TCG Trusted Network Connect

TNC IF-MAP Binding for SOAP

Specification Version 1.1
Revision 5
18 May 2009
Published

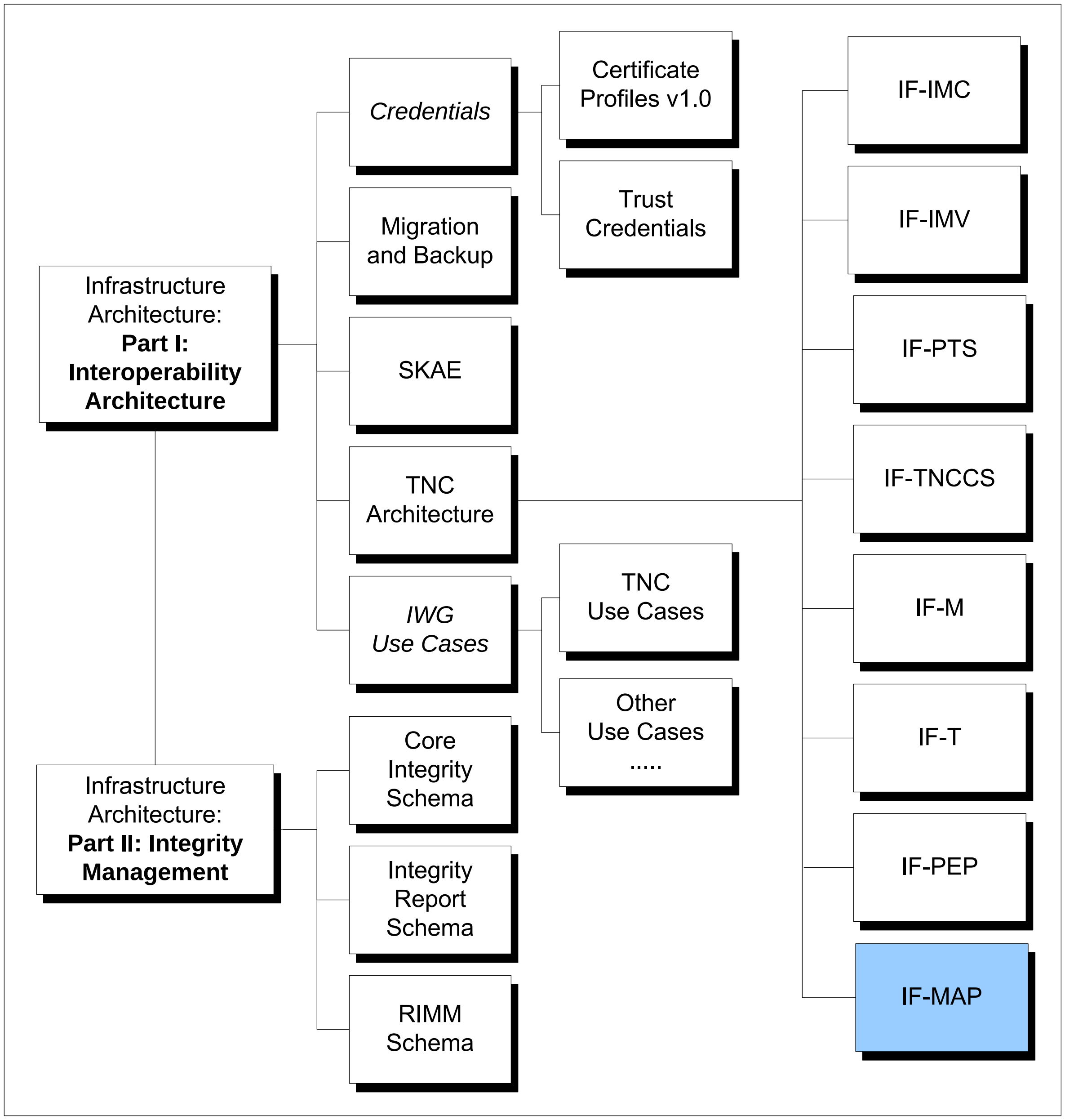
Contact:

admin@trustedcomputinggroup.org

TCG PUBLISHED

TCG

IWG TNC Document Roadmap



Acknowledgements

The TCG wishes to thank all those who contributed to this specification. This document builds on considerable work done in the various working groups in the TCG.

Scott Kelly	**Aruba Networks**
Amit Agarwal	**Avaya**
Jeffery Dion	**Boeing**
Steven Venema	**Boeing**
Peter Wrobel	**CESG**
Mark Townsend	**Enterasys**
Mahalingam Mani	**Avaya**
Michael McDaniels	**Extreme Networks**
Hannes Burmeister	**Fachhochschule Hannover**
Michael Klaas	**Fachhochschule Hannover**
Jussi Salzwedel	**Fachhochschule Hannover**
Hidenobu Ito	**Fujitsu Limited**
Seigo Kotani	**Fujitsu Limited**
Houcheng Lee	**Fujitsu Limited**
Sung Lee	**Fujitsu Limited**
Mauricio Sanchez	**Hewlett-Packard**
Ren Lanfang	**Huawei Technologies**
Jiwei Wei	**Huawei Technologies**
Han Yin	**Huawei Technologies**
Diana Arroyo	**IBM**
Guha Prasad Venkataraman	**IBM**
Sean Convery	**Identity Engines**
Chris Hessing	**Identity Engines**
Morteza Ansari	**Infoblox**
Stuart Bailey (Editor)	**Infoblox**
Rod Murchison	**Infoblox**
Ivan Pulleyn	**Infoblox**
Ravi Sahita	**Intel Corporation**
Ned Smith	**Intel Corporation**
Josh Howlett	**JANET (UK)**
Yan Avlasov	**Juniper Networks**
Roger Chickering (Editor)	**Juniper Networks**
Charles Goldberg	**Juniper Networks**
Steve Hanna (TNC co-chair)	**Juniper Networks**
Clifford Kahn	**Juniper Networks**
PJ Kirner (Editor)	**Juniper Networks**
Lisa Lorenzin (Editor)	**Juniper Networks**
John Jerrim	**Lancope**
Tom Price	**Lumeta**
Matt Webster	**Lumeta**
Ryan Hurst	**Microsoft**
Sandilya Garimella	**Motorola**
Meenakshi Kaushik	**Nortel**
Paul Sangster (TNC co-chair)	**Symantec Corporation**
Hannes Burmeister	**University of Applied Sciences and Arts in Hannover**
Michael Klaas	**University of Applied Sciences and**

