I2NSF
Internet Draft
Intended status: Informational

Y. Xie
L. Xia
J. Wu
Huawei

Expires: October 29 2015 April 29, 2015

Interface to Network Security Functions Demo Outline Design
draft-xie-i2nsf-demo-outline-design-00.txt

Abstract

This document describes the outline design of an Interface to network Security Functions (I2NSF) demo, aim to enhance understanding of the I2NSF architecture and justify its feasibility. The I2NSF demo enables the interaction between I2NSF client, I2NSF controller and NSF/vNSF by using NETCONF protocol and YANG model. The advantage of it is to ensure that such demo outline design will be able to share and reuse consensually designed functions, thereby increasing interoperability.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on October 29, 2015.

`Copyright Notice

Table of Contents

1. Introduction

A standard interface to express, monitor, and manage security policies for physical and virtual distributed security functions that may be running on different premises is increasingly demanded by Enterprises, residential, and mobile customers. The Interface to Network Security Functions (I2NSF) whose ultimate goal is to establish how to communicate with vNSF and how to get performance data or report out of vNSF well satisfies the demands.

Derived from [I-D.dunbar-i2nsf-problem-statement], it should have two types of I2NSF interface to consider: first, interface between I2NSF user/client and network controller (Service-oriented API), second, north-bound interface (Functional-oriented API) provided by the network security functions (NSFs) [I-D.xia-i2nsf-capability-interface-im].

This draft is focused on the outline design of an I2NSF demo including the design of I2NSF client, I2NSF controller and NSF/vNSF. NETCONF protocol and YANG model are used for the I2NSF demo realization. The demo aims to enhance understanding of the I2NSF architecture and justify its feasibility. Section 3 describes outline design of I2NSF demo. Section 4 describes protocol between Client, Controller, and NSF/vNSF. Section 5 gives an example definition of the Service-oriented API and Functional-oriented API by YANG model.

2. Conventions used in this document

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].

2.1. Terminology

I2NSF - Interface to Network Security Functions

NETCONF - Network Configuration Protocol

YANG - a data modeling language for the NETCONF network configuration protocol

RPC - Remote Procedure Control

SSH - Secure Shell

NSF - Network Security Function

vNSF - virtual Network Security Function

DB - Database

FW - Firewall

Shorewall - It is an open source firewall tool for Linux that builds upon the Netfilter (iptables / ipchains) system built into the Linux kernel

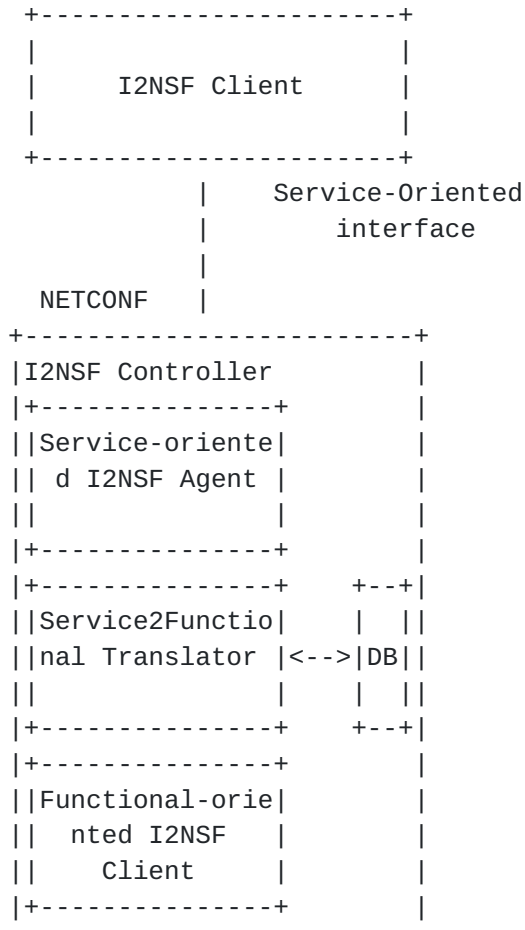
UFW - Uncomplicated Firewall

VM - Virtual Machine

3. Design of the I2NSF Demo

This section describes the design of the I2NSF demo. It first provides an architectural overview of the demo. Then, specific design of Client, Controller and NSF/vNSF is presented.

