Network Working Group   H. Schulzrinne
Request for Comments: 4745   Columbia U.
Category: Standards Track   H. Tschofenig
                                        Siemens Networks GmbH & Co KG
                                                             J. Morris
                                                                   CDT
                                                            J. Cuellar
                                                               Siemens
                                                               J. Polk
                                                          J. Rosenberg
                                                                 Cisco
                                                         February 2007

Common Policy: A Document Format for Expressing Privacy Preferences

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This document defines a framework for authorization policies
   controlling access to application-specific data. This framework
   combines common location- and presence-specific authorization
   aspects. An XML schema specifies the language in which common policy
   rules are represented. The common policy framework can be extended
   to other application domains.

Table of Contents

1. Introduction

This document defines a framework for creating authorization policies
for access to application-specific data. This framework is the
result of combining the common aspects of single authorization
systems that more specifically control access to presence and
location information and that previously had been developed
separately. The benefit of combining these two authorization systems
is two-fold. First, it allows building a system that enhances the
value of presence with location information in a natural way and
reuses the same underlying authorization mechanism. Second, it
encourages a more generic authorization framework with mechanisms for
extensibility. The applicability of the framework specified in this
document is not limited to policies controlling access to presence
and location information data, but can be extended to other
application domains.

The general framework defined in this document is intended to be
accompanied and enhanced by application-specific policies specified
elsewhere. The common policy framework described here is enhanced by
domain-specific policy documents, including presence [7] and location
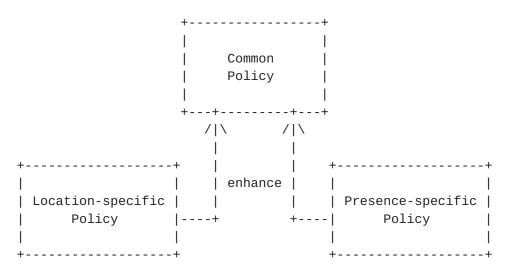[8]. This relationship is shown in Figure 1.

```
                       +-----------------+
                       |                 |
                       |     Common      |
                       |     Policy      |
                       |                 |
                       +---+---------+---+
                          /|\       /|\
                           |         |
     +-------------------+ |         | +-------------------+
     |                   | | enhance | |                   |
     | Location-specific | |         | | Presence-specific |
     |      Policy       | |----+  +----|      Policy       |
     |                   | |         | |                   |
     +-------------------+           | +-------------------+
```

Figure 1: Common Policy Enhancements

This document starts with an introduction to the terminology in
Section 2, an illustration of basic modes of operation in Section 3,
a description of goals (see Section 4) and non-goals (see Section 5)
of the policy framework, followed by the data model in Section 6.
The structure of a rule, namely, conditions, actions, and
transformations, is described in Sections 7, 8, and 9. The procedure
for combining permissions is explained in Section 10 and used when
conditions for more than one rule are satisfied. A short description

of meta policies is given in Section 11. An example is provided in
Section 12. The XML schema will be discussed in Section 13. IANA
considerations in Section 15 follow security considerations in
Section 14.

2. Terminology

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 [1].

This document introduces the following terms:

PT - Presentity / Target: The PT is the entity about whom
     information has been requested.

RM - Rule Maker: The RM is an entity that creates the authorization
     rules that restrict access to data items.

PS - (Authorization) Policy Server: This entity has access to both
     the authorization policies and the data items. In location-
     specific applications, the entity PS is labeled as location
     server (LS).

WR - Watcher / Recipient: This entity requests access to data items
     of the PT. An access operation might be a read, a write, or any
     other operation.

A policy is given by a 'rule set' that contains an unordered list of
'rules'. A 'rule' has a 'conditions', an 'actions', and a
'transformations' part.

The term 'permission' indicates the action and transformation
components of a 'rule'.

The term 'using protocol' is defined in [9]. It refers to the
protocol used to request access to and to return privacy-sensitive
data items.

3. Modes of Operation

The abstract sequence of operations can roughly be described as
follows. The PS receives a query for data items for a particular PT,
via the using protocol. The using protocol (or more precisely, the
authentication protocol) provides the identity of the requestor,
either at the time of the query or at the subscription time. The
authenticated identity of the WR, together with other information
provided by the using protocol or generally available to the server,

is then used for searching through the rule set. All matching rules
are combined according to a permission combining algorithm described
in Section 10. The combined rules are applied to the application
data, resulting in the application of privacy based on the
transformation policies. The resulting application data is returned
to the WR.

Three different modes of operation can be distinguished:

3.1. Passive Request-Response - PS as Server (Responder)

In a passive request-response mode, the WR queries the PS for data
items about the PT. Examples of protocols following this mode of
operation include HTTP, FTP, LDAP, finger, and various remote
procedure call (RPC) protocols, including Sun RPC, Distributed
Computing Environment (DCE), Distributed Component Object Model
(DCOM), common object request broker architecture (Corba), and Simple
Object Access Protocol (SOAP). The PS uses the rule set to determine
whether the WR is authorized to access the PT's information, refusing
the request if necessary. Furthermore, the PS might filter
information by removing elements or by reducing the resolution of
elements.