	Arts in Hannover
Jussi Salzwedel	**University of Applied Sciences and Arts in Hannover**
Brad Upson	**UNH InterOperability Lab**
Lauren Giroux	**US National Security Agency**
Chris Salter	**US National Security Agency**
Thomas Hardjono	**Wave Systems**
Greg Kazmierczak	**Wave Systems**

Special thanks to the members of the TNC contributing to this document:

Morteza Ansari	Infoblox
John Jerrim	Lancope
Meenakshi Kaushik	Nortel Networks
PJ Kirner	Juniper Networks
Mahalingam Mani	Avaya
Ivan Pulleyn	Infoblox
Mauricio Sanchez	HP

Table of Contents

1 Introduction

1.1 Scope and Audience

The Trusted Network Connect Working Group (TNC-WG) has defined an open solution architecture that enables network operators to enforce policies regarding the security state of endpoints in order to determine whether to grant access to a requested network infrastructure. Part of the TNC architecture is IF-MAP, a standard interface between the Metadata Access Point and other elements of the TNC architecture. This document defines and specifies IF-MAP.

Architects, designers, developers and technologists who wish to implement, use, or understand IF-MAP should read this document carefully. Before reading this document any further, the reader should review and understand the TNC architecture as described in [1].

1.2 Keywords

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [2]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

1.3 Changes since IF-MAP 1.0

The InvalidSessionID errorCode was added to the schema. invalidSessionID was described in IF-MAP 1.0, but was missing from the schema.

The new-session and attach-session elements moved from the SOAP Header to the SOAP Body. In addition, the session-id and publisher-id elements in responses to new-session and attach-session requests moved from the SOAP Header to the SOAP Body. In all other requests and responses the session-id element remains in the SOAP Header. As a result, IF-MAP 1.1 is not backwards compatible with IF-MAP 1.0.

2 Background

2.1 MAP Servers and Clients

A MAP (Metadata Access Point) is a TNC element providing the MAP Server function, which stores state information about devices, users, and flows in a network. This information includes registered address bindings, authentication status, endpoint policy compliance status, endpoint behavior, and authorization status. For example, the user joe has authenticated through an 802.1X switch using an endpoint with MAC address 00:11:22:33:44:55. An endpoint assessment has revealed that the endpoint has the proper anti-virus software installed and enabled, as required by policy. The endpoint has subsequently been assigned IPv4 address 192.0.2.4, and has been engaged in instant messaging (IM) traffic with the corporate IM server 192.0.2.69.

MAP Clients may publish information to a MAP, search the information in a MAP, and subscribe to notifications from a MAP when information stored in the server changes. A single MAP Client may publish, search, and subscribe. Often, however, a MAP Client is either a publisher or a subscriber. For example, a TNC Server publishes information about the policy compliance of an endpoint and a Flow Controller (such as a layer 3 firewall) subscribes to notification of changes to this information. When the TNC Server detects that the endpoint is no longer policy compliant, the TNC Server updates the information in the MAP Server. The MAP Server notifies the Flow Controller. The Flow Controller blocks access to the network by the newly non-compliant device. In this example, both the TNC Server and the Flow Controller are MAP Clients. The TNC Server is a publisher. The Flow Controller is a subscriber.

IF-MAP is the protocol used for communication between MAP Clients and Servers.

2.2 Operational Scope of IF-MAP

A MAP allows elements in the TNC architecture to share and correlate stateful runtime metadata. This data *augments* other sources of data for security related decision making. Searches and subscriptions using IF-MAP return data which *nominally* reflects recent metadata values and relationships as reported by MAP Clients. IF-MAP is not a trusted system of absolute truth but rather a reflection of actual IF-MAP operations by systems which may or may not be behaving as intended with respect to a given operational situation. Validation of proper MAP Client behavior for a specific use case (e.g. correctly reporting a de-provisioning operation via IF-MAP) is out of the scope of this specification. No global transactional guarantees are provided for IF-MAP 1.0 (e.g. ordering of publish requests). IF-MAP does not provide historical information.

2.3 Supported Use Cases

Use cases that this version of IF-MAP supports:

- A Policy Decision Point (PDP) publishes access requestor (AR) authentication, VLAN, and other status to a MAP.
- A TNC Server publishes AR compliance status after performing an integrity check.
- A DHCP server assigns a lease to the AR. DHCP server publishes address and lease information associated with the AR.
- A Flow Controller detects a previously unseen flow from an AR and queries a MAP Server to obtain authentication and compliance status associated with this AR in order to make enforcement decisions about the new flow.
- A Flow Controller subscribes to notifications from a MAP Server about changes in authentication, compliance, vulnerability, or other status for an AR so the Flow Controller can make appropriate enforcement adjustments to an existing flow.

- A Sensor (i.e. an intrusion detection server) publishes information related to an AR or flows originated from an AR (vulnerability detection, flow classification, flow compliance, etc) to a MAP Server.

- A PDP queries a MAP for metadata that a MAP Client has associated with an AR (e.g. flow classification or vulnerability information). The PDP uses the metadata to make appropriate policy decisions. The PDP subscribes to notifications from the MAP Server about changes to the AR's metadata so the PDP can adjust the AR's access when the AR's metadata changes.

2.4 Requirements

The following are the requirements which IF-MAP must meet in order to successfully play its role in the TNC architecture. These are stated as general requirements, with specific requirements called out as appropriate.

1. **Meets the needs of the TNC architecture**

   IF-MAP must support all the functions and use cases described in the TNC architecture as they apply to the relationship between the MAP and any TNC element.

   Specific requirements include:

   - IF-MAP must support both synchronous response and asynchronous notification queries.
   - IF-MAP must support updates to metadata items. While directory protocols like LDAP are optimized for centralized updates and distributed reads, IF-MAP is required to have stronger support for distributed updates.

2. **Secure**

   - Communication between MAP Clients and Servers MUST be authenticated and integrity-protected against unauthorized modifications enroute.
   - Communication between MAP Clients and Servers MUST provide confidentiality against unauthorized disclosure.
   - Communication between MAP Clients and Servers MUST not be susceptible to replay attacks.

3. **Extensible**

   IF-MAP needs to expand over time as new features and supported network, message, and authentication technologies are added to the TNC architecture. IF-MAP must allow new features to be added easily, providing for a smooth transition and allowing newer and older architectural components to work together.

4. **Easy to use and implement**

   IF-MAP should be easy for MAP Client and Server vendors to use and implement. It should allow them to enhance existing products to support the TNC architecture and integrate legacy code without requiring substantial changes.

