SUPA
Network Working Group
Internet Draft
Intended status: Informational
Expires: January 8, 2017

J. Bi
Tsinghua University
G. Karagiannis
J. Strassner
Huawei Technologies
M. Klyus
NetCracker
Q. Sun
China Telecom
Luis M. Contreras
Telefonica
July 8, 2016

Problem Statement for Simplified Use of Policy Abstractions (SUPA)
draft-bi-supa-problem-statement-01

Abstract

Simplified Use of Policy Abstractions (SUPA) defines a set of rules that define how services are designed, delivered, and operated within an operator's environment independent of any one particular service or networking device. SUPA expresses policy rules using a generic policy information model, which serves as a unifying influence to enable different data model implementations to be simultaneously developed.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The rapid growth in the variety and importance of traffic flowing over increasingly complex enterprise and service provider network architectures makes the task of network operations and management applications and deploying new services much more difficult. In addition, network operators want to deploy new services quickly and efficiently. Two possible mechanisms for dealing with this growing difficulty are the use of software abstractions to simplify the design and configuration of monitoring and control operations and the use of programmatic control over the configuration and operation of such networks. Policy-based management can be used to combine these two mechanisms into an extensible framework.

Policy rules can be used to express high-level network operator requirements directly, or from a set of management applications, to a network management or element system. The network management or element system can then control the configuration and/or monitoring of network elements and services.

Simplified Use of Policy Abstractions (SUPA) will define a generic policy information model (GPIM) [SUPA-info-model] for use in network operations and management applications. The GPIM defines concepts and terminology needed by policy management indepednent of the form and content of the policy rule. The ECA Policy Rule Information Model (EPRIM) [SUPA-info-model] extends the GPIM to define how to build policy rules according to the event-condition-action paradigm.

Both the GPIM and the EPRIM are targeted at controlling the configuration and monitoring of network elements throughout the service development and deployment lifecycle. The GPIM and the EPRIM will both be translated into corresponding YANG [RFC6020] modules that define policy concepts, terminology, and rules in a generic and interoperable manner; additional YANG modules may also be defined from the GPIM and/or EPRIM to manage specific functions.

The key benefit of policy management is that it enables different network elements and services to be instructed to behave the same way, even if they are programmed differently. Management applications will benefit from using policy rules that enable scalable and consistent programmatic control over the configuration and monitoring of network elements and services.

1.1. Problem Statement

Network operators must construct networks of increasing size and complexity in order to improve their availability and quality, as more and more business services depend on them.

Currently, different technologies and network elements require different forms of the same policy that governs the production of network configuration snippets. The power of policy management is its applicability to many different types of systems, services, and networking devices. This provides significant improvements in configuration agility, error detection, and uptime for operators.

Many different types of actors can be identified that can use a policy management system, including applications, end-users, developers, network administrators, and operators. Each of these actors typically has different skills and uses different concepts and terminologies. For example, an operator may want to express that only Platinum and Gold users can use streaming and interactive multimedia applications. As a second example, an operator may want to define a more concrete policy rule that looks at the number of dropped packets. If, for example, this number exceeds a certain threshold value, then the applied queuing, dropping and scheduling algorithms could be changed in order to reduce the number of dropped packets. The power of SUPA is that both of these examples may be abstracted. For example, in the latter example, different thresholds and algorithms could be defined for different classes of service.

1.2. Proposed Solution

SUPA enables network operators to express policies to control network configuration and monitoring data models in a generic manner. The configuration and monitoring processes are independent of device, as well as domain or type of application, and result in configuration according to YANG data models.

Both of the examples in section 1.1 can be referred to as "policy rules", but they take very different forms, since they are defined at different levels of abstraction and likely authored by different actors. The first example described a very abstract policy rule, and did not contain any technology-specific terms, while the second example included more concrete policy rules and likely used technical terms of a general (e.g., IP address range and port numbers) as well as vendor-specific nature (e.g., specific algorithms implemented in a particular device). Furthermore, these two policy rules could affect each other. For example, Gold and Platinum users might need different device configurations to give the proper QoS markings to their streaming multimedia traffic. This is very difficult to do if a common policy framework does not exist.

Note that SUPA is not limited to any one type of technology. While the above two policies could be considered "QoS" policies, other examples include:

- network elements must not accept passwords for logins

- all SNMP agents in this network must drop all SNMP traffic unless it is originating from, or targeting, the management network

- Periodically perform workload consolidation if average CPU utilization falls below X%

The above three examples are not QoS related; this emphasizes the utility of the SUPA approach in being able to provide policies to control different types of network element configuration and/or monitoring snippets.

There are many types of policies. SUPA differentiates between "management policies" and "embedded policies". Management policies are used to control the configuration of network elements. Management policies can be interpreted externally to network elements, and the interpretation typically results in configuration changes of collections of network elements. In contrast, "embedded policies" are policies that are embedded in the configuration of network elements, and are usually interpreted on network elements in isolation. Since embedded policies are interpreted in the network device, they are typically composed in a very specific fashion to run at near-realtime timescales.

1.3. Value of the SUPA Approach

SUPA will achieve an optimization and reduction in the amount of work required to define and implement policy-based data models in the IETF. This is due to the generic and extensible framework provided by SUPA.

SUPA defines policy independent of where it is located. Other WGs are working on embedding policy in the configuration of a network element; SUPA is working on defining policies that can be interpreted external to network elements (i.e., management policies). Hence, SUPA policies can be used to define the behavior of and interaction between embedded policies.

Since the GPIM defines common policy terminology and concepts, it can be used to both define more specific policies as part of a data model as well as derive a (more abstract) information model from a (more specific) data model.

This latter approach may be of use in discovering common structures that occur in data models that have been designed in isolation of each other.

The SUPA policy framework defines a set of consistent, flexible, and scalable mechanisms for monitoring and controlling resources and services. It may be used to create a management and operations interface that can enable existing IETF data models, such as those from I2RS and L3SM, to be managed in a unified way that is independent of application domain, technology and vendor. Resource and service management become more effective, because policy defines the context that different operations, such as configuration and monitoring, are applied to.

2. Application of Generic Policy-based Management

This section provides examples of how SUPA can be used to define different types of policies. Examples applied to various domains, including system management, operations management, access control, routing, and service function chaining, are also included. Note that typical use cases and the applicability of SUPA policy models are provided in [SUPA-Applicability].

ECA policies are rules that consist of an event clause, a condition clause, and an action clause.

Network Service Management Example

```
Event:     too many interface alarms received from an
           L3VPN service
Condition: alarms resolve to the same interface within a
           specified time period
Action:    if error rate exceeds x% then put L3VPN service
           to Error State and migrate users to one or more
           new L3VPNs
```

Security Management Example

```
Event:     anomalous traffic detected in network
Condition: determine the severity of the traffic
Action:    apply one or more actions to affected NEs based
           on the type of the traffic detected (along with
           other factors, such as the type of resource
           being attacked if the traffic is determined to
           be an attack)
```

Traffic Management Examples

```
Event:     edge link close to being overloaded by
           incoming traffic
Condition: if link utilization exceeds Y% or if link
           utilization average is increasing over a
           specified time period
Action:    change routing configuration to other peers
           that have better metrics


Event:     edge link close to be overloaded by
           outgoing traffic
Condition: if link utilization exceeds Z% or if link
           utilization average is increasing over a
           specified time period
Action:    reconfigure affected nodes to use source-based
           routing to balance traffic across multiple links
```

Service Management Examples

```
Event:     alarm received or periodic time period check
Condition: CPU utilization level comparison
Action:    no violation: no action
           violation:
             1) determine workload profile in time interval
             2) determine complementary workloads (e.g.,
                whose peaks are at different times in day)
             3) combine workloads (e.g., using integer
                programming)

Event:     alarm received or periodic time check
Condition: if DSCP == AFxy and
           throughput < T% or packet loss > P%
Action:    no: no action
           yes: remark to AFx'y'; reconfigure queuing;
           configure shaping to S pps; ...
```

Note: it is possible to construct an ECA policy rule that is directly tied to configuration parameters.

3. Conclusions: the Value of SUPA

SUPA can be used to define high-level, possibly network-wide policies to create interoperable network element configuration snippets. SUPA expresses policies and associated concepts using a generic policy information model, and produces generic policy YANG data modules. SUPA focuses on management policies that control the configuration of network elements. Management policies can be interpreted outside of network elements, and the interpretation typically results in configuration changes to collections of network elements.

Policies embedded in the configuration of network elements are not in the scope of SUPA. In contrast to policies targeted by SUPA, embedded policies are usually interpreted on network elements in isolation, and often at timescales that require the representation of embedded policies to be optimized for a specific purpose.

The SUPA information model generalizes common concepts from multiple technology-specific data models, and makes it reusable. Conceptually, SUPA can be used to interface and manage existing and future data models produced by other IETF working groups. In addition, by defining an object-oriented information model with metdata, the characteristics and behavior of data models can be better defined.

4. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting detailed configuration states of network elements. This places additional security requirements on SUPA (e.g., authorization, and authentication of network services) that needs further investigation. Moreover, policy interpretation can lead to corner cases and side effects that should be carefully examined, e.g., in case policy rules are conflicting with each other.

5. IANA Considerations

This document has no actions for IANA.

6. Contributors

The following people all contributed to creating this document, listed in alphabetical order:

Parviz Yegani, Juniper Networks
Vikram Choudhary, Huawei Technologies
Diego Lopez, Telefonica I+D
Dan Romascanu, Avaya
J. Schoenwaelder, Jacobs University, Germany
Will(Shucheng) Liu, Huawei Technologies
Tina Tsou
Jean-Francois Tremblay

7. Acknowledgments

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members: H. Rafiee, J. Saperia, C. Zhou, Spencer Dawkins, Xing Li, Chongfeng Xie, Benoit Claise, Ian Farrer, Marc Blancet, Zhen Cao, Hosnieh Rafiee, Mehmet Ersue, Simon Perreault, Fernando Gont, Jose Saldana, Tom Taylor, Kostas Pentikousis, Juergen Schoenwaelder, Eric Voit, Scott O. Bradner, Marco Liebsch, Scott Cadzow, Marie-Jose Montpetit.

8. References

8.1. Informative References

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

[SUPA-info-model] J. Strassner, J. Halpern, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", IETF Internet draft, draft-ietf-supa-generic-policy-info-model-00, June 2016

[SUPA-Applicability] N. Vadrevu, D. Zhang, S. Zhu, Y. Cheng, "Applicability of SUPA", IETF Internet draft, draft-vadrevu-supa-applicability-06, March 2016

Authors' Addresses

Georgios Karagiannis
Huawei Technologies
Hansaallee 205,
40549 Dusseldorf,
Germany
Email: Georgios.Karagiannis@huawei.com

Maxim Klyus, Ed.
NetCracker
Kozhevnicheskaya str.,7 Bldg. #1
Moscow, Russia
E-mail: klyus@netcracker.com

Qiong Sun
China Telecom
No.118 Xizhimennei street, Xicheng District
Beijing 100035
P.R. China
Email: sunqiong@ctbri.com.cn

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, Sur-3 building, 3rd floor
Madrid 28050
Spain
Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

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

Jun Bi
Tsinghua University
Network Research Center, Tsinghua University
Beijing 100084
China
EMail: junbi@tsinghua.edu.cn

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 20, 2017

J. Halpern
Ericsson
J. Strassner
Huawei Technologies

July 20, 2016

Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)

draft-ietf-supa-generic-policy-data-model-00

Abstract

This document defines two YANG policy data models. The first is a generic policy model that is meant to be extended on an application-specific basis. The second is an exemplary extension of the first generic policy model, and defines rules as event-condition-action policies. Both models are independent of the level of abstraction of the content and meaning of a policy.

Status of this Memo

Copyright Notice

Table of Contents

1. Overview

This document defines two YANG [RFC6020] [RFC6991] policy data models. The first is a generic policy model that is meant to be extended on an application-specific basis. It is derived from the Generic Policy Information Model (GPIM) defined in [1]. The second is an exemplary extension of the first (generic policy) model, and defines policy rules as event-condition-action tuples. Both models are independent of the level of abstraction of the content and meaning of a policy.

The GPIM defines a common framework as a set of model elements (e.g., classes, attributes, and relationships) that specify a common set of policy management concepts that are independent of the type of policy (e.g., imperative, procedural, declarative, or otherwise). The first YANG data model is a translation of the GPIM to a YANG module. The Eca Policy Rule Information Model (EPRIM), also defined in [1], extends the GPIM to represent policy rules that use the Event-Condition-Action (ECA) paradigm. The second YANG data model maps the EPRIM to YANG. The second YANG data model MAY be used to augment the functionality of the first YANG data 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 [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

3. Terminology

This section defines acronyms, terms, and symbology used in the rest of this document.

3.1. Acronyms

CNF	Conjunctive Normal Form
DNF	Disjunctive Normal Form
ECA	Event-Condition-Action
EPRIM	(SUPA) ECA Policy Rule Information Model
GPIM	(SUPA) Generic Policy Information Model
NETCONF	Network Configuration protocol
OAM&P	Operations, Administration, Management, and Provisioning
OCL	Object Constraint Language
OID	Object IDentifier
SUPA	Simplified Use of Policy Abstractions
UML	Unified Modeling Language
URI	Uniform Resource Identifier

3.2. Definitions

Action: a set of purposeful activities that have a set of associated behavior.

Boolean Clause: a logical statement that evaluates to either TRUE or FALSE. Also called Boolean Expression.

