

I2NSF
Internet Draft
Intended status: Standard Track

John Strassner
Liang Xia
Huawei

Expires: August 2015

February 10, 2015

Interface to Network Security Functions Information Model
draft-strassner-i2nfs-info-model-00.txt

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/lid-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 August 10, 2015.

Copyright Notice

Copyright (c) 2015 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

Internet-Draft

I2NSF Information Model

February 2015

Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Abstract

This document describes an information model that defines the salient managed entities and their relationships in an Interface to Network Security Function (I2NSF) architecture. The information model is independent of platform, language, and protocol, and serves as a common consensual lexicon for the I2NFS architecture as well as clients using this architecture. This enables multiple application-specific data models (which are dependent on platform, language, and/or protocol) to be built from this information model. The advantage of doing so is to ensure that such data models will be able to share and reuse consensually defined concepts, thereby increasing interoperability.

Table of Contents

1.	Introduction	3
2.	Conventions used in this document	3
2.1.	Acronyms	4
2.2.	Definitions	4
2.2.1.	Information Model	4
2.2.2.	Data Model.....	5
2.2.3.	Inheritance	5
2.2.4.	Relationship	5
2.2.4.1.	Association	5
2.2.4.2.	Aggregation	5
2.2.4.3.	Composition	5
2.2.5.	Multiplicity	6
2.3.	Symbology	6
2.3.1.	Basic Symbols	6
2.3.2.	Relationship Multiplicity	6
2.3.3.	Relationship Naming	7
2.3.4.	Relationship Navigability	7
2.3.5.	Relationships Implemented as Classes	7
3.	Design of the I2NSF Information Model	8
3.1.	Overview	8
3.2.	I2NSF Clients	10
3.3.	Security Policy Layer	10
3.4.	Context-Aware Policy Engine	10

3.5. Security Capability Layer	11
4. I2NSF Information Model Hierarchy	11
5. Usage Examples of the I2NSF Information Model	11

6. Security Considerations	11
7. IANA Considerations	12
8. References	12
8.1. Normative References	12
8.2. Informative References	12
9. Acknowledgments	12

1. Introduction

Networks and networked applications are becoming increasingly diverse and complex. Concurrently, operators are struggling to deploy new services quickly, and require more powerful and robust network management services and applications. Both of these problems are exacerbated by the proliferation of vendor-specific devices and data models.

While Yang offers an easier way to build data models, it lacks several software abstractions that facilitate representing high-level entities and relationships between such entities. UML Information Models are the de facto method for defining entities and relationships in a technologically-neutral manner. The use of an information model increases interoperability by defining a common lexicon of concepts, terms, entities, and relationships that all clients and applications can use. Data Models created in Yang, as well as in other technologies, can use the terms and concepts defined in the information model to ensure that concepts defined in different technologies can be identified and translated to each other.

The I2NFS Information Model will define managed objects and mechanisms to address the operational, administrative, and management aspects of the managed objects.

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. Acronyms

AAA - Authentication, Authorization, Accounting

ABAC - Attribute Based Access Control

I2NSF - Interface to Network Security Functions

Netconf - Network Configuration protocol

NSF - Network Security Function

OAM - Operational, Administrative, and Management

PAP - Policy Administration Point

PBAC - Policy Based Access Control

PDP - Policy Decision Point

PEP - Policy Enforcement Point

PIP - Policy Information Point

PR - Policy Repository

PXP - Policy Execution Point

RBAC - Role Based Access Control

UML - Unified Modeling Language

Yang - A data definition language for use with Netconf

2.2. Definitions

This section defines important terms that are used in this document.

2.2.1. Information Model

An information model is a representation of concepts of interest to an environment in a form that is independent of platform, language, and protocol.

2.2.2. Data Model

A data model is a representation of concepts of interest to an environment in a form that is dependent on platform, language, and/or protocol (typically, but not necessarily, all three).

2.2.3. Inheritance

Inheritance makes an entity at a lower level of abstraction (e.g., the subclass) a type of an entity at a higher level of abstraction (e.g., the superclass). A subclass does NOT change the characteristics or behavior of the superclass that it inherits from. However, a subclass MAY add new attributes and relationships that distinguish it from the attributes and relationships defined by its superclass.