3.2. Active Request-Response - PS as Client (Initiator)

Alternatively, the PS may contact the WR and convey data items.
Examples include HTTP, SIP session setup (INVITE request), H.323
session setup or SMTP.

3.3. Event Notification

Event notification adds a subscription phase to the "Active Request-
Response - PS as Client (Initiator)" mode of operation. A watcher or
subscriber asks to be added to the notification list for a particular
presentity or event. When the presentity changes state or the event
occurs, the PS sends a message to the WR containing the updated
state. (Presence is a special case of event notification; thus, we
often use the term interchangeably.)

In addition, the subscriber may itself add a filter to the
subscription, limiting the rate or content of the notifications. If
an event, after filtering by the rule-maker-provided rules and by the
subscriber-provided rules, only produces the same notification
content that was sent previously, no event notification is sent.

A single PS may authorize access to data items in more than one mode.
Rather than having different rule sets for different modes all three
modes are supported with a one rule set schema. Specific instances
of the rule set can omit elements that are only applicable to the
subscription model.

4. Goals and Assumptions

   Below, we summarize our design goals and constraints.

   Table representation:

      Each rule must be representable as a row in a relational database.
      This design goal should allow efficient policy implementation by
      utilizing standard database optimization techniques.

   Permit only:

      Rules only provide permissions rather than denying them. Removing
      a rule can never increase permissions. Depending on the
      interpretation of 'deny' and 'permit' rules, the ordering of rules
      might matter, making updating rule sets more complicated since
      such update mechanisms would have to support insertion at specific
      locations in the rule set. Additionally, it would make
      distributed rule sets more complicated. Hence, only 'permit'
      actions are allowed that result in more efficient rule processing.
      This also implies that rule ordering is not important.
      Consequently, to make a policy decision requires processing all
      rules.

   Additive permissions:

      A query for access to data items is matched against the rules in
      the rule database. If several rules match, then the overall
      permissions granted to the WR are the union of those permissions.
      A more detailed discussion is provided in Section 10.

   Upgradeable:

      It should be possible to add additional rules later, without
      breaking PSs that have not been upgraded. Any such upgrades must
      not degrade privacy constraints, but PSs not yet upgraded may
      reveal less information than the rule maker would have chosen.

Capability support:

   In addition to the previous goal, a RM should be able to determine
   which extensions are supported by the PS. The mechanism used to
   determine the capability of a PS is outside the scope of this
   specification.

Protocol-independent:

   The rule set supports constraints on both notifications or queries
   as well as subscriptions for event-based systems such as presence
   systems.

No false assurance:

   It appears more dangerous to give the user the impression that the
   system will prevent disclosure automatically, but fail to do so
   with a significant probability of operator error or
   misunderstanding, than to force the user to explicitly invoke
   simpler rules. For example, rules based on weekday and time-of-
   day ranges seem particularly subject to misinterpretation and
   false assumptions on part of the RM. (For example, a non-
   technical RM would probably assume that the rules are based on the
   time zone of his current location, which may not be known to other
   components of the system.)

[5](#). Non-Goals

   We explicitly decided that a number of possibly worthwhile
   capabilities are beyond the scope of this first version. Future
   versions may include these capabilities, using the extension
   mechanism described in this document. Non-goals include:

No external references:

   Attributes within specific rules cannot refer to external rule
   sets, databases, directories, or other network elements. Any such
   external reference would make simple database implementation
   difficult and hence they are not supported in this version.

No regular expressions:

   Conditions are matched on equality or 'greater-than'-style
   comparisons, not regular expressions, partial matches such as the
   SQL LIKE operator (e.g., LIKE "%foo%"), or glob-style matches
   ("*@example.com"). Most of these are better expressed as explicit
   elements.

No repeat times:

> Repeat times (e.g., every day from 9 am to 4 pm) are difficult to
> make work correctly, due to the different time zones that PT, WR,
> PS, and RM may occupy. It appears that suggestions for including
> time intervals are often based on supporting work/non-work
> distinctions, which unfortunately are difficult to capture by time
> alone. Note that this feature must not be confused with the
> 'Validity' element that provides a mechanism to restrict the
> lifetime of a rule.

6. Basic Data Model and Processing

A rule set (or synonymously, a policy) consists of zero or more
rules. The ordering of these rules is irrelevant. The rule set can
be stored at the PS and conveyed from RM to PS as a single document,
in subsets or as individual rules. A rule consists of three parts:
conditions (see Section 7), actions (see Section 8), and
transformations (see Section 9).

The conditions part is a set of expressions, each of which evaluates
to either TRUE or FALSE. When a WR asks for information about a PT,
the PS goes through each rule in the rule set. For each rule, it
evaluates the expressions in the conditions part. If all of the
expressions evaluate to TRUE, then the rule is applicable to this
request. Generally, each expression specifies a condition based on
some variable that is associated with the context of the request.
These variables can include the identity of the WR, the domain of the
WR, the time of day, or even external variables, such as the
temperature or the mood of the PT.

