SAVI
Internet-Draft
Intended status: Experimental
Expires: February 17, 2018

C. An
J. Yang
J. Wu
J. Bi
CERNET

August 16, 2017

A Yang Data Model for SAVI Management
draft-an-savi-yang-02

Abstract

This document contains a specification of YANG modules for the management of SAVI (Source Address Validation Improvements) protocol.

The core SAVI data module ietf-savi serves as a framework for configuring and managing SAVI instance and provides common building blocks. It is expected to be augmented by additional YANG modules for specific IP address assignment methods.

The other four modules augment the core SAVI data module and define data models for different IP address assignment methods. Module ietf-savi-fcfs defines module specific for Stateless Address Auto Configuration (SLAAC), module ietf-savi-dhcpv4 and ietf-savi-dhcpv6 define modules specific for Dynamic Host Configuration Protocol version 4 and version 6 (DHCPv4 and DHCPv6), and module ietf-savi-send defines module specific for Secure Neighbor Discovery (SEND).

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 February 17, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Source Address Validation Improvement protocol was developed to complement ingress filtering with finer-grained, standard IP source address validation([RFC7039]). A SAVI protocol instance is located on the path of hosts' packets, enforcing the hosts' use of legitimate IP source addresses.

SAVI protocol determines whether the IP address obtaining process is legitimate according to IP address assignment method. For links with Stateless Address Auto Configuration (SLAAC), the process is defined in [RFC6620]. For links with Dynamic Host Configuration Protocol (DHCP), the process is defined in [RFC7513]. For links with Secure Neighbor Discovery (SEND), the process is defined in [RFC7219].

This document contains a core SAVI data module serving as a framework for configuring and managing SAVI instance and provides common building blocks. The other four modules augment the core SAVI data module and define data models for different IP address assignment methods.

- Module "ietf-savi" defines a core data module which provides generic components of SAVI data model, and is intended as a basis for future data model development covering more IP address assignment methods.

- Module "ietf-savi-fcfs" augments the "ietf-savi" module with additional data specific to SAVI FCFS ([RFC6620]).

- Module "ietf-savi-dhcp4" augments the "ietf-savi" module with additional data specific to SAVI DHCP ([RFC7513]) for IPv4 address assignment.

- Module "ietf-savi-dhcp6" augments the "ietf-savi" module with additional data specific to SAVI DHCP ([RFC7513]) for IPv6 address assignment.

- Module "ietf-savi-send" augments the "ietf-savi" module with additional data specific to SAVI SEND ([RFC7219]).

2. Terminology and Notation

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].

The following terms are defined in RFC6241 [RFC6241]:

- client,
- message,
- protocol operation,
- server

The following terms are defined in RFC6020 [RFC6020]:

- augment,
- configuration data,
- container,
- data model,
- data node,
- leaf,
- list,
- mandatory node,
- module

The following terms are defined in [RFC7039].

- IP Address Assignment Methods,
- SAVI method,
- Binding Anchors,
- SAVI instance

The following terms are defined in [RFC6620].

- SAVI FCFS,
- Validating Ports (VPs),
- Trusted Ports (TPs),
- Lifetime

- Status: either NO_BIND, TENTATIVE, VALID, TESTING_VP, or TESTING_TP-LT,
- Creation time,
- TENT_LT,
- DEFAULT_LT,
- T_WAIT

The following terms are defined in [RFC7513].

- SAVI DHCP,
- Binding entry: A rule that associates an IP address with a binding anchor,
- Binding State Table (BST): The data structure that contains the binding entries,
- Binding entry limit: The maximum number of binding entries that may be associated with a binding anchor,
- Status: either NO_BIND, INIT_BIND, BOUND, DETECTION , RECOVERY, or VERIFY,
- Trust Attribute,
- DHCP-Trust Attribute,
- DHCP-Snooping Attribute,
- Data-Snooping Attribute,
- Validating Attribute,
- MAX_DHCP_RESPONSE_TIME,
- MAX_LEASEQUERY_DELAY,
- DETECTION_TIMEOUT,
- DATA_SNOOPING_INTERVAL,
- OFFLINK_DELAY

The following terms are defined in [RFC6620].

- SAVI SEND,
- Validating Ports (VPs),
- Trusted Ports (TPs),
- Status: either TENTATIVE_DAD, TENTATIVE_NUD, VALID, TESTING_VP, or TESTING_VP',
- TENT_LT,
- DEFAULT_LT

2.1. Glossary of New Terms

system-controlled entry: An entry of a list in state data ("config false") that is created by the system independently of what has been explicitly configured. See Section 4.1 for details.

user-controlled entry: An entry of a list in state data ("config false") that is created and deleted as a direct consequence of certain configuration changes. See Section 4.1 for details.

2.2. Tree Diagrams

Simplified graphical representation of the data tree is presented in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only), "-x" RPC operations, and "-n" notifications.
- Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2.3. Prefixes in Data Node Names

In this document, names of data nodes, RPC operations and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC7223]
savi	ietf-savi	Section 6
savi-fcfs	ietf-savi-fcfs	Section 7
savi-dhcpv4	ietf-savi-dhcpv4	Section 8
savi-dhcpv6	ietf-savi-dhcpv6	Section 9
savi-send	ietf-savi-send	Section 10
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

3. Objectives

The initial design of the SAVI data model was driven by the following objectives:

- The data model should be suitable for different IP address assignment method proposed now, and can be augmented to support new IP address assignment method in different scenarios, such as WLAN, IPv4/IPv6 Transition Network, etc.

- The data model should be suitable for the common address families, in particular IPv4 and IPv6.

- A simple IP assignment system, such as one that uses only static IP, should be configurable in a simple way, which are called savi-manual.