3.1. Overall architecture description



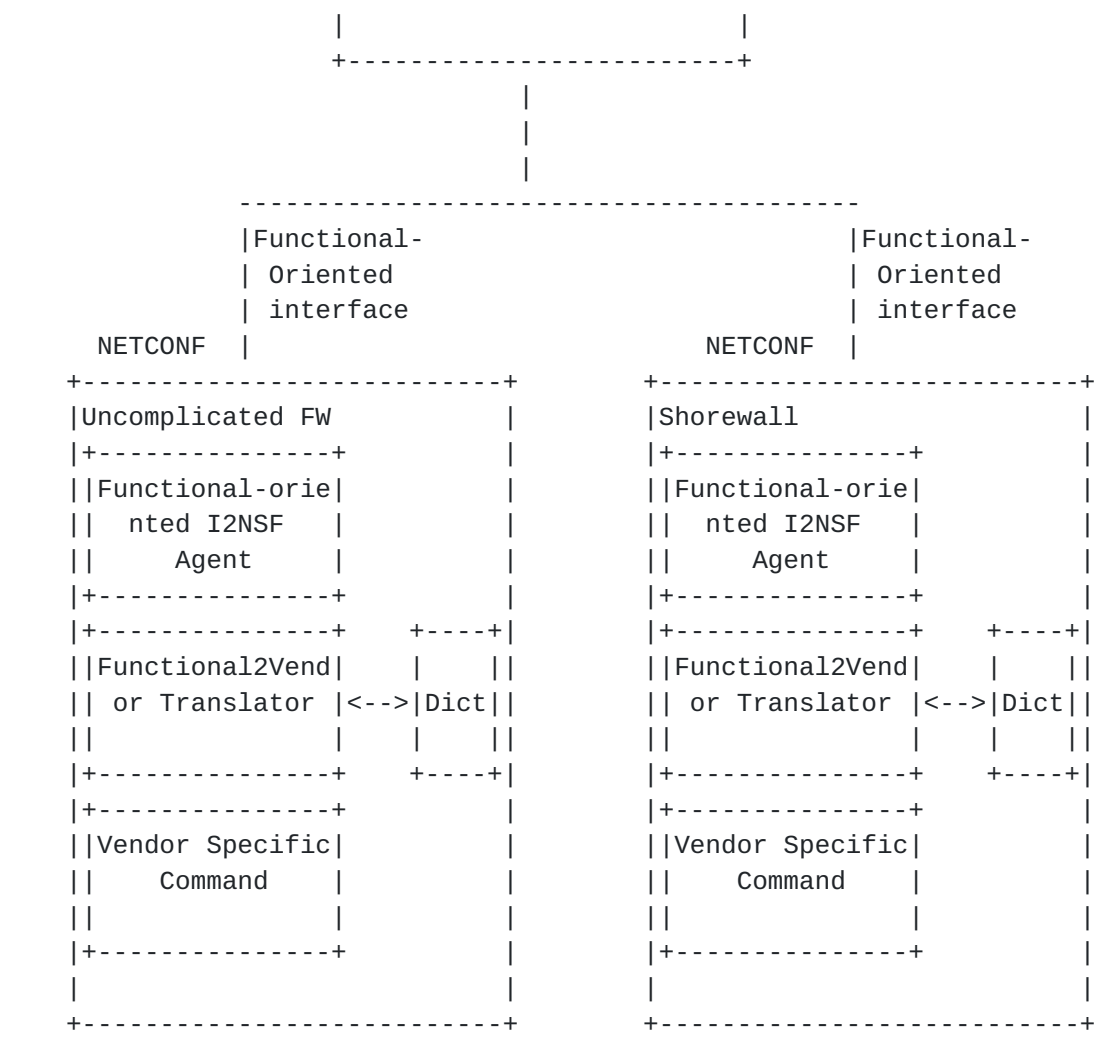


Figure 1. Overall Architecture

The I2NSF demo architecture and its basic functions are as follows:

1) Input configuration policy in the command line module of I2NSF Client;

2) I2NSF Service API parameters are composed according to YANG model;

3) Represent the composed I2NSF Service API parameters by XML;

4) Reliable and ordered data transmission supported by NETCONF protocol;

5) Receive data in NETCONF module of I2NSF Controller;