Assuming that the rule is applicable to the request, the actions and
transformations (commonly referred to as permissions) in the rule
specify how the PS is supposed to handle this request. If the
request is to view the location of the PT, or to view its presence,
the typical action is "permit", which allows the request to proceed.

Assuming the action allows the request to proceed, the
transformations part of the rule specifies how the information about
the PT -- their location information, their presence, etc. -- is
modified before being presented to the WR. These transformations are
in the form of positive permissions. That is, they always specify a
piece of information that is allowed to be seen by the WR. When a PS
processes a request, it takes the transformations specified across
all rules that match, and creates the union of them. For computing
this union, the data type, such as Integer, Boolean, Set, or the
Undef data type, plays a role. The details of the algorithm for
combining permissions is described in Section 10. The resulting

union effectively represents a "mask" -- it defines what information
is exposed to the WR. This mask is applied to the actual location or
presence data for the PT, and the data that is permitted by the mask
is shown to the WR. If the WR requests a subset of information only
(such as city-level civic location data only, instead of the full
civic location information), the information delivered to the WR MUST
be the intersection of the permissions granted to the WR and the data
requested by the WR.

Rules are encoded in XML. To this end, Section 13 contains an XML
schema defining the Common Policy Markup Language. This, however, is
purely an exchange format between RM and PS. The format does not
imply that the RM or the PS use this format internally, e.g., in
matching a query with the policy rules. The rules are designed so
that a PS can translate the rules into a relational database table,
with each rule represented by one row in the database. The database
representation is by no means mandatory; we will use it as a
convenient and widely-understood example of an internal
representation. The database model has the advantage that operations
on rows have tightly defined meanings. In addition, it appears
plausible that larger-scale implementations will employ a backend
database to store and query rules, as they can then benefit from
existing optimized indexing, access control, scaling, and integrity
constraint mechanisms. Smaller-scale implementations may well choose
different implementations, e.g., a simple traversal of the set of
rules.

6.1. Identification of Rules

Each rule is equipped with a parameter that identifies the rule.
This rule identifier is an opaque token chosen by the RM. A RM MUST
NOT use the same identifier for two rules that are available to the
PS at the same time for a given PT. If more than one RM modifies the
same rule set, then it needs to be ensured that a unique identifier
is chosen for each rule. A RM can accomplish this goal by retrieving
the already specified rule set and choosing a new identifier for a
rule that is different from the existing rule set.

6.2. Extensions

The policy framework defined in this document is meant to be
extensible towards specific application domains. Such an extension
is accomplished by defining conditions, actions, and transformations
that are specific to the desired application domain. Each extension
MUST define its own namespace.

Extensions cannot change the schema defined in this document, and
this schema is not expected to change except via revision to this
specification. Therefore, no versioning procedures for this schema
or namespace are provided.

7. Conditions

The access to data items needs to be matched with the rule set stored
at the PS. Each instance of a request has different attributes
(e.g., the identity of the requestor) that are used for
authorization. A rule in a rule set might have a number of
conditions that need to be met before executing the remaining parts
of a rule (i.e., actions and transformations). Details about rule
matching are described in Section 10. This document specifies only a
few conditions (i.e., identity, sphere, and validity). Further
condition elements can be added via extensions to this document. If
a child element of the <conditions> element is in a namespace that is
not known or not supported, then this child element evaluates to
FALSE.

As noted in Section 5, conditions are matched on equality or "greater
than" style comparisons, rather than regular expressions. Equality
is determined according to the rules for the data type associated
with the element in the schema given in Section 13, unless explicit
comparison steps are included in this document. For xs:anyURI types,
readers may wish to consult [2] for its discussion xs:anyURI, as well
as the text in Section 13.

7.1. Identity Condition

7.1.1. Overview

The identity condition restricts matching of a rule either to a
single entity or a group of entities. Only authenticated entities
can be matched; acceptable means of authentication are defined in
protocol-specific documents. If the <identity> element is absent,
identities are not considered, and thus, other conditions in the rule
apply to any user, authenticated or not.

The <identity> condition is considered TRUE if any of its child
elements (e.g., the <one/> and the <many/> elements defined in this
document) evaluate to TRUE, i.e., the results of the individual child
element are combined using a logical OR.

If a child element of the <identity> element is in a namespace that
is not known or not supported, then this child element evaluates to
FALSE.

7.1.2. Matching One Entity

   The <one> element matches the authenticated identity (as contained in
   the 'id' attribute) of exactly one entity or user. For
   considerations regarding the 'id' attribute, refer to Section 7.2.

   An example is shown below:

   <?xml version="1.0" encoding="UTF-8"?>
   <ruleset xmlns="urn:ietf:params:xml:ns:common-policy">

       <rule id="f3g44r1">
           <conditions>
               <identity>
                   <one id="sip:alice@example.com"/>
                   <one id="tel:+1-212-555-1234" />
                   <one id="mailto:bob@example.net" />
               </identity>
           </conditions>
           <actions/>
           <transformations/>
       </rule>

   </ruleset>

   This example matches if the authenticated identity of the WR is
   either sip:alice@example.com, tel:+1-212-555-1234, or
   mailto:bob@example.net.