2.2.4. Relationship

A relationship is a generic term that represents how a first set of entities interact with a second set of entities. A recursive relationship sets the first and second entity to the same entity. There are three basic types of relationships, as defined in the subsections below.

2.2.4.1. Association

An association represents a generic dependency between a first and a second set of entities.

2.2.4.2. Aggregation

An aggregation is a stronger type (i.e., more restricted semantically) of association, and represents a whole-part dependency between a first and a second set of entities. In other words, three objects are implied by an aggregation: the first entity, the second entity, and a new third entity that represents the combination of the first and second entities. The entity owning the aggregation is referred to as the "aggregate", and the entity that is aggregated is referred to as the "part".

2.2.4.3. Composition

A composition is a stronger type (i.e., more restricted semantically) of aggregation, and represents a whole-part dependency with two important behaviors. First, an instance of the part is included in at most one instance of the aggregate at a time. Second, any action performed on the composite entity (i.e., the aggregate) is

propagated to its constituent part objects. For example, if the composite entity is deleted, then all of its constituent part entities are also deleted. This is not true of aggregations or associations - in both, only the entity being deleted is actually removed, and the other entities are unaffected.

2.2.5. Multiplicity

A specification of the range of allowable cardinalities that a set of entities may assume. This is always a pair of ranges, such as 1 - 1 or 0..n - 2..5.

2.3. Symbology

In order to unambiguously represent information in this document, ASCII art will be used. This art will use the following special symbols.

2.3.1. Basic Symbols

Indentation: this will be used to indicate hierarchy levels

Inheritance: represented by --|> (e.g., subclass -I> superclass)

Association: represented by --A- (e.g., ClassA --A- ClassB)

Aggregation: represented by --Ag- (e.g., ClassA --Ag- ClassB)

Composition: represented by --C- (e.g., ClassA --C- ClassB)

2.3.2. Relationship Multiplicity

Relationships MUST have a specified multiplicity, and can be represented as follows:

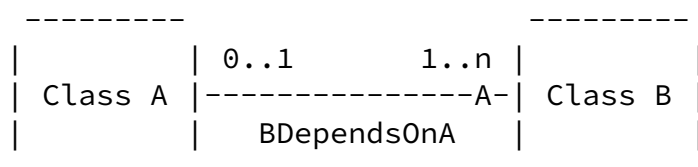


Figure 1. Illustrating a Non-Directed Association.

2.3.3. Relationship Naming

Relationships MUST be named. This is to facilitate their implementation as named managed objects. The name SHOULD be in InitCaps. In Figure 1, Class B depends on Class A (e.g., an event must happen to A before B can take action).

2.3.4. Relationship Navigability

Relationships MAY indicate a constraint on which set of entities can communicate with the other set of entities in a relationship. If there is no such constraint, then the symbology in [Section 2.3.1](#), illustrated by Figure 1 in [Section 2.3.2](#), is sufficient. Otherwise, an arrow, denoted by the right angle character (i.e., ">"), is used. Relationships with constraint and a specified multiplicity can be represented as follows:

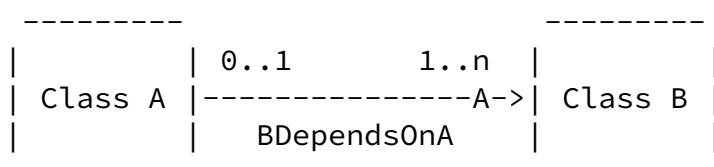


Figure 2. Illustrating a Directed Association.

In this case, Entities of Class A can communicate with Class B; the reverse is not allowed.

2.3.5. Relationships Implemented as Classes

A relationship MAY be realized as a class. This is typically called an "association class", regardless of whether the association is an association, an aggregation, or a composition. This is illustrated as follows:

Strassner, et al. Expires August 10, 2015 [Page 7]

Internet-Draft I2NSF Information Model February 2015

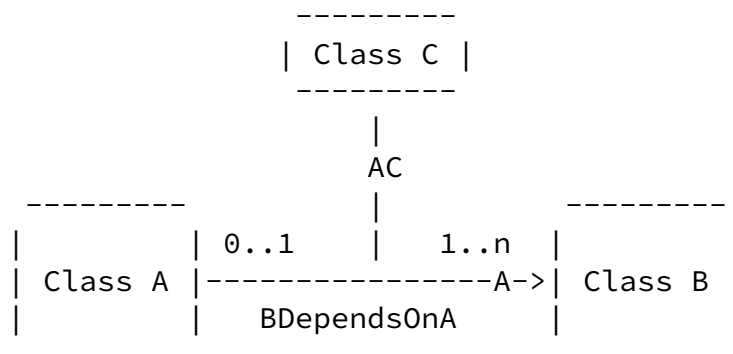


Figure 3. Illustrating an Association Class.