4. The Design of the SAVI Data Model

The SAVI data model consists of five YANG modules. The first module, "ietf-savi", defines the generic components of a SAVI system. The other four modules, "ietf-savi-fcfs", "ietf-savi-dhcpv4", "ietf-savi-

dhcpv6" and "ietf-savi-send", augment the "ietf-savi" module with additional data nodes that are needed for the specific IP address assignment method, respectively. Figures 1 and 2 show abridged views of the configuration and state data hierarchies. See Appendix A for the complete data trees.

```
  +--rw savi
     +--rw savi-instances
     |  +--rw savi-instance* [savi-method]
     |     +--rw savi-method           string
     |     +--rw enable?               boolean
     |     +--rw preference?           uint32
     |     +--rw savi-fcfs:params
     |     |  +--...
     |     +--rw savi-dhcpv4:params
     |     |  +--...
     |     +--rw savi-dhcpv6:params
     |     |  +--...
     |     +--rw savi-send:params
     |        +--...
     +--rw interfaces
     |  +--rw interface* [ifname]
     |     +--rw ifname               if:interface-ref
     |     +--rw filtering-enabled?   boolean
     +--rw binding-table
        +--rw ipv4
        |  +--rw binding-entry* [ifname address]
        |     +--rw address           inet:ipv4-address
        |     +--rw ifname            if:interface-ref
        |     +--rw mac?              yang:mac-address
        |     +--rw lifetime          yang:timeticks
        |     +--rw creationtime      yang:timestamp
        |     +--rw binding-method    string
        +--rw ipv6
           +--rw binding-entry* [ifname address]
              +--rw address           inet:ipv6-address
              +--rw ifname            if:interface-ref
              +--rw mac?              yang:mac-address
              +--rw lifetime          yang:timeticks
              +--rw creationtime      yang:timestamp
              +--rw binding-method    string
```

Figure 1: Configuration data hierarchy.

```
  +--ro savi-state
     +--ro savi-instances
  |  |  +--ro savi-instance* [savi-method]
  |  |     +--ro savi-method                        string
  |  |     +--ro preference?                        uint32
  |  |     +--ro savi-fcfs:binding-state-table
  |  |     |  +--ro savi-fcfs:binding-state-entry* [ifname address]
  |  |     |     +--...
  |  |     +--ro savi-dhcpv4:binding-state-table
  |  |     |  +--ro savi-dhcpv4:binding-state-entry* [ifname address]
  |  |     |     +--...
  |  |     +--ro savi-dhcpv6:binding-state-table
  |  |     |  +--ro savi-dhcpv6:binding-state-entry* [ifname address]
  |  |     |     +--...
  |  |     +--ro savi-send:binding-state-table
  |  |        +--ro savi-send:binding-state-entry* [ifname address]
  |  |           +--...
  |  +--ro binding-table
  |  |  +--ro ipv4
  |  |  |  +--ro binding-entry* [ifname address]
  |  |  |     +--...
  |  |  +--ro ipv6
  |  |     +--ro binding-entry* [ifname address]
  |  |        +--...
  |  +--ro statistics
  |     +--ro bst-entry-volume?   uint32
  |     +--ro bst-entry-counts?   uint32
  |     +--ro filtering-pks
  |        +--ro if-filtering-pks* [ifname]
  |           +--ro ifname           if:interface-ref
  |           +--ro filtering-pks?   uint32
```

Figure 2: State data hierarchy.

As can be seen from Figures 1 and 2, the SAVI data model includes several generic components: SAVI instance, binding table, binding state table, interface attribute, and statistics. Section 5 describes these components in more detail.

4.1. System-Controlled and User-Controlled List Entries

The SAVI data model defines several lists in the schema tree, such as "binding-table".

In such a list, the server creates the required item as a so-called system-controlled entry in state data, i.e., inside the "binding-table" container.

Additional entries may be created in the configuration by a client, e.g., via the NETCONF protocol. These are so-called user-controlled entries. If the server accepts a configured user-controlled entry, then this entry also appears in the state data version of the list.

Corresponding entries in both versions of the list (in state data and configuration) have the same value of the list key.

A client may also provide supplemental configuration of system-controlled entries. To do so, the client creates a new entry in the configuration with the desired contents. In order to bind this entry to the corresponding entry in the state data list, the key of the configuration entry has to be set to the same value as the key of the state entry.

Deleting a user-controlled entry from the configuration list results in the removal of the corresponding entry in the state data list. In contrast, if a system-controlled entry is deleted from the configuration list, only the extra configuration specified in that entry is removed but the corresponding state data entry remains in the list.

5. Basic Building Blocks

This section presents the basic building blocks of the SAVI data model.

5.1. SAVI Instance

SAVI data model supports one or more IP address assignment method. Each SAVI method runs as a SAVI instance. Each SAVI instance has separate configuration and state data. The SAVI instance can be set to enable or disable and be configured with preference value. When multiple SAVI instance running in the same system, the binding entry with high preference will be used to filter packets.

5.2. Binding Table

Entries in binding table are used to filter packets. Each binding entry includes source IP address, mac address, interface name, lifetime, creation time, binding method. Entries will be inserted or deleted by SAVI instance. And an entry can also be inserted or deleted by client if it is a manual binding entry.

5.3. Binding State Table

There is a binding state table for each IP address assignment method. Each binding state entry includes source IP address, mac address, interface name, state, lifetime, and other parameters specific for the SAVI method. For different SAVI method, the state is different. e.g. for SAVI FCFS, the state includes NO_BIND, TENTATIVE, VALID, TESTING_VP, and TESTING_TP-LT, and for SAVI DHCP, the state includes NO_BIND, INIT_BIND, BOUND, DETECTION , RECOVERY, and VERIFY.

5.4. Interface Attribute

There is corresponding interface attribute for each SAVI method. Such as for SAVI FCFS, the interface attribute includes Validating Port and Trusted Port, for SAVI DHCP, the interface attributes includes Trust Attribute, DHCP-Trust Attribute, DHCP-Snooping Attribute, Data-Snooping Attribute, and Validating Attribute.