7.1.3. Matching Multiple Entities

   The <many> element is a mechanism to perform authorization decisions
   based on the domain part of the authenticated identity. As such, it
   allows matching a large and possibly unknown number of users within a
   domain.

   Furthermore, it is possible to include one or multiple <except>
   elements to exclude either individual users or users belonging to a
   specific domain. Excluding individual entities is implemented using
   a <except id="..."/> statement. The semantic of the 'id' attribute
   of the <except> element has the same meaning as the 'id' attribute of
   the <one> element (see Section 7.2). Excluding users belonging to a
   specific domain is implemented using the <except domain="..."/>
   element that excludes any user from the indicated domain.

   If multiple <except> elements are listed as child elements of the
   <many> element, then the result of each <except> element is combined
   using a logical OR.

Common policy MUST either use UTF-8 or UTF-16 to store domain names
in the 'domain' attribute. For non-IDNs (Internationalized Domain
Names), lowercase ASCII SHOULD be used. For the comparison operation
between the value stored in the 'domain' attribute and the domain
value provided via the using protocol (referred to as "protocol
domain identifier"), the following rules are applicable:

1. Translate percent-encoding for either string.

2. Convert both domain strings using the ToASCII operation described
    in RFC 3490 [3].

3. Compare the two domain strings for ASCII equality, for each
    label. If the string comparison for each label indicates
    equality, the comparison succeeds. Otherwise, the domains are
    not equal.

If the conversion fails in step (2), the domains are not equal.

7.1.3.1. Matching Any Authenticated Identity

The <many/> element without any child elements or attributes matches
any authenticated user.

The following example shows such a rule that matches any
authenticated user:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy">

    <rule id="f3g44r5">
        <conditions>
            <identity>
              <many/>
            </identity>
        </conditions>
        <actions/>
        <transformations/>
    </rule>

</ruleset>
```

7.1.3.2. **Matching Any Authenticated Identity Except Enumerated
          Domains/Identities**

   The <many> element enclosing one or more <except domain="..."/>
   elements matches any user from any domain except those enumerated.
   The <except id="..."/> element excludes particular users. The
   semantics of the 'id' attribute of the <except> element is described
   in Section 7.2. The results of the child elements of the <many>
   element are combined using a logical OR.

   An example is shown below:

   <?xml version="1.0" encoding="UTF-8"?>
   <ruleset xmlns="urn:ietf:params:xml:ns:common-policy">

       <rule id="f3g44r1">
           <conditions>
               <sphere value="work"/>
               <identity>
                   <many>
                       <except domain="example.com"/>
                       <except domain="example.org"/>
                       <except id="sip:alice@bad.example.net"/>
                       <except id="sip:bob@good.example.net"/>
                       <except id="tel:+1-212-555-1234" />
                       <except id="sip:alice@example.com"/>
                   </many>
               </identity>
               <validity>
                   <from>2003-12-24T17:00:00+01:00</from>
                   <until>2003-12-24T19:00:00+01:00</until>
               </validity>
           </conditions>
           <actions/>
           <transformations/>
       </rule>

   </ruleset>

   This example matches all users except any user in example.com, or any
   user in example.org or the particular users alice@bad.example.net,
   bob@good.example.net, and the user with the telephone number
   'tel:+1-212-555-1234'. The last 'except' element is redundant since
   alice@example.com is already excluded through the first line.

7.1.3.3. Matching Any Authenticated Identity within a Domain Except Enumerated Identities

The <many> element with a 'domain' attribute and zero or more <except id="..."/> elements matches any authenticated user from the indicated domain except those explicitly enumerated. The semantics of the 'id' attribute of the <except> element is described in Section 7.2.

It is nonsensical to have domains in the 'id' attribute that do not match the value of the 'domain' attribute in the enclosing <many> element.

An example is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy">

    <rule id="f3g44r1">
        <conditions>
            <identity>
                <many domain="example.com">
                    <except id="sip:alice@example.com"/>
                    <except id="sip:bob@example.com"/>
                </many>
            </identity>
        </conditions>
        <actions/>
        <transformations/>
    </rule>

</ruleset>
```

This example matches any user within example.com (such as carol@example.com) except alice@example.com and bob@example.com.

7.2. Single Entity

The 'id' attribute used in the <one> and in the <except> element refers to a single entity. In the subsequent text, we use the term 'single-user entity' as a placeholder for the <one> and the <except> element. The <except> element fulfills the purpose of excluding elements from the solution set.

A single-user entity matches the authenticated identity (as contained in the 'id' attribute) of exactly one entity or user. If there is a match, the single-user entity is considered TRUE. The single-user entity MUST NOT contain a 'domain' attribute.

The 'id' attribute contains an identity that MUST first be expressed
as a URI. Applications using this framework must describe how the
identities they are using can be expressed as URIs.

7.3. Sphere

The <sphere> element belongs to the group of condition elements. It
can be used to indicate a state (e.g., 'work', 'home', 'meeting',
'travel') the PT is currently in. A sphere condition matches only if
the PT is currently in the state indicated. The state may be
conveyed by manual configuration or by some protocol. For example,
RPID [10] provides the ability to inform the PS of its current
sphere. The application domain needs to describe in more detail how
the sphere state is determined. Switching from one sphere to another
causes a switch between different modes of visibility. As a result,
different subsets of rules might be applicable.