6) Transform Service API into Functional API in adapter module of I2NSF Controller;

7) Construct YANG model with Functional API parameters, and transmit the data by NETCONF protocol;

8) Receive data in NETCONF module of NSF/vNSF;

9) Fetch I2NSF Functional API parameter values in YANG model deposed module of NSF/vNSF;

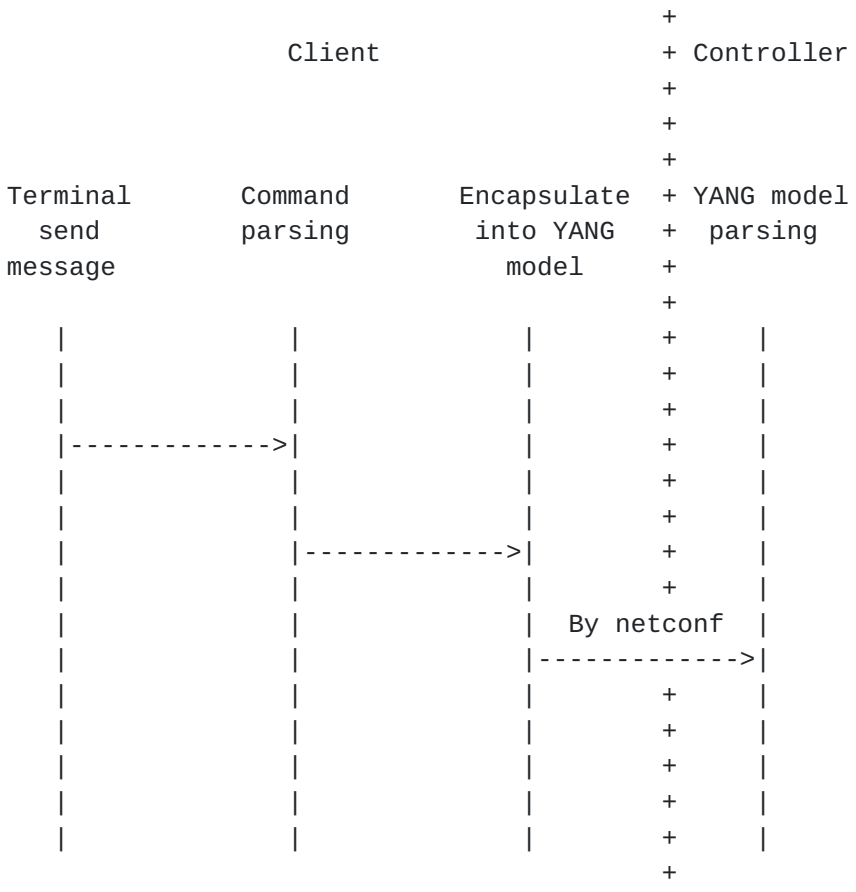
10) Match API KEY with different vendor FW KEY in KEY mapping module (aka Dict) of NSF/vNSF;

11) IP address mapping and MAC address mapping in Value mapping module of NSF/vNSF;

12) Different vendor FW parameters are assembled into command line form in command assembling module of NSF/vNSF;

13) Send Vendor specific command in sequence in command send module of NSF/vNSF;