5.5. SAVI Statistics

The SAVI Statistics contains counters for the collection of statistics, including volume and count of binding table, count of packets dropped because of IP address validation.

6. Definition of ietf-savi module

```
<CODE BEGINS> file "ietf-savi@2017-08-15.yang"
module ietf-savi {
    namespace "urn:ietf:params:xml:ns:yang:ietf-savi";
    prefix savi;
    import ietf-yang-types {
        prefix yang;
    }
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }
    organization "IETF SAVI Working Group";
    contact
        "
        WG Web:  <http://datatracker.ietf.org/wg/savi/charter>
        Editor:   Changqing An
                  <mailto:acq@tsinghua.edu.cn>
               ";
    description
```

```
        "This YANG module defines essential components for the management
        of a savi subsystem.";


    revision 2017-08-15{
        description "Initial revision.";
        reference   "DRAFT XXX: A YANG Data Model for SAVI Management.";
    }

     /* Identities */

    identity binding-state {
        description "Base identity for the sates of binding entry.";
    }

     /* Groupings */

    grouping binding-entry {
        description "This grouping provides basic parameters of a binding
entry.";

        leaf ifname {
            type if:interface-ref;
            description "The name of the interface.";

        }
        leaf mac {
            type yang:mac-address;
            description "The binding source mac address.";
        }
        leaf lifetime {
            type yang:timeticks;
            mandatory true;
            description
            "The remaining lifetime of the entry.";
        }
    }


    grouping binding-table {
        description "This grouping defines binding table for both IPv4 and
IPv6.";
        container binding-table {
            description "Container for binding table.";

            container ipv4 {
                description "Container for binding table for IPv4 protocol.";
                list binding-entry {
                    key "ifname address";
```

```
                description "Definition of a binding entry";
```

```
                    leaf address {
                        type inet:ipv4-address;
                        description "IPv4 address of the binding host.";
                    }
                    uses binding-entry;
                    leaf creationtime {
                        type yang:timestamp;
                        mandatory true;
                        description "The value of the local clock when the
entry was firstly created.";
                    }
                    leaf binding-method {
                        type string;
                        mandatory true;
                        description "IP address assignment methods.";
                    }
                }
            }

            container ipv6 {
                description "Container for binding table for IPv4 protocol.";
                list binding-entry {
                    key "ifname address";
                    description "Definition of a binding entry";
                    leaf address {
                        type inet:ipv6-address;
                        description "IPv6 address of the binding host.";
                    }
                    uses binding-entry;
                    leaf creationtime {
                        type yang:timestamp;
                        mandatory true;
                        description "The value of the local clock when the
entry was firstly created.";
                    }
                    leaf binding-method {
                        type string;
                        mandatory true;
                        description "IP address assignment methods.";
                    }
                }
            }
        }
    }

     /* State data */

    container savi-state {
```

```
        config false;
        description "State data of the savi subsystem.";
```

```
        container savi-instances {
            description "Container of parameters for each savi method.";
            list savi-instance {
                key savi-method;
                description "A list of parameters for each savi method.";
                leaf savi-method {
                    type string;
                    description "IP address assignment methods.";
                }
                leaf preference {
                    type uint32;
                    description "Preference of the savi method.";
                }
            }
        }

        uses binding-table;
        container statistics {
            description "Container of statistics parameters for savi
subsystem.";
            leaf bst-entry-volume {
                type uint32;
                description "The volume of the the binding state table.";
            }
                leaf bst-entry-counts {
                type uint32;
                description "The count of the binding state table.";
                }
            container filtering-pks {
                description "Container of parameters for counting filtering
packets.";
                list if-filtering-pks {
                    key ifname;
                    description "A list of parameters for counting filtering
packets.";
                    leaf ifname {
                        type if:interface-ref;
                        description "The name of the interface.";
                    }
                    leaf filtering-pks {
                        type uint32;
                        description "The count of filtering packets.";
                    }
                }
            }
        }
    }
```

```
     /* Configuration Data */

    container savi {
```

```
        description "Configuration data of the savi subsystem.";
        container savi-instances {
            description "Container of parameters for each savi method.";
            list savi-instance {
                key savi-method;
                description "A list of parameters for each savi method.";
                leaf savi-method {
                    type string;
                    description "IP address assignment methods.";
                }
                leaf enable {
                    type boolean;
                    description "If the savi method is enabled?";
                }
                leaf preference {
                    type uint32;
                    description "Preference of the savi method.";
                }
            }
        }

        container if-filtering-attributes {
            description "Container for defining filtering attributes of each
interface, common for every savi instance.";
            list if-filtering-attribute {
                key ifname;
                description "A list of filtering attributes for each
interface.";
                leaf ifname {
                    type if:interface-ref;
                    description "The name of the interface.";
                }
                leaf filtering-enabled {
                    type boolean;
                    default true;
                    description "If the filtering attribute is enabled? ";
                }
            }
        }
                /* Binding table for manual entry which can be configured by
operators*/
        uses binding-table {
            when "/savi/savi-instances/savi-instance[savi-method = 'savi-
manual']/enable = 'true'";
        }
    } //container savi
}
```