5. **Unambiguous**

   There should be clarity and lack of ambiguity for identification of specific entities (ARs, flows, etc.) for which metadata exists and which are interacting with the MAP Server. For example layer 2 and 3 flows, users, ARs and all other instances of TNC elements should be uniquely identifiable within an IF-MAP implementation.

6. **Scalable and Efficient**

IF-MAP is intended to be used for interfacing thousands of networking devices to a distributed IF-MAP service in a large organization in which thousands of updates occur per second. It is expected that within years of publication of IF-MAP 1.0, MAP Servers will be required to serve millions of networking devices. Therefore, the IF-MAP specification should not place implementation burdens on clients or servers that would prevent scaling to meet the demands of a large organization.

2.5 IF-MAP in TNC Architecture

The three-role TNC architecture is extended by:

- Adding a MAP Client role which includes Flow Controllers and other devices like DHCP servers, providing Flow Controller and/or Sensor functions, that are not required to communicate directly with a PDP but may share metadata regarding network defense and provisioning via IF-MAP.
- Adding a Metadata Access Point role which coordinates metadata exchange.

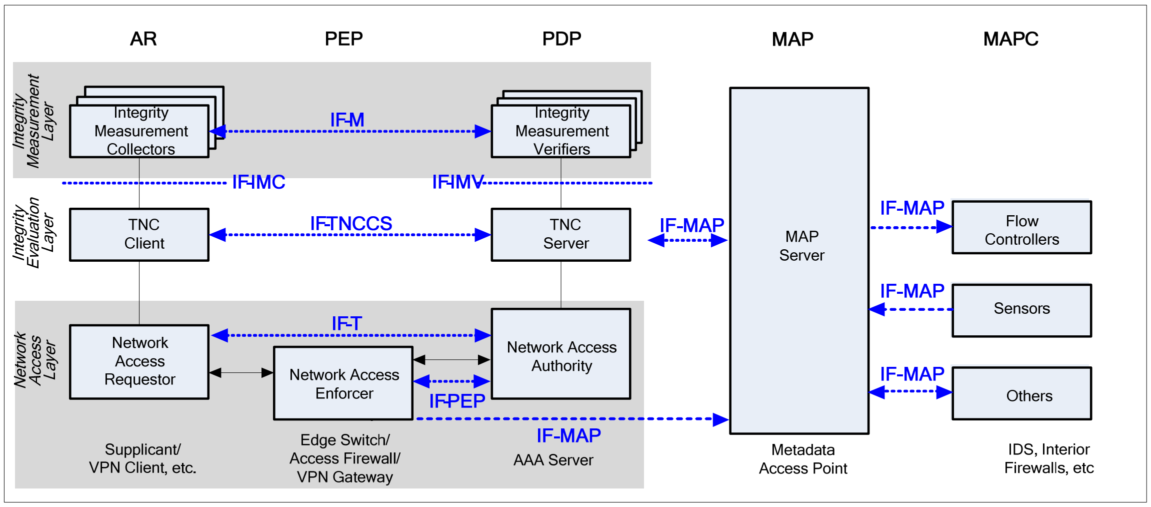


Figure 1

The Metadata Access Point role of the TNC architecture is performed by an element that has a MAP Server function. All other elements in the TNC architecture which use IF-MAP (PEPs, Flow Controllers, Sensors, PDP, etc) are MAP Clients. In fact, any network element can use IF-MAP as long it is authorized to do so.

2.6 Data Model

In IF-MAP there are two types of data: identifier and metadata. There is one type of relationship: link. All IF-MAP operations and data types are represented as XML documents.

2.6.1 Identifier

IF-MAP specifies an **identifier** as a single, globally UNIQUE value within a space of values described by an identifier type specified in the IF-MAP XML schema. An identifier or set of identifiers is required for all IF-MAP metadata operations. The schemas for all types of identifiers are defined by this document in order to provide a unified namespace for metadata.

For example, the IPAddressType identifier type schema element defines an identifier space consisting of all possible IP addresses.

All identifiers in an identifier space implicitly exist at all times and are always legal for any operation within the limits of a client's authorization policy. In other words, an identifier does not need to be explicitly created before being used (and in fact there are no operations in IF-MAP for creating and destroying identifiers).

2.6.2 Link

IF-MAP specifies a **link** as an unnamed, bi-directional binding relationship between two **identifiers**. For example, a DHCP server might create a link between a mac-address identifier and an ip-address identifier.

2.6.3 Metadata

In IF-MAP, metadata is represented as typed values which are well described by schema. Each instance of metadata in a MAP Server is associated with a particular **identifier** or **link**. There are two types of metadata:

- Standard Metadata
- Vendor-specific Metadata

The schema for standard metadata is completely defined by this specification. Vendor-specific metadata is used for use cases not defined by this specification. All MAP Clients and Servers MUST support both standard and vendor-specific metadata. A MAP Client MUST ignore vendor-specific metadata that it does not understand. A MAP Server MUST allow storage and retrieval of vendor-specific metadata. All MAP Server implementations SHOULD support XML Schema validation using standard URIs as described by [3] and [4]. All MAP Clients SHOULD likewise validate documents using the same mechanisms.

A metadata type may be single-valued or multi-valued. A MAP Client updating a multi-valued metadata type adds new instances of the type. A MAP Client updating a single-valued metadata type replaces any previous instance of the type being updated.

3 IF-MAP Interface

3.1 String Encoding

MAP Clients and MAP Servers MUST store string fields in UTF-8.

3.2 Identifiers

Identifiers provide a unified namespace that can be used to refer to specific metadata items. There are several types of identifiers. All MAP Server and Client implementations MUST support the entire set of identifiers.

There are no explicit limits on the size of an identifier. MAP Clients and Servers MUST support identifiers up to 1000 bytes in length, and SHOULD support longer identifiers. If a MAP Server receives an identifier from a client that exceeds its maximum identifier length, the MAP Server SHOULD respond with an IdentifierTooLong error (see section 0). If a MAP Client receives an identifier from a server that exceeds its maximum identifier length, the MAP Client SHOULD treat the operation as having failed and log an administrator-viewable message.

3.2.1 access-request

An access request represents a request for access to a network by a logical endpoint. A single endpoint may request access to multiple networks, in which case multiple access-request identifiers for the same endpoint may be stored in the MAP database. A single organization may have several administrative domains which provision access-requests.

IF-MAP specifies an *AccessRequestType* consisting of administrative domain string and name string. A MAP Server MUST process two **access-request** identifiers as equivalent if and only if ALL corresponding fields in the two **access-request** identifiers are equivalent.