The content of the 'value' attribute of the <sphere> element MAY
contain more than one token. The individual tokens MUST be separated
by a blank character. A logical OR is used for the matching the
tokens against the sphere settings of the PT. As an example, if the
content of the 'value' attribute in the sphere attribute contains two
tokens 'work' and 'home' then this part of the rule matches if the
sphere for a particular PT is either 'work' OR 'home'. To compare
the content of the 'value' attribute in the <sphere> element with the
stored state information about the PT's sphere setting a
case-insensitive string comparison MUST be used for each individual
token. There is neither a registry for these values nor a language-
specific indication of the sphere content. As such, the tokens are
treated as opaque strings.

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy">

  <rule id="f3g44r2">
    <conditions>
      <sphere value="work"/>
      <identity>
        <one id="sip:andrew@example.com"/>
      </identity>
    </conditions>
    <actions/>
    <transformations/>
  </rule>
```

```
     <rule id="y6y55r2">
       <conditions>
         <sphere value="home"/>
         <identity>
            <one id="sip:allison@example.com"/>
         </identity>
       </conditions>
       <actions/>
       <transformations/>
     </rule>

     <rule id="z6y55r2">
       <conditions>
         <identity>
               <one id="sip:john@doe.example.com"/>
         </identity>
         <sphere value="home work"/>
       </conditions>
       <actions/>
       <transformations/>
     </rule>
  </ruleset>
```

The rule example above illustrates that the rule with the entity
andrew@example.com matches if the sphere is been set to 'work'. In
the second rule, the entity allison@example.com matches if the sphere
is set to 'home'. The third rule also matches since the value in the
sphere element also contains the token 'home'.

7.4. Validity

The <validity> element is the third condition element specified in
this document. It expresses the rule validity period by two
attributes, a starting and an ending time. The validity condition is
TRUE if the current time is greater than or equal to at least one
<from> child, but less than the <until> child after it. This
represents a logical OR operation across each <from> and <until>
pair. Times are expressed in XML dateTime format. A rule maker
might not always have access to the PS to invalidate some rules that
grant permissions. Hence, this mechanism allows invalidating granted
permissions automatically without further interaction between the
rule maker and the PS. The PS does not remove the rules; instead the
rule maker has to clean them up.

An example of a rule fragment is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy">

  <rule id="f3g44r3">
    <conditions>
        <validity>
            <from>2003-08-15T10:20:00.000-05:00</from>
            <until>2003-09-15T10:20:00.000-05:00</until>
        </validity>
    </conditions>
    <actions/>
    <transformations/>
  </rule>
</ruleset>
```

The <validity> element MUST have the <from> and <until> subelements
in pairs. Multiple <from> and <until> elements might appear in pairs
(i.e., without nesting of <from> and <until> elements). Using
multiple <validity> elements as subelements of the <conditions>
element is not useful since all subelements of the <conditions>
element are combined as a logical AND.

8. Actions

While conditions are the 'if'-part of rules, actions and
transformations form their 'then'-part. The actions and
transformations parts of a rule determine which operations the PS
MUST execute after having received from a WR a data access request
that matches all conditions of this rule. Actions and
transformations only permit certain operations; there is no 'deny'
functionality. Transformations exclusively specify PS-side
operations that lead to a modification of the data items requested by
the WR. Regarding location data items, for instance, a
transformation could force the PS to lower the precision of the
location information that is returned to the WR.

Actions, on the other hand, specify all remaining types of operations
the PS is obliged to execute, i.e., all operations that are not of
transformation type. Actions are defined by application-specific
usages of this framework. The reader is referred to the
corresponding extensions to see examples of such elements.

9. Transformations

   Two sub-parts follow the conditions part of a rule: transformations
   and actions. As defined in Section 8, transformations specify
   operations that the PS MUST execute and that modify the result that
   is returned to the WR. This functionality is particularly helpful in
   reducing the granularity of information provided to the WR, as, for
   example, required for location privacy. Transformations are defined
   by application-specific usages of this framework.

   A simple transformation example is provided in Section 10.

10. Procedure for Combining Permissions

10.1. Introduction

   This section describes how rules are selected and how actions and
   permissions are determined. When a PS receives a request for access
   to privacy-sensitive data, the request is matched against the rule
   set. A rule matches if all conditions contained as child elements in
   the <conditions> element of a rule evaluate to TRUE. Each type of
   condition defines when it is TRUE. All rules where the conditions
   match the request form the matching rule set. The permissions in the
   matching rule set are combined using a set of combining rules (CRs)
   described in Section 10.2.

10.2. Combining Rules (CRs)

   Each type of permission is combined across all matching rules. Each
   type of action or transformation is combined separately and
   independently. The combining rules generate a combined permission.
   The combining rules depend only on the data type of permission. If a
   particular permission type has no value in a rule, it assumes the
   lowest possible value for that permission for the purpose of
   computing the combined permission. That value is given by the data
   type for booleans (FALSE) and sets (empty set), and MUST be defined
   by any extension to the Common Policy for other data types.