<CODE ENDS>

7. Definition of ietf-savi-fcfs module

```
<CODE BEGINS> file "ietf-savi-fcfs@2017-08-15.yang"
module ietf-savi-fcfs {
    namespace "urn:ietf:params:xml:ns:yang:ietf-savi-fcfs";
    prefix savi-fcfs;
    import ietf-yang-types {
        prefix yang;
    }
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }

    import ietf-savi {
        prefix savi;
    }
    organization "IETF SAVI Working Group";
    contact
        "
        WG Web:  <http://datatracker.ietf.org/wg/savi/charter>
        Editor:   Changqing An
                  <mailto:acq@tsinghua.edu.cn>
        ";
    description
        "
        The Yang data module defined for SAVI FCFS.
        ";

    revision 2017-08-15 {
        description "Initial revision.";
        reference "DRAFT XXX: A YANG Data Model for SAVI Management";
    }

     /* Identities */

    identity savi-fcfs-state {
        base savi:binding-state;
        description "Base identity for the sates definition of SAVI FCFS.";
    }
    identity tentative {
        base savi-fcfs-state;
        description "A state defined in SAVI FCFS.";
    }
    identity valid {
```

```
        base savi-fcfs-state;
        description "A state defined in SAVI FCFS.";
    }
    identity testing_vp {
        base savi-fcfs-state;
        description "A state defined in SAVI FCFS.";
    }
    identity testing_vp-lt {
        base savi-fcfs-state;
        description "A state defined in SAVI FCFS.";
    }

     /* State data */

    augment "/savi:savi-state/savi:savi-instances/savi:savi-instance" {
        when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-
method = 'savi-fcfs'";
        description "Binding state table specific for SAVI FCFS.";
        container binding-state-table {
            description "Binding state table specific for SAVI FCFS.";
            list binding-state-entry {
                key "ifname address";
                description "A binding status entry specific for SAVI FCFS.";
                leaf address {
                    type inet:ipv6-address;
                    description "The binding source IP address.";
                }
                uses savi:binding-entry;
                leaf state {
                    type identityref {
                        base savi-fcfs-state;
                    }
                    description "State of the entry as defined in SAVI FCFS:
NO_BIND, TENTATIVE, VALID, TESTING_VP, TESTING_TP-LT";
                }
            }
        }
    }

     /* Configuration Data */

    augment "/savi:savi/savi:savi-instances/savi:savi-instance" {
        when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-
method = 'savi-fcfs'";
        description "Parameters specific to SAVI FCFS.";
        container params {
            description "Parameters specific to SAVI FCFS.";
            leaf tent_lt {
```

```
                type yang:timeticks;
                default 50;
                description "A default value defined in SAVI FCFS.";
```

```
                reference "TENT_LT from [RFC6620].";
            }

            leaf default_lt {
                type yang:timeticks;
                default 30000;
                description "A default value defined in SAVI FCFS.";
                reference "DEFAULT_LT from [RFC6620]";
            }

            leaf twait {
                type yang:timeticks;
                default 25;
                description "A default value defined in SAVI FCFS";
                reference "T_WAIT from [RFC6620].";
            }
            container if-attributes {
                description "Interface attributes specific to SAVI SEND.";
                list if-attribute {
                    key ifname;
                        description "A list of attributes for each interface.";
                    leaf ifname {
                        type if:interface-ref;
                        description "The name of the interface.";
                    }
                    leaf validating {
                        type boolean;
                        must .=not(../trust);
                        default true;
                        description "SAVI FCFS processing is performed in the
port.";
                    }

                    leaf trust {
                        type boolean;
                        must .=not(../validating);
                        default false;
                        description "SAVI FCFS processing is not performed in
the port.";
                    }
                } //list
            } //container
        } //container
    } //augment
}

<CODE ENDS>
```

8. Definition of ietf-savi-dhcpv4 module

```
<CODE BEGINS> file "ietf-savi-dhcpv4@2017-08-15.yang"
module ietf-savi-dhcpv4 {
    namespace "urn:ietf:params:xml:ns:yang:ietf-savi-dhcpv4";
    prefix savi-dhcpv4;
    import ietf-yang-types {
        prefix yang;
    }
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }
    import ietf-savi {
        prefix savi;
    }
    organization "IETF SAVI Working Group";
    contact
        "
        WG Web:  <http://datatracker.ietf.org/wg/savi/charter>
        Editor:   Changqing An
                  <mailto:acq@tsinghua.edu.cn>
                ";
    description
        "
        The Yang data module defined for SAVI DHCPv4.
        ";

    revision 2017-08-15 {
        description "Initial revision.";
        reference "DRAFT XXX: A YANG Data Model for SAVI Management";
    }

     /* Identities */

    identity savi-dhcp-state {
        base savi:binding-state;
        description "Base identity for the sates definition of SAVI DHCPv4.";

    }
    identity no_bind {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv4.";
    }
    identity init_bind {
        base savi-dhcp-state;
```

```
        description "A state defined in SAVI DHCPv4.";
    }
    identity bind {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv4.";
    }
    identity detection {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv4.";
    }
    identity recovery {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv4.";
    }
    identity verify {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv4.";
    }

     /* State data */

    augment "/savi:savi-state/savi:savi-instances/savi:savi-instance" {
    when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-method =
'savi-dhcpv4'";
        description "Binding state table specific for SAVI DHCPv4.";
        container binding-state-table {
            description "Binding state table specific for SAVI DHCPv4.";
            list binding-state-entry {
                key "ifname address";
                description "A binding state entry specific for SAVI DHCPv4.";
                leaf address {
                    type inet:ipv4-address;
                    description "The binding source IP address.";
                }
                uses savi:binding-entry;
                leaf state {
                    type identityref {
                        base savi-dhcp-state;
                    }
                    description "State of the entry as defined in SAVI DHCP:
NO_BIND, INIT_BIND, BOUND, DETECTION , RECOVERY, VERIFY.";
                }
                leaf tid {
                    type uint32;
                    description "The Transaction ID of the corresponding DHCP
transaction.";
                }
                leaf timeouts {
```

```
                    when "/savi:savi/savi:savi-instances/savi:savi-instance/
params/if-attributes/if-attribute/data-snooping = 'true'";
                    type uint32;
                    description "the number of timeouts that expired in the
current state";
```

```
                }
            }
        }
    }

     /* Configuration Data */

    augment "/savi:savi/savi:savi-instances/savi:savi-instance" {
        when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-
method = 'savi-dhcpv4'";
        description "Parameters specific to SAVI DHCPv4";
        container params {
            description "Parameters specific to SAVI DHCPv4";
            leaf max-dhcp-responsetime {
                type yang:timeticks;
                default 12000;
                description "Maximum Solicit timeout value. Default is 120s.";
                reference "SOL_MAX_RT from [RFC3315]";
            }
            leaf max-leasequery-delay {
                type yang:timeticks;
                default 1000;
                description "Maximum LEASEQUERY timeout value. Default is
10s.";
                reference "LQ_MAX_RT from [RFC5007]";
            }
            leaf datasnooping-interval {
                type yang:timeticks;
                default 6000;
                description
                "Minimum interval between two successive EVE_DATA_UNMATCH
                events triggered by an attachment. Recommended interval:
                60s and configurable.";
                reference "DATA_SNOOPING_INTERVAL from [RFC7513]";
            }
            leaf offlink-delay {
                type yang:timeticks;
                default 3000;
                description
                "Period after a client is last detected before the binding
                anchor is being removed.  Recommended delay: 30s.";
                reference "OFFLINK_DELAY from [RFC7513].";
            }
            leaf detection-timeout {
                type yang:timeticks;
                default 50;
                description
                "Maximum duration of a hardware address verification step
```

```
                in the VERIFY state.";
                reference "DETECTION_TIMEOUT from [RFC7513]";
```

```
            }
            container if-attributes {
                description "Interface attributes specific to SAVI DHCPv4.";
                list if-attribute {
                    key ifname;
                        description "A list of attributes for each interface.";
                    leaf ifname {
                        type if:interface-ref;
                        description "The name of the interface.";
                    }
                    leaf trust-attribute {
                        type boolean;
                        default false;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf dhcp-trust {
                        type boolean;
                        default false;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf dhcp-snooping {
                        type boolean;
                        default true;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf data-snooping {
                        type boolean;
                        default false;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf validating {
                        type boolean;
                        default true;
                        description "An attribute defined in SAVI DHCP.";
                    }
                } //list
            } //container
        } //container
    } //augment
}

<CODE ENDS>
```

