SIMPLE
Internet-Draft
Expires: August 22, 2005

J. Rosenberg
Cisco Systems

February 21, 2005

An Extensible Markup Language (XML) Representation for Expressing Policy Capabilities
draft-rosenberg-simple-common-policy-caps-02

Status of this Memo

Copyright Notice

Abstract

An important component of presence and location services is policy. Policy systems allow the presentity or location target to grant access to specific pieces of information to specific watchers or requestors. These policy systems can be extremely simple, allowing a user to accept or block requests based solely on the identity of the requestor, to extremely complex, allowing for time based rules that grant or deny specific pieces of information. Policy systems often

support vendor proprietary features. To allow for interoperability between clients which set such policies, and servers which execute them, it is necessary for clients to be able to determine the capabilities of the server to which it is connected. This specification defines an Extensible Markup Language (XML) based format for expressing such capabilities.

Table of Contents

1. Terminology 3
2. Introduction 3
3. Overview of Operation 3
4. Structure of Policy Capabilities 4
5. XML Schema 5
6. Example Document 6
7. Usage with XCAP 6
7.1 Application Unique ID 6
7.2 XML Schema 7
7.3 Default Namespace 7
7.4 MIME Type 7
7.5 Validation Constraints 7
7.6 Data Semantics 7
7.7 Naming Conventions 7
7.8 Resource Interdependencies 7
7.9 Authorization Policies 7
8. Security Considerations 8
9. IANA Considerations 8
9.1 XCAP Application Usage ID 8
9.2 MIME Type Registration 8
9.3 URN Sub-Namespace Registrations 9
9.4 XML Schema Registration 10
10. References 10
10.1 Normative References 10
10.2 Informative References 11
Author's Address 12
Intellectual Property and Copyright Statements 13

1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant implementations.

2. Introduction

An important component of presence [10] and location services [11] is policy. Policy systems allow the presentity or location target (referred to generically as the Presentity Target (PT)) to grant access to specific pieces of information to specific watchers or requestors (referred to as a WR) [5]. These policy systems can be extremely simple, allowing a PT to accept or block requests based solely on the identity of the WR, to extremely complex, allowing for time based rules that grant or deny specific pieces of information. [5] specifies a generic format for representing these policies, using the Extensible Markup Language (XML). These policies consist of conditions, actions, and transformations. That specification defines very few actual conditions, actions or transformations. Rather, it leaves such definitions to actual policy systems, such as [5] for location services, and [13] for presence services.

In addition to the conditions, actions and transformations specificed in the documents referenced above, policy systems often support vendor proprietary features. It is also anticipated that future specifications will be continually developed that add new types of policies. This presents an interoperability challenge. Clients may support policies that are not supported by the servers they are using. This could lead to protocol failures or poor user experiences.

To address this problem, it is necessary for a capability declaration system to be put in place. This specification defines a general purpose format for representing policy capabilities within the framework established in [5].

3. Overview of Operation

This specification defines an XML-based document format that allows a server to represent its capabilities. When a client, acting as an agent of a PT, starts up, it obtains this document from its policy server. This specification does not prescribe a singular means of transporting such a document between the server and the client. It is anticipated that different systems may use different techniques. However, for systems that make use of the XML Configuration Access Protocol (XCAP) [4], Section 7 defines an application usage that

allows for the transfer of the document using XCAP.

Once the document has been obtained by the client, it can determine which actions, conditions and transformations are understood by the server. This set is matched against those supported by the client. Those actions, conditions and transformations supported by the client, but not by the server, can be "greyed out" from a user interface, for example.

It is anticipated that the capabilities of the server can change over time. As a result, it is RECOMMENDED that clients obtain a fresh copy of the capabilities document each time they start.

4. Structure of Policy Capabilities

A policy capabilities document is an XML [6] document that MUST be well-formed and SHOULD be valid. Policy capabilities documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying policy capabilities documents and document fragments. The namespace URI for elements defined for this purpose is a URN [2], using the namespace identifier 'ietf' defined by [3] and extended by [7]. This URN is:

urn:ietf:params:xml:ns:policy-capabilities

A policy capabilities document is structured much like a policy document [5]. The root element is <policy-capabilities>. This element has three children - <conditions>, <actions>, and <transformations>. Each of these contain a list of the condition, action, and transformation capabilities, respectively. Generally speaking, each specific condition, action or transformation element (referred to as a capability element) is empty, unless it requires additional content to further refine the capability.