   For boolean permissions, the resulting permission is TRUE if and only
   if at least one permission in the matching rule set has a value of
   TRUE and FALSE otherwise. For integer, real-valued and date-time
   permissions, the resulting permission is the maximum value across the
   permission values in the matching set of rules. For sets, it is the
   union of values across the permissions in the matching rule set.

10.3. Example

   In the following example we illustrate the process of combining
   permissions. We will consider three conditions for our purpose,
   namely those of name identity (WR-ID), sphere, and validity
   (from,until). The ID column is used as a rule identifier. For
   editorial reasons we omit the domain part of the WR's identity.

   We use two actions in our example, namely X and Y. The values of X
   and Y are of data types Boolean and Integer, respectively.

   The transformation, referred to as Z, uses values that can be set
   either to '+' (or 3), 'o' (or 2) or '-' (or 1). Permission Z allows
   us to show the granularity reduction whereby a value of '+' shows the
   corresponding information unrestricted, and '-' shows nothing. This
   permission might be related to location information or other presence
   attributes like mood. Internally, we use the data type Integer for
   computing the permission of this attribute.

   The label 'NULL' in the table indicates that no value is available
   for a particular cell.

```
      Conditions                    Actions/Transformations
   +-------------------------------+--------------------+
   | Id  WR-ID    sphere from until |  X       Y    Z    |
   +-------------------------------+--------------------+
   |  1   bob      home   A1   A2   |  TRUE    10   o    |
   |  2   alice    work   A1   A2   |  FALSE   5    +    |
   |  3   bob      work   A1   A2   |  TRUE    3    -    |
   |  4   tom      work   A1   A2   |  TRUE    5    +    |
   |  5   bob      work   A1   A3   |  NULL    12   o    |
   |  6   bob      work   B1   B2   |  FALSE   10   -    |
   +-------------------------------+--------------------+
```

   Again for editorial reasons, we use the following abbreviations for
   the two <validity> attributes 'from' and 'until':

```
   A1=2003-12-24T17:00:00+01:00
   A2=2003-12-24T21:00:00+01:00
   A3=2003-12-24T23:30:00+01:00
   B1=2003-12-22T17:00:00+01:00
   B2=2003-12-23T17:00:00+01:00
```

   Note that B1 < B2 < A1 < A2 < A3.

   The entity 'bob' acts as a WR and requests data items. The rule set
   consists of the six rules shown in the table and identified by the
   values 1 to 6 in the 'Id' column. The PS receives the query at

2003-12-24T17:15:00+01:00, which falls between A1 and A2. In our
example, we assume that the sphere value of the PT is currently set
to 'work'.

As a first step, it is necessary to determine which rules fire by
evaluating the conditions part of each of them.

Rule 1 does not match since the sphere condition does not match.
Rule 2 does not match as the identity of the WR (here 'alice') does
not equal 'bob'. Rule 3 matches since all conditions evaluate to
TRUE. Rule 4 does not match as the identity of the WR (here 'tom')
does not equal 'bob'. Rule 5 matches. Rule 6 does not match since
the rule is not valid anymore.

Only rules 3 and 5 fire. We use the actions and transformations part
of these two rules to determine the combined permission, as shown
below.

```
          Actions/Transformations
   +-----+-----------------------+
   | Id  | X        Y       Z    |
   +-----+-----------------------+
   |  3  | TRUE      3      -    |
   |  5  | NULL     12      o    |
   +-----+-----------------------+
```

Each column is treated independently. The combined value of X is set
to TRUE since the NULL value equals FALSE according to the
description in Section 10.2. For the column with the name Y, we
apply the maximum of 3 and 12, so that the combined value of Y is 12.
For column Z, we again compute the maximum of 'o' and '-' (i.e., 2
and 1) which is 'o' (2).

The combined permission for all three columns is therefore:

```
          Actions/Transformations
        +-----------------------+
        | X        Y       Z    |
        +-----------------------+
        | TRUE     12      o    |
        +-----------------------+
```

11. Meta Policies

   Meta policies authorize a rule maker to insert, update, or delete a
   particular rule or an entire rule set. Some authorization policies
   are required to prevent unauthorized modification of rule sets. Meta
   policies are outside the scope of this document.

   A simple implementation could restrict access to the rule set only to
   the PT but more sophisticated mechanisms could be useful. As an
   example of such policies, one could think of parents configuring the
   policies for their children.

12. Example

   This section gives an example of an XML document valid with respect
   to the XML schema defined in Section 13. Semantically richer
   examples can be found in documents that extend this schema with
   application-domain-specific data (e.g., location or presence
   information).

   Below a rule is shown with a condition that matches for a given
   authenticated identity (bob@example.com) and within a given time
   period. Additionally, the rule matches only if the target has set
   its sphere to 'work'.