Condition: a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to make a decision. A Condition, when used in the context of a Policy Rule, is used to determine whether or not the set of Actions in that Policy Rul can be executed or not.

Constraint: A constraint is a limitation or restriction. Constraints may be added to any type of object (e.g., events, conditions, and actions in Policy Rules).

Constraint Programming: a type of programming that uses constraints to define relations between variables in order to find a feasible (and not necessarily optimal) solution.

Data Model: a data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol (typically one or more of these).

ECA: Event - Condition - Action policy.

Event: an Event is defined as any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. An Event, when used in the context of a Policy Rule, is used to determine whether the condition clause of an imperative Policy Rule can be evaluated or not.

Information Model: an information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol.

Metadata: is data that provides descriptive and/or prescriptive information about the object(s) to which it is attached.

Policy Rule: A Policy Rule is a set of rules that are used to manage and control the changing or maintaining of the state of one or more managed objects.

3.3. Symbology

The following representation is used to describe YANG data modules defined in this draft.

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

4. Design of the SUPA Policy Data Models

This will be completed in the next version of this draft. Three important points are:

- different policy models have common semantics
- capture those semantics within a common framework (GPIM)
- extend these semantics with a specific ECA example (EPRIM)

5. SUPA Policy Data Model YANG Module

The SUPA YANG data model module is divided into two main parts:

1) a set of containers that represent the objects that make updated a Policy Rule and its Policy Rule Components
2) a set of containers that represent the objects that define and apply metadata to Policy Rules and/or Policy Rule Components