9. Definition of ietf-savi-dhcpv6 module

```
<CODE BEGINS> file "ietf-savi-dhcpv6@2017-08-15.yang"
module ietf-savi-dhcpv6 {
    namespace "urn:ietf:params:xml:ns:yang:ietf-savi-dhcpv6";
    prefix savi-dhcpv6;
    import ietf-yang-types {
        prefix yang;
    }
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }

    import ietf-savi {
        prefix savi;
    }
    organization "IETF SAVI Working Group";
    contact
        "
        WG Web:  <http://datatracker.ietf.org/wg/savi/charter>
        Editor:   Changqing An
                  <mailto:acq@tsinghua.edu.cn>
        ";
    description
        "
        The Yang data module defined for SAVI DHCPv6.
        ";

    revision 2017-08-15 {
        description "Initial revision.";
        reference "DRAFT XXX: A YANG Data Model for SAVI Management";
    }

     /* Identities */

    identity savi-dhcp-state {
        base savi:binding-state;
        description "Base identity for the sates definition of SAVI DHCPv6.";
    }
    identity no_bind {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv6.";
    }
    identity init_bind {
        base savi-dhcp-state;
```

```
        description "A state defined in SAVI DHCPv6.";
    }
    identity bind {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv6.";
    }
    identity detection {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv6.";
    }
    identity recovery {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv6.";
    }
    identity verify {
        base savi-dhcp-state;
        description "A state defined in SAVI DHCPv6.";
    }


     /* State data */

    augment "/savi:savi-state/savi:savi-instances/savi:savi-instance" {
    when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-method =
'savi-dhcpv6'";
        description "Binding state table specific for SAVI DHCPv6.";
        container binding-state-table {
            description "Binding state table specific for SAVI DHCPv6.";
            list binding-state-entry {
                key "ifname address";
                description "A binding state entry specific for SAVI DHCPv6.";
                leaf address {
                    type inet:ipv6-address;
                    description "The binding source IP address.";
                }
                uses savi:binding-entry;
                leaf state {
                    type identityref {
                        base savi-dhcp-state;
                    }
                    description "State of the entry as defined in SAVI DHCP:
NO_BIND, INIT_BIND, BOUND, DETECTION , RECOVERY, VERIFY.";
                }
                leaf tid {
                    type uint32;
                    description "The Transaction ID of the corresponding DHCP
transaction.";
                }
```

```
                leaf timeouts {
                    when "/savi:savi/savi:savi-instances/savi:savi-instance/
params/if-attributes/if-attribute/data-snooping = 'true'";
                    type uint32;
```

```
                    description "The number of timeouts that expired in the
current state.";
                }
            }
        }
    }

     /* Configuration Data */

    augment "/savi:savi/savi:savi-instances/savi:savi-instance" {
        when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-
method = 'savi-dhcpv6'";
        description "Parameters specific to SAVI DHCPv6";
        container params {
            description "Parameters specific to SAVI DHCPv6";
            leaf max-dhcp-responsetime {
                type yang:timeticks;
                default 12000;
                description "Maximum Solicit timeout value. Default is 120s.";
                reference "SOL_MAX_RT from [RFC3315]";
            }
            leaf max-leasequery-delay {
                type yang:timeticks;
                default 1000;
                description "Maximum LEASEQUERY timeout value. Default is
10s.";
                reference "LQ_MAX_RT from [RFC5007]";
            }
            leaf datasnooping-interval {
                type yang:timeticks;
                default 6000;
                description
                "Minimum interval between two successive EVE_DATA_UNMATCH
                events triggered by an attachment. Recommended interval:
                60s and configurable.";
                reference "DATA_SNOOPING_INTERVAL from [RFC7513]";
            }
            leaf offlink-delay {
                type yang:timeticks;
                default 3000;
                description
                "Period after a client is last detected before the binding
                anchor is being removed.  Recommended delay: 30s.";
                reference "OFFLINK_DELAY from [RFC7513].";
            }
            leaf detection-timeout {
                type yang:timeticks;
                default 50;
```

```
            description
             "Maximum duration of a hardware address verification step
             in the VERIFY state.";
```

```
                reference "DETECTION_TIMEOUT from [RFC7513]";
            }
            container if-attributes {
                description "Interface attributes specific to SAVI DHCPv6.";
                list if-attribute {
                    key ifname;
                        description "A list of attributes for each interface.";
                    leaf ifname {
                        type if:interface-ref;
                        description "The name of the interface.";
                    }
                    leaf trust-attribute {
                        type boolean;
                        default false;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf dhcp-trust {
                        type boolean;
                        default false;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf dhcp-snooping {
                        type boolean;
                        default true;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf data-snooping {
                        type boolean;
                        default false;
                        description "An attribute defined in SAVI DHCP.";
                    }
                    leaf validating {
                        type boolean;
                        default true;
                        description "An attribute defined in SAVI DHCP.";
                    }
                } //list
            } //container
        } //container
    } //augment
}

<CODE ENDS>
```