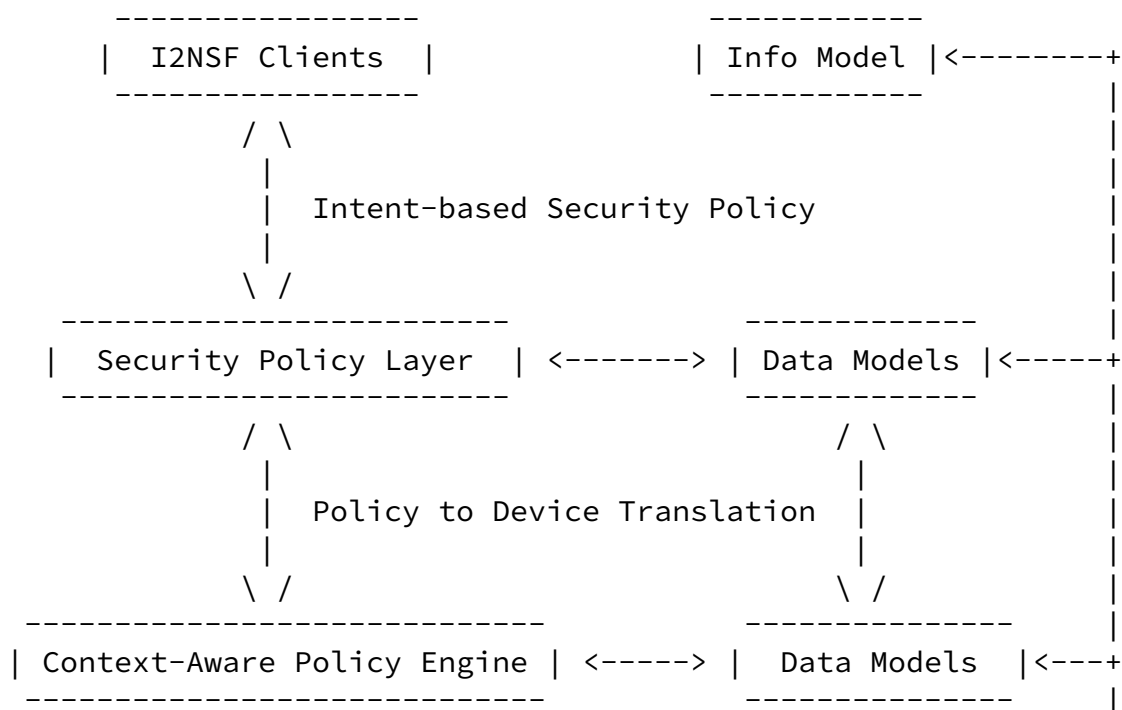
In Figure 3, Class C is an association class, and represents the realization of the association named BDependsOnA. Conceptually, this means that the relationship between Classes A and B is complex, and requires a class to represent it. For example, class attributes MAY be used to define how Class B depends on Class A.

3. Design of the I2NSF Information Model

This section describes the I2NSF Information Model. It first provides an architectural overview of the main entities involved. Then, it defines an object-oriented information model for representing the entities and their relationships.

3.1. Overview

The operation of I2NSF may be conceptualized as shown in Figure 4.