3.1.1. I2NSF Workflow



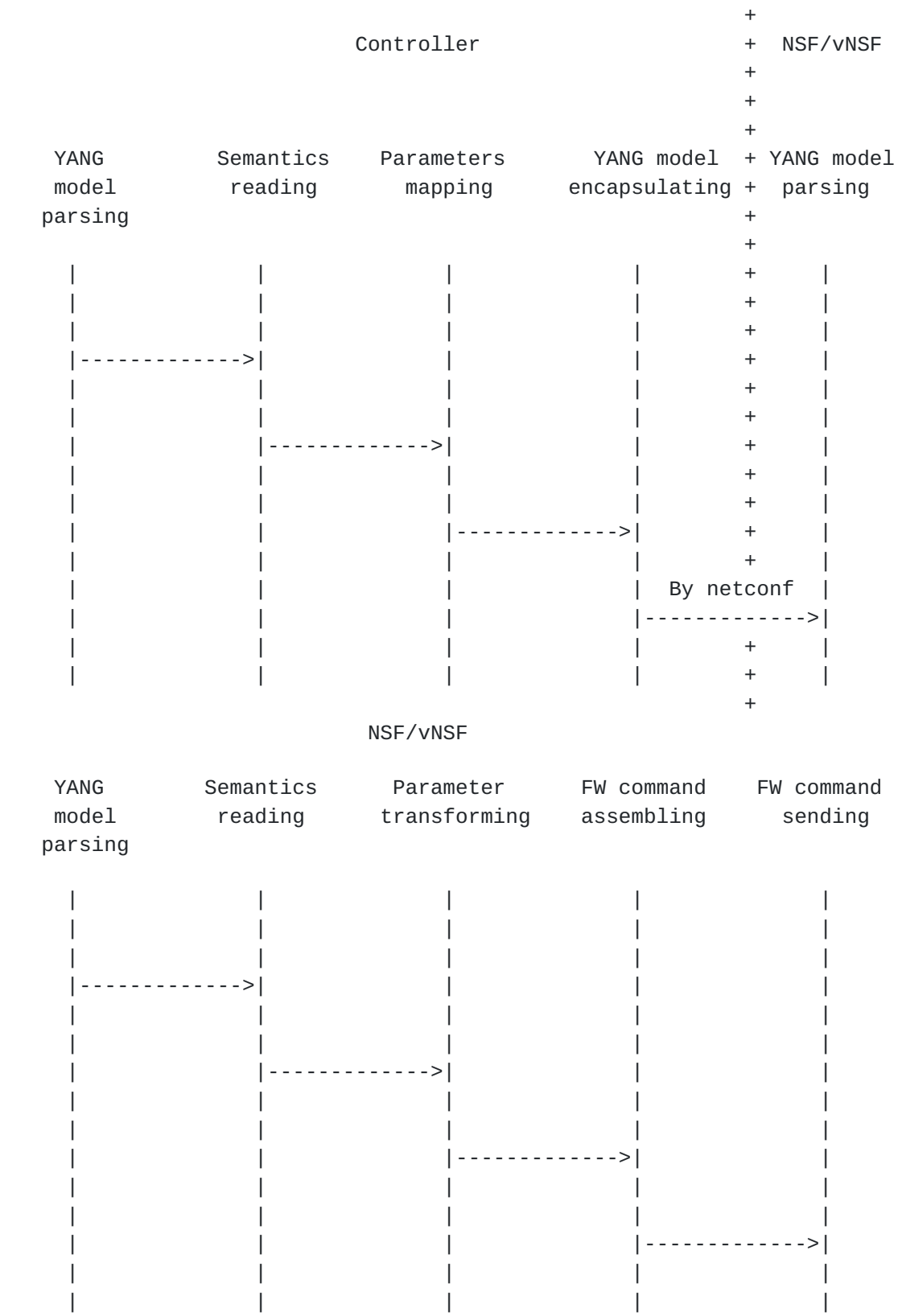


Figure 2. I2NSF Workflow

3.2. I2NSF Client

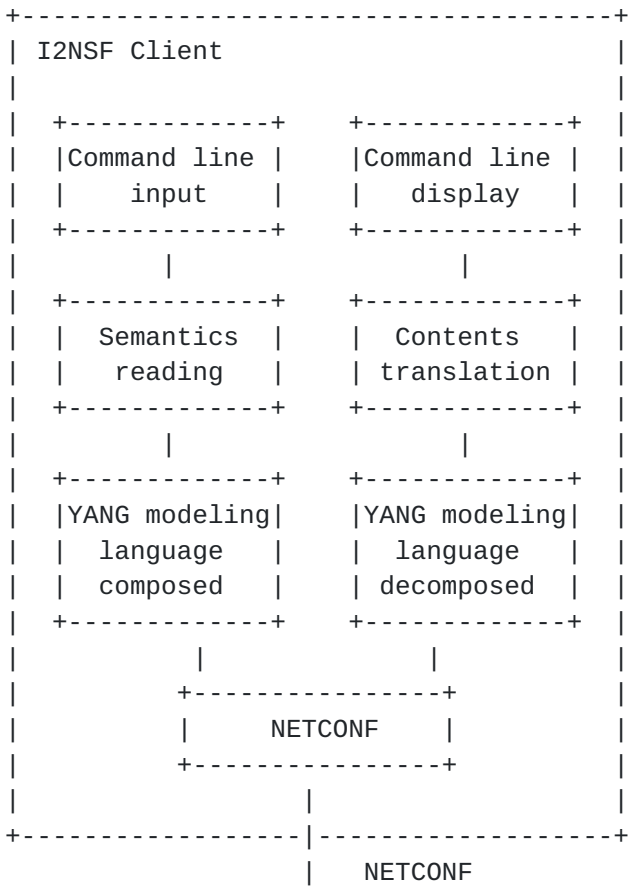


Figure 3. I2NSF Client Modules

I2NSF Client consists of seven modules:

1) Command line input module;

2) Semantics reading module;

3) YANG modeling language composed module;

4) YANG modeling language decomposed module;

5) Contents translation module;

6) Command line display module;

7) NETCONF module;

3.2.1. I2NSF Service API Format

A policy rule can be described as:

<tenant> <user_group> <role> <object> <action> <condition>.

<tenant> is used to uniquely identify tenant that has parameters such as tenant_name and tenant_ID. The default value is NULL

<user_group> has parameters such as Device(s), Place(s), User(s).

<role> is the role of user.

<object> are selected traffics that network behaviors enforced on.

<action> are network behaviors enforced on selected traffic.

<condition> means condition for action, with parameters such as Rate, Time, Timeout, Throughput, State.

Therein,

Example 1: John_0001 is allowed to access rtsp://video-server/mrbean.ts.

<tenant> = NULL

<user_group> = John_0001

<role> = marketing

<object> = rtsp://video-server/mrbean.ts

<action> = allow

<condition> = NULL

3.2.2. I2NSF Client Sub-module Design

1) Command line input module is responsible for receiving input command.

Input: John_0001 is allowed to access rtsp://video-server/mrbean.ts

2) Semantics reading module is responsible for analyzing commands and fetching I2NSF Service API parameter values.

Output:

```
        <tenant>                 = NULL

        user_group.user.user_name= John_0001

        role                     = marketing

        object.url_group.url     = rtsp://video-server/mrbean.ts

        action                   = allow

        condition                = NULL
```

3) YANG modeling language composed module is responsible for encapsulating I2NSF Service API parameter types and values into YANG model.

Input:

```
        <tenant>                 = NULL

        user_group.user.user_name= John_0001

        role                     = marketing

        object.url_group.url     = rtsp://video-server/mrbean.ts

        action                   = allow

        condition                = NULL
```

Output:

```
<tenant></tenant>
<user_group>
<user>
  <user_name>John_0001</user_name>
</user>
</user_group>
<role>marketing</role>
<object>
<url_group>
```

```
  <url>rtsp://video-server/mrbean.ts <http://video-server/MrBean.ts></url>
</url_group>
</object>
<action>allow</action>
<condition></condition>
......
```

4) YANG modeling language decomposed module is responsible for parsing YANG model and fetching I2NSF Service API parameter types and values.

5) Contents translation module is responsible for translating I2NSF Service API parameter types and values into readable contents to display.

6) Command line display module is responsible for displaying data information from I2NSF Controller.

7) NETCONF module is responsible for communicating between Client and Controller.

3.3. I2NSF Controller

I2NSF Controller consists of Service-oriented I2NSF Agent, Service2Functional Translator, Functional-oriented I2NSF Client.

3.3.1. Service-oriented I2NSF Agent

Service-oriented I2NSF Agent consists of NETCONF module, YANG model decomposed and composed modules.

NETCONF module is responsible for communicating with I2NSF Client.

YANG modeling language decomposed module parse YANG model into <key, value> pairs. It can map YANG model into Service2Functional KEY.

Input:

```
<tenant></tenant>
<user_group>
<user>
  <user_name>John_0001</user_name>
</user>
```

```
</user_group>
<role>marketing</role>
<object>
<url_group>
  <url>rtsp://video-server/mrbean.ts <http://video-server/MrBean.ts></url>
</url_group>
</object>
<action>allow</action>
<condition></condition>
.......
```

Output:

```
        <tenant>                 = NULL

        user_group.user.user_name= John_0001

        role                     = marketing

        object.url_group.url     = rtsp://video-server/mrbean.ts

        action                   = allow

        condition                = NULL
```

YANG modeling language composed module encapsulate <key, value> pairs into YANG model. It maps Service2Functional KEY into YANG model.

