithm {

                type ike-crypto:ike-encryption-algorithm;
                default aes-cbc-256;
                description "Encryption algorithm of ike protocol";
            }
            leaf prf-algorithm {
                type ike-crypto:ike-prf-algorithm;
                default hmac-sha2-256;
```





```
                description "Prf algorithm of ike protocol";
            }
            leaf dh-group {

                type ike-crypto:ike-dh-group;
                must "dh-group != 'dh-group-none'" {
                    error-message "DH Group MUST be configurated";
                    description "DH Group MUST be configurated";
                }
                default dh-group-2;

                description "DH group of ike protocol";
            }
            leaf reauth-interval {

                type uint32 {
                    range "60..604800";
                }
                units "Seconds";
```

```
                default 86400;
                description "Reauth interval time of IKE protocol";
            }
            leaf life-time {
                type uint32 {
                    range "60..604800";
                }
                units "Seconds";
                default 86400;
                description "IKE SA life time";
            }

        } //End of IKEProposal
    }
    container ike-peer {
        description "IKE peer information";

        list ike-peer-entries {

            key "peer-name";
            description "IKE peer information";

            leaf peer-name {
                type string;
                mandatory true;
                description "Name of IKE peer";
            }

            leaf ike-proposal-number {
```





```
                type ike-proposal-number-ref;
                description "IKE proposal number referenced by IKE peer";
            }

            leaf PresharedKey {
                type string;
                description "Preshare key";
            }

            leaf nat-traversal {
                type boolean;
                default false;
```

```
            description "Enable/Disable nat traversal";
        }

        choice local-id-type {
            default ip;
            case ip {
                leaf ip {
                    type empty;
                    description "IP address";
                }
            }
            case fqdn {
                leaf fqdn {
                    type empty;
                    description "Fully Qualifed Domain name ";
                }
            }
            case dn {
                leaf dn {
                    type empty;
                    description "Domain name";
                }
            }
            case user_fqdn {
                leaf user_fqdn {
                    type empty;
                    description "User FQDN";
                }
            }
            description "Local ID type";
        }
        leaf local-id {
            type string;
            description "Local ID Name. When IP is used as local ID type,
                         it is ignored. If it is not configurated,
                         global local name will be used.";
```





```
        }
        leaf remote-id {
            type "string";
            description "ID of IKE peer";
        }
```

```
            leaf low-remote-address {
                type inet:ip-address;
                description "Low range of remote address";
            }
            leaf high-remote-address {
                type inet:ip-address;
                description "High range of remote address";
            }
            leaf certificate {
                type string;
                description "Certificate file name";
            }
            leaf auth-address-begin {
                type inet:ip-address;
                description "The begin range of authenticated peer address";
            }
            leaf auth-address-end {
                type inet:ip-address;
                description "The end range of authenticated peer address";
            }
        }

    }//End of IKEPeerEntries
    container ipsec-policy {
        description "IPsec policy information";

        grouping policy-content {
            leaf local-address {
                type inet:ip-address;
                description
                    "Local address used by IKE when negotiate with peer,
                     if it is not configed, the interface address with bind
                     this ipsec policy will be used.";
            }
            leaf binding-interface-name {
                type string;
                description "The interface that the policy is already bind with
            }
            choice acl {
                case acl-number {
                    leaf acl-number {
                        type uint32 {
                            range "3000..3999";
```

```
                }
                description "Config common acl as IPsec traffic selecto
            }

        }
        case advance-acl {
            leaf advance-acl {
                type string {
                    length "1..32";
                }
                description "Config advance acl as IPsec traffic select
            }
        }
        description "Config acl as IPsec traffic selector";
    }


    leaf pfs {
        type ike-crypto:ike-dh-group;
        default dh-group-none;
        description
            "Whether choose different DH group with IKE SA when create
             ipsec SA to increase perfect forwarding security";
    }

    leaf peer-name {
        type ike-peer-name-ref;
        description "The ike peer binding with this policy";
    }

    uses ipsec-common-configuration {
        description "The common configuration of IPsec SA";
    }

    list ipsec-proposal {
        key "proposal-name";
        max-elements "6";
        description "The ipsec-proposals binding with the policy";

        leaf proposal-name {
            type ipsec-proposal-name-ref;
            description "The ipsec-proposals binding with the policy";
        }
    }
    description "IPsec policy content";
}

list policy-entries {
```

```
            key "policy-name sequence-number";
            description "IPsec policy information";

            leaf policy-name {
                type string;
                mandatory true;
                description "IPsec policy group name";

            }
            leaf sequence-number {
                type uint32;
                mandatory true;
                description "IPsec policy sequence number";
            }
            choice policy-mode {

                case isakmp {
                    leaf isakmp {
                        type empty;
                        description "Common ISAKMP IPsec policy";
                    }
                    uses policy-content {
                        description "common ipsec policy content";
                    }
                }
                case template {
                    leaf template {
                        type empty;
                        description "ISAKMP IPsec policy created using template
                    }
                    leaf template-name {
                        type ipsec-policy-template-name-ref;
                        mandatory true;
                        description
                            "The IPsec policy template name which is used to cr
                    }
                }
                default isakmp;
                description "IPsec policy mode";