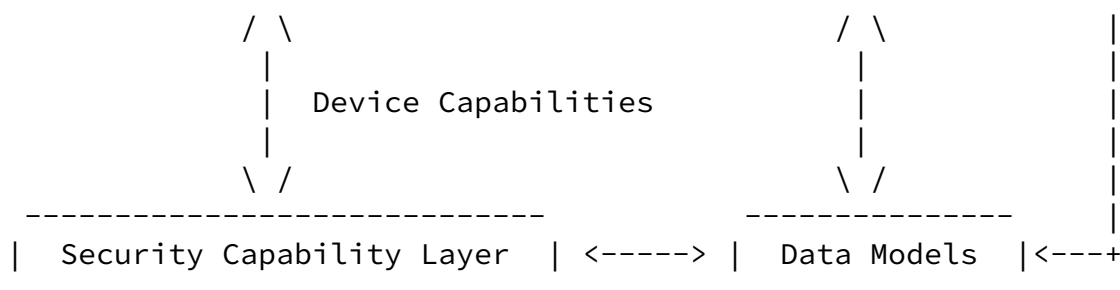


Figure 4. Conceptual Operation of the I2NSF.

An I2NSF client is a user, application, process, or similar entity that wants to invoke a physical or virtual Network Security Function for OAM purposes. This results from the I2NSF Security Policy Layer exposing a set of security capabilities of one or more network devices. The I2NSF client does not have to be aware of which set of security devices is present or is being used; all the I2NSF client needs to be cognizant of is which network security functions are required for a given flow.

The Context-Aware Policy Engine uses information about the subject and target of the policy, the current context, the type of operation desired, and any other required information, and translates the (high-level) intent of the I2NSF client to the capabilities of the network security devices.

The Intent-based Security Policy is a declarative specification of the security functions required by an I2NSF client. Since multiple

devices can have non-interoperable data models that describe their capabilities, I2NSF uses an information model to represent all concepts that are used by I2NSF clients, applications, and the system itself. This enables each of the three layers shown in Figure 4 to (in principle) have their own data model to represent their functionality in an efficient manner. The information model serves as a lexicon that enables different entities in the I2NSF architecture to map the same or similar terms to different expressions from different data models.

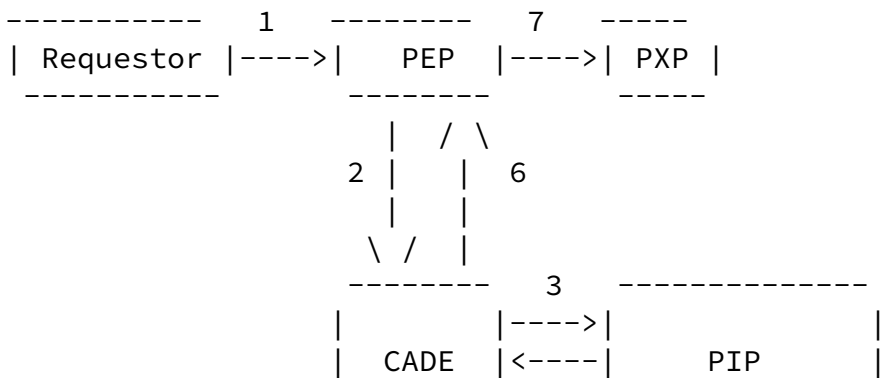
The following subsections define important architectural entities in each layer in more detail.

3.2. I2NSF Clients

3.3. Security Policy Layer

3.4. Context-Aware Policy Engine

The purpose of I2NSF is to ensure that only properly authorized and validated requests to perform operations on Resources and Services are allowed.



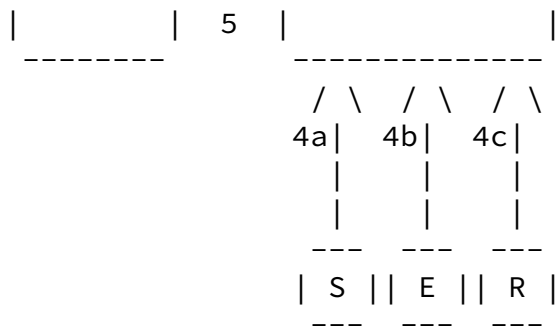


Figure 5. High-Level Overview of the I2NSF Access Control Engine.

3.5. Security Capability Layer

4. I2NSF Information Model Hierarchy

TBD

5. Usage Examples of the I2NSF Information Model

TBD

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.

8.2. Informative References

9. Acknowledgments

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

Authors' Addresses

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95138
USA
email: john.sc.strassner@huawei.com

Liang Xia
Huawei Technologies
email: Frank.xialiang@huawei.com