10. Definition of ietf-savi-send module

```
<CODE BEGINS> file "ietf-savi-send@2017-08-15.yang"
module ietf-savi-send {
    namespace "urn:ietf:params:xml:ns:yang:ietf-savi-send";
    prefix savi-send;
    import ietf-yang-types {
        prefix yang;
    }
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }
    import ietf-savi {
        prefix savi;
    }
    organization "IETF SAVI Working Group";
    contact
        "
        WG Web:  <http://datatracker.ietf.org/wg/savi/charter>
        Editor:   Changqing An
                  <mailto:acq@tsinghua.edu.cn>
        ";
    description
        "
        The Yang data module defined for SAVI SEND.
        ";

    revision 2017-08-15 {
        description "Initial revision.";
        reference "DRAFT XXX: A YANG Data Model for SAVI Management";
    }

     /* Identities */

    identity savi-send-state {
        base savi:binding-state;
        description "Base identity for the sates definition of SAVI SEND.";
    }
    identity tentative-dad {
        base savi-send-state;
        description "A state defined in SAVI SEND.";
    }
    identity tentative-nud {
        base savi-send-state;
```

```
        description "A state defined in SAVI SEND.";
    }
    identity valid {
        base savi-send-state;
        description "A state defined in SAVI SEND.";
    }

    identity testing_vp {
        base savi-send-state;
        description "A state defined in SAVI SEND.";
    }
    identity testing_vp_1 {
        base savi-send-state;
        description "A state defined in SAVI SEND.";
    }

     /* State data */

    augment "/savi:savi-state/savi:savi-instances/savi:savi-instance" {
        when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-
method = 'savi-send'";
        description "Binding state table specific for SAVI SEND.";
        container binding-state-table {
            description "Binding state table specific for SAVI SEND.";
            list binding-state-entry {
                key "ifname address";
                description "A binding state entry specific for SAVI SEND.";
                leaf address {
                    type inet:ipv6-address;
                    description "The binding source IP address.";
                }
                uses savi:binding-entry;
                leaf alternative-if {
                    type if:interface-ref;
                    description "Alternative interface is a parameter defined
in SAVI SEND.";
                }

                leaf state {
                    type identityref {
                        base savi-send-state;
                    }
                    description "State of the entry as defined in SAVI SEND:
TENTATIVE_DAD, TENTATIVE_NUD, VALID, TESTING_VP, TESTING_VP'";
                }
            }
        }
    }
```

```
     /* Configuration Data */
```

```
    augment "/savi:savi/savi:savi-instances/savi:savi-instance" {
        when "/savi:savi/savi:savi-instances/savi:savi-instance/savi:savi-
method = 'savi-send'";
        description "Parameters specific to SAVI SEND.";
        container params {
            description "Parameters specific to SAVI SEND.";
            leaf tent_lt {
                type yang:timeticks;
                default 50;
                description "A default value defined in SAVI SEND.";
                reference "TENT_LT from [RFC7219].";
            }

            leaf default_lt {
                type yang:timeticks;
                default 30000;
                description "A default value defined in SAVI SEND.";
                reference "DEFAULT_LT from [RFC7219]";
            }
            container if-attributes {
                description "Interface attributes specific to SAVI SEND.";
                list if-attribute {
                    key ifname;
                        description "A list of attributes for each interface.";
                    leaf ifname {
                        type if:interface-ref;
                        description "The name of the interface.";
                    }
                    leaf validating {
                        type boolean;
                        must .=not(../trust);
                        default true;
                        description "SAVI SEND processing is performed in the
port.";
                    }

                    leaf trust {
                        type boolean;
                        must .=not(../validating);
                        default false;
                        description "SAVI SEND processing is not performed in
the port.";
                    }
                } //list
            } //container
        } //container
    } //augment
}
```

<CODE ENDS>

11. Security Considerations

Configuration and state data conforming to the SAVI yang data model (defined in this document) are 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 the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

A number of data nodes defined in the YANG modules belonging to the configuration part of the SAVI data model are writable/creatable/deletable (i.e., "config true" in YANG terms, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations to these data nodes, such as "edit-config", can have negative effects on the network if the protocol operations are not properly protected.