When utilizing the *AccessRequestType* specified in IF-MAP, a MAP Client MAY specify an administrative domain. A MAP Server MUST process the administrative domain field as CASE SENSITIVE. A MAP Server MUST process administrative domains which are 1) unspecified, 2) specified as an empty string as equivalent (i.e. belonging to the same administrative domain).

The name field MUST be of the form “publisher-id:UID” where Publisher-ID refers to a specific MAP Client, usually a PDP (see example section 8.2) . The UID may be a simple ordinal value, even monotonically increasing starting at one.

```
<xsd:complexType name="AccessRequestType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
```

3.2.2 device

A device represents a physical or virtual asset which is attempting to gain entry to a network.

IF-MAP specifies a *DeviceType* consisting of either an aik-name string or a name string. A MAP Server MUST process two **device** identifiers as equivalent if and only if the corresponding names in the two **device** identifiers are equivalent in both type (i.e. name or aik-name) and value.

The name field MUST be of the form “publisher-id:UID” where publisher-id refers to a specific MAP Client. The UID may be a simple ordinal value, even monotonically increasing starting at one.

For multiple virtual devices operating on the same physical device, each virtual device SHOULD have its own device identifier in IF-MAP.

```
<xsd:complexType name="DeviceType">
```

```
  <xsd:choice>
    <xsd:element name="aik-name" type="xsd:string"/>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:choice>
</xsd:complexType>
```

3.2.3 identity

An identity represents an end-user. A single organization may have several administrative domains which provision end-user identities. Identities may take several different syntactic forms.

IF-MAP specifies an IdentityType consisting of administrative domain string, name string, type enumeration, and other-type-definition string. A MAP Server MUST process two identities as equivalent if and only if ALL corresponding fields in the two identities are equivalent.

When utilizing the IdentityType specified in IF-MAP, a MAP Client MAY specify an administrative domain. A MAP Server MUST process the administrative domain field as CASE SENSITIVE. It is up to MAP Clients to perform canonicalization of the administrative domain field if it is derived from a case insensitive external source. A MAP Server MUST process administrative domains which are 1) unspecified, 2) specified as an empty string as equivalent (i.e. belonging to the same administrative domain).

A MAP Client MUST specify a name field consisting of a non-empty string. A MAP Client MUST specify a type in order to provide a hint to MAP Clients how to parse the name field. A MAP Server MUST process the name field according to the syntax specified by the type enumeration. A MAP Server MUST process two syntactically equal name fields as distinct if they are associated with distinct types. For example an identity with an unspecified administrative domain, type=username, and name=”foo.bar” MUST be considered distinct from an identity with an unspecified administrative domain, type=dns-name name=”foo.bar” since the types are distinct. Furthermore, the uniqueness of the name field is dependent on the syntax of the type enumeration. For example an identity with an unspecified administrative domain, type=”dns-name” and name=”foo.bar” is usually processed as equivalent to an identity with an unspecified administrative domain, type=”dns-name” name=”FOO.BAR” since the type dns-name typically specifies that the name field must be processed as case insensitive.

Because the exact syntax and equivalence (i.e. matching) rules for each of the identity types may vary widely from organization to organization, the choice is left to the MAP Server implementation. MAP Servers SHOULD provide both pre-configured and configurable options for both syntax and equivalence rules of all the various identity types.

The following identity types are supported:

- **aik-name**
- **distinguished-name**
- **dns-name**
- **email-address**
- **kerberos-principal**
- **username**
- **sip-uri**
- **tel-uri**
- **other**

If a MAP Client specifies identity type as other, the client MUST specify a non-empty string for the other-type-definition field. The other-type-definition field’s value MUST take one of two forms:

1. “Vendor-ID:Name”: A vendor-defined type. Vendor-ID is a DNS name owned by the vendor, and Name is the type name
2. “Name”: A TCG-defined type. A TCG-defined type may be specified in a future version of IF-MAP or in a supplement to IF-MAP.

```
<xsd:complexType name="IdentityType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="type" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="aik-name"/>
        <xsd:enumeration value="distinguished-name"/>
        <xsd:enumeration value="dns-name"/>
        <xsd:enumeration value="email-address"/>
        <xsd:enumeration value="kerberos-principal"/>
        <xsd:enumeration value="trusted-platform-module"/>
        <xsd:enumeration value="username"/>
        <xsd:enumeration value="sip-uri"/>
        <xsd:enumeration value="tel-uri"/>
        <xsd:enumeration value="other"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="other-type-definition" type="xsd:string"/>
</xsd:complexType>
```

3.2.4 ip-address

An ip-address identifier represents a single IP address. Due to the fact that many networks are deployed using overlapping IP address spaces, an additional “administrative-domain” qualifier can optionally be added to an ip-address identifier in order to uniquely identify that address. The administrative-domain is a string whose format is organizationally defined.

Both IPv4 and IPv6 address identifiers are supported. These two classes of addresses are differentiated by use of the “type” attribute, which can have either the value “IPv4” or “IPv6”. In the absence of a “type” attribute, “IPv4” is assumed.

IP addresses MUST be canonicalized by MAP Clients. MAP Servers MUST reject IP addresses which are not in canonical form.

The canonical form of an IPv4 address is dot-decimal notation (i.e. dotted quad notation) consisting of four dot-separated decimal numbers between 0 and 255. No leading 0s are allowed except that the number 0 is represented by a single 0 character.

IPv4 address => octet “.” octet “.” octet “.” octet

octet => 0..255

As specified in [9], the canonical form of an IPv6 address is x:x:x:x:x:x:x:x, where the 'x's are the lowercase hexadecimal values of the eight 16-bit pieces of the address.

Examples:

2001:db8:7654:3210:fedc:ba98:7654:3210

2001:db8:0:0:8:800:200c:417a

Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field.

A MAP Client MAY specify an administrative domain. A MAP Server MUST process the administrative domain field as CASE SENSITIVE. A MAP Server MUST process administrative domains which are 1) unspecified, 2) specified as an empty string as equivalent (i.e. belonging to the same administrative domain).

```
<xsd:complexType name="IPAddressType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
  <xsd:attribute name="type">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="IPv4"/>
        <xsd:enumeration value="IPv6"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>
```

3.2.5 mac-address

A mac-address identifier represents an Ethernet MAC address. Due to the prevalence of virtual MAC addresses, an optional administrative-domain qualifier can be added to a MAC address in order to uniquely identify the address. The administrative-domain is a string whose format is organizationally defined.