This specification defines three capability elements - <identity>, <validity>, and <sphere> matching the three conditions defined in [5]. Other specifications that define additional policies MUST also define matching capability elements. When such elements are defined, that specification MUST indicate, for each capability element, the corresponding action, condition, or permission elements which can be placed into a common policy document [5]. The name of the capability element need not match the name of the corresponding condition, action or transformation, although using a matching name is RECOMMENDED.

A server constructing a document to represent its capabilities MUST include all of its capabilities, even if those capabilities represent mandatory-to-implement features. However, the server MAY indicate

differing sets of capabilities to different users. As such, the set of capabilities combines both the ability and the willingness to support those capabilities.

5. XML Schema

```
<xs:schema
 targetNamespace="urn:ietf:params:xml:ns:policy-capabilities"
 xmlns="urn:ietf:params:xml:ns:policy-capabilities"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="policy-capabilities">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="conditions" minOccurs="0">
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="identity"/>
       <xs:element ref="sphere"/>
       <xs:element ref="validity"/>
       <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="actions" minOccurs="0">
     <xs:complexType>
      <xs:sequence>
       <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="transformations" minOccurs="0">
     <xs:complexType>
      <xs:sequence>
       <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:complexType name="emptyType">
  <xs:complexContent>
   <xs:restriction base="xs:anyType"/>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="validity" type="emptyType"/>
```

```
 <xs:element name="sphere" type="emptyType"/>
 <xs:element name="identity" type="emptyType"/>
</xs:schema>
```

6. Example Document

The following document indicates that the identity, validity and sphere conditions are supported. It also indicates that a vendor-specific capability, called <happy>, is supported, a vendor specific action capability, <log>, is supported, and a vendor-specific transformation capability - <min-security> is supported. The specification which defines these capabilities would indicate the specific elements which could be included in a policy document.

```
<?xml version="1.0" encoding="UTF-8"?>
<policy-capabilities
 xmlns="urn:ietf:params:xml:ns:policy-capabilities"
 xmlns:cp="urn:ietf:params:xml:ns:policy-capabilities"
 xmlns:vpp="urn:ietf:params:xml:ns:vpp"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <conditions>
  <identity/>
  <sphere/>
  <validity/>
  <vpp:happy/>
 </conditions>
 <actions>
   <vpp:log/>
 </actions>
 <transformations>
   <vpp:min-security/>
 </transformations>
</policy-capabilities>
```

7. Usage with XCAP

The following section defines the details necessary for clients to read supported permissions documents from a server using XCAP.

7.1 Application Unique ID

XCAP requires application usages to define an application unique ID (AUID) in either the IETF tree or a vendor tree. This specification

defines the "policy-capabilities" AUID within the IETF tree, via the IANA registration in Section 9.

7.2 XML Schema

The schema is defined in Section 5.

7.3 Default Namespace

The default namespace used in evaluating a URI is urn:ietf:params:xml:ns:policy-capabilities.

7.4 MIME Type

The MIME type for this document is "application/policy-caps+xml".

7.5 Validation Constraints

This specification does not introduce any additional validation constraints beyond those defined in the schema.

7.6 Data Semantics

Semantics for the document content are provided in Section 4.

7.7 Naming Conventions

When a client starts, it can fetch the capabilities of the server in one of two places. If the server capabilities differ on a user by user basis, the capabilities for user foo can be found in the document with filename "cap.xml" in the user's home directory for this application usage. A client SHOULD check this file first. If this document doesn't exist, the client should next check for the system wide permissions by reading the document with filename "cap.xml" in the global directory for this application usage.

7.8 Resource Interdependencies

Policy capability documents are usually either created automatically by the server, or modified by administrator to reflect the features of a server. For those users that have access to the full capabilities of the server, a change in the server-wide capabilities, expressed in the "cap.xml" file in the global directory, MUST be reflected in any "cap.xml" documents in user's home directories.