3.3.2. Service2Functional Translator

Service2Functional Translator does the mapping between YANG model and Functional API according to the database, which means the mapping between Client command and Functional API. It consists of Service2Functional KEY mapping and Service2Functional Value mapping.

3.3.2.1. Data in the database

SRC_DB: "John_0001"---"192.168.1.100"

ACT_DB: "allow"---"permit", "deny"---"deny"

PROTO_DB: "rtsp://video-server/mrbean.ts"---"tcp", "rtsp://video-server/france24.ts"---"tcp", "rtsp://video-server/mrbean.ts,rtsp://video-server/france24.ts"---"tcp"

PORT_DB: "rtsp://video-server/mrbean.ts"---"554", "rtsp://video-server/france24.ts"---"8554", "rtsp://video-server/mrbean.ts,rtsp://video-server/france24.ts"---"554","8554"

DEST_DB: "rtsp://video-server/mrbean.ts"---"100.1.1.100", "rtsp://video-server/france24.ts"---"100.1.1.100", "rtsp://video-server/mrbean.ts,rtsp://video-server/france24.ts"---"100.1.1.100"

3.3.2.2. Service2Functional KEY Mapping

KEY lexicon deposits the correspondence between Service API and Functional API.

Example 1:

<Key in Service API>	<Key in Functional API>
user_group.user.user_name	objects.address_group.address.src.src_ip_addr
object.url_group.url	objects.address_group.address.dst.dst_ip_addr
	objects.service.protocol
	objects.service.port
action	actions

3.3.2.3. Service2Functional Value Mapping

ValueDB deposit actual value in the network corresponding to value in Service API, such as IP/MAC/URL address, protocol+ port number and so on.

For example:

objects.address_group.address.src.src_ip_addr =192.168.1.1

3.3.3. Functional-oriented I2NSF Client

Functional-oriented I2NSF Client consists of YANG modeling language composed module, NETCONF module, and YANG modeling language decomposed module.

YANG modeling language composed module encapsulates actual value in the network into YANG model, and then transports it to NSF/vNSF by NETCONF.

Input

objects.address_group.address.src.src_ip_addr=192.168.1.1

objects.address_group.address.dst.dst_ip_addr=100.1.1.100

objects.service.protocol=tcp

objects.service.port=554

actions=permit

Output

```
<objects>
<address_group>
<address>
<src>
  <src_ip_addr>192.168.1.1</src_ip_addr>
</src>
<dst>
  <dst_ip_addr>100.1.1.100</dst_ip_addr>
</dst>
</address>
</address_group>
<service>
  <protocol>tcp</protocol>
  <port>554</port>
</service>
</objects>
<actions>permit</actions>
......
```

NETCONF module is responsible for communicating with NSF/vNSF.

3.4. NSF/vNSF

NSF/vNSF consists of Functional-oriented I2NSF Agent, Functional2Vendor Translator and Vendor Specific Command.

3.4.1. Functional-oriented I2NSF Agent

Functional-oriented I2NSF Agent is responsible for communicating with I2NSF Controller and the protocol used is NETCONF. It consists of NETCONF module, YANG modeling language decomposed and composed modules.

YANG modeling language decomposed module parses YANG model into <key, value> pairs.

Input:

```
<objects>
<address_group>
<address>
<src>
  <src_ip_addr>192.168.1.1</src_ip_addr>
</src>
<dst>
  <dst_ip_addr>100.1.1.100</dst_ip_addr>
</dst>
</address>
</address_group>
<service>
  <protocol>tcp</protocol>
  <port>554</port>
</service>
</objects>
<actions>permit</actions>
......
```

Output:

<key= objects.address_group.address.src.src_ip_addr ,
value=192.168.1.100>

.....

YANG modeling language composed module encapsulates the <key value> pairs into YANG model.

3.4.2. Functional2Vendor Translator

As the core part of the NSF/vNSF, Functional2Vendor Translator comprises a) lexicon module, b) key mapping module, c) value mapping module, d) command assembling module and e) message parsing module.

Lexicon module deposits the correspondence between the keys of YANG model and the keys ruled by the specific FW syntax.