MAP Clients and Servers MUST specify the MAC address as six groups of two lowercase hexadecimal digits, separated by colons (:) in transmission order, e.g. 01:23:45:67:89:ab.

A MAP Client MAY specify an administrative domain. A MAP Server MUST process the administrative domain field as CASE SENSITIVE. A MAP Server MUST process administrative domains which are 1) unspecified, 2) specified as an empty string as equivalent (i.e. belonging to the same administrative domain).

```
<xsd:complexType name="MACAddressType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
```

3.3 Metadata

All metadata types are derived from the MetadataType type from the http://www.trustedcomputinggroup.org/2006/IFMAP-METADATA/1 schema.

A metadata type may be either single-valued or multi-valued. When a MAP Client updates an instance of a single-valued metadata type, any previous instance of that metadata type associated with the same identifier or link is replaced. If a MAP Client updates an instance of a multi-valued metadata type, the new values are appended to the old values.

The SingleValueMetadataType and MultiValueMetadataType types are used as base classes for single-valued and multi-valued metadata types respectively.

```
<xsd:complexType name="SingleValueMetadataType"
  abstract="true">
  <xsd:complexContent>
    <xsd:extension base="MetadataType">
      <xsd:attribute name="cardinality"
        default="singleValue">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="singleValue"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="MultiValueMetadataType"
  abstract="true">
  <xsd:complexContent>
    <xsd:extension base="MetadataType">
      <xsd:attribute name="cardinality"
        default="multiValue">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="multiValue"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The MAP Server determines whether a metadata type is single-valued or multi-valued using the cardinality attribute of an instance received in a publish request. If the MAP Server has access to the schema that defines the metadata, the MAP Server can tell whether a metadata type is single-valued or multi-valued even if the cardinality attribute is missing by getting the default value from the schema. If the MAP Server does not have the schema that defines the metadata and the cardinality attribute is missing, the MAP Server MUST treat the metadata type as multi-valued. For this reason, MAP Clients using vendor-specific metadata types MUST include cardinality attributes in instances of single-valued metadata types.

If a MAP Client sends a publish request in which the cardinality attribute differs from the cardinality in the schema, the MAP Server MUST respond with an InvalidMetadata error.

There are no explicit limits on the size of metadata. MAP Clients and Servers MUST support metadata up to 100000 bytes in length, and SHOULD support longer metadata. If a MAP Server receives metadata from a MAP Client that exceeds its maximum metadata length, the MAP Server SHOULD respond with a MetadataTooLong error (see section 0). If a MAP Client receives metadata from a MAP Server that exceeds its maximum metadata length, the MAP Client SHOULD treat the operation as having failed and log an administrator-viewable message.

3.3.1 IF-MAP Standard Metadata types

3.3.1.1 **access-request-device** *(Link)*
Recommended for: access-request and device

access-request-device metadata is placed on a link to associate an access-request identifier with a device identifier. It is a simple metadata type and does not have any content.

A PDP may create the association after provisioning network access for an access-request. A Flow Controller may match links marked with access-request-device to search for device-attribute metadata.

```
<xsd:element name="access-request-device">
```

```
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SingleValueMetadataType"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.2 access-request-ip *(Link)*

Recommended for: access-request and ip-address

access-request-ip metadata is placed on a link to associate an access-request identifier with a ip-address identifier. It is a simple metadata type and does not have any content.

A PDP or IF-MAP-aware DHCP server may create the association when an ip-address is provisioned for an access-request. A Flow Controller may attempt to match links marked with access-request-ip to search for capability, device-attribute, and role metadata associated with a specific ip-address when making access control decisions.

```
<xsd:element name="access-request-ip">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SingleValueMetadataType"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.3 access-request-mac *(Link)*

Recommended for: access-request and mac-address

access-request-mac metadata is placed on a link to associate a specific access-request identifier with a specific mac-address identifier. It is a simple metadata type and does not have any content.

A PDP or PEP may create the association when an endpoint is authenticated and granted access. A Flow Controller may attempt to match links marked with access-request-mac to search for capability, device-attribute, and role metadata associated with a specific mac-address when making access control decisions. A DHCP server may attempt to match links marked with access-request-mac to search for access-requests when making decisions regarding provisioning IP addresses to endpoints.

```
<xsd:element name="access-request-mac">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SingleValueMetadataType"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.4 authenticated-as *(Link)*

Recommended for: access-request and identity

authenticated-as metadata is placed on a link to associate a specific access-request identifier with a specific identity. It is a simple metadata type and does not have any content.

A PDP may create the association when an endpoint is authenticated and granted access. A Flow Controller may attempt to match links marked with authenticated-as metadata to search for roles associated with an identity when making access control decisions.

An access-request SHOULD have only one authenticated-as link. If an access-request originates from a multi-user endpoint, a PDP may be able to determine that more than one user is using the endpoint. However, creating multiple authenticated-as links, one for each user, will have the effect of combining the roles of all users for the purpose of access control decisions regarding the endpoint. Properly handling multi-user endpoints is beyond the scope of IF-MAP 1.0.

```
<xsd:element name="authenticated-as">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SingleValueMetadataType"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.5 authenticated-by *(Link)*

Recommended for: access-request and ip-address

authenticated-by metadata is placed on a link to associate a specific access-request identifier with a specific ip-address identifier representing a PDP or other authenticating element. It is a simple metadata type and does not have any content.

A PDP may create the association when an endpoint is authenticated and granted access. A PDP may attempt to match links marked with authenticated-by to subscribe for changes associated with specific endpoints when making access control decisions.

```
<xsd:element name="authenticated-by">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SingleValueMetadataType"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.6 capability

Recommended for: access-request

capability metadata refers to a collection of privileges assigned to an endpoint as a result of an access request. Privileges are defined outside the scope of IF-MAP; however, capability is the name used in IF-MAP to reference those privileges for use in controlling access.

For example, an organization might define Trusted and Untrusted capabilities which are attached to an access-request. A PDP or Flow Controller may use capabilities from an access-request identifier when making access control decisions.

Different parts of an organization may be empowered to designate capabilities. In order to distinguish the same capability name assigned by different parts of the organization, an optional administrative-domain may be specified to further qualify capability names.

```
<xsd:element name="capability">
  <xsd:complexType>
    <xsd:complexContent>
         <xsd:extension base="MultiValueMetadataType">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"
            minOccurs="1" maxOccurs="1"/>
          <xsd:element name="administrative-domain"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.7 device-attribute *(Link)*

Recommended for: access-request and device

A device-attribute is an attribute assigned to a specific endpoint. The values of device-attribute metadata are defined outside the scope of IF-MAP; however, device-attribute is used in IF-MAP to reference those values for use in controlling access.

For example, an organization might define AntiVirusRunning and PatchesUpToDate attributes which are attached to a device identifier. A PDP or Flow Controller may use those attributes when making access control decisions.

