INTERNET-DRAFT
<draft-brunner-policy-core-ext-00.txt>
Expires in six months

M. Brunner
J. Quittek
NEC Corporation

November 17, 2000

Policy Framework Core Info Model Extensions
<draft-brunner-policy-core-ext-00.txt>

Status of this Memo

Abstract

This memo should help starting the discussion on extensions of the Policy Framework Core Information Model (PCIM). It does not replace nor change concepts used in PCIM, but tries to add concepts, which could help different application areas to define policies more easily by reusing some of the concepts proposed in this draft. The draft basically tries to generalize concepts introduced by other drafts such as the Policy Framework QoS Information Model [PQIM]. The focus lies in means for specifying general conditions and actions. Additionally, group ordering, events, and variables are introduced.

1. Introduction

PCIM as proposed today is a well defined as a high-level information model for policies. However, many concepts are useful for the simple case, whereas more complicated policies poses problems using the

classes available. The other problem lies in a set of application areas using the basic model and adapt and enhance it to area specific needs. It turns out that many concepts are developed again and again, e.g., filters, variables, simple conditions etc.

The goal of this draft is to extend PCIM with general constructs, which may be used in other areas. Basically, they center around generic definition of conditions and actions with the knowlege of the structure of the modeling language. They are heavily influenced by constructs used in [PQIM], [PMPLS], and [PIPSEC].

The draft in its current state is far from being complete. e.g., the definition of class do not contain any formal property definition yet. However, the proposed concepts and the list of open issues should help discussing useful extensions to the core information model.

Other authors are very welcome to contribute to make the policy framework core information model useful in various areas.

2. PolicyGroup Ordering

In PCIM, the policyRule has a property Priority, which allows a manager to give a rule priority over others. However, it is not specified what the scope of the priority is, in the system or in the group.

We propose a class OrderedPolicyGroup, which has a property priority as well. This allows for prioritizing groups over other groups. Furthermore, groups within the OrderPolicyGroup are ordered via the rule's or group's priority property. Additionally, the scope of the priority is only within that group, not globally as it could be read in PCIM. Groups in OrderPolicyGroups not being of type OrderPolicyGroup or a derivative of it, have the least priority.

Note, that the functionality could be placed in the aggregation PolicyGroupinPolicyGroup as well. (What is preferred?)

NAME	OrderedPolicyGroup
DERIVED FROM	PolicyGroup (defined in [PCIM])
ABSTRACT	False
PROPERTIES	Priority [int]

3. Conditions

3.1. BasicPolicyCondition

This class refines the basic structure of the policyCondition class

defined in [PCIM] by using the triplet <variable>, <operator> and <variable>. The SimplePolicyCondition of PQIM is similar in structure, and a special case of this one, where the right variable has a constant value.

The operator is relational and the binding is given by the types of both sides. Whether the operator is statically of dynamically bound (variable type or value type) is for further study.

If one of the variables evaluate to undefined, the condition evaluates to false.

What kind of RelOperators do we want?

For example, the RelOperator property values may be
- equal, notequal, lessthan, greaterthan, lessthanequal, greaterthanequal, for totally ordered values
- startswith, endswith, equals, contains, for strings
- contains, in, equals, includes, intersects, for sets

NAME	BasicPolicyCondition
DERIVED FROM	PolicyCondition (defined in [PCIM])
ABSTRACT	False
PROPERTIES	firstVariable [ref PolicyVariable], RelOperator [String], secondVariable [ref PolicyVariable]

3.2. Class PolicyObjectTypeSelector

Basically, PCIM uses the policyRole property in the PolicyRule class in order to select a set of objects the rule should apply to. The selection may be refined by conditions.

This kind of condition is special in the sense, that it compares the set of objects available in the context (basically specified by the role) whether they are of the type specified in this condition. So the rule containing this selector in the condition list operates only on objects of the specific type. This is very convenient, because it allows a policy designer to specify the actions with the knowledge, that only objects of a specific type will pass the condition.

NAME	PolicyObjectTypeSelector
DERIVED FROM	policyCondition (defined in [PCIM])
ABSTRACT	False
PROPERTIES	SelectorClassName

3.3. Class PolicyAssociatedObjectSelector

The PolicyAssociatedObjectSelector provides means to select objects only if they are associated with objects of a certain type and the AssocObjectPolicyCondition evaluates to true.

Note that the class has a association to PolicyConditions (PolicyConditionInAssocObjectSelector). With that it is possible to specify again a PolicyAssociationObjectSelector as the condition. This allows to recursivly hop throught associated object.

Furthermore, note that the condition within this selector has a different context, compared to the condition itself. Conceptually, the PolicyAssociatedObjectSelector takes a set of objects and checks for all of them, whether they are in an association of type AssociationClassName where the object in this association is of type AssocObjectClassName and the policy condition evaluates to true.

NAME	PolicyAssociatedObjectSelector
DERIVED FROM	policyCondition (defined in [PCIM])
ABSTRACT	False
PROPERTIES	AssociationClassName, AssocObjectClassName,

3.4. Association PolicyConditionInAssocObjectSelector

This association ties together the PolicyAssociatedObjectSelector with a PolicyCondition.

NAME	PolicyConditionInAssocObjectSelector
ABSTRACT	False
PROPERTIES	selector [ref PolicyObjectTypeSelector[0..1]], condition [ref PolicyCondition[0..1]]

4. Variables

Variables may be global or local. We propose to use only local variables, which means a variable is always used in the context of an object.

4.1. Class PolicyAttributeVariable

The PolicyAttributeVariable has an attribute name and a class name as properties. The class name specifies in what scope the attribute name is valid. The class name is only used in cases, where attribute names are not globally unique.

Conceptually, if the variable is used in a condition, it evaluates to the value of the attribute named by AttributeName. Note, that this variable may be undefined. This means the basic condition it is contained in evaluates to false.

NAME	PolicyAttributeVariable
DERIVED FROM	Policy (defined in [PCIM])
ABSTRACT	False
PROPERTIES	AttrClassName, AttributeName

5. Actions

5.1. Class PolicyAttributeSetAction

The AttributeSetAction is used to set or change a value of an attribute. The action has again a class name and an attribute name as property. The class name specifies what type of objects to set, and the attribute name specifies what attribute to change. Note again, in case of globally unique attribute names, the class name is not used in the action.

The semantic behind the AttributeSetAction is, that setting the attribute of an object results in the configuration of the real object, represented by that object.

If the attribute is not compatible with the type of the value, the action is not performed.

NAME	PolicyAttributeSetAction
DERIVED FROM	PolicyAction (defined in [PCIM])
ABSTRACT	False
PROPERTIES	AttrSetClassName, AttributeSetName, value

6. Events

6.1. PolicyEvent

An event is a mean for triggering a rule to be evaluated. The source of the event may be in- or outside of the policy engine. Therefore, an event has in its basic version no source identification associated. However, classes derived from the base class may add the source, if needed. The PolicyEvent has a time and date when it happened. It has a name. The event name is used to differ the events. It could be usefule to add a context string, which make different policy models more independet and less error prone, because of global name spaces.

The semantic behind the event is open, but see the next section for conditions on events. However, typically the name will also define the semantic of the event.

NAME	PolicyEvent
DERIVED FROM	Policy (defined in [PCIM])
ABSTRACT	False
PROPERTIES	event_time, event_date, event_name

6.2. PolicyEventCondition

The event condition evaluates to true if an event happened with the name specified in cond_event_name. In general, there is no time constrain defined, until when the rule containing the event condition needs to be evaluated. But depending on the execution environment of the policy rules, this may immediately trigger the evaluation of the policy rule containing the event condition.

After evaluating all rules in the system, containing the event condition in its condition tree, the event does not exist anymore, and the condition is evaluating to false for all further evaluations until the event is created once more.

Note that, with this definition of the semantic, we do not run into problems of nested conditions and rules, which contain associations to the same event condition. However, more advanced semantics are possible, but may produce difficult to implement semantic.

NAME	PolicyEventCondition
DERIVED FROM	PolicyCondition (defined in [PCIM])
ABSTRACT	False
PROPERTIES	cond_event_name

6.3. PolicyEventCreateAction

The PolicyEventCreateAction creates an event with a certain name. This may be useful in cases, where you have multi-level policy systems running in the same execution environment.

NAME	PolicyEventCreateAction
DERIVED FROM	PolicyAction (defined in [PCIM])
ABSTRACT	False
PROPERTIES	create_event_name

7. Security Considerations

The security considerations for this document are the same as those of the [PCIM] and are not further addressed in this version of the draft.

8. Open Issues

8.1. Multi-Object Selectors

There are many cases where you want to select two or more objects, in order to associate the two objects, or all objects in a set to each other.

8.2. Creation of Objects

It could be very beneficial for policies to create objects just with the type information, and set the attributes afterwards. However, what is the semantic of the creation of an object?

8.3. Creation of Associations

Section 8.1, already mention the problem of multi-object selection. One goal could be the creation of a new association, between these objects. This action would not be difficult to specify, but it is not useful without the multi-object selection.

8.4. Modification of Associations

Until now, we have no means other than area specific actions, which allow a policy designer to implement the modification of an association, or its properties.

8.5. Expressions

How can we set or change attributes and parameters of action by an expression (e.g., set bandwidth to 50% of the existing one)

8.6. IP specific filters

Filters are used many times, why not specifing it in the PCIM extension.

9. References

[PCIM] B. Moore, E. Ellesson, J. Strassner, "Policy Core Information Model -- Version 1 Specification", Internet Draft, <draft-ietf-policy-core-info-model-06.txt>, May, 2000

[PQIM] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, "Policy Framework QoS Information Model", Internet draft, <draft-ietf-policy-qos-info-model-01.txt>, April 2000

[PMPLS] K. Isoyama, M. Brunner, M. Yoshida, J. Quittek, R. Chadha,

A. Poylisher, G. Mykoniatis, A. Kind,F. Reichmeyer,
"Policy Framework MPLS Information Model for QoS and TE"
<draft-chadha-policy-mpls-te-01.txt>, December 2000.

[PIPSEC] J. Jason, "IPsec Configuration Policy Model", Internet draft
<draft-ietf-ipsp-config-policy-model-01.txt>, July 2000.

10. Authors' Addresses

Marcus Brunner, Juergen Quittek
NEC Europe Ltd.
C&C Research Laboratories
Adenauerplatz 6
D-69115 Heidelberg, Germany
Phone: +49 (0)6221 905110
Fax: +49 (0)6221 9051155
Email: [brunner|quittek]@ccrle.nec.de

Full Copyright Statement