< This will be finished in version 02 >

```
<CODE BEGINS> file "ietf-supa-policydatamodel@2016-03-21.yang"

module ietf-supa-policydatamodel {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-supa-policydatamodel";
    prefix supa-pdm;

    import ietf-yang-types {
        prefix yang;
    }

    organization "IETF";
        contact
            "Editor: Joel Halpern
             email: jmh@joelhalpern.com;
             Editor: John Strassner
             email: strazpdj@gmail.com;";

    description
        "This module defines a data model for generic high level
         definition of policies to be applied to a network.
         This module is derived from and aligns with
         draft-strassner-supa-generic-policy-info-model-04.
         Details on all classes, associations, and attributes
         can be found there.
         Copyright (c) 2015 IETF Trust and the persons identified
         as the document authors.  All rights reserved.
         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and
         subject to the license terms contained in, the Simplified
         BSD License set forth in Section 4.c of the IETF Trust's
         Legal Provisions Relating to IETF Documents
         (http://trustee.ietf.org/license-info).";
```

```
    revision 2016-07-20 {
        description
            "Conversion to WG draft, 20160720.
             Fixed pyang 1.1 compilation errors. Fixed must clause
             derefencing used in grouping statements. Reformatted
             and expanded descriptions. Fixed various typos.
             Initial version, 20160321";
        reference
            "draft-ietf-supa-policy-data-model-00";
    }


    typedef policy-constraint-language-list {
        type enumeration {
            enum "undefined" {
                description
                    "This may be used as an initialization and/or
                     an error state.";
            }
            enum "OCL2.4" {
                description
                    "Object Constraint Language v2.4. This is a
                     declarative language for describing rules for
                     defining constraints and query expressions.";
            }
            enum "OCL2.x" {
                description
                    "Object Constraint Language, v2.0 through 2.3.1.";
            }
            enum "OCL1.x" {
                description
                    "Object Constraint Language, any version prior
                     to v2.0.";
            }
            enum "QVT1.2R" {
                description
                    "QVT Relational Language.";
            }
            enum "QVT1.2O" {
                description
                    "QVT Operational language.";
            }
            enum "Alloy" {
                description
                    "A language for defining structures and
                     and relations using constraints.";
            }
        }
```

```
        description
            "The language used to encode the constraints
             relevant to the relationship between the metadata
             and the underlying policy object.";
    }

    typedef policy-data-type-id-encoding-list {
        type enumeration {
            enum "undefined" {
                description
                    "This can be used for either initialization
                     or for signifying an error.";
            }
            enum "String" {
                description
                    "The clause is directly present in
                     the content.";
            }
            enum "GUID" {
                description
                    "The clause is referenced by this GUID.";
            }
            enum "UUID" {
                description
                    "The clause is referenced by this UUID.";
            }
            enum "URI" {
                description
                    "The clause is referenced by this URI.";
            }
            enum "FQDN" {
                description
                    "The clause is referenced by this FQDN.";
            }
        }
        description
            "The list of possible data types used to represent object
             IDs in the SUPA policy hierarchy.";
    }

    typedef policy-data-type-encoding-list {
        type enumeration {
            enum "undefined" {
                description
                    "This can be used for either initialization
                     or for signifying an error.";
            }
            enum "string" {
                description
                    "This represents a string data type.";
            }
```

```
            enum "integer" {
                description
                    "This represents an integer data type.";
            }
            enum "boolean" {
                description
                    "This represents a Boolean data type.";
            }
            enum "floating point" {
                description
                    "This represents a floating point data type.";
            }
            enum "date-and-time" {
                description
                    "This represents a data type that can specify
                     date and/or time.";
            }
            enum "GUID" {
                description
                    "This represents a GUID data type.";
            }
            enum "UUID" {
                description
                    "This represents a UUID data type.";
            }
            enum "URI" {
                description
                    "This represents a Uniform Resource Identifier
                     (URI) data type.";
            }
            enum "DN" {
                description
                    "This represents a Distinguished Name (DN)
                     data type.";
            }
            enum "NULL" {
                description
                    "This represents a NULL data type. NULL means the
                     absence of an actual value. NULL is frequently
                     used to represent a missing or invalid value.";
            }
        }
        description
            "The set of allowable data types used to encode
             multi-valued SUPA Policy attributes.";
    }

// identities are used in this model as a means to provide simple
// reflection to allow an instance-identifier to be tested as to what
// class it represents. In turn, this allows must clauses to specify
// that the target of a particular instance-identifier leaf must be a
// specific class, or within a certain branch of the inheritance tree.
```

```
// This depends upon the ability to refine the entity class default
// value. The entity class should be read-only. Howeverm as this is
// the target of a MUST condition, it cannot be config-false. Also,
// it appears that we cannot put a MUST condition on its definition,
// as the default (actual) value changes at each inheritance.

    identity POLICY-OBJECT-TYPE {
        description
            "The identity corresponding to a SUPAPolicyObject
             object instance.";
    }

    grouping supa-policy-object-type {
        leaf supa-policy-ID {
            type string;
            mandatory true;
            description
                "The string identifier of this policy object.
                 It must be unique within the policy system.";
        }
        leaf entity-class {
            type identityref {
                base POLICY-OBJECT-TYPE;
            }
            default POLICY-OBJECT-TYPE;
            description
                "The identifier of the class of this grouping.";
        }
        leaf supa-policy-object-ID-encoding {
            type policy-data-type-id-encoding-list;
            mandatory true;
            description
                "The encoding used by the supa-object-ID.";
        }
        leaf supa-policy-object-description {
            type string;
            description
                "Human readable description of the characteristics
                 and behavior of this policy object.";
        }
        leaf supa-policy-name {
            type string;
            description
                "A human-readable name for this policy.";
        }
        leaf-list supa-has-policy-metadata-agg {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-METADATA-ASSOC)";
```

```
            description
                "The SUPAPolicyObject object instance that aggregates
                 this set of SUPAPolicyMetadata object instances. As
                 there are attributes on this association, the
                 instance-identifier MUST point to an instance using
                 the grouping supa-has-policy-metadata-detail (which
                 includes subclasses of this association class).";
        }
        description
            "This is the superclass for all SUPA objects. It is
             used to define common attributes and relationships
             that all SUPA subclasses inherit.";
    }

    identity POLICY-COMPONENT-TYPE {
        base POLICY-OBJECT-TYPE;
        description
            "The identity corresponding to a
             SUPAPolicyComponentStructure object instance.";
    }

    grouping supa-policy-component-structure-type {
        uses supa-policy-object-type {
            refine entity-class {
                default POLICY-COMPONENT-TYPE;
            }
        }
        leaf supa-has-policy-component-decorator-part {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-COMPONENT-DECORATOR-ASSOC)";
            mandatory true;
            description
                "A reference to the association class for relating
                 policy component decorators to the policy components
                 they decorate. This is the set of
                 SUPAPolicyComponentStructure object instances that are
                 aggregated by a SUPAPolicyComponentDecorator object
                 instance. As there are attributes on this association,
                 the instance-identifier MUST point to an instance
                 using the specified grouping. This defines the object
                 class that this instance-identifier points to.";
       }
       description
          "A superclass for all objects that represent different types
           of components of a Policy Rule. Important subclasses include
           the SUPAPolicyClause and the SUPAPolicyComponentDecorator.
           This object is the root of the decorator pattern; as such,
           it enables all subclasses to be decorated.";
    }
```

```
    identity POLICY-COMPONENT-DECORATOR-TYPE {
        base POLICY-COMPONENT-TYPE;
        description
            "The identity corresponding to a
             SUPAPolicyComponentDecorator object instance.";
    }

    grouping supa-policy-component-decorator-type {
        uses supa-policy-component-structure-type {
            refine entity-class {
                default POLICY-COMPONENT-DECORATOR-TYPE;
            }
        }
        leaf-list supa-has-policy-component-decorator-agg {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-COMPONENT-DECORATOR-ASSOC)";
            max-elements 1;
            description
                "The SUPAPolicyComponentDecorator object instance
                 that aggregates this set of
                 SUPAPolicyComponentStructure object instances. This
                 is a list of associations to the SUPA policy components
                 that this decorator decorates. As there are attributes
                 on this association, the instance-identifier MUST
                 point to an instance using the specified grouping.
                 This defines the object class that this
                 instance-identifier points to.";
        }
        leaf-list supa-decorator-constraints {
            type string;
            description
               "A constraint expression applying to this
                decorator, allowing specification of details not
                captured in its subclasses, using an appropriate
                constraint language.";
        }
        leaf supa-has-decorator-constraint-encoding {
            type policy-constraint-language-list;
            description
               "The language in which the constraints on the
                policy component decorator is expressed.";
        }
        description
           "This object implements the decorator pattern, which
            enables all or part of one or more objects to wrap
            another concrete object.";
    }
```

```
    identity POLICY-COMPONENT-CLAUSE-TYPE {
        base POLICY-COMPONENT-TYPE;
        description
            "The identity corresponding to a SUPAPolicyClause
             object instance.";
    }

    grouping supa-policy-clause-type {
        uses supa-policy-component-structure-type {
            refine entity-class {
               default POLICY-COMPONENT-CLAUSE-TYPE;
            }
        }
        leaf supa-policy-clause-exec-status {
            type enumeration {
                enum "Unknown" {
                    description
                       "This may be used as an initialization and/or
                        an error state.";
                }
                enum "Completed" {
                    description
                       "This signifies that this particular policy
                        clause has run successfully, and is now idle.";
                }
                enum "Working" {
                    description
                       "This signifies that this particular policy
                        clause is currently in use, and no errors have
                        been reported.";
                }
                enum "Not Working" {
                    description
                       "This signifies that this particular policy
                        clause is currently in use, but one or more
                        errors have been reported.";
                }
                enum "Available" {
                    description
                       "This signifies that this particular policy
                        clause could be used, but currently is not
                        in use.";
                }
                enum "In Test" {
                    description
                       "This signifies that this particular policy
                        clause is not for use in operational policies.";
                }
```

```
                enum "Disabled" {
                    description
                       "This signifies that this particular policy
                        clause is not available for use.";
                }
            }
            description "This describes whether this policy clause is in
                use and if so whether it is working properly.";
        }
        leaf-list supa-has-policy-clause-part {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-CLAUSE-ASSOC)";
            min-elements 1;
            description
                "The set of SUPAPolicyClause object instances that are
                 aggregated by this SUPAPolicyStructure (i.e., this
                 SUPA Policy Rule) object instance. This defines the
                 object class that this instance-identifier points to.";
        }
        description "The parent class for all SUPA Policy Clauses.";
    }

    identity POLICY-ENCODED-CLAUSE-TYPE {
        base POLICY-COMPONENT-CLAUSE-TYPE;
        description
            "The identity corresponding to a SUPAPolicyEncodedClause
             object instance.";
    }

    grouping supa-encoded-clause-type {
        uses supa-policy-clause-type {
            refine entity-class {
                default POLICY-ENCODED-CLAUSE-TYPE;
            }
        }
        leaf supa-encoded-clause-content {
            type string;
            mandatory true;
            description
               "Either a reference to a source for this clause or the
                string representation of the clause.";
        }
        leaf supa-encoded-clause-encoding {
            type policy-data-type-id-encoding-list;
            mandatory true;
            description
               "The encoding for the encoding clause content.";
        }
```

```
        leaf supa-encoded-clause-language {
            type enumeration {
                enum "undefined" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
                enum "CLI" {
                    description
                        "This defines the language as a type of Command
                         Line Interface.";
                }
                enum "TL1" {
                    description
                        "This defines the language as a type of
                         Transaction Language 1.";
                }
                enum "YANG" {
                    description
                        "This defines the language as a type of YANG.";
                }
            }
            mandatory true;
            description
               "Indicates the lanaguage used for this object instance.";
        }
        leaf supa-encoded-clause-response {
            type boolean;
            description
               "If present, this represents the success or failure
                of the last invocation of this clause.";
        }
        description
           "This class refines the behavior of the supa-policy-clause
            by encoding the contents of the clause into the attributes
            of this object. This enables clauses that are not based on
            other SUPA objects to be modeled.";
    }

    container supa-encoding-clause-container {
        description
            "This is a container to collect all object instances of
             type SUPAEncodedClause.";
        list supa-encoding-clause-list {
            key supa-policy-ID;
            uses supa-encoded-clause-type;
            description
                "List of all instances of supa-encoding-clause-type.
                 If a module defines subclasses of the encoding clause,
                 those will be stored in a separate container.";
        }
    }
```

```
    identity POLICY-COMPONENT-TERM-TYPE {
        base POLICY-COMPONENT-DECORATOR-TYPE;
        description
            "The identity corresponding to a
             SUPAPolicyComponentDecorator object instance.";
    }

    grouping supa-policy-term-type {
        uses supa-policy-component-decorator-type {
            refine entity-class {
               default POLICY-COMPONENT-TERM-TYPE;
            }
        }
        leaf supa-policy-term-is-negated {
            type boolean;
            description
               "If the value of this attribute is true, then
                this particular term is negated.";
        }
        description
           "This is the superclass of all SUPA policy objects that are
            used to test or set the value of a variable.";
    }

    identity POLICY-COMPONENT-VARIABLE-TYPE {
        base POLICY-COMPONENT-TERM-TYPE;
        description
            "The identity corresponding to a SUPAPolicyVariable
             object instance.";
    }

    grouping supa-policy-variable-type {
        uses supa-policy-term-type {
            refine entity-class {
               default POLICY-COMPONENT-VARIABLE-TYPE;
            }
        }
        leaf supa-policy-variable-name {
             type string;
             description
                "A human-readable name for this policy variable.";
         }
        description
           "This is one formulation of a SUPA Policy Clause. It uses
            an object, defined in the SUPA hierarchy, to represent the
            variable portion of a SUPA Policy Clause. The attribute
            defined by the supa-policy-variable-name specifies an
            attribute whose content should be compared to a value,
            which is typically specified by supa-policy-value-type.";
    }
```

```
    container supa-policy-variable-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyVariable.";
        list supa-policy-variable-list {
            key supa-policy-ID;
            uses supa-policy-variable-type;
            description
                "List of all instances of supa-policy-variable-type.
                 If a module defines subclasses of this class,
                 those will be stored in a separate container.";
        }
    }

    identity POLICY-COMPONENT-OPERATOR-TYPE {
        base POLICY-COMPONENT-TERM-TYPE;
        description
            "The identity corresponding to a SUPAPolicyOperator
             object instance.";
    }

    grouping supa-policy-operator-type {
        uses supa-policy-term-type {
            refine entity-class {
               default POLICY-COMPONENT-OPERATOR-TYPE;
            }
        }
        leaf supa-policy-value-op-type {
            type enumeration {
                enum "unknown" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
                enum "greater than" {
                    description
                        "A greater-than operator.";
                }
                enum "greater than or equal to" {
                    description
                        "A greater-than-or-equal-to operator.";
                }
                enum "less than" {
                    description
                        "A less-than operator.";
                }
                enum "less than or equal to" {
                    description
                        "A less-than-or-equal-to operator.";
                }
```

```
                enum "equal to" {
                    description
                        "An equal-to operator.";
                }
                enum "not equal to"{
                    description
                        "A not-equal-to operator.";
                }
                enum "IN" {
                    description
                        "An operator that determines whether a given
                         value matches any of the specified values.";
                }
                enum "NOT IN" {
                    description
                        "An operator that determines whether a given
                         value does not match any of the specified
                         values.";
                }
                enum "SET" {
                    description
                        "An operator that makes the value of the
                         result equal to the input value.";
                }
                enum "CLEAR"{
                    description
                        "An operator that deletes the value of the
                         specified object.";
                }
                enum "BETWEEN" {
                    description
                        "An operator that determines whether a given
                         value is within a specified range of values.";
                }
            }
            mandatory true;
            description
                "The type of operator used to compare the variable
                 and value portions of this SUPA Policy Clause.";
        }
        description
           "This is one formulation of a SUPA Policy Clause. It uses
            an object, defined in the SUPA hierarchy, to represent the
            operator portion of a SUPA Policy Clause. The attribute
            defined by the supa-policy-op-type specifies an attribute
            whose content defines the type of operator used to compare
            the variable and value portions of this policy clause.";
    }
```

```
    container supa-policy-operator-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyOperator.";
        list supa-policy-operator-list {
            key supa-policy-ID;
            uses supa-policy-operator-type;
            description
                "List of all instances of supa-policy-operator-type.
                 If a module defines subclasses of this class,
                 those will be stored in a separate container.";
        }
    }

    identity POLICY-COMPONENT-VALUE-TYPE {
        base POLICY-COMPONENT-TERM-TYPE;
        description
            "The identity corresponding to a SUPAPolicyValue
             object instance.";
    }

    grouping supa-policy-value-type {
        uses supa-policy-term-type {
            refine entity-class {
               default POLICY-COMPONENT-VALUE-TYPE;
            }
        }
        leaf-list supa-policy-value-content {
            type string;
            description
               "The content of the value portion of this SUPA Policy
                Clause. The data type of the content is specified in
                the supa-policy-value-encoding.";
        }
        leaf supa-policy-value-encoding {
            type policy-data-type-encoding-list;
            description
                "The data type of the supa-policy-value-content.";
        }
        description
           "This is one formulation of a SUPA Policy Clause. It uses
            an object, defined in the SUPA hierarchy, to represent the
            value portion of a SUPA Policy Clause. The attribute
            defined by the supa-policy-value-content specifies an
            attribute whose content should be compared to a variable,
            which is typically specified by supa-policy-variable-type.";
    }
```

```
    container supa-policy-value-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyValue.";
        list supa-policy-value-list {
            key supa-policy-ID;
            uses supa-policy-value-type;
            description
                "List of all instances of supa-policy-value-type.
                 If a module defines subclasses of this class,
                 those will be stored in a separate container.";
        }
    }

    identity POLICY-GENERIC-DECORATED-TYPE {
        base POLICY-COMPONENT-DECORATOR-TYPE;
        description
            "The identity corresponding to a
             SUPAGenericDecoratedComponent object instance.";
    }

    grouping supa-policy-generic-decorated-type {
        uses supa-policy-component-decorator-type {
            refine entity-class {
               default POLICY-GENERIC-DECORATED-TYPE;
            }
        }
        leaf-list supa-policy-generic-decorated-content {
            type string;
            description
               "The content of this SUPA Policy Clause. The data type
                of this attribute is specified in the
                supa-policy-generic-decorated-encoding.";
        }
        leaf supa-policy-generic-decorated-encoding {
            type policy-data-type-encoding-list;
            description
                "The data type of the
                 supa-policy-generic-decorated-content attribute.";
        }
        description
           "This object enables a generic object to be defined and
            used as a decorator in a SUPA Policy Clause.
            This should not be confused with the SUPAEncodedClause
            class. This class represents a single, atomic,
            vendor-specific object that defines a portion of a SUPA
            Policy Clause, whereas a SUPA Policy Encoded Clause
            represents the entire policy clause.";
    }
```

```
    container supa-policy-generic-decorated-container {
        description
            "This is a container to collect all object instances of
             type SUPAGenericDecoratedComponent.";
        list supa-encoding-clause-list {
            key supa-policy-ID;
            uses supa-policy-generic-decorated-type;
            description
                "List of all instances of
                 supa-policy-generic-decorated-type. If a module
                 defines subclasses of this class, those will be
                 stored in a separate container.";
        }
    }

    identity POLICY-COLLECTION {
        base POLICY-COMPONENT-DECORATOR-TYPE;
        description
            "The identity corresponding to a SUPAPolicyCollection
             object instance.";
    }

    grouping supa-policy-collection {
        uses supa-policy-component-decorator-type {
            refine entity-class { default POLICY-COLLECTION;
            }
        }
        leaf-list supa-policy-collection-content {
            type string;
            description
               "The content of this collection object. The data type
                is specified in supa-policy-collection-encoding.";
        }
        leaf supa-policy-collection-encoding {
            type enumeration {
                enum "undefined" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
                enum "by regex" {
                    description
                        "This defines the data type of the content of
                         this collection instance to be a regular
                         expression that contains all or part of a
                         string to match the class name of the object
                         that is to be collected by this instance of a
                         SUPAPolicyCollection class.";
                }
```

```
                enum "by URI" {
                    description
                        "This defines the data type of the content of
                         this collection instance to be a Uniform
                         Resource Identifier. It identifies the object
                         instance that is to be collected by this
                         instance of a SUPAPolicyCollection class.";
                }
            }
            mandatory true;
            description
                "The data type of the supa-policy-collection-content.";
        }
        leaf supa-policy-collection-function {
            type enumeration {
                enum "undefined" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
                enum "event collection" {
                    description
                        "This collection contains objects that are used
                         to populate the event clause of a
                         SUPA Policy.";
                }
                enum "condition collection" {
                    description
                        "This collection contains objects that are used
                         to populate the condition clause of a
                         SUPA Policy.";
                }
                enum "action collection" {
                     description
                        "This collection contains objects that are used
                         to populate the action clause of a
                         SUPA Policy.";
                }
              enum "logic collection" {
                    description
                        "This collection contains objects that define
                         logic for processing a SUPA Policy.";
                }
            }
            description
                "Defines how this collection instance is to be used.";
        }
```

```
        leaf supa-policy-collection-is-ordered {
            type boolean;
            description
                "If the value of this leaf is true, then all elements
                 in this collection are ordered.";
        }
        leaf supa-policy-collection-type {
            type enumeration {
                enum "undefined" {
                    description
                       "This may be used as an initialization and/or
                        an error state.";
                }
                enum "set" {
                    description
                       "An unordered collection of elements that MUST
                        NOT have duplicates.";
                }
                enum "bag" {
                    description
                       "An unordered collection of elements that MAY
                        have duplicates.";
                }
                enum "dictionary" {
                    description
                       "A list of values that is interpreted as a set
                        of pairs, with the first entry of each pair
                        interpreted as a dictionary key, and the
                        second entry interpreted as a value for that
                        key. As a result, collections using this value
                        of supa-policy-collection-type MUST have
                        supa-policy-collection-is-ordered set to true.";
                }
            }
            mandatory true;
            description
                "The type of the supa-policy-collection.";
        }
        description
           "This enables a collection of arbitrary objects to be
            defined and used in a SUPA Policy Clause.
            This should not be confused with the SUPAEncodedClause
            class. This class represents a single, atomic, object that
            defines a portion of a SUPA Policy Clause, whereas a SUPA
            Policy Encoded Clause represents the entire policy clause.";
    }
```

```
    container supa-policy-collection-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyCollection.";
        list supa-policy-collection-list {
            key supa-policy-ID;
            uses supa-policy-collection;
            description
                "List of all instances of supa-policy-collection.
                 If a module defines subclasses of this class,
                 those will be stored in a separate container.";
        }
    }

    identity POLICY-STRUCTURE-TYPE {
        base POLICY-OBJECT-TYPE;
        description
            "The identity corresponding to a SUPAPolicyStructure
             object instance.";
    }

    grouping supa-policy-structure-type {
        uses supa-policy-object-type {
            refine entity-class {
                default POLICY-STRUCTURE-TYPE;
            }
        }
        leaf supa-policy-admin-status {
            type enumeration {
                enum "unknown" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
                enum "enabled" {
                    description
                        "This SUPA Policy Rule has been
                         administratively enabled.";
                }
                enum "disabled" {
                    description
                        "This SUPA Policy Rule has been
                         administratively disabled.";
                }
                enum "in test" {
                    description
                        "This SUPA Policy Rule has been
                         administratively placed into test mode, and
                         SHOULD NOT be used as part of an operational
                         policy rule.";
                }
            }
```

```
            mandatory true;
            description
                "The current admnistrative status of this SUPA POLICY
                 Rule.";
        }
        leaf supa-policy-continuum-level {
            type uint32;
            description
                "This is the current level of abstraction of this
                 particular SUPA Policy Rule.";
        }
        leaf supa-policy-deploy-status {
            type enumeration {
                enum "undefined" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
                enum "deployed and enabled" {
                    description
                        "This SUPA Policy Rule has been deployed and
                         enabled.";
                }
                enum "disabled" {
                    description
                        "This SUPA Policy Rule has been
                         administratively disabled.";
                }
                enum "in test" {
                    description
                        "This SUPA Policy Rule has been
                         administratively placed into test mode, and
                         SHOULD NOT be used as part of an operational
                         policy rule.";
                }
            }
            mandatory true;
            description
                "This is the current level of abstraction of this
                 particular SUPA Policy Rule.";
        }
        leaf supa-policy-exec-status {
            type enumeration {
                enum "undefined" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
```

```
                enum "operational success" {
                    description
                        "This SUPA Policy Rule has been executed in
                         operational mode, and produced no errors.";
                }
                enum "operational failure" {
                    description
                        "This SUPA Policy Rule has been executed in
                         operational mode, but has produced at least
                         one error.";
                }
                enum "currently in operation" {
                    description
                        "This SUPA Policy Rule is currently still
                         executing in operational mode.";
                }
                enum "ready" {
                    description
                        "This SUPA Policy Rule is ready to be
                         executed in operational mode.";
                }
                enum "test success" {
                    description
                        "This SUPA Policy Rule has been executed in
                         test mode, and produced no errors.";
                }
                enum "test failure" {
                    description
                        "This SUPA Policy Rule has been executed in
                         test mode, but has produced at least
                         one error.";
                }
                enum "currently in test" {
                    description
                        "This SUPA Policy Rule is currently still
                         executing in test mode.";
                }
            }
            mandatory true;
            description
                "This is the current level of abstraction of this
                 particular SUPA Policy Rule.";
        }
        leaf supa-policy-exec-fail-strategy {
            type enumeration {
                enum "undefined" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
```

```
                enum "rollback all" {
                    description
                        "This means that execution of this SUPA
                         Policy Rule is stopped, rollback of all
                         actions (whether successful or not) is
                         attempted, and all SUPA Policy Rules that
                         otherwise would have executed are ignored.";
                }
                enum "rollback failure" {
                    description
                        "This means that execution of this SUPA
                         Policy Rule is stopped, and rollback is
                         attempted for only the SUPA Policy Rule that
                         failed to execute correctly.";
                }
                enum "stop execution" {
                    description
                        "This means that execution of this SUPA Policy
                         Rule SHOULD be stopped.";
                }
                enum "ignore" {
                    description
                        "This means that any failures produced by this
                         SUPA Policy Rule SHOULD be ignored.";
                }
            }
            mandatory true;
            description
                "This defines what actions, if any, should be taken by
                 this particular SUPA Policy Rule if it fails to
                 execute correctly. Some implementations may not be
                 able to accommodate the rollback failure option;
                 hence, this option may be skipped.";
        }
        leaf-list supa-has-policy-source-agg {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-SOURCE-ASSOC)";
            description
                "The SUPAPolicyStructure (i.e., the type of SUPA
                 Policy Rule) object instance that aggregates this set
                 set of SUPAPolicySource object instances. This
                 defines the object class that this instance-identifier
                 points to.";
        }
        leaf-list supa-has-policy-target-agg {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-TARGET-ASSOC)";
```

```
            description
                 "This represents the aggregation of Policy Target
                 objects by this particular SUPA Policy Rule. It is
                 the SUPAPolicyStructure object instance that
                 aggregates this set of SUPAPolicyTarget object
                 instances. This defines the object class that
                 this instance-identifier points to.";
        }
        leaf-list supa-has-policy-clause-agg {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-CLAUSE-ASSOC)";
            description
                "The SUPAPolicyStructure object instance that
                 aggregates this set of SUPAPolicyClause object
                 instances. This defines the object class that
                 this instance-identifier points to.";
        }
        leaf-list supa-has-policy-exec-action-assoc-src-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-EXEC-ACTION-ASSOC)";
            description
                "This associates a SUPAPolicyStructure (i.e., a SUPA
                 Policy Rule) object instance to zero or more SUPA
                 Policy Actions to be used to correct errors caused if
                 this SUPA Policy Rule does not execute correctly.";
        }
        leaf-list supa-has-policy-exec-action-assoc-dst-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-EXEC-ACTION-ASSOC)";
            min-elements 1;
            description
                "The set of zero or more SUPA Policy Actions to be used
                 by this particular SUPAPolicyStructure (i.e., SUPA
                 Policy Rule to correct errors caused if this SUPA
                 Policy Rule does not execute correctly.";
        }
        description
           "A superclass for all objects that represent different types
            of Policy Rules. Currently, this is limited to a single
            type - the event-condition-action (ECA) policy rule.
            A SUPA Policy may be an individual policy, or a set of
            policies. This is supported by applying the composite
            pattern to this class.";
    }
```

```
    identity POLICY-SOURCE-TYPE {
        base POLICY-OBJECT-TYPE;
        description
            "The identity corresponding to a SUPAPolicySource
             object instance.";
    }

    grouping supa-policy-source-type {
        uses supa-policy-object-type {
            refine entity-class {
                default POLICY-SOURCE-TYPE;
            }
        }
        leaf-list supa-has-policy-source-part {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                   SUPA-HAS-POLICY-SOURCE-ASSOC)";
            description
                "This represents the aggregation of one or more SUPA
                 Policy Source objects to this particular SUPA Policy
                 Rule object. In other words, it is the set of
                 SUPAPolicySource object instances that are aggregated
                 by this SUPAPolicyStructure (i.e., this SUPA Policy
                 Rule). This defines the object class that this
                 instance-identifier points to.";
        }
        description
           "This object defines a set of managed entities that
            authored, or are otherwise responsible for, this SUPA
            Policy Rule. Note that a SUPA Policy Source does not
            evaluate or execute SUPAPolicies. Its primary use is for
            auditability and the implementation of deontic and/or
            alethic logic.";
    }

    identity POLICY-TARGET-TYPE {
        base POLICY-OBJECT-TYPE;
        description
            "The identity corresponding to a SUPAPolicyTarget
             object instance.";
    }

    grouping supa-policy-target-type {
        uses supa-policy-object-type {
            refine entity-class {
                default POLICY-TARGET-TYPE;
            }
        }
```

```
        leaf-list supa-has-policy-target-part {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-TARGET-ASSOC)";
            description
                "This represents the aggregation of one or more SUPA
                 Policy Target objects to this particular SUPA Policy
                 Rule object. In other words, it is the set of
                 SUPAPolicyTarget object instances that are aggregated
                 by this SUPAPolicyStructure (i.e., this SUPA Policy
                 Rule). This defines the object class that this
                 instance-identifier points to.";
        }
        description
           "This object defines a set of managed entities that a
            SUPA Policy Rule is applied to.";
    }

    identity POLICY-METADATA-TYPE {
        description
            "The identity corresponding to a SUPAPolicyMetadata
             object instance.";
    }

    grouping supa-policy-metadata-type {
        leaf supa-policy-metadata-id {
            type string;
            mandatory true;
            description
                "This represents part of the object identifier of an
                 instance of this class. It defines the content of the
                 object identifier.";
        }
        leaf entity-class {
            type identityref {
                base POLICY-METADATA-TYPE;
            }
            default POLICY-METADATA-TYPE;
            description
               "The identifier of the class of this grouping.";
        }
        leaf supa-policy-metadata-id-encoding {
            type policy-data-type-id-encoding-list;
            mandatory true;
            description
                "This represents part of the object identifier of an
                 instance of this class. It defines the format of the
                 object identifier.";
        }
```

```
        leaf supa-policy-metadata-description {
            type string;
            description
                "This contains a free-form textual description of this
                 metadata object.";
        }
        leaf supa-policy-metadata-name {
            type string;
            description
                "This contains a human-readable name for this
                 metadata object.";
        }
        leaf-list supa-has-policy-metadata-part {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-METADATA-ASSOC)";
            description
                "This represents the set of SUPAPolicyMetadata object
                 instances that are aggregated by this SUPAPolicyObject
                 object instance (i.e., this is the set of policy
                 metadata aggregated by this SUPAPolicyObject). As
                 there are attributes on this association, the
                 instance-identifier MUST point to an instance using
                 the grouping supa-has-policy-metadata-detail (which
                 includes the subclasses of the association class).";
        }
        leaf supa-policy-metadata-decorator-part {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  SUPA-HAS-POLICY-COMPONENT-DECORATOR-ASSOC)";
            mandatory true;
            description
               "This object implements the decorator pattern, which is
                applied to SUPA metadata objects. This enables all or
                part of one or more metadata objects to wrap another
                concrete metadata object.";
        }
        description
           "This is the superclass of all metadata classes. Metadata
            is information that describes and/or prescribes the
            characteristics and behavior of another object that is
            not an inherent, distinguishing characteristics or
            behavior of that object.";
    }

    identity POLICY-METADATA-CONCRETE-TYPE {
        base POLICY-METADATA-TYPE;
        description
            "The identity corresponding to a SUPAPolicyConcreteMetadata
             object instance.";
    }
```

```
    grouping supa-policy-concrete-metadata-type {
        uses supa-policy-metadata-type {
            refine entity-class {
                default POLICY-METADATA-TYPE;
            }
        }
        leaf supa-policy-metadata-valid-period-end {
            type yang:date-and-time;
            description
                "This defines the ending date and time that this
                 metadata object is valid for.";
        }
        leaf supa-policy-metadata-valid-period-start {
            type yang:date-and-time;
            description
                "This defines the starting date and time that this
                 metadata object is valid for.";
        }
        description
           "This is a concrete class that will be wrapped by concrete
            instances of the SUPA Policy Metadata Decorator class. It
            can be viewed as a container for metadata that will be
            attached to a subclass of SUPA Policy Object. It may
            contain all or part of one or more metadata subclasses.";
    }

    container supa-policy-concrete-metadata-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyConcreteMetadata.";
        list supa-policy-concrete-metadata-list {
            key supa-policy-metadata-id;
            uses supa-policy-concrete-metadata-type;
            description
                "A list of all supa-policy-metadata instances in the
                 system.";
        }
    }

    identity POLICY-METADATA-DECORATOR-TYPE {
        base POLICY-METADATA-TYPE;
        description
            "The identity corresponding to a
             SUPAPolicyMetadataDecorator object instance.";
    }
```

```
    grouping supa-policy-metadata-decorator-type {
        uses supa-policy-metadata-type {
            refine entity-class {
                default POLICY-METADATA-DECORATOR-TYPE;
            }
        }
        leaf-list supa-policy-metadata-decorator-agg {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                   SUPA-HAS-POLICY-COMPONENT-DECORATOR-ASSOC)";
            max-elements 1;
            description
                "This represents the decorator pattern being applied to
                 metadata. This is the aggregate part (i.e., the
                 concrete subclass of the SUPAPolicyMetadataDecorator
                 class that wraps a concrete subclass of
                 SUPAPolicyMetadata; currently, the only such class is
                 SUPAPolicyConcreteMetadata).";
        }
        description
           "This object implements the decorator pattern, which is
            applied to SUPA metadata objects. This enables all or part
            of one or more metadata objects to wrap another concrete
            metadata object.";
    }

    identity POLICY-METADATA-DECORATOR-ACCESS-TYPE {
        base POLICY-METADATA-DECORATOR-TYPE;
        description
            "The identity corresponding to a
             SUPAPolicyAccessMetadataDef object instance.";
    }

    grouping supa-policy-metadata-decorator-access-type {
        uses supa-policy-metadata-decorator-type {
            refine entity-class {
                default POLICY-METADATA-DECORATOR-ACCESS-TYPE;
            }
        }
        leaf supa-policy-metadata-access-priv-def {
            type enumeration {
                enum "undefined" {
                    description
                        "This may be used as an initialization and/or
                         an error state.";
                }
                enum "read only" {
                    description
                        "This defines access as read only for ALL SUPA
                         Policy object instances that are adorned with
                         this metadata object.";
                }
```

```
                enum "read write" {
                    description
                        "This defines access as read and/or write for
                         ALL SUPA Policy object instances that are
                         adorned with this metadata object.";
                }
                enum "specified by MAC" {
                    description
                        "This defines access as defined by an external
                         Mandatory Access Control model. The name and
                         location of this model are specified in the
                         supa-policy-metadata-access-priv-model-name
                         and supa-policy-metadata-access-priv-model-ref
                         attributes of this metadata object.";
                }
                enum "specified by DAC" {
                    description
                        "This defines access as defined by an external
                         Discretionary Access Control model. The name
                         and location of this model are specified in the
                         supa-policy-metadata-access-priv-model-name
                         and supa-policy-metadata-access-priv-model-ref
                         attributes of this metadata object.";
                }
                enum "specified by RBAC" {
                    description
                        "This defines access as defined by an external
                         Role Based Access Control model. The name
                         and location of this model are specified in the
                         supa-policy-metadata-access-priv-model-name
                         and supa-policy-metadata-access-priv-model-ref
                         attributes of this metadata object.";
                }
                enum "specified by ABAC" {
                    description
                        "This defines access as defined by an external
                         Attribute Based Access Control model. The name
                         and location of this model are specified in the
                         supa-policy-metadata-access-priv-model-name
                         and supa-policy-metadata-access-priv-model-ref
                         attributes of this metadata object.";
                }
                enum "specified by custom" {
                    description
                        "This defines access as defined by an external
                         Custom Access Control model. The name and
                         location of this model are specified in the
                         supa-policy-metadata-access-priv-model-name
                         and supa-policy-metadata-access-priv-model-ref
                         attributes of this metadata object.";
                }
            }
```

```
            description
                "This defines the type of access control model that is
                 used by this object instance.";
        }
        leaf supa-policy-metadata-access-priv-model-name {
            type string;
            description
                "This contains the name of the access control model
                 being used. If the value of the
                 supa-policy-metadata-access-priv-model-ref is 0-2,
                 then the value of this attribute is not applicable.
                 Otherwise, the text in this class attribute should be
                 interpreted according to the value of the
                 supa-policy-metadata-access-priv-model-ref class
                 attribute.";
        }
        leaf supa-policy-metadata-access-priv-model-ref {
            type enumeration {
                enum "undefined" {
                    description
                        "This can be used for either initialization
                        or for signifying an error.";
                }
                enum "URI"  {
                    description
                        "The clause is referenced by this URI.";
                }
                enum "GUID"  {
                    description
                        "The clause is referenced by this GUID.";
                }
                enum "UUID"  {
                    description
                        "The clause is referenced by this UUID.";
                }
                enum "FQDN"  {
                    description
                        "The clause is referenced by this FQDN.";
                }
            }
            description
                "This defines the data type of the
                 supa-policy-metadata-access-priv-model-name
                 attribute.";
        }
        description
            "This is a concrete class that defines metadata for access
             control information that can be added to a SUPA Policy
             object. This is done using the SUPAHasPolicyMetadata
             aggregation.";
    }
```

```
    container supa-policy-metadata-decorator-access-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyAccessMetadataDef.";
        list supa-policy-metadata-decorator-access-list {
            key supa-policy-metadata-id;
            uses supa-policy-metadata-decorator-type;
            description
                "A list of all supa-policy-metadata-decorator-access
                instances in the system.  Instances of subclasses
                will be in a separate list.";
        }
    }

    identity POLICY-METADATA-DECORATOR-VERSION-TYPE {
        base POLICY-METADATA-DECORATOR-TYPE;
        description
            "The identity corresponding to a
             SUPAPolicyVersionMetadataDef object instance.";
    }

    grouping supa-policy-metadata-decorator-version-type {
        uses supa-policy-metadata-decorator-type {
            refine entity-class {
                default POLICY-METADATA-DECORATOR-VERSION-TYPE;
            }
        }
        leaf supa-policy-metadata-version-major {
            type string;
            description
                "This contains a string (typically representing an
                 integer in the overall version format) that indicates
                 a significant increase in functionality is present in
                 this version.";
        }
        leaf supa-policy-metadata-version-minor {
            type string;
            description
                "This contains a string (typically representing an
                 integer in the overall version format) that indicates
                 that this release contains a set of features and/or bug
                 fixes that collectively do not warrant incrementing the
                 supa-policy-metadata-version-major attribute.";
        }
        leaf supa-policy-metadata-version-rel-type {
            type enumeration {
                enum "undefined" {
                    description
                        "This can be used for either initialization
                        or for signifying an error.";
                }
```

```
                enum "internal" {
                    description
                        "This indicates that this version should only
                         be used for internal (development) purposes.";
                }
                enum "alpha" {
                    description
                        "This indicates that this version is considered
                         to be alpha quality.";
                }
                enum "beta" {
                    description
                        "This indicates that this version is considered
                         to be beta quality.";
                }
                enum "release candidate" {
                    description
                        "This indicates that this version is considered
                         to be a candidate for full production.";
                }
                enum "release production" {
                    description
                        "This indicates that this version is considered
                         to be ready for full production.";
                }
                enum "maintenance" {
                    description
                        "This indicates that this version is considered
                         to be for maintenance purposes.";
                }
            }
            description
                "This defines the type of this version’s release.";
        }
        leaf supa-policy-metadata-version-rel-type-num {
            type string;
            description
                "This contains a string (typically representing an
                 integer in the overall version format) that indicates
                 a significant increase in functionality is present in
                 this version.";
        }
        description
            "This is a concrete class that defines metadata for version
             control information that can be added to a SUPA Policy
             object. This is done using the SUPAHasPolicyMetadata
             aggregation.";
    }
```

```
    container supa-policy-metadata-decorator-version-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyVersionMetadataDef.";
        list supa-policy-metadata-decorator-version-list {
            key supa-policy-metadata-id;
            uses supa-policy-metadata-decorator-type;
            description
                "A list of all supa-policy-metadata-decorator-version
                instances in the system. Instances of subclasses
                will be in a separate list.";
        }
    }

    identity SUPA-HAS-POLICY-METADATA-ASSOC {
        description
            "The identity corresponding to a
             SUPAHasPolicyMetadataDetail association class
             object instance.";
    }

    grouping supa-has-policy-metadata-detail {
        leaf supa-policy-ID {
            type string;
            description
                "This is a globally unique ID for this association
                 instance in the overall policy system.";
        }
        leaf entity-class {
            type identityref {
                base SUPA-HAS-POLICY-METADATA-ASSOC;
            }
            default SUPA-HAS-POLICY-METADATA-ASSOC;
            description
                "The identifier of the class of this assocation.";
        }
        leaf supa-has-policy-metadata-object-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  POLICY-OBJECT-TYPE)";
            description
                "This is a reference from the SUPAPolicyObject object
                 instance that is aggregating SUPAPolicyMetadata object
                 instances using the SUPAHasPolicyMetadata aggregation.
                 This SUPAPolicyMetadataDetail association class is
                 used to define part of the semantics of the
                 SUPAHasPolicyMetadata aggregation. For example, it can
                 define which SUPAPolicyMetadata object instances can
                 be aggregated by this particular SUPAPolicyObject
                 object instance.";
        }
```

```
        leaf supa-has-policy-metadata-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  POLICY-METADATA-TYPE)";
            description
                "This is a reference from the SUPAPolicyMetadata object
                 instance(s) that are being aggregated by this
                 SUPAPolicyObject object instance using the
                 SUPAHasPolicyMetadata aggregation. The class
                 SUPAPolicyMetadataDetail association class is used to
                 define part of the semantics of the
                 SUPAHasPolicyMetadata aggregation. For example, it can
                 define which SUPAPolicyMetadata object instances can
                 be aggregated by this particular SUPAPolicyObject
                 object instance.";
        }
        leaf supa-policy-metadata-detail-is-applicable {
            type boolean;
            description
                "This attributes controls whether the associated
                 metadata is currently considered applciable to this
                 policy object; this enables metadata to be turned on
                 and off when needed without disturbing the structure
                 of the object that the metadata applies to.";
        }
        leaf-list supa-policy-metadata-detail-constraint {
            type string;
            description
                "A list of constraints, expressed as strings
                in the language defined by the
                supa-policy-metadata-detail-encoding.";
        }
        leaf supa-policy-metadata-detail-encoding {
            type string;
            description
                "The langauge used to encode the constraints
                relevant to the relationship between the metadata
                and the underlying policy object.";
        }
        description
            "This is a concrete association class that defines the
             semantics of the SUPAPolicyMetadata aggregation. This
             enables the attributes and relationships of the
             SUPAPolicyMetadataDetail class to be used to constrain
             which SUPAPolicyMetadata objects can be aggregated by
             this particular SUPAPolicyObject instance.";
    }
```

```
    container supa-policy-metadata-detail-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyMetadataDetail.";
        list supa-policy-metadata-detail-list {
            key supa-policy-ID;
            uses supa-has-policy-metadata-detail;
            description
                "This is a list of all supa-policy-metadata-detail
                instances in the system.  Instances of subclasses
                will be in a separate list.
                Note that this policy is made concrete for exemplary
                purposes.  To be useful, it almost certainly needs
                refinement.";
        }
    }

    identity SUPA-HAS-POLICY-COMPONENT-DECORATOR-ASSOC {
        description
            "The identity corresponding to a SUPAHasMetadataDecorator
             association class object instance.";
    }

    grouping supa-has-decorator-policy-component-detail {
        leaf supa-policy-ID {
            type string;
            description
                "This is a globally unique ID for this association
                 instance in the overall policy system.";
        }
        leaf entity-class {
            type identityref {
                base SUPA-HAS-POLICY-COMPONENT-DECORATOR-ASSOC;
            }
            default SUPA-HAS-POLICY-COMPONENT-DECORATOR-ASSOC;
            description
                "The identifier of the class of this assocation.";
        }
        leaf supa-policy-component-decorator-ptr {
            type instance-identifier;
             must "derived-from-or-self (deref(.)/entity-class,
                   SUPA-POLICY-COMPONENT-DECORATOR-TYPE)";
            description
                "This associates the SUPAPolicyComponentStructure
                 object instance participating in a
                 SUPAHasDecoratedPolicyComponent aggregation to the
                 SUPAHasDecoratedPolicyComponentDetail association
                 class that provides the semantics of this aggregation.
                 This defines the object class that this
                 instance-identifier points to.";
        }
```

```
        leaf supa-policy-component-ptr {
            type instance-identifier;
             must "derived-from-or-self (deref(.)/entity-class,
                   SUPA-POLICY-COMPONENT-TYPE)";
            description
                "This associates the SUPAPolicyComponentDecorator
                 object instance participating in a
                 SUPAHasDecoratedPolicyComponent aggregation to the
                 SUPAHasDecoratedPolicyComponentDetail association
                 class that provides the semantics of this aggregation.
                 This defines the object class that this
                 instance-identifier points to.";
        }
        leaf-list supa-has-decorator-constraint {
            type string;
            description
                "A constraint expression applying to this association
                 between a policy component decorator and the
                 decorated component.";
        }
        leaf supa-has-decorator-constraint-encoding {
            type string;
            description
                "The language in which the constraints on the
                 policy component-decoration is expressed.";
        }
        description
            "This is a concrete association class that defines the
             semantics of the SUPAHasDecoratedPolicyComponent
             aggregation. The purpose of this class is to use the
             Decorator pattern to detemine which
             SUPAPolicyComponentDecorator object instances, if any,
             are required to augment the functionality of the concrete
             subclass of SUPAPolicyClause that is being used.";
    }

    container supa-policy-component-decorator-detail-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyComponentDecoratorDetail.";
        list supa-policy-component-decorator-detail-list {
            key supa-policy-ID;
            uses supa-has-decorator-policy-component-detail;
            description
                "This is a list of all
                 supa-policy-component-decorator-details.";
        }
    }
```

```
    identity SUPA-HAS-POLICY-SOURCE-ASSOC {
        description
            "The identity corresponding to a SUPAHasPolicySource
             association class object instance.";
    }

    grouping supa-has-policy-source-detail {
        leaf supa-policy-ID {
            type string;
            description
                "This is a globally unique ID for this association
                 instance in the overall policy system.";
        }
        leaf entity-class {
            type identityref {
                base SUPA-HAS-POLICY-SOURCE-ASSOC;
            }
            default SUPA-HAS-POLICY-SOURCE-ASSOC;
            description
                "The identifier of the class of this assocation.";
        }
        leaf supa-policy-source-structure-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  POLICY-STRUCTURE-TYPE)";
            description
                "This associates the SUPAPolicyStructure object
                 instance participating in a SUPAHasPolicySource
                 aggregation to the SUPAHasPolicySourceDetail
                 association class that provides the semantics of
                 this aggregation. This defines the object class
                 that this instance-identifier points to.";
        }
        leaf supa-policy-source-ptr {
            type instance-identifier;
             must "derived-from-or-self (deref(.)/entity-class,
                   SUPA-POLICY-SOURCE-TYPE)";
            description
                "This associates the SUPAPolicySource object
                 instance participating in a SUPAHasPolicySource
                 aggregation to the SUPAHasPolicySourceDetail
                 association class that provides the semantics of
                 this aggregation. This defines the object class
                 that this instance-identifier points to.";
        }
        leaf supa-policy-source-is-authenticated {
            type boolean;
            description
                "If the value of this attribute is true, then this
                 SUPAPolicySource object has been authenticated by
                 this particular SUPAPolicyStructure object.";
        }
```

```
        leaf supa-policy-source-is-trusted {
            type boolean;
            description
                "If the value of this attribute is true, then this
                 SUPAPolicySource object has been verified to be
                 trusted by this particular SUPAPolicyStructure
                 object.";
        }
        description
            "This is an association class, and defines the semantics of
             the SUPAHasPolicySource aggregation. The attributes and
             relationships of this class can be used to define which
             SUPAPolicySource objects can be attached to which
             particular set of SUPAPolicyStructure objects.";
    }

    container supa-policy-source-detail-container {
    description
            "This is a container to collect all object instances of
             type SUPAPolicySourceDetail.";
        list supa-policy-source-detail-list {
            key supa-policy-ID;
            uses supa-has-policy-source-detail;
            description
                "This is a list of all supa-policy-source-detail
                 objects.";
        }
    }

    identity SUPA-HAS-POLICY-TARGET-ASSOC {
        description
            "The identity corresponding to a SUPAHasPolicyTarget
             association class object instance.";
    }

    grouping supa-has-policy-target-detail {
        leaf supa-policy-ID {
            type string;
            description
                "This is a globally unique ID for this association
                 instance in the overall policy system.";
        }
        leaf entity-class {
            type identityref {
                base SUPA-HAS-POLICY-TARGET-ASSOC;
            }
            default SUPA-HAS-POLICY-TARGET-ASSOC;
            description
                "The identifier of the class of this assocation.";
        }
```

```
        leaf supa-policy-target-structure-ptr {
            type instance-identifier;
             must "derived-from-or-self (deref(.)/entity-class,
                   POLICY-STRUCTURE-TYPE)";
            description
                "This associates the SUPAPolicyStructure object
                 instance participating in a SUPAHasPolicyTarget
                 aggregation to the SUPAHasPolicyTargetDetail
                 association class that provides the semantics of
                 this aggregation. This defines the object class
                 that this instance-identifier points to.";
        }
        leaf supa-policy-target-ptr {
            type instance-identifier;
             must "derived-from-or-self (deref(.)/entity-class,
                   SUPA-POLICY-TARGET-TYPE)";
            description
                "This associates the SUPAPolicyTarget object
                 instance participating in a SUPAHasPolicyTarget
                 aggregation to the SUPAHasPolicyTargetDetail
                 association class that provides the semantics of
                 this aggregation. This defines the object class
                 that this instance-identifier points to.";
        }
        leaf supa-policy-source-is-authenticated {
            type boolean;
            description
                "If the value of this attribute is true, then this
                 SUPAPolicyTarget object has been authenticated by
                 this particular SUPAPolicyStructure object.";
        }
        leaf supa-policy-source-is-enabled {
            type boolean;
            description
                "If the value of this attribute is true, then this
                 SUPAPolicyTarget object is able to be used as a
                 SUPAPolicyTarget. This means that it has agreed to
                 play the role of a SUPAPolicyTarget, and that it is
                 able to either process (directly or with the aid of a
                 proxy) SUPAPolicies, or receive the results of a
                 processed SUPAPolicy and apply those results to
                 itself.";
        }
        description
            "This is an association class, and defines the semantics of
             the SUPAHasPolicyTarget aggregation. The attributes and
             relationships of this class can be used to define which
             SUPAPolicyTarget objects can be attached to which
             particular set of SUPAPolicyStructure objects.";
    }
```

```
    container supa-policy-target-detail-container {
    description
            "This is a container to collect all object instances of
             type SUPAPolicyTargetDetail.";
        list supa-policy-target-detail-list {
            key supa-policy-ID;
            uses supa-has-policy-target-detail;
            description
                "This is a list of all supa-policy-target-detail
                 objects.";
        }
    }

    identity SUPA-HAS-POLICY-CLAUSE-ASSOC {
        description
            "The identity corresponding to a SUPAHasPolicyClause
             association class object instance.";
    }

    grouping supa-has-policy-clause-detail {
        leaf supa-policy-ID {
            type string;
            description
                "This is a globally unique ID for this association
                 instance in the overall policy system.";
        }
        leaf entity-class {
            type identityref {
                base SUPA-HAS-POLICY-CLAUSE-ASSOC;
            }
            default SUPA-HAS-POLICY-CLAUSE-ASSOC;
            description
                "The identifier of the class of this assocation.";
        }
        leaf supa-policy-clause-structure-ptr {
            type instance-identifier;
             must "derived-from-or-self (deref(.)/entity-class,
                   POLICY-STRUCTURE-TYPE)";
            description
                "This associates the SUPAPolicyStructure object
                 instance participating in a SUPAHasPolicyClause
                 aggregation to the SUPAHasPolicyClauseDetail
                 association class that provides the semantics of
                 this aggregation. This defines the object class
                 that this instance-identifier points to.";
        }
        leaf supa-policy-clause-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                   SUPA-POLICY-CLAUSE-TYPE)";
```

```
            description
                "This associates the SUPAPolicyClause object
                 instance participating in a SUPAHasPolicyClause
                 aggregation to the SUPAHasPolicyClauseDetail
                 association class that provides the semantics of
                 this aggregation. This defines the object class
                 that this instance-identifier points to.";
        }
        description
            "This is an association class, and defines the semantics of
             the SUPAHasPolicyClause aggregation. The attributes and
             relationships of this class can be used to define which
             SUPAPolicyTarget objects can be attached to which
             particular set of SUPAPolicyStructure objects.
             Every SUPAPolicyStructure object instance MUST aggregate
             at least one SUPAPolicyClause object instance. However,
             the converse is NOT true. For example, a SUPAPolicyClause
             could be instantiated and then stored for later use in a
             policy repository.";
    }

    container supa-policy-clause-detail-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolicyClauseDetail.";
        list supa-policy-clause-detail-list {
            key supa-policy-ID;
            uses supa-has-policy-clause-detail;
            description
                "This is a list of all supa-policy-clause-detail
                 objects.";
        }
    }

    identity SUPA-HAS-POLICY-EXEC-ACTION-ASSOC {
        description
            "The identity corresponding to a
             SUPAHasPolExecFailActionToTake association class
             object instance.";
    }

    grouping supa-has-policy-exec-action-detail {
        leaf supa-policy-ID {
            type string;
            description
                "This is a globally unique ID for this association
                 instance in the overall policy system.";
        }
        leaf entity-class {
            type identityref {
                base SUPA-HAS-POLICY-EXEC-ACTION-ASSOC;
            }
```

```
            default SUPA-HAS-POLICY-EXEC-ACTION-ASSOC;
            description
                "The identifier of the class of this assocation.";
        }
        leaf supa-policy-structure-action-src-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  POLICY-STRUCTURE-TYPE)";
            description
                "This associates the SUPAPolicyStructure object
                 instance participating in a
                 SUPAHasPolExecFailActionToTake association to the
                 SUPAHasPolExecFailActionToTakeDetail association
                 class that provides the semantics of this
                 aggregation. This defines the object class that
                 this instance-identifier points to.";
        }
        leaf supa-policy-structure-action-dst-ptr {
            type instance-identifier;
            must "derived-from-or-self (deref(.)/entity-class,
                  POLICY-STRUCTURE-TYPE)";
            description
                "This associates a SUPAPolicyAction object
                 instance participating in a
                 SUPAHasPolExecFailActionToTake association to the
                 SUPAHasPolExecFailActionToTakeDetail association
                 class that provides the semantics of this
                 aggregation. This defines the object class that
                 this instance-identifier points to.";
        }
        leaf supa-policy-exec-fail-take-action-encoding {
            type policy-data-type-id-encoding-list;
            description
                "This defines how to find the set of SUPA Policy
                 Action objects contained in each element of the
                 supa-policy-exec-fail-take-action-name attribute
                 object.";
        }
        leaf-list supa-policy-exec-fail-take-action-name {
            type string;
            description
                "This identifies the set of SUPA Policy Actions to take
                 if the SUPAPolicyStructure object that owns this
                 association failed to execute properly. The
                 interpretation of this string attribute is defined by
                 the supa-policy-exec-fail-take-action-encoding class
                 attribute.";
        }
```

```
        description
            "This is an association class, and defines the semantics of
             the SUPAHasPolExecFailTakeAction association. The
             attributes and relationships of this class can be used to
             determine which SUPA Policy Action objects are executed in
             response to a failure of the SUPAPolicyStructure object
             instance that owns this association.";
    }

    container supa-policy-exec-fail-take-action-detail-container {
        description
            "This is a container to collect all object instances of
             type SUPAPolExecFailActionToTakeDetail.";
        list supa-policy-exec-fail-take-action-detail-list {
            key supa-policy-ID;
            uses supa-has-policy-exec-action-detail;
            description
                "This is a list of all
                 supa-has-policy-exec-action-detail objects.";
        }
    }
}

<CODE ENDS>
```

6. IANA Considerations

No IANA considerations exist for this document.

7. Security Considerations

TBD

8. Acknowledgments

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: Qin Wu.

9. References

This section defines normative and informative references for this document.

9.1. Normative References

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

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

[RFC6991]   Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

9.2. Informative References

[1]   Strassner, J., Halpern, J., Coleman, J., "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-strassner-supa-generic-policy-info-model-05 March 21, 2016

Authors' Addresses

Joel Halpern
Ericsson
P. O. Box 6049
Leesburg, VA 20178
Email: joel.halpern@ericsson.com

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

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2017

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
X. Long

July 6, 2016

Northbound Interfaces for Security Policy Controllers : A Framework and Information Model

draft-kumar-i2nsf-controller-northbound-framework-00

Abstract

This document provides a framework and information model for the definition of northbound interfaces for a security policy controller. The interfaces are based on user-intent instead of vendor-specific or device-centric approaches that would require deep knowledge of vendor products and their security features. The document identifies the common interfaces needed to enforce the user-intent-based policies onto network security functions (NSFs) irrespective of how those functions are realized. The function may be physical or virtual in nature and may be implemented in networking or dedicated appliances.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Programming security policies in a network is a fairly complex task and requires very deep knowledge of the vendors' devices in order to implement a security policy. This has been the biggest challenge for both Service Providers and Enterprise, henceforth known as end-customers, to keep up-to-date with the security of their networks and assets. The challenge is amplified due to virtualization because security appliances come in both physical and virtual forms and are supplied by a variety of vendors who have their own proprietary interfaces to manage and implement the security policies on their devices.

Even if an end-customer deploys a single vendor solution across its entire network, it is difficult to manage security policies due to the complexity of network security features available in the devices. The end-customer may use a vendor-provided management system that gives some abstraction in the form of GUI and helps in provisioning and managing security policies. The single vendor approach is highly restrictive in today’s network as explained below:

- o The end-customer cannot rely on a single vendor because one vendor may not be able keep up to date with its security needs.

- o The large end-customer may have a presence across different sites and regions and that may mean it is not possible to have a single vendor solution due to technical or business reasons.

- o If and when the end-customer migrates from one vendor to another, it is not possible to migrate security policies from one management system to another without complex manual work.

- o Due to virtualization within data centers, end-customers are using physical and virtual forms of security functions with a wide variety of vendors, including open source, to control their costs.

- o The end-customer might choose various devices in the network (such as routers, switches, firewall devices, and overlay-networks) as enforcement points for security policies for any reason (such as network design simplicity, cost, most-effective place, scale and performance).

In order to provide the end-customer with a solution where they can deploy security policies across different vendors and devices whether physical or virtual, the Interface to Network Security Functions (I2NSF) working group in the IETF is defining a set of northbound interfaces.

This document discusses the requirements for these northbound interfaces and describes a framework and information model so that these interfaces can be easily used by end-customer security administrators without knowledge of specific security devices or features. We refer to this as "user-intent-based interfaces".

2. Conventions Used in this Document

BSS: Business Support System.

CMDB: Configuration Management Database.

Controller: Used interchangeably with Service Provider Security Controller or management system throughout this document.

FW: Firewall.

IDS: Intrusion Detection System.

IPS: Intrusion Protection System.

LDAP: Lightweight Directory Access Protocol.

NSF: Network Security Function, defined by [I-D.ietf-i2nsf-problem-and-use-cases].

OSS: Operation Support System.

RBAC: Role Based Access Control.

SIEM: Security Information and Event Management.

URL: Universal Resource Locator.

vNSF: Refers to NSF being instantiated on Virtual Machines.

3. Security Provisioning Framework

The IETF I2NSF working group has defined a framework for Interfaces to Network Security Functions that defines following terminology:

Client: A client could be a GUI system used by a security administrator, an OSS/BSS system used by an end-customer, or a security controller system or application in the end-customer's management system.

Client-Facing Interface: A client-facing interface is an interface used to configure and manage security a framework across the entire network independent of device-specific interface so that same interface can be used for any device from any vendor.

The "Client Facing Interface" ensures that an end-customer can deploy any device from any vendor and still be able to use same consistent interface. In essence, these interfaces give a framework to manage end-customer's security policies. Henceforth in this document, we "security policy management interface" interchangeably when we refer to these northbound interfaces.

3.1. Deployment Models for Implementing Security Policies

This document describes a framework for security policy management interfaces. This document does not describe a framework for southbound interface: those may be defined in another draft.

Traditionally, medium and larger end-customers deploy management systems to manage their security policies. This approach may not be suitable for modern datacenters that are virtualized and manage their resources using controllers.

There are two different deployment models:

a. Management without an explicit management system for control of devices and NSFs. In this deployment, the security policy controller acts as a NSF policy management system that takes information passed over the northbound policy interface and translates into data on the I2NSF southbound interface. The I2NSF interfaces are implemented by security device/function vendors. This would usually be done by having an I2NSF agent embedded in the security device or NSF. This deployment model is shown in Figure 1.

```
                            RESTful API
                    SUPA or I2NSF Policy Management
                                  ^
    Northbound                    |
    Security Policy Interface     |
    (Independent of individual    |
     NSFs, devices, and vendors)  |
                                  |
                    ------------------------------
                   |                              |
                   |  Security Policy Controller  |
                   |                              |
                    ------------------------------
                         |                 ^
    Southbound Security  |   I2NSF         |
    Capability Interface |   NSF-facing    |
    (Specific to NSFs)   |   Interface     |
                     ..............................
                         |                 |
                         v                 |


                   -------------     -------------
                  | I2NSF Agent |   | I2NSF Agent |
                  |-------------|   |-------------|
                  |             |---|             |
                  |     NSF     |   |     NSF     |
        NSFs      |             |   |             |
    (virtual       -------------\   /-------------
       and               |       \ /       |
    physical)            |        X        |
                         |       / \       |
                   -------------/   \-------------
                  | I2NSF Agent |   | I2NSF Agent |
                  |-------------|   |-------------|
                  |             |---|             |
                  |     NSF     |   |     NSF     |
                  |             |   |             |
                   -------------     -------------
```

Figure 1: Deployment without Management System

b. Management with an explicit management system for control of devices and NSFs. This model is similar to the model above except that security policy controller interacts with a dedicated management system which could either proxy I2NSF southbound interfaces or could provide a layer where security devices or

NSFs do not support an I2NSF agent to process I2NSF southbound interfaces. This deployment model is shown in Figure 2.

```
                            RESTful API
                    SUPA or I2NSF Policy Management
                                  ^
    Northbound                    |
    Security Policy Interface     |
    (Independent of individual    |
     NSFs, devices, and vendors)  |
                                  |
                    ------------------------------
                   |                              |
                   |  Security Policy Controller  |
                   |                              |
                    ------------------------------
                         |                 ^
    Southbound Security  |   I2NSF         |
    Capability Interface |   NSF-facing    |
    (Specific to NSFs)   |   Interface     |
                     ..............................
                         |                 |
                         v                 |
                    ------------------------------
                   |                              |
                   |      I2NSF Proxy Agent /     |
                   |      Management System       |
                   |                              |
                    ------------------------------
                         |                 ^
                         |  Proprietary    |
                         |  Functional     |
                         |  Interface      |
                     ..............................
                         |                 |
                         v                 |

                   -------------     -------------
                  |             |---|             |
                  |     NSF     |   |     NSF     |
        NSFs      |             |   |             |
    (virtual       -------------\   /-------------
       and               |       \ /       |
    physical)            |        X        |
                         |       / \       |
                   -------------/   \-------------
                  |             |---|             |
```

```
                  |     NSF     |   |     NSF     |
                  |             |   |             |
                   -------------     -------------
```

Figure 2: Deployment with Management System or I2NSF Proxy Agent

Although the deployment models discussed here don't necessarily affect the northbound security policy interface, they do give an overall context for defining a security policy interface based on abstraction.

3.2. Client Perspective on Security Policy Configuration and Management

In order to provide I2NSF northbound interface for security policies to client that are not specific to any vendor, device or feature implementation, it is important that security policies shall be configured and managed from a client's perspective. We refer to this as the user-intent based model since it is primarily driven by how security administrators view security policies from the deployment perspective.

The client perspective ensures that policy management is not only easy to understand for them (the actual users), but is also independent of vendor, device, and specific implementation which is the foremost goal for a northbound interface.

4. Functional Requirements for the Northbound Interface

As mentioned earlier, it is important that the northbound interface be primarily driven by user-intent which is what a client understands well. In order to define this interface, we must understand the requirements and framework used by the security administrator.

A security policy that is based on user-intent is completely agnostic of how this policy is enforced in the end-customer's network. The security controller may choose to implement such a policy on any device (router, switch, firewall) in a physical or virtual form factor. The security policy controller's implementation is outside the scope of this document and the I2NSF working group.

At a high level, the objects that are required in order to express and build the security policies fall into the following categories:

- o Multi-tenancy and RBAC for policy management

- o Policy lifecycle management

- o Policy endpoint groups
- o Policy rules
- o Policy actions
- o Third party integration
- o Telemetry data

The above categories are by no means a complete list and may not be sufficient for all use-cases and all end-customers, but should be a good start for a wide variety of use-cases in both Service Provider networks and Enterprise networks.

The following sections provide further details on the above mentioned security policies categories.

4.1. Multi-Tenancy and RBAC for Policy Management

An end-customer that uses security policies may have internal tenants and would like to have a framework wherein each tenant manages its own security policies to provide isolation across different tenants.

An end-customer may be a cloud service provider with multi-tenant deployments where each tenant is a different organization and must allow complete isolation across different tenants.

The RBAC objects and method needed to build such a framework is defined below.

Policy-Tenant: An entity that owns and manages the security policies.

Policy-User: A user within a Policy-Tenant authorized to manage security policies for that tenant.

Policy-Authorization-Role: A role assigned to a Policy-User that determines whether the user has read-write access, read-only access, or no access for certain resources.

Authentication and Authorization Scheme: There must be a scheme for a Policy-User to be authenticated and authorized to use the policy controller.

4.2. Policy Lifecycle Management

In order to provide more sophisticated security framework, there should be a mechanism to express that a policy becomes dynamically active/enforced or inactive based on either security administrator intervention or an event.

One example of dynamic policy management is when the security administrator pre-configures all the security policies, but the policies get activated/enforced or deactivated based on dynamic threats faced by the end-customer. Basically, a threat event may activate certain inactive policies, and once a new event indicates that the threat has gone away, the policies become inactive again.

The northbound interface should support the following mechanisms for policy enforcement:

Admin-Enforced: The policy, once configured, remains active/enforced until removed by the security administrator.

Time-Enforced: The policy configuration specifies the time profile that determines when policy is activated/enforced.

Event-Enforced: The policy configuration specifies the event profile that determines when policy is activated/enforced.

4.3. Policy Endpoint Groups

Typically, when the security administrator configures a security policy, the intention is to apply this policy to certain subsets of the network. The subsets may be identified based on criteria such as users, devices, and applications. We refer to such a subset of the network as a "Policy Endpoint Group".

One of the biggest challenges for a security administrator is how to make sure that security policies remain effective while constant changes are happening to the "Policy Endpoint Group" for various reasons (e.g., organizational changes). If the policy is created based on static information such as user names, application, or network subnets, then every time that this static information changes policies would need to be updated. For example, if a policy is created that allows access to an application only from the group of Human Resource users (the HR-users group), then each time the HR-users group changes, the policy needs to be updated.

Changes to policy could be highly taxing to the end-customer for various operational reasons. The policy management framework must allow "Policy Endpoint Group" to be dynamic in nature so that changes to the group (HR-users in our example) automatically result in updates to its content.

We call these dynamic Policy Endpoint Groups "Meta-data Driven Groups". The meta-data is a tag associated with endpoint information such as users, applications, and devices. The mapping from meta-data to dynamic content could come either from standards-based or proprietary tools. The security controller could use any available mechanisms to derive this mapping and to make automatic updates to the policy content if the mapping information changes.

The northbound policy interface must support endpoint groups for user-intent based policy management. The following meta-data driven groups are typically used for configuring security polices:

User-Group: This group identifies a set of users based on a tag or on static information. The tag to user information is dynamically derived from systems such as Active Directory or LDAP. For example, an end-customer may have different user-groups, such as HR-users, Finance-users, Engineering-users, to classify a set of users in each department.

Device-Group: This group identifies a set of devices based on a tag or on static information. The tag to device information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all machines running one operating system into one group and machines running another operating system into another group.

Application-Group: This group identifies a set of applications based on a tag or on static information. The tag to application information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all applications running in the Legal department into one group and all applications running under a specific operating system into another group.

Location-Group: This group identifies a set of locations based on a tag or on static information. The tag to location information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all sites/locations in a geographic region as one group.

4.4. Policy Rules

The security policy rules can be as simple as specifying a match for the user or application specified through "Policy Endpoint Group" and take one of the "Policy Actions" or more complicated rules that specify how two different "Policy Endpoint Groups" interact with each other. The northbound interface must support mechanisms to allow the following rule matches.

Policy Endpoint Groups: The rule must allow a way to match either a single or a member of a list of "Policy Endpoint Groups".

There must also be a way to express whether a group is a source or a destination so that the security administrator can apply the rule in only one direction of a communication.

There must also be a way to express a match between two "Policy Endpoint Groups" so that a policy can be effective for communication between two groups.

Direction: The rule must allow a way to express whether the security administrator wants to match the "Policy Endpoint Group" as the source or destination. The default should be to match both directions if the direction rule is not specified in the policy.

Threats: The rule should allow the security administrator to express a match for threats that come either in the form of feeds (such as botnet feeds, GeoIP feeds, URL feeds, or feeds from a SIEM) or speciality security appliances.

The threat could be from malware and this requires a way to match for virus signatures or file hashes.

4.5. Policy Actions

The security administrator must be able to configure a variety of actions within a security policy. Typically, security policy specifies a simple action of "deny" or "permit" if a particular rule is matched. Although this may be enough for most of the simple policies, the I2NSF northbound interface must also provide a more comprehensive set of actions so that the interface can be used effectively across various security functions.

Permit: This action means continue processing the next rule or allow the packet to pass if this is the last rule.

Deny: This action means stop further rule processing and drop the packet.

Drop connection: This action means stop further rule processing, drop the packet, and drop connection (for example, by sending a TCP reset).

Log: This action means create a log entry whenever a rule is matched.

Authenticate connection: This action means that whenever a new connection is established it should be authenticated.

Quarantine/Redirect: This action may be relevant for event driven policy where certain events would activate a configured policy that quarantines or redirects certain packet flows.

4.6. Third-Party Integration

The security policies in the end-customer's network may require the use of specialty devices such as honeypots, behavioral analytics, or SIEM in the network, and may also involve threat feeds, virus signatures, and malicious file hashes as part of comprehensive security policies.

The northbound interface must allow the security administrator to configure these threat sources and any other information to provide integration and fold this into policy management.

4.7. Telemetry Data

One of the most important aspect of security is to have visibility into the networks. As threats become more sophisticated, the security administrator must be able to gather different types of telemetry data from various devices in the network. The collected data could simply be logged or sent to security analysis engines for behavioral analysis and for threat detection.

The northbound interface must allow the security administrator to collect various kinds of data from NSFs. The data source could be syslog, flow records, policy violation records, and other available data.

5. Operational Requirements for the Northbound Interface

5.1. API Version

The northbound interface must support a version number for each RESTful API. This is very important because the client application and the controller application will most likely come from different vendors. Even if the vendor is same, it is hard to imagine that two different applications would be released in lock step.

Without API versioning, it hard to debug and figure out issues if application breaks. Although API versioning does not guarantee that applications will always work, it helps in debugging if the problem is caused by an API mismatch.

5.2. API Extensiblity

Abstraction and standardization of the northbound interface is of tremendous value to end-customers as it gives them the flexibility of deploying any vendors' NSF. However this might also look like as an obstacle to innovation.

If an NSF vendor comes up with new feature or functionality that can't be expressed through the currently defined northbound interface, there must be a way to extend existing APIs or to create a new API that is relevant for that NSF vendor only.

6. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

7. Acknowledgements

The editors would like to thank Adrian Farrel for helpful discussions and advice.

8. Normative References

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-00 (work in progress), February 2016.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Xiaobo Long
4 Cottonwood Lane
Warren, NJ 07059
US

Email: long.xiaobo@gmail.com

M. Klyus
NetCracker
J.Bi
Tsinghua University
C. Xie
China Telecom
July 8, 2016

SUPA policy-based management framework
draft-liu-supa-policy-based-management-framework-01.txt

Status of this Memo

Copyright Notice

Abstract

Simplified Use of Policy Abstractions (SUPA) defines a set of rules that define how services are designed, delivered, and operated within an operator’s environment independent of any one particular service or networking device. This document describes the SUPA basic architecture, its elements and interfaces.

Table of Contents

1. Introduction

The rapid growth in the variety and importance of traffic flowing over increasingly complex enterprise and service provider network architectures makes the task of network operations and management applications and deploying new services much more difficult. In addition, network operators want to deploy new services quickly and efficiently. Two possible mechanisms for dealing with this growing difficulty are the use of software abstractions to simplify the design and configuration of monitoring and control operations and the use of programmatic control over the configuration and operation of such networks. Policy-based management can be used to combine these two mechanisms into an extensible framework.

Policy rules can be used to express high-level network operator requirements directly, or from a set of management applications, to a network management or element system. The network management or element system can then control the configuration and/or monitoring of network elements and services.

Simplified Use of Policy Abstractions (SUPA) will define a generic policy information model (GPIM) [SUPA-info-model] for use in network operations and management applications. The GPIM defines concepts and terminology needed by policy management indepednent of the form and content of the policy rule. The ECA Policy Rule Information Model (EPRIM) [SUPA-info-model] extends the GPIM to define how to build policy rules according to the event-condition-action paradigm.

Both the GPIM and the EPRIM are targeted at controlling the configuration and monitoring of network elements throughout the service development and deployment lifecycle. The GPIM and the EPRIM will both be translated into corresponding YANG [RFC6020] modules that define policy concepts, terminology, and rules in a generic and interoperable manner; additional YANG modules may also be defined from the GPIM and/or EPRIM to manage specific functions.

The key benefit of policy management is that it enables different network elements and services to be instructed to behave the same way, even if they are programmed differently. Management applications will benefit from using policy rules that enable scalable and consistent programmatic control over the configuration and monitoring of network elements and services.

2. Framework for Generic Policy-based Management

This section briefly describes the design and operation of the SUPA policy-based management framework.

2.1. Overview

Figure 1 shows a simplified functional architecture of how SUPA is used to define policies for creating network element configuration and monitoring snippets. SUPA uses the GPIM to define a consensual vocabulary that different actors can use to interact with network elements and services. The EPRIM defines a generic structure for imperative policies. The GPIM, as well as the combination of the GPIM and EPRIM, are converted to generic YANG data modules. The IETF produces the modules, and IANA is used to register the module and changes to it.

In one possible approach, SUPA Generic & ECA Policy YANG Data modules together with the Resource and Service YANG data models specified in IETF (which define the specific elements that will be controlled by policies) are used by the Service Interface Logic. This Service Interface Logic creates appropriate input mechanisms for the operator to define policies   (e.g., a web form or a script) for creating and managing the network configuration. The operator interacts with the interface, which is then translated to configuration snippets.

Note that YANG models may not exist. In this case, the SUPA generic policy YANG data modules serve as an extensible basis to develop new YANG data models for the Service Interface Logic to create appropriate input mechanisms for the operator to define policies. This transfers the work specified by the Resource and Service YANG data models specified in IETF into the Service Interface Logic, which is then translated to configuration snippets.

```
                          +---------------------+
        +----------+       \| SUPA Generic Policy |
        |   IETF   |---+----+  Information Model  |
        +----------+   |   /|                     |
                       |    +---------+-----------+
                       |              |
           Assignments |              | Defines Policy Concepts
            and Manage |              |
             Content   |             \|/
                       |    +---------+-----------+
                       |   \| SUPA GPIM and EPRIM |
                       +----+    Generic YANG     |
                           /|    Data Modules     |
                            +---------+-----------+
                                      *
                                      *  Possible
                                      *  Approach
                                      *
     +--------------------------------*----------------------------------+
     |  Management System             *                                  |
     |                                *                                  |
     |                               \*/                                 |
     |             Fills   +----------+----------+    +---------------+  |
     | +--------+  Forms  \|  Service Interface  |/   | Resource and  |/ |   +--
----+
     | |Operator|----------+        Logic        +----| Service YANG  |------| I
ETF |
     | +--------+  Runs   /|  (locally defined   |\   |  Data Models  |\ |   +--
-----
     |            Scripts  | forms, scripts,...) |    +---------------+  |
     |                     +----------+----------+                       |
```

```
     |                                |                                   |
     |                              \|/                                   |
     |                        +-------+--------+                          |
     |                        |  Local Devices |                          |
     |                        | and Management |                          |
     |                        |     Systems    |                          |
     |                        +----------------+                          |
     +--------------------------------------------------------------------+
```

Figure 1 SUPA Framework

Figure 1 is exemplary. The Operator actor shown in Figure 1 can interact with SUPA in other ways not shown in Figure 1. In addition, other actors (e.g., an application developer) that can interact with SUPA are not shown for simplicity.

The EPRIM defines an Event-Condition-Action (ECA) policy as an example of imperative policies. An ECA policy rule is activated when its event clause is true; the condition clause is then evaluated and, if true, signals the execution of one or more actions in the action clause. Imperative policy rules require additional management functions, which are explained in section 2.2 below.

Figure 2 shows a SUPA Policy Model creating and communicating policy rules to two different Network Manager and Network Controller elements.

The Generic Policy Information Model (GPIM) was used to construct policies. The GPIM defines generic policy concepts, as well as two types of policies: ECA policy rules and declarative policy statements.

An ECA policy rule is activated when its event clause is true; the condition clause is then evaluated and, if true, signals the execution of one or more actions in the action clause. This type of policy explicitly defines the current and desired states of the system being managed.

A set of Generic Policy Data Models are then created from the GPIM. These YANG data model policies are then used to control the configuration of network elements that model the service(s) to be managed using policy.

```
     +-----------------------------------------------------------------+
     |                       SUPA Policy Model                         |
     |                                                                 |
     |              +----------------------------------+               |
     |              | Generic Policy Information Model |               |
     |              +----+------------------------+----+               |
     |                   D                        D                    |
     |                   D                       \ /                   |
     |                   D           +------------+--------------+     |
     |                   D           | ECAPolicyRule Information |     |
     |                   D           | Model (EPRIM)             |     |
     |                   D           +------------+--------------+     |
     |  +----------------D------------------------D----------------+   |
     |  |                D SUPA Policy Data Model D                |   |
     |  |               \ /                       D                |   |
     |  |+---------------+-----------+            D                |   |
     |  || Generic Policy Data Model |            D                |   |
     |  |+-------------------+-------+            D                |   |
     |  |                    D                    D                |   |
     |  |                   \ /                  \ /               |   |
     |  |                 +--+--------------------+--------------+ |   |
     |  |                 |    ECA PolicyRule Data Model         | |   |
     |  |                 +--------------------------------------+ |   |
     |  +------------------------------+---------------------------+   |
     |                                 |                               |
     |                                 |                               |
     +---------------------------------|-------------------------------+
                                       |
                                       |   NETCONF/RESTCONF
                         +-------------+--------+
                         C                      C
                         C                      C
                        \ /                    \ /
        +----------------+-----------+  +-------+--------------------+
        | Network Manager/Controller |  | Network Manager/Controller |
        |   +--------------------+   |  |   +---------------------+  |
        |   |  Network Resource  |   |  |   |  Network Resource   |  |
        |   |     Data Model     |   |  |   |       Data Model    |  |
        |   +--------------------+   |  |   +---------------------+  |
        +---+---+---+----------------+  +-----+---+---+--------------+
           / \ / \ / \                       / \ / \ / \
            C   C   C                         C   C   C
            C   C   C                         C   C   C
            C   C   C                         C   C   C
           \ / \ / \ /                       \ / \ / \ /
           NE1 NE2 NEn                       NE1 NE2 NEn
```

Figure 2 SUPA Policy Model Framework

In Figure 2:
A double-headed arrow with Cs means communication;
A double-headed arrow with Ds means derived from.

The network elements used in this framework are:

SUPA Policy Model: represents one or more policy modules that contain the following entities:

Generic Policy Information Model: a model for defining policy rules that are independent of data repository, data definition, query, and implementation languages, and protocol. This model is abstract and is used for design; it MUST be turned into a data model for implementation.

Generic Policy Data Model: a model of policy rules for that are dependent of data repository, data definition, query, and implementation languages, and protocol.

ECA Policy Rule Information Data Model (EPRIM): represents a policy rule as a statement that consists of an event clause, a condition clause, and an action clause. This type of Policy Rule explicitly defines the current and desired states of the system being managed. This model is abstract and is used for design; it MUST be turned into a data model for implementation.

ECA Policy Rule Data Model: a model of policy rules derived from EPRIM, consist of an event clause, a condition clause, and an action clause.

NM/NC: Network Manager / Controller, which represents one or more entities that are able to control the operation and management of a network infrastructure (e.g., a network topology that consists of Network Elements).

Network Resource Data Model: a model of the physical and virtual network topology including the resource attributes (e.g., data rate or latency of links) and operational parameters needed to support service deployment over the network topology. An example of a network resource data model can be found in [ID.draft-contreras-supa-yang-network-topo].

Network Element (NE), which can interact with local or remote NM/NC in order to exchange information, such as configuration information, policy enforcement capabilities, and network status.

Relationship among Policy, Service and Resource models can be illustrated by the figure below.

```
      +---------------+                   +-----------------+
      |   Policy      |        (1)        |   Service       |
      |               |*******************|                 |
      | ( SUPA )      |                   |(L3SM, ...  )    |
      +---------------+                   +-----------------+
                *                                   **
                 **                               **
                   *                            **
             (2)    **                        **   (3)
                      *                     **
                       **                 **
                         *              **
                     +--------------------+
                     |     Resource       |
                     |                    |
                     | (Inventory,...)    |
                     +--------------------+
```

Figure 3 Relationship among Policy, Service and Resource

In Figure 3:
(1) policy relies on and is able to adjust service
(2) policy relies on network ability provided by resource and is able to adjust resource
(3) resource relies on network ability and is able to reserve and consume/occupy resource

2.2. Operation

SUPA can be used to define various types of policies, including policies that affect services and/or the configuration of individual or groups of network elements. SUPA can be used by a centralized and/or distributed set of entities for creating, managing, interacting with, and retiring policy rules.

The SUPA scope is limited to policy information and data models. SUPA will not define network resource data models or network service data models; both are out of scope. Instead, SUPA will make use of network resource data models defined by other WGs or SDOs.

Declarative policies that specify the goals to achieve but not how to achieve those goals (also called "intent-based" policies) are out of scope for the initial phase of SUPA.

2.3. The GPIM and the EPRIM

The GPIM provides a common vocabulary for representing concepts that are common to expressing different types of policy, but which are independent of language, protocol, repository, and level of abstraction.

This enables different policies at different levels of abstraction to form a continuum, where more abstract policies can be translated into more concrete policies, and vice-versa. For example, the information model can be extended by generalizing concepts from an existing data model into the GPIM; the GPIM extensions can then be used by other data models.

The SUPA working group develops models for expressing policy at different levels of abstraction. Specifically, two models are envisioned (both of which are contained in the Generic Policy Information Model block in Figure 1:

1. a generic model (the GPIM) that defines concepts and vocabulary needed by policy management systems independent of the form and content of the policy

2. a more specific model (the EPRIM) that refines the GPIM to specify policy rules in an event-condition-action form

2.4. Creation of Generic YANG Modules

An information model is abstract. As such, it cannot be directly instantiated (i.e., objects cannot be created directly from it). Therefore, both the GPIM, as well as the combination of the GPIM and the EPRIM, are translated to generic YANG modules.

SUPA will provide guidelines for translating the GPIM (or the combination of the GPIM and the EPRIM) into concrete YANG data models that define how to manage and communicate policies between systems. Multiple imperative policy YANG data models may be instantiated from the GPIM (or the combination of the GPIM and the EPRIM). In particular, SUPA will specify a set of YANG data models that will consist of a base policy model for representing policy management concepts independent of the type or structure of a policy, and as well, an extension for defining policy rules according to the ECA paradigm.

The process of developing the GPIM, EPRIM and the derived/translated YANG data models is realized following the sequence shown below. After completing this process and if the implementation of the YANG data models requires it, the GPIM and EPRIM and the derived/translated YANG data models are updated and synchronized.

(1)=>(2)=>(3)=>(4)=>(3’)=>(2’)=>(1’)

Where, (1)=GPIM; (2)=EPRIM; (3)=YANG data models; (4)= Implementation; (3’)= update of YANG data models; (2’)=update of EPRIM; (1’) = update of GPIM

The YANG module derived from the GPIM contains concepts and terminology for the common operation and administration of policy-based systems, as well as an extensible structure for policy rules of different paradigms. The YANG module derived from the EPRIM extends the generic nature of the GPIM to represent policies using an event-condition-action structure.

3. Security Considerations

TBD

4. IANA Considerations

This document has no actions for IANA.

5. Contributors

The following people all contributed to creating this document, listed in alphabetical order:

TBD.

6. Acknowledgments

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: Andy Bierman.

Part of the initial draft of this document was picked up from previous documents, and this section lists the acknowledgements from them.

From "SUPA Value Proposition" [Klyus2016]

The following people all contributed to creating this document, listed in alphabetical order:

Vikram Choudhary, Huawei Technologies
Luis M. Contreras, Telefonica I+D
Dan Romascanu, Avaya
J. Schoenwaelder, Jacobs University, Germany
Qiong Sun, China Telecom
Parviz Yegani, Juniper Networks

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: H. Rafiee, J. Saperia and C. Zhou.

The authors of "SUPA Value Proposition" [Klyus2016] were:

Maxim Klyus, Ed. , NetCracker
John Strassner, Ed. , Huawei Technologies
Will(Shucheng) Liu, Huawei Technologies
Georgios Karagiannis, Huawei Technologies
Jun Bi, Tsinghua University

The initial draft of this document merged one document, and this section lists the acknowledgements from it.

From "Problem Statement for Simplified Use of Policy Abstractions (SUPA)" [Karagiannis2015]

The authors of this draft would like to thank the following persons for the provided valuable feedback and contributions: Diego Lopez, Spencer Dawkins, Jun Bi, Xing Li, Chongfeng Xie, Benoit Claise, Ian Farrer, Marc Blancet, Zhen Cao, Hosnieh Rafiee, Mehmet Ersue, Simon Perreault, Fernando Gont, Jose Saldana, Tom Taylor, Kostas Pentikousis, Juergen Schoenwaelder, John Strassner, Eric Voit, Scott O. Bradner, Marco Liebsch, Scott Cadzow, Marie-Jose Montpetit. Tina Tsou, Will Liu and Jean-Francois Tremblay contributed to an early version of this draft.

The authors of "Problem Statement for Simplified Use of Policy Abstractions (SUPA)" [Karagiannis2015] were:

Georgios Karagiannis, Huawei Technologies
Qiong Sun, China Telecom
Luis M. Contreras, Telefonica
Parviz Yegani, Juniper
John Strassner, Huawei Technologies
Jun Bi, Tsinghua University

From "The Framework of Simplified Use of Policy Abstractions (SUPA)" [Zhou2015]

The authors of this draft would like to thank the following persons for the provided valuable feedback: Diego Lopez, Jose Saldana, Spencer Dawkins, Jun Bi, Xing Li, Chongfeng Xie, Benoit Claise, Ian Farrer, Marc Blancet, Zhen Cao, Hosnieh Rafiee, Mehmet Ersue, Mohamed Boucadair, Jean Francois Tremblay, Tom Taylor, Tina Tsou, Georgios Karagiannis, John Strassner, Raghav Rao, Jing Huang.

Early version of this draft can be found here:
https://tools.ietf.org/html/draft-zhou-supa-architecture-00
At the early stage of SUPA, we think quite some issues are left open, it is not so suitable to call this draft as "architecture". We would like to rename it to "framework". Later there may be a dedicated architecture document.

The authors of "The Framework of Simplified Use of Policy Abstractions (SUPA)" [Zhou2015] were:

Cathy Zhou, Huawei Technologies
Luis M. Contreras, Telefonica
Qiong Sun, China Telecom
Parviz Yegani, Juniper

7. References

7.1. Normative References

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

7.2. Informative References

[RFC3198] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., Waldbusser, S., "Terminology for Policy-Based Management", RFC 3198, November, 2001

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

[RFC7285] R. Alimi, R. Penno, Y. Yang, S. Kiesel, S. Previdi, W. Roome, S. Shalunov, R. Woundy "Application-Layer Traffic Optimization (ALTO) Protocol", September 2014

[SUPA-info-model] J. Strassner, J. Halpern, J. Coleman, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", IETF Internet draft, draft-strassner-supa-generic-policy-info-model-04, February 2016

[TR235] J. Strassner, ed., "ZOOM Policy Architecture and Information Model Snapshot", TR245, part of the TM Forum ZOOM project, October 26, 2014

[Karagiannis2015] G. Karagiannis, ed., "Problem Statement for Simplified Use of Policy Abstractions (SUPA)", IETF Internet draft, draft-karagiannis-supa-problem-statement-07, June 5, 2015

[Klyus2016] M. Klyus, ed., "SUPA Value Proposition", IETF Internet draft, draft-klyus-supa-value-proposition-00, Mar 21, 2016

[Zhou2015] C. Zhou, ed., "The Framework of Simplified Use of Policy Abstractions (SUPA)", draft-zhou-supa-framework-02, May 08, 2015

Authors' Addresses

Will(Shucheng) Liu
Huawei Technologies
Bantian, Longgang District, Shenzhen 518129
P.R. China

Email: liushucheng@huawei.com

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95138 USA

Email: john.sc.strassner@huawei.com

Georgios Karagiannis
Huawei Technologies
Hansaallee 205, 40549 Dusseldorf
Germany

Email: Georgios.Karagiannis@huawei.com

Maxim Klyus
NetCracker
Kozhevnicheskaya str.,7 Bldg. #1
Moscow, Russia

E-mail: klyus@netcracker.com

Jun Bi
Tsinghua University
Network Research Center, Tsinghua University
Beijing 100084
P.R. China

Email: junbi@tsinghua.edu.cn

Chongfeng Xie
China Telecom Beijing Research Institute
China Telecom Beijing Information Science&Technology Innovation Park
Beiqijia Town Changping District Beijing 102209 China
Email: xiechf@ctbri.com.cn