```
<xsd:element name="device-attribute">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="MultiValueMetadataType">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"
            minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.8 event

Recommended for: access-request, identity, ip-address or mac-address

event metadata refers to activity of interest detected on the network. Examples include network traffic that matches the profile of a virus attack, excessive network traffic originating from a particular endpoint, and the use of a specific protocol such as an Instant Messaging protocol.

The event-recorded-time element refers to the time at which this event was first detected. The time of publication of the metadata to the MAP Server provides some temporal data, but does not accurately reflect the start time. For example, by the time an event is correlated and reported by the IPS/IPS to the MAP Server, several seconds, minutes, or even hours may have passed since the first action on the part of the endpoint that resulted in the event being created.

The magnitude element indicates how widespread the effects of the activity are. Magnitude ranges from 0 to 100, with higher magnitudes indicating more widespread effects.

The confidence element indicates how confident the MAP Client that published the event is that it accurately describes the activity of interest. Confidence ranges from 0 to 100, with higher values indicating higher confidence.

The significance element indicates how important the event is.

The following event types are supported:

- p2p: Peer-to-peer traffic was detected
- cve: Common Vulnerabilities and Exposures. Event types of “cve” MUST identify the vulnerability using the vulnerability-uri element.
- botnet infection: The endpoint is infected with a botnet and appears to be under remote control

- worm infection: The endpoint is infected with a worm which is trying to infect other endpoints in the network
- excessive flows: The endpoint is enaged in an unreasonable or unexpected amount of network traffic
- behavioral change: The endpoint's profile has changed. For example, an endpoint which initially behaved like an IP phone is now behaving like a personal computer.
- policy violation: The endpoint has violated a policy. For example, the network policy may forbid the use of instant messaging, and a Sensor detected instant messaging traffic.
- other: Extension point for other kinds of events. If a MAP Client specifies event type as other, the client MUST specify a non-empty string for the other-type-definition field. The other-type-definition field's value MUST take one of two forms:
    1. "Vendor-ID:Name": A vendor-defined type. Vendor-ID is a DNS name owned by the vendor, and Name is the type name
    2. "Name": A TCG-defined type. A TCG-defined type may be specified in a future version of IF-MAP or in a supplement to IF-MAP.

The value of the information element is a human consumable informational string. Any machine consumable information should be put into a vendor-specific metadata schema rather than in the information element.

The vulnerability-uri element is used for events with type cve to indicate which vulnerability was detected.

Additional details about the event may be added by vendor and implementation specific metadata. To accomplish this, a MAP Client publishes auxiliary metadata from a vendor-specific schema, and correlates the event metadata with the vendor-specific metadata using the name element and publisher-id attribute of the event metadata.

MAP Clients that publish events are responsible for aggregating events to limit the number of events published for a particular policy violation in order to avoid flooding the MAP Server with redundant events.

```
<xsd:element name="event">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="MultiValueMetadataType">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"
            minOccurs="1" maxOccurs="1"/>
          <xsd:element name="event-recorded-time"
            type="xsd:dateTime" minOccurs="1"
            maxOccurs="1"/>
          <xsd:element name="magnitude" minOccurs="1"
            maxOccurs="1">
            <xsd:simpleType>
              <xsd:restriction base="xsd:integer">
                <xsd:minInclusive value="0"/>
                <xsd:maxInclusive value="100"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="confidence" minOccurs="1"
            maxOccurs="1">
            <xsd:simpleType>
              <xsd:restriction base="xsd:integer">
```

```
                <xsd:minInclusive value="0"/>
                <xsd:maxInclusive value="100"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="significance" minOccurs="1"
            maxOccurs="1">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="critical"/>
                <xsd:enumeration value="important"/>
                <xsd:enumeration value="informational"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="type" minOccurs="0"
            maxOccurs="1">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="p2p"/>
                <xsd:enumeration value="cve"/>
                <xsd:enumeration value="botnet infection"/>
                <xsd:enumeration value="worm infection"/>
                <xsd:enumeration value="excessive flows"/>
                <xsd:enumeration value="behavioral change"/>
                <xsd:enumeration value="policy violation"/>
                <xsd:enumeration value="other"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="other-type-definition"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
          <xsd:element name="information"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
          <xsd:element name="vulnerability-uri"
            type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.9 layer2-information *(Link)*

Recommended for: access-request and ip-address

layer2-information is placed on a link between an access-request and an ip-address, where the ip-address specifies the IP address of the PEP through which access is occurring. layer2-information includes vlan, which specifies the VLAN assigned to the access request; port, which specifies the port on the layer 2 PEP that the access-request originates from; and an optional administrative-domain, which may be used to distinguish between two instances of the same VLAN number in different parts of a network.

Common usage:

- A PDP attaches layer2-information metadata to an access-request/ip-address link when assigning the AR to a specific VLAN.

```
<xsd:element name="layer2-information">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SingleValueMetadataType">
        <xsd:sequence>
          <xsd:element name="vlan" type="xsd:integer"
            minOccurs="1" maxOccurs="1"/>
          <xsd:element name="port" type="xsd:integer"
            minOccurs="1" maxOccurs="1"/>
          <xsd:element name="administrative-domain"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.10 ip-mac *(Link)*

Recommended for: ip-address and mac-address

ip-mac is a binding between an ip-address and a mac-address which is valid for a specific time duration and optionally sanctioned by a specific DHCP server.

A DHCP server can attach ip-mac to a link between an ip-address and a mac-address when leasing that IP.

If an endpoint has one network adapter with more than one IP address, a MAP Client may publish an ip-mac link between each ip-address and the adapter's MAC address.

```
<xsd:element name="ip-mac">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SingleValueMetadataType">
        <xsd:sequence>
          <xsd:element name="start-time"
            type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
          <xsd:element name="end-time" type="xsd:dateTime"
            minOccurs="0" maxOccurs="1"/>
          <xsd:element name="dhcp-server"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.3.1.11 role *(Link)*

Recommended for: access-request and identity

A role is the name of a collection of privileges assigned to a specific access-request and identity. Privileges are defined outside the scope of IF-MAP; however, role is a name used in IF-MAP to reference those privileges for use in controlling access.

For example, an organization might designate Employee and Contractor roles which are attached to links between access-request and identity identifiers. A PDP or Flow Controller may use roles from these links when making access control decisions.

Different parts of an organization may be empowered to designate roles. In order to distinguish the same role name assigned by different parts of the organization, an optional administrative-domain may be specified to further qualify role names.