12. IANA Considerations

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:ietf-savi
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-savi-fcfs
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-savi-dhcpv4
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-savi-dhcpv6
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-savi-send
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

This document registers the following YANG modules in the YANG Module Names registry [RFC6020]:

```
name:         ietf-savi
namespace:    urn:ietf:params:xml:ns:yang:ietf-savi
prefix:       savi
reference:    RFC XXXX

name:         ietf-savi-fcfs
namespace:    urn:ietf:params:xml:ns:yang:ietf-savi-fcfs
prefix:       savi-fcfs
reference:    RFC XXXX

name:         ietf-savi-dhcpv4
namespace:    urn:ietf:params:xml:ns:yang:ietf-savi-dhcpv4
prefix:       savi-dhcpv4
reference:    RFC XXXX

name:         ietf-savi-dhcpv6
namespace:    urn:ietf:params:xml:ns:yang:ietf-savi-dhcpv6
prefix:       savi-dhcpv6
reference:    RFC XXXX

name:         ietf-savi-send
namespace:    urn:ietf:params:xml:ns:yang:ietf-savi-send
prefix:       savi-send
reference:    RFC XXXX
```

13. Contributors

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6620] Nordmark, E., Bagnulo, M., and E. Levy-Abegnoli, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", RFC 6620, DOI 10.17487/RFC6620, May 2012, <http://www.rfc-editor.org/info/rfc6620>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <http://www.rfc-editor.org/info/rfc7039>.

[RFC7219] Bagnulo, M. and A. Garcia-Martinez, "SEcure Neighbor Discovery (SEND) Source Address Validation Improvement (SAVI)", RFC 7219, DOI 10.17487/RFC7219, May 2014, <http://www.rfc-editor.org/info/rfc7219>.

[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>.

[RFC7513] Bi, J., Wu, J., Yao, G., and F. Baker, "Source Address Validation Improvement (SAVI) Solution for DHCP", RFC 7513, DOI 10.17487/RFC7513, May 2015, <http://www.rfc-editor.org/info/rfc7513>.

14.2. Informative References

[RFC2223] Postel, J. and J. Reynolds, "Instructions to RFC Authors", RFC 2223, DOI 10.17487/RFC2223, October 1997, <http://www.rfc-editor.org/info/rfc2223>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <http://www.rfc-editor.org/info/rfc2863>.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, DOI 10.17487/RFC3410, December 2002, <http://www.rfc-editor.org/info/rfc3410>.

[RFC4181] Heard, C., Ed., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, DOI 10.17487/RFC4181, September 2005, <http://www.rfc-editor.org/info/rfc4181>.

[RFC4293] Routhier, S., Ed., "Management Information Base for the Internet Protocol (IP)", RFC 4293, DOI 10.17487/RFC4293, April 2006, <http://www.rfc-editor.org/info/rfc4293>.

14.3. URL References

[idguidelines] IETF Internet Drafts editor, "http://www.ietf.org/ietf/1id-guidelines.txt".

[idnits] IETF Internet Drafts editor, "http://www.ietf.org/ID-Checklist.html".

[ietf] IETF Tools Team, "http://tools.ietf.org".

[ops] the IETF OPS Area, "http://www.ops.ietf.org".

[xml2rfc] XML2RFC tools and documentation, "http://xml.resource.org".