7.9 Authorization Policies

This application usage does not use the default XCAP authorization

policies.

A user cannot modify the supported permissions document, they can only read it. Write access is granted only to administrators.

A user can read the "cap.xml" document in the global directory, but cannot modify it. Write access is granted only to administrators.

8. Security Considerations

Policy capability documents reveal capability information about a server. This information can potentially be used by an enterprise to determine the features found in competitive products. However, such information could just as easily be obtained through other means, for example, by signing up as a legitimate user of the competitive service. Because supported permission documents can vary by user to user, they can also reveal information about the grade of service offered to a particular user. However, this information does not appear particularly sensitive. As a result, encryption of these documents is not terribly important.

If an attacker can modify the contents of a supported permission document as it passes from client to server, the attacker can remove capability elements, therefore reducing the level of service received by the client. This can therefore form a type of denial-of-service attack. As a result, systems which transfer these documents SHOULD provide for message integrity.

9. IANA Considerations

There are several IANA considerations associated with this specification.

9.1 XCAP Application Usage ID

This section registers an XCAP Application Unique ID (AUID) according to the IANA procedures defined in [4].

Name of the AUID: policy-capabilities

Description: Policy capability documents describe the capabilities of a policy server to support different conditions, actions, and transformations, as defined in [5].

9.2 MIME Type Registration

This specification requests the registration of a new MIME type

according to the procedures of RFC 2048 [8] and guidelines in RFC 3023 [9].

MIME media type name: application

MIME subtype name: policy-caps+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [9].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [9].

Security considerations: See Section 10 of RFC 3023 [9] and Section 8 of RFC XXXX [[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number of this specification]].

Interoperability considerations: none.

Published specification: RFC XXXX [[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number of this specification]]

Applications which use this media type: This document type has been used to support capabilities for presence and geolocation.

Additional Information:

Magic Number: None

File Extension: .pcp

Macintosh file type code: "TEXT"

Personal and email address for further information: Jonathan Rosenberg, jdrosen@jdrosen.net

Intended usage: COMMON

Author/Change controller: The IETF.

9.3 URN Sub-Namespace Registrations

This section registers a new XML namespace, as per the guidelines in [7]

URI: The URI for this namespace is
urn:ietf:params:xml:ns:policy-capabilities.

Registrant Contact: IETF, SIMPLE working group, (simple@ietf.org),
Jonathan Rosenberg (jdrosen@jdrosen.net).

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>Policy Capabilities Namespace</title>
</head>
<body>
  <h1>Namespace for Policy Capabilities</h1>
  <h2>urn:ietf:params:xml:ns:policy-capabilities</h2>
  <p>See <a href="[URL of published RFC]">RFCXXXX[[NOTE
TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this
specification.</a>.</p>
</body>
</html>
END
```

9.4 XML Schema Registration

This section registers an XML schema as per the procedures in [7].

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

Registrant Contact: IETF, SIMPLE working group, (simple@ietf.org),
Jonathan Rosenberg (jdrosen@jdrosen.net).

The XML for this schema can be found as the sole content of
Section 5.

10. References

10.1 Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement

Levels", BCP 14, RFC 2119, March 1997.

[2] Moats, R., "URN Syntax", RFC 2141, May 1997.

[3] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[4] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-06 (work in progress), February 2005.

[5] Schulzrinne, H., "A Document Format for Expressing Privacy Preferences", draft-ietf-geopriv-common-policy-03 (work in progress), October 2004.

[6] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C FirstEdition REC-xml-20001006, October 2000.

[7] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[8] Freed, N., Klensin, J. and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 2048, November 1996.

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

10.2 Informative References

[10] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

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

[12] Schulzrinne, H., "A Document Format for Expressing Privacy Preferences for Location Information", draft-ietf-geopriv-policy-05 (work in progress), November 2004.

[13] Rosenberg, J., "Presence Authorization Rules", draft-ietf-simple-presence-rules-01 (work in progress), October 2004.

Author's Address

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@cisco.com
URI: http://www.jdrosen.net

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.