```
<xsd:element name="role">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="MultiValueMetadataType">
        <xsd:sequence>
          <xsd:element name="administrative-domain"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
          <xsd:element name="name" type="xsd:string"
            minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

3.4 Example:

The figure below represents valid identifier and metadata relationships that could arise on a MAP during a network security scenario. For more details on how this state could emerge on a MAP, see Section 8. In the figure, ovals represent identifiers, lines between ovals represent links, and squares represent metadata which may be attached to either identifiers or links.

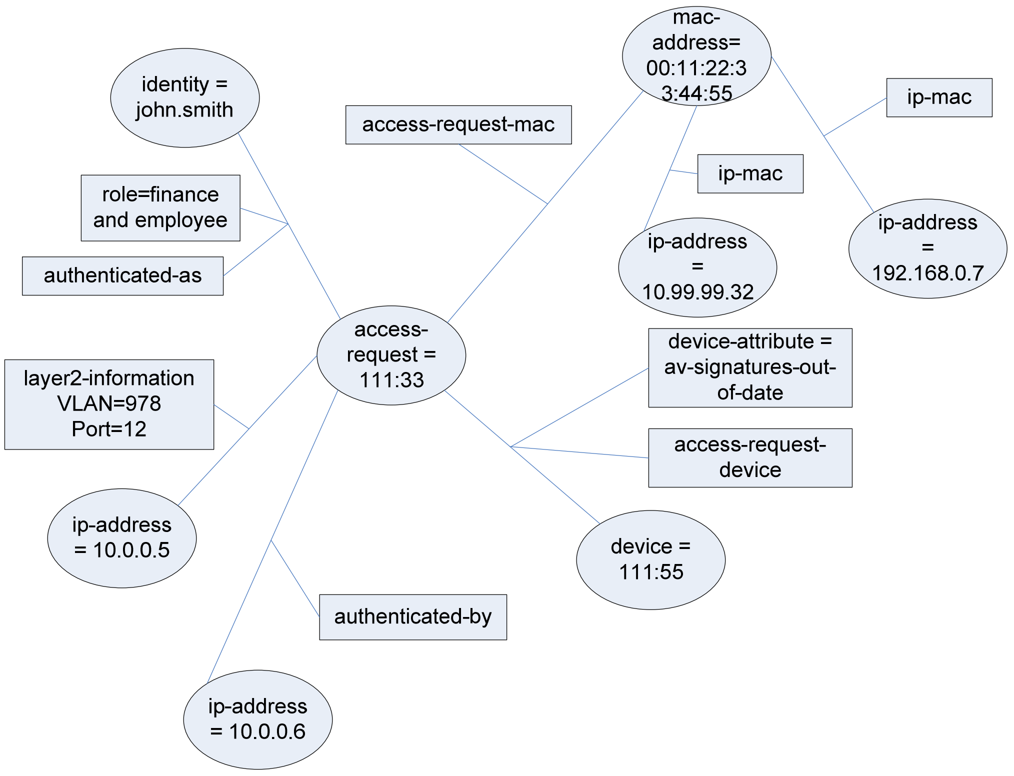


Figure 2

3.5 Filters

An IF-MAP request may include filters to specify elements to which the request applies. A MAP Server MUST consider a filter consisting of the empty string as a request to match nothing.

```
<xsd:simpleType name="FilterType">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

IF-MAP filter syntax is a subset of XPath [5] and is defined by the following grammar:

```
Filter             ::= ElemOrExpr
ElemOrExpr         ::= ElemExpr ( "or" ElemExpr )*
ElemExpr           ::= ( ( QName ) ( "[" PredOrExpr "]" )? )
                       | ( "[" PredOrExpr "]" )
PredOrExpr         ::= PredAndExpr ( "or" PredAndExpr )*
PredAndExpr        ::= PrimaryPredExpr ( "and" PrimaryPredExpr )*
PrimaryPredExpr    ::= PredExpr | ParenPredExpr
ParenPredExpr      ::= "(" PredOrExpr ")"
PredExpr           ::= Selector Operation Value
Selector           ::= ( ( ElemSelector ) ( "/" AttributeSelector )? )
                       | AttributeSelector
ElemSelector       ::= QName ( "/" QName )*
AttributeSelector ::= "@" QName
Operation          ::= "=" | "!=" | "<" | ">" | "<=" | ">="
Value              ::= Literal
Literal            ::= NumericLiteral | StringLiteral
NumericLiteral     ::= IntegerLiteral | DecimalLiteral | DoubleLiteral
IntegerLiteral     ::= Digits
DecimalLiteral     ::= ("." Digits) | (Digits "." [0-9]*)
DoubleLiteral      ::= (("." Digits)
                       | (Digits ("." [0-9]*)?)) [eE] [+-]? Digits
StringLiteral      ::= ('"' (EscapeQuot | [^"])* '"')
                       | ("'" (EscapeApos | [^'])* "'")
EscapeQuot         ::= '""'
EscapeApos         ::= "''"
Digits             ::= [0-9]+
QName              ::= [http://www.w3.org/TR/REC-xml-names/#NT-QName]
```

A filter consists of a set of ElemExprs, which are expressions for selecting elements. ElemExprs may be combined using “or” and parentheses. ElemExprs may not be combined using “and” since such combinations would typically result in filters that match no elements.

An ElemExpr may select elements based on element name, attributes and subelements, or both. To select elements by name, an ElemExpr with a QName is used. To select elements by attributes and subelements, a set of PredExprs (for “predicate expressions”) inside square brackets is used. PredExprs may be combined using “and”, “or”, and parentheses.

A QName is the name of an XML element or attribute. A QName may optionally contain a prefix followed by a “:” character.

A PredExpr consists of a Selector, an Operation, and a Value. A Selector is a path expression that specifies matching subelements and/or attributes. Subelements are matched against QNames, and attributes are matched against QNames preceded by “@” prefixes.

Subelements and attributes may be matched against Values using the Operations “=”, “!=”, “<”, “>”, “<=”, and “>=”. If both the Selector and the Value are Numeric, then numeric comparisons are used. Otherwise, case-sensitive string comparisons are used.

Given the xmlns attribute xmlns:meta=”http://www.trustedcomputinggroup.org/2006/IFMAP/1”, here are some example IF-MAP filters:

- meta:role
- meta:layer2-information[administrative-domain="Main Campus" and vlan > 30]
- meta:role[name="sales"] or meta:layer2-information[vlan=42]
- meta:event[confidence > 50 and significance=”critical”]
- [@publisher-id=”my publisher id”]

Given a vendor-specific metadata schema and xmlns attribute xmlns:vend=”http://example.com /IFMAP/metadata-schema/1”

- vend:ike-policy[@gateway=”1.2.3.4” and phase1/@identity=”joe”]

3.6 XML Validation

MAP Servers and Clients have the ability to assert that IF-MAP XML message documents adhere to their specified XML schema. A MAP Server SHOULD perform XML validation on non-metadata (i.e. operations and identifiers). A MAP Server MAY perform XML validation on metadata. MAP Servers and Clients SHOULD inform each other about whether transmitted XML has been validated.

If a MAP Client does not validate XML, any MAP Servers it communicates with SHOULD validate XML. If a MAP Server does validate XML, any MAP Clients it communicates with MAY validate XML.

In order to allow MAP Clients and Servers to communicate about whether or not XML has been validated, IF-MAP requests and responses extend ValidationType.