            }
```

```
        }

        list policy-template-entries {
            key "policy-name sequence-number";
            description "IPsec policy template define";
```





```
            leaf policy-name {
                type string {
                    length "1..15";
                }
                mandatory true;
                description "IPsec policy template name";
            }
            leaf sequence-number {
                type uint32;
                mandatory true;
                description "Sequence number of policy template";

            }
            uses policy-content {
                description "common ipsec policy content";
            }
        }

    }
    container ipsec-interface-map {
        description "The map information between IPsec policy and interface";

        list policy-interface {
            key "interface-name";
            description "The map information between IPsec policy and interface

            leaf interface-name {
                type string;
                mandatory true;
                description "Interface name which will bind IPsec policy";
            }
            leaf policy-name {
                type ipsec-policy-name-ref;
                mandatory true;
                description "IPsec policy name";
```

```
            }
            leaf generate-unr-route {
                type boolean;
                default false;
                description "Whether generate UNR route";
            }
        }
    }

    container ike-sa {
        config false;
        description "IKE SA informations";
```





```
        list ike-sa-entries {

            key "initiator-spi responder-spi";

            description "IKE SA informations";

            leaf remote-address {
                type inet:ip-address;
                description "The IP address of the remote peer";
            }
            leaf local-address {
                type inet:ip-address;
                description "The IP address of local";
            }
            leaf initiator-spi {
                type uint64;
                description "The SPI of initiator";
            }
            leaf responder-spi {
                type uint64;
                description "The SPI of responder";
            }
            leaf remote-id {
                type string;
                description "The ID of the remote peer";
            }
            leaf local-id {
                type string;
```

```
                description "The ID of local";
            }
            leaf auth-method {
                type ike-auth-method;
                description "The authentication method of IKE peer";
            }
            leaf integrity-algorithm {
                type ike-crypto:ike-integrity-algorithm;
                description "The integrity algorithm chosen by IKE negotiation"
            }
            leaf encryption-algorithm {
                type ike-crypto:ike-encryption-algorithm;
                description "The encryption algorithm chosen by IKE negotiation
            }
            leaf prf-algorithm {
                type ike-crypto:ike-prf-algorithm;
                description "The PRF algorithm chosen by IKE negotiation";
            }
            leaf dh-group {
                type ike-crypto:ike-dh-group;
```





```
                description "The DH group chosen by IKE negotiation";
            }
            leaf sa-established-time {
                type string;
                description "The establish time of the IKE SA";
            }
            leaf remaining-time {
                type uint32;
                description "The remain life time of IKE SA";
            }

        }

    }
    container ipsec-sa {
        config false;
        description "IPsec SA information";

        list ipsec-sa-entries {

            description "IPsec SA information";
```

```
            leaf remote-address {
                type inet:ip-address;
                description "The IP address of the remote tunnel end-point";
            }
            leaf local-address {
                type inet:ip-address;
                description "The IP address of local tunnel end-point";
            }
            leaf responder-spi {
                type uint32;
                description "The SPI of responder";
            }

            leaf remaining-time {
                type uint32;
                description "The remain life time of IPsec SA";
            }

        }

    }

    rpc reset-ike-sa {
        description "Reset IKE SA";
        input {
```





```
            uses choose-ipsec-peer;
            description "Reset IKE SA";
        }
        output {
            leaf status {
                type string;
                description "Operation status";
            }
        }
    }
    rpc reset-ipsec-sa {
        description "Reset IPsec SA";
        input {
            choice sa-info {
```

```
                case parameters {
                    uses choose-ipsec-peer {
                        refine "peer-info" {
                            mandatory true;
                        }
                    }
                    leaf protocol {
                        type ipsec-type:ipsec-protocol;
                        mandatory true;
                        description "SA protocol";
                    }
                    leaf spi {

                        type ipsec-type:ipsec-spi;
                        mandatory true;
                        description "SA SPI";
                    }
                    description "Reset according to special parameters";
                }

                case remote-peer {
                    container remote-peer {
                        uses choose-ipsec-peer;
                        description "Reset according to remote peer";
                    }
                }
                case policy {
                    leaf policy {
                        type ipsec-policy-name-ref;
                        description "Reset according to IPsec policy name";
                    }
                }
                description "Reset according to special information";
```





```
            }

        }
        output {
            leaf status {
                type string;
                description "Operation status";
```