Appendix A. The Complete Data Trees

This appendix presents the complete configuration and state data trees of the SAVI data model. See Section 2.2 for an explanation of the symbols used. Data type of every leaf node is shown near the right end of the corresponding line.

```
module: ietf-savi
   +--ro savi-state
   |  +--ro savi-instances
   |  |  +--ro savi-instance* [savi-method]
   |  |     +--ro savi-method                          string
   |  |     +--ro preference?                          uint32
   |  |     +--ro savi-fcfs:binding-state-table
   |  |     |  +--ro savi-fcfs:binding-state-entry* [ifname address]
   |  |     |     +--ro savi-fcfs:address     inet:ipv6-address
   |  |     |     +--ro savi-fcfs:ifname      if:interface-ref
   |  |     |     +--ro savi-fcfs:mac?        yang:mac-address
   |  |     |     +--ro savi-fcfs:lifetime    yang:timeticks
   |  |     |     +--ro savi-fcfs:state?      identityref
   |  |     +--ro savi-dhcpv4:binding-state-table
   |  |     |  +--ro savi-dhcpv4:binding-state-entry* [ifname address]
   |  |     |     +--ro savi-dhcpv4:address     inet:ipv4-address
   |  |     |     +--ro savi-dhcpv4:ifname      if:interface-ref
   |  |     |     +--ro savi-dhcpv4:mac?        yang:mac-address
   |  |     |     +--ro savi-dhcpv4:lifetime    yang:timeticks
   |  |     |     +--ro savi-dhcpv4:state?      identityref
   |  |     |     +--ro savi-dhcpv4:tid?        uint32
   |  |     |     +--ro savi-dhcpv4:timeouts?   uint32
   |  |     +--ro savi-dhcpv6:binding-state-table
   |  |     |  +--ro savi-dhcpv6:binding-state-entry* [ifname address]
   |  |     |     +--ro savi-dhcpv6:address     inet:ipv6-address
   |  |     |     +--ro savi-dhcpv6:ifname      if:interface-ref
   |  |     |     +--ro savi-dhcpv6:mac?        yang:mac-address
   |  |     |     +--ro savi-dhcpv6:lifetime    yang:timeticks
   |  |     |     +--ro savi-dhcpv6:state?      identityref
   |  |     |     +--ro savi-dhcpv6:tid?        uint32
   |  |     |     +--ro savi-dhcpv6:timeouts?   uint32
   |  |     +--ro savi-send:binding-state-table
   |  |        +--ro savi-send:binding-state-entry* [ifname address]
   |  |           +--ro savi-send:address           inet:ipv6-address
   |  |           +--ro savi-send:ifname            if:interface-ref
   |  |           +--ro savi-send:mac?              yang:mac-address
   |  |           +--ro savi-send:lifetime          yang:timeticks
   |  |           +--ro savi-send:alternative-if?   if:interface-ref
   |  |           +--ro savi-send:state?            identityref
   |  +--ro binding-table
```

```
   |  |  +--ro ipv4
   |  |  |  +--ro binding-entry* [ifname address]
   |  |  |     +--ro address           inet:ipv4-address
   |  |  |     +--ro ifname            if:interface-ref
   |  |  |     +--ro mac?              yang:mac-address
   |  |  |     +--ro lifetime          yang:timeticks
   |  |  |     +--ro creationtime      yang:timestamp
   |  |  |     +--ro binding-method    string
   |  |  +--ro ipv6
   |  |     +--ro binding-entry* [ifname address]
   |  |        +--ro address           inet:ipv6-address
   |  |        +--ro ifname            if:interface-ref
   |  |        +--ro mac?              yang:mac-address
   |  |        +--ro lifetime          yang:timeticks
   |  |        +--ro creationtime      yang:timestamp
   |  |        +--ro binding-method    string
   |  +--ro statistics
   |     +--ro bst-entry-volume?   uint32
   |     +--ro bst-entry-counts?   uint32
   |     +--ro filtering-pks
   |        +--ro if-filtering-pks* [ifname]
   |           +--ro ifname           if:interface-ref
   |           +--ro filtering-pks?   uint32
   +--rw savi
      +--rw savi-instances
      |  +--rw savi-instance* [savi-method]
      |     +--rw savi-method           string
      |     +--rw enable?               boolean
      |     +--rw preference?           uint32
      |     +--rw savi-fcfs:params
      |     |  +--rw savi-fcfs:tent_lt?         yang:timeticks
      |     |  +--rw savi-fcfs:default_lt?      yang:timeticks
      |     |  +--rw savi-fcfs:twait?           yang:timeticks
      |     |  +--rw savi-fcfs:if-attributes
      |     |     +--rw savi-fcfs:if-attribute* [ifname]
      |     |        +--rw savi-fcfs:ifname        if:interface-ref
      |     |        +--rw savi-fcfs:validating?   boolean
      |     |        +--rw savi-fcfs:trust?        boolean
      |     +--rw savi-dhcpv4:params
      |     |  +--rw savi-dhcpv4:max-dhcp-responsetime?   yang:timeticks
      |     |  +--rw savi-dhcpv4:max-leasequery-delay?    yang:timeticks
      |     |  +--rw savi-dhcpv4:datasnooping-interval?   yang:timeticks
      |     |  +--rw savi-dhcpv4:offlink-delay?           yang:timeticks
      |     |  +--rw savi-dhcpv4:detection-timeout?       yang:timeticks
      |     |  +--rw savi-dhcpv4:if-attributes
      |     |     +--rw savi-dhcpv4:if-attribute* [ifname]
      |     |        +--rw savi-dhcpv4:ifname             if:interface-ref
      |     |        +--rw savi-dhcpv4:trust-attribute?   boolean
```

```
|     |        +--rw savi-dhcpv4:dhcp-trust?        boolean
|     |        +--rw savi-dhcpv4:dhcp-snooping?     boolean
|     |        +--rw savi-dhcpv4:data-snooping?     boolean
|     |        +--rw savi-dhcpv4:validating?        boolean
|     +--rw savi-dhcpv6:params
|     |  +--rw savi-dhcpv6:max-dhcp-responsetime?   yang:timeticks
|     |  +--rw savi-dhcpv6:max-leasequery-delay?    yang:timeticks
|     |  +--rw savi-dhcpv6:datasnooping-interval?   yang:timeticks
|     |  +--rw savi-dhcpv6:offlink-delay?           yang:timeticks
|     |  +--rw savi-dhcpv6:detection-timeout?       yang:timeticks
|     |  +--rw savi-dhcpv6:if-attributes
|     |     +--rw savi-dhcpv6:if-attribute* [ifname]
|     |        +--rw savi-dhcpv6:ifname             if:interface-ref
|     |        +--rw savi-dhcpv6:trust-attribute?   boolean
|     |        +--rw savi-dhcpv6:dhcp-trust?        boolean
|     |        +--rw savi-dhcpv6:dhcp-snooping?     boolean
|     |        +--rw savi-dhcpv6:data-snooping?     boolean
|     |        +--rw savi-dhcpv6:validating?        boolean
|     +--rw savi-send:params
|        +--rw savi-send:tent_lt?         yang:timeticks
|        +--rw savi-send:default_lt?      yang:timeticks
|        +--rw savi-send:if-attributes
|           +--rw savi-send:if-attribute* [ifname]
|              +--rw savi-send:ifname        if:interface-ref
|              +--rw savi-send:validating?   boolean
|              +--rw savi-send:trust?        boolean
+--rw if-filtering-attributes
|  +--rw if-filtering-attribute* [ifname]
|     +--rw ifname               if:interface-ref
|     +--rw filtering-enabled?   boolean
+--rw binding-table
   +--rw ipv4
   |  +--rw binding-entry* [ifname address]
   |     +--rw address           inet:ipv4-address
   |     +--rw ifname            if:interface-ref
   |     +--rw mac?              yang:mac-address
   |     +--rw lifetime          yang:timeticks
   |     +--rw creationtime      yang:timestamp
   |     +--rw binding-method    string
   +--rw ipv6
      +--rw binding-entry* [ifname address]
         +--rw address           inet:ipv6-address
         +--rw ifname            if:interface-ref
         +--rw mac?              yang:mac-address
         +--rw lifetime          yang:timeticks
         +--rw creationtime      yang:timestamp
         +--rw binding-method    string
```

Appendix B. Change Log

Authors' Addresses

Changqing An
CERNET
Network Research Center, Tsinghua University
Beijing 100084
China

Phone: +86 10 62603113
EMail: acq@tsinghua.edu.cn

Jiahai Yang
CERNET
Network Research Center, Tsinghua University
Beijing 100084
China

Phone: +86 10 62783492
EMail: yang@cernet.edu.cn

Jianping Wu
CERNET
Network Research Center, Tsinghua University
Beijing 100084
China

EMail: jianping@cernet.edu.cn

Jun Bi
CERNET
Network Research Center, Tsinghua University
Beijing 100084
China

EMail: junbi@cernet.edu.cn