```
<xsd:complexType name="ValidationType" abstract="true">
  <xsd:attribute name="validation" use="optional">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="None"/>
        <xsd:enumeration value="BaseOnly"/>
        <xsd:enumeration value="MetadataOnly"/>
        <xsd:enumeration value="All"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>
```

3.7 Responses

IF-MAP involves the client sending requests to the server, and the server sending responses back to the client. A MAP Server MUST respond to every MAP Client request. If a MAP Client request cannot be processed, the MAP Server MUST respond with an errorResult element and appropriate error code and message. Responses to search and poll requests include additional results which are described in detail in section 3.8.3.

```
<xsd:complexType name="ResponseType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType">
      <xsd:choice>
        <xsd:element name="errorResult" type="ErrorResultType"/>
        <xsd:element name="pollResult" type="PollResultType"/>
        <xsd:element name="searchResult"
type="SearchResultType"/>
        <xsd:element name="subscribeReceived"/>
        <xsd:element name="publishReceived"/>
        <xsd:element name="purgePublisherReceived"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

3.7.1 Error Codes in responses

A response from a MAP Server may indicate an error by including an errorResult element:

```
<xsd:complexType name="ErrorResultType">
  <xsd:sequence>
    <xsd:element name="errorString" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="errorCode" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="AccessDenied"/>
        <xsd:enumeration value="Failure"/>
        <xsd:enumeration value="InvalidIdentifier"/>
        <xsd:enumeration value="InvalidIdentifierType"/>
        <xsd:enumeration value="IdentifierTooLong"/>
        <xsd:enumeration value="InvalidMetadata"/>
        <xsd:enumeration value="InvalidMetadataListType"/>
        <xsd:enumeration value="InvalidSchemaVersion"/>
        <xsd:enumeration value="InvalidSessionID"/>
        <xsd:enumeration value="MetadataTooLong"/>
        <xsd:enumeration value="SearchResultsTooBig"/>
        <xsd:enumeration value="SystemError"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="name"/>
</xsd:complexType>
```

The meanings of the error codes are:

Code	**Meaning**
InvalidIdentifier	**Syntax of an identifier in the request is invalid. For example, an ip-address identifier of 1.2.3.A would result in an InvalidIdentifier error.**
InvalidIdentifierType	**An unrecognized identifier type was specified within an identifier element in the request.**
IdentifierTooLong	**The MAP Client specified an identifier that exceeds the maximum identifier size supported by the server.**
InvalidMetadata	**Invalid metadata value in the request. This**

	can happen if the specified metadata does not match the schema associated with its type.
InvalidSchemaVersion	**The MAP Server was unable to process the metadata specified in the request because the MAP Client and MAP Server do not agree on the schema version.**
InvalidSessionID	**The MAP Client specified an invalid session-id.**
MetadataTooLong	**The MAP Client specified metadata that exceeds the maximum identifier size supported by the server.**
AccessDenied	**The MAP Client issued an unauthorized request. Which requests are allowed from a particular client depends on configuration settings of the MAP Server which are beyond the scope of this document.**
SearchResultsTooBig	**A search or subscribe command generates too much data.**
SystemError	**The MAP Server was unable to successfully process a request because of a system error on the server**
Failure	**The MAP Server was unable to successfully process a request for an unspecified reason.**

The name attribute in an errorResult element is used for poll results to indicate the name of the subscription that caused an error (see section 3.8.5).

3.7.2 Error Strings in responses

If present, the errorString in an errorResult element contains a human readable string further describing the error that occurred.

3.7.3 Logging of Errors in Responses

A MAP Client SHOULD log errors so that administrators can trace the causes of IF-MAP errors. Log messages SHOULD include the error code as well as the errorString if present.

3.8 Requests

MAP Clients interact with MAP Servers by sending requests.

3.8.1 publish

A publish request may create, modify, or delete metadata associated with one or more identifiers or links. Publish requests are for publishing metadata and not creating or destroying identifiers or links. Identifiers and links are never explicitly created or destroyed. Conceptually, all identifiers exist at all times (see section 2.6.1) and links between identifiers have no meaning without having metadata attached to them. A successful metadata publish MUST result in a publishReceived message. Otherwise, the response MUST contain an errorResult element with an errorCode attribute indicating the cause of the failure.

There are two subtypes of publish: update and delete. A MAP Server MUST process valid publish messages which contain exclusively update messages. A MAP Server MUST process valid publish messages which contain exclusively delete messages. A MAP Server MUST process valid publish messages which contain both update and delete messages.

```
<xsd:complexType name="PublishType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="identifier" type="IdentifierType"/>
      <xsd:element name="link" type="LinkType"/>
    </xsd:choice>
    <xsd:element name="metadata" type="MetadataListType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="DeleteType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="identifier" type="IdentifierType"/>
      <xsd:element name="link" type="LinkType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="filter" type="FilterType"
    use="optional"/>
</xsd:complexType>

<xsd:complexType name="PublishRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType">
      <xsd:sequence>
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
          <xsd:element name="update" type="PublishType"/>
          <xsd:element name="delete" type="DeleteType"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

3.8.1.1 update

update adds or changes metadata associated with identifiers and links. An update request for SingleValueMetadataType metadata MUST replace the previous value for that metadata type if it exists on the MAP Server. An update request for MultiValueMetadataType metadata MUST append the new value to a list of values for that metadata type on the MAP Server.

For example, a PDP publishes a list of roles to a user:

```
<?xml version="1.0"?>
<ifmap:publish
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <update>
    <identifier>
      <identity name="joe" type="