Example 1: Shorewall lexicon

<Key in Functional API>	<Key in the FW>
objects.address_group.address.src.src_ip_addr	SOURCE
objects.address_group.address.dst.dst_ip_addr	DEST

Example 2: UFW lexicon

<Key in Functional API>	<Key in the FW>
objects.address_group.address.src.src_ip_addr	from
objects.address_group.address.dst.dst_ip_addr	to

Key mapping module is responsible for the translation between the keys of YANG model and the keys ruled by the FW according to the lexicon module.

Input:

<key= objects.address_group.address.src.src_ip_addr, value=192.168.1.100>

.....

Output:

<key= SOURCE, value=192.168.1.100>

.....

Value mapping module is responsible for the translation between the values of YANG model and the values ruled by the FW.

For example: (Shorewall)

Input:

<key= SOURCE, value=192.168.1.100>

.....

Output

<key= SOURCE, value=cli:192.168.1.100>

.....

Command assembling module assembles different vendors FW values into command line form.

Shorewall command assembling format:

#ACTION SOURCE DEST PROTO DEST PORT(S)

Input

<key= SOURCE, value=cli:192.168.1.100>

.....

Output (Shorewall)

ACCEPT cli:192.168.1.100 ser:100.1.1.100 tcp 554

UFW command assembling format:

ufw allow|deny|reject|logging|show [proto protocol] [from ADDRESS [port PORT]] [to ADDRESS [port PORT]]

Output (UFW):

ufw route allow in on eth1 out on eth2 from 192.168.1.100 to 100.1.1.100 port 554 proto tcp

Message parsing module parses the upward report commands of different vendors FW into <key, value> pairs.

3.4.3. Vendor Specific Command

Vendor Specific Command is responsible for communicating with FW configuration policy executive layer, including the sending down commands and upward report commands.

4. Transport Protocol

4.1. Protocol between Client, Controller, and NSF/vNSF

As shown in Figure 1, Service-oriented API connects I2NSF Client with I2NSF Controller Functional-oriented API connects I2NSF Controller with NSF/vNSF.

The protocol used in Service-oriented API and Functional-oriented API is NETCONF whose protocol stack has four layers: Content, Operations, RPC, and Transport [RFC4741]. The set of operations includes <get>, <get-config>, <edit-config>, etc. The configuration data and state data are separated and both of them are expressed by XML. NETCONF protocol is connection-oriented and the connection should provide reliable, ordered transmission. NETCONF provides the fundamental programming features for comfortable and robust automation of network services.

5. YANG model

YANG is a full, formal contract language with rich syntax and semantics to build applications on. YANG is a NETCONF data modeling language which is able to model configuration data, state data, operations, and notifications. YANG definitions can directly map to XML content.

6. Security Considerations

TBD

7. IANA Considerations

TBD

8. References

8.1. Normative References

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

[RFC4741] Enns, R., Ed., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

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

8.2. Informative References

[INCITS359 RBAC] NIST/INCITS, "American National Standard for Information Technology - Role Based Access Control", INCITS 359, April, 2003

[I-D.zarny-i2nsf-data-center-use-cases] Zarny, M., et.al., "I2NSF Data Center Use Cases", Work in Progress, October 2014.

[I-D.qi-i2nsf-access-network-usecase] Qi, M., et.al., "Integrated Security with Access Network Use Case", Work in Progress, October, 2014.

[I-D.pastor-i2nsf-access-usecases] Pastor, A., et.al., "Access Use Cases for an Open OAM Interface to Virtualized Security Services", Work in Progress, October, 2014.

[I-D.dunbar-i2nsf-problem-statement] Dunbar, L., et.al., "Interface to Network Security Functions Problem Statement", Work in Progress, September, 2014.

[I-D.xia-i2nsf-capability-interface-im] Liang Xia, "Information Model of Interface to Network Security Functions Capability Interface", Work in Progress, January, 2015.

[I-D.strassner-i2nfs-info-model] Strassner, et.al., "Interface to Network Security Functions Information Model", February, 2015.

[I-D.xia-i2nsf-service-interface-dm] Liang Xia, et.al., "Data Model of Interface to Network Security Functions Service Interface", February, 2015.

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Yuming Xie
Huawei Technologies
Email: yuming.xie@huawei.com

Liang Xia (Frank)
Huawei Technologies
Email: Frank.xialiang@huawei.com

Jun Wu
Huawei Technologies
Email: junwu.wu@huawei.com