```
            }
        }
    }

    notification dpd-failure{
        description "IKE peer DPD detect failure";
        leaf peer-id {
            type string;
            description "Peer ID";
        }
    }

    notification peer-authentication-failure {
        description "Peer authentication fail when negotication";
        leaf peer-id {
            type string;
            description "The ID of remote peer";
        }
    }

    notification ike-reauth-failure {
        description "IKE peer reauthentication fail";
        leaf peer-id {
            type string;
            description "The ID of remote peer";
        }
    }

    notification ike-rekey-failure {
        description "IKE SA rekey failure";
        leaf peer-id {
            type string;
            description "The ID of remote peer";
        }
        leaf old-i-spi {
            type uint64;
            description "old SPI";
        }
        leaf old-r-spi {
            type uint64;
            description "old SPI";
```

```
        }
    }

    notification ipsec-rekey-failure {
        description "IPsec SA rekey failure";
        leaf peer-id {
            type string;
            description "The ID of remote peer";
        }
        leaf old-inbound-spi {
            type ipsec-type:ipsec-spi;
            description "old inbound SPI";
        }
        leaf old-outbound-spi {
            type ipsec-type:ipsec-spi;
            description "old outbound SPI";
        }
    }


}
```

## 3.2. IKE Algorithm Yang Module

```
   module ietf-ike-crypto {
     namespace "urn:ietf:params:xml:ns:yang:ietf-ike-crypto";
     prefix ike-crypto;

     organization "Huawei Technologies India Pvt Ltd";
     contact
       "stonewater.wang@huawei.com";
     description
       "IKE Crypto Yang";
     reference "RFC 7296: Internet Key Exchange Protocol Version 2";

     revision 2015-04-18 {
       description
         "Initial revision.";
       reference "RFC 7296: Internet Key Exchange Protocol Version 2";
     }

     typedef ike-integrity-algorithm {
       type enumeration {
         enum "hmac-md5-96" {
           description
             "HMAC-MD5-96 Integrity Algorithm";
         }
         enum "hmac-sha1-96" {
```

```
          description
            "HMAC-SHA1-96 Integrity Algorithm";
        }
        enum "hmac-sha2-256" {
          description
            "HMAC-SHA2-256 Integrity Algorithm";
        }
        enum "hmac-sha2-384" {
          description
            "HMAC-SHA2-384 Integrity Algorithm";
        }
        enum "hmac-sha2-512" {
          description
            "HMAC-SHA2-512 Integrity Algorithm";
        }
      }
      description
        "typedef for ike integrity algorithm.";
    }

    typedef ike-encryption-algorithm {
      type enumeration {
        enum "des-cbc" {
          description
            "DES-CBC Encryption algorithm";
        }
        enum "3des-cbc" {
          description
            "3DES-CBC Encryption algorithm";
        }
        enum "aes-cbc-128" {
          description
            "AES-CBC-128 Encryption algorithm";
        }
        enum "aes-cbc-192" {
          description
            "AES-CBC-192 Encryption algorithm";
        }
        enum "aes-cbc-256" {
          description
            "AES-CBC-256 Encryption algorithm";
        }
      }
```

```
      description
        "typedef for ike encryption algorithm.";
    }

    typedef ike-prf-algorithm {
```





```
      type enumeration {
        enum "hmac-md5-96" {
          description
            "HMAC-MD5-96 PRF Algorithm";
        }
        enum "hmac-sha1-96" {
          description
            "HMAC-SHA1-96 PRF Algorithm";
        }
        enum "hmac-sha2-256" {
          description
            "HMAC-SHA2-256 PRF Algorithm";
        }
        enum "hmac-sha2-384" {
          description
            "HMAC-SHA2-384 PRF Algorithm";
        }
        enum "hmac-sha2-512" {
          description
            "HMAC-SHA2-512 PRF Algorithm";
        }
      }
      description
        "typedef for ike prf algorithm.";
    }

    typedef ike-dh-group {
      type enumeration {
        enum "dh-group-none" {
          description
            "None Diffie-Hellman group";
        }
        enum "dh-group-1" {
          description
            "768 bits Diffie-Hellman group";
        }
```

```
        enum "dh-group-2" {
          description
            "1024 bits Diffie-Hellman group";
        }
        enum "dh-group-5" {
          description
            "1536 bits Diffie-Hellman group";
        }
        enum "dh-group-14" {
          description
            "2048 bits Diffie-Hellman group";
        }
```





```
      }
      description
        "typedef for ike dh group";
    }
  }
```

## 4. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-ike XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ike-crypto XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-ike namespace: urn:ietf:params:xml:ns:yang:ietf-ike prefix: ike reference: [RFC7296]

name: ietf-ike-crypto namespace: urn:ietf:params:xml:ns:yang:ietf-ike-crypto prefix: ike-crypto reference: [RFC7296]

## 5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol[RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content. There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations.

## 6. Acknowledgements





## 7. Normative References

[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.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

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

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, October 2014.

Authors' Addresses

Honglei Wang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: stonewater.wang@huawei.com

Vijay Kumar Nagaraj
Huawei Technologies
Huawei Technologies India Pvt Ltd
Bangalore 560008
India

Email: vijay.kn@huawei.com





Xia Chen
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: xiachen@huawei.com