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy">

    <rule id="f3g44r1">
        <conditions>
            <identity>
                <one id="sip:bob@example.com"/>
            </identity>
            <sphere value="work"/>
            <validity>
                <from>2003-12-24T17:00:00+01:00</from>
                <until>2003-12-24T19:00:00+01:00</until>
            </validity>
        </conditions>
        <actions/>
        <transformations/>
    </rule>
</ruleset>
```

13. XML Schema Definition

   This section provides the XML schema definition for the common policy
   markup language described in this document.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:common-policy"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- /ruleset -->
    <xs:element name="ruleset">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="xs:anyType">
                    <xs:sequence>
                        <xs:element name="rule" type="cp:ruleType"
                        minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!-- /ruleset/rule -->
    <xs:complexType name="ruleType">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                <xs:sequence>
                    <xs:element name="conditions"
                    type="cp:conditionsType" minOccurs="0"/>
                    <xs:element name="actions"
                    type="cp:extensibleType" minOccurs="0"/>
                    <xs:element name="transformations"
                    type="cp:extensibleType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="id" type="xs:ID" use="required"/>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <!-- //rule/conditions -->
    <xs:complexType name="conditionsType">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                <xs:choice maxOccurs="unbounded">
                    <xs:element name="identity"
                    type="cp:identityType" minOccurs="0"/>
                    <xs:element name="sphere"
                    type="cp:sphereType" minOccurs="0"/>
```

```
                  <xs:element name="validity"
                  type="cp:validityType" minOccurs="0"/>
                  <xs:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
              </xs:choice>
          </xs:restriction>
      </xs:complexContent>
</xs:complexType>
<!-- //conditions/identity -->
<xs:complexType name="identityType">
    <xs:complexContent>
        <xs:restriction base="xs:anyType">
            <xs:choice  minOccurs="1" maxOccurs="unbounded">
                <xs:element name="one" type="cp:oneType"/>
                <xs:element name="many" type="cp:manyType"/>
                <xs:any namespace="##other" processContents="lax"/>
            </xs:choice>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
<!-- //identity/one -->
<xs:complexType name="oneType">
    <xs:complexContent>
        <xs:restriction base="xs:anyType">
            <xs:sequence>
                <xs:any namespace="##other"
                minOccurs="0" processContents="lax"/>
            </xs:sequence>
            <xs:attribute name="id"
            type="xs:anyURI" use="required"/>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
<!-- //identity/many -->
<xs:complexType name="manyType">
    <xs:complexContent>
        <xs:restriction base="xs:anyType">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="except" type="cp:exceptType"/>
                <xs:any namespace="##other"
                minOccurs="0" processContents="lax"/>
            </xs:choice>
            <xs:attribute name="domain"
            use="optional" type="xs:string"/>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
<!-- //many/except -->
```

```
    <xs:complexType name="exceptType">
        <xs:attribute name="domain" type="xs:string" use="optional"/>
        <xs:attribute name="id" type="xs:anyURI" use="optional"/>
    </xs:complexType>
    <!-- //conditions/sphere -->
    <xs:complexType name="sphereType">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                <xs:attribute name="value"
                type="xs:string" use="required"/>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <!-- //conditions/validity -->
    <xs:complexType name="validityType">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                <xs:sequence minOccurs="1" maxOccurs="unbounded">
                    <xs:element name="from" type="xs:dateTime"/>
                    <xs:element name="until" type="xs:dateTime"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <!-- //rule/actions or //rule/transformations -->
    <xs:complexType name="extensibleType">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                <xs:sequence>
                    <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

14. Security Considerations

This document describes a framework for policies. This framework is
intended to be enhanced elsewhere by application-domain-specific
data. Security considerations are to a great extent application-data
dependent, and therefore need to be covered by documents that extend
the framework defined in this specification. However, new action and
transformation permissions along with their allowed values must be
defined in a way so that the usage of the permissions combining rules
of Section 10 does not lower the level of privacy protection. See
Section 10 for more details on this privacy issue.

15. IANA Considerations

This section registers a new XML namespace, a new XML schema, and a
new MIME type. This section registers a new XML namespace per the
procedures in [4].

15.1. Common Policy Namespace Registration

URI: urn:ietf:params:xml:ns:common-policy

Registrant Contact: IETF GEOPRIV working group, Henning Schulzrinne
   (hgs+geopriv@cs.columbia.edu).

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1"/>
  <title>Common Policy Namespace</title>
</head>
<body>
  <h1>Namespace for Common Authorization Policies</h1>
  <h2>urn:ietf:params:xml:ns:common-policy</h2>
<p>See <a href="ftp://ftp.rfc-editor.org/in-notes/rfc4745.txt">
  RFC 4745</a>.</p>
</body>
</html>
END
```

15.2. **Content-type Registration for 'application/auth-policy+xml'**

   This specification requests the registration of a new MIME type
   according to the procedures of RFC 4288 [5] and guidelines in RFC
   3023 [6].

   MIME media type name: application

   MIME subtype name: auth-policy+xml

   Mandatory parameters: none

   Optional parameters: charset

      Indicates the character encoding of enclosed XML.

   Encoding considerations:

      Uses XML, which can employ 8-bit characters, depending on the
      character encoding used. See RFC 3023 [6], Section 3.2.

   Security considerations:

      This content type is designed to carry authorization policies.
      Appropriate precautions should be adopted to limit disclosure of
      this information. Please refer to Section 14 of RFC 4745 and to
      the security considerations described in Section 10 of RFC 3023
      [6] for more information.

   Interoperability considerations: None

   Published specification: RFC 4745

   Applications which use this media type:

      Presence- and location-based systems

   Additional information:

      Magic Number: None

      File Extension: .apxml

      Macintosh file type code: 'TEXT'

   Personal and email address for further information:
      Hannes Tschofenig, Hannes.Tschofenig@siemens.com

   Intended usage: LIMITED USE

   Author:

      This specification is a work item of the IETF GEOPRIV working
      group, with mailing list address <geopriv@ietf.org>.

   Change controller:

      The IESG <iesg@ietf.org>

[15.3](#). Common Policy Schema Registration

   URI: urn:ietf:params:xml:schema:common-policy

   Registrant Contact: IETF GEOPRIV working group, Henning Schulzrinne
      (hgs+geopriv@cs.columbia.edu).

   XML: The XML schema to be registered is contained in [Section 13](#).
      Its first line is

   <?xml version="1.0" encoding="UTF-8"?>

   and its last line is

   </xs:schema>

[16](#). References

[16.1](#). Normative References

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

   [2] Duerst, M. and M. Suignard, "Internationalized Resource
        Identifiers (IRIs)", [RFC 3987](#), January 2005.

   [3] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing
        Domain Names in Applications (IDNA)", [RFC 3490](#), March 2003.

   [4] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#),
        January 2004.

   [5] Freed, N. and J. Klensin, "Media Type Specifications and
        Registration Procedures", [BCP 13](#), [RFC 4288](#), December 2005.

   [6] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types",
        [RFC 3023](#), January 2001.

16.2. Informative References

[7]   Rosenberg, J., "Presence Authorization Rules", Work in Progress,
      June 2006.

[8]   Schulzrinne, H., Tschofenig, H., Morris, J., Cuellar, J., and J.
      Polk, "A Document Format for Expressing Privacy Preferences for
      Location Information", Work in Progress, February 2006.

[9]   Cuellar, J., Morris, J., Mulligan, D., Peterson, J., and J.
      Polk, "Geopriv Requirements", RFC 3693, February 2004.

[10] Schulzrinne, H., Gurbani, V., Kyzivat, P., and J. Rosenberg,
      "RPID: Rich Presence Extensions to the Presence Information Data
      Format (PIDF)", RFC 4480, July 2006.

Appendix A. Contributors

   We would like to thank Christian Guenther for his help with initial
   versions of this document.

Appendix B. Acknowledgments

   This document is partially based on the discussions within the IETF
   GEOPRIV working group. Discussions at the Geopriv Interim Meeting
   2003 in Washington, D.C., helped the working group to make progress
   on the authorization policies based on the discussions among the
   participants.

   We particularly want to thank Allison Mankin <mankin@psg.com>,
   Randall Gellens <rg+ietf@qualcomm.com>, Andrew Newton
   <anewton@ecotroph.net>, Ted Hardie <hardie@qualcomm.com>, and Jon
   Peterson <jon.peterson@neustar.biz> for discussing a number of
   details with us. They helped us to improve the quality of this
   document. Allison, Ted, and Andrew also helped us to make good
   progress with the internationalization support of the identifier/
   domain attributes.

   Furthermore, we would like to thank the IETF SIMPLE working group for
   their discussions of J. Rosenberg's draft on presence authorization
   policies. We would also like to thank Stefan Berg, Murugaraj
   Shanmugam, Christian Schmidt, Martin Thomson, Markus Isomaki, Aki
   Niemi, Eva Maria Leppanen, Josip Matanovic, and Mark Baker for their
   comments. Martin Thomson helped us with the XML schema. Mark Baker
   provided a review of the media type. Scott Brim provided a review on
   behalf of the General Area Review Team.

Authors' Addresses

    Henning Schulzrinne
    Columbia University
    Department of Computer Science
    450 Computer Science Building
    New York, NY 10027
    USA

    Phone: +1 212 939 7042
    EMail: schulzrinne@cs.columbia.edu
    URI: http://www.cs.columbia.edu/~hgs

    Hannes Tschofenig
    Siemens Networks GmbH & Co KG
    Otto-Hahn-Ring 6
    Munich, Bavaria 81739
    Germany

    EMail: Hannes.Tschofenig@siemens.com
    URI: http://www.tschofenig.com

    John B. Morris, Jr.
    Center for Democracy and Technology
    1634 I Street NW, Suite 1100
    Washington, DC 20006
    USA

    EMail: jmorris@cdt.org
    URI: http://www.cdt.org

    Jorge R. Cuellar
    Siemens
    Otto-Hahn-Ring 6
    Munich, Bavaria 81739
    Germany

    EMail: Jorge.Cuellar@siemens.com

James Polk
Cisco
2200 East President George Bush Turnpike
Richardson, Texas 75082
USA

EMail: jmpolk@cisco.com

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, New York 07054
USA

EMail: jdrosen@cisco.com
URI: http://www.jdrosen.net

Full Copyright Statement

Intellectual Property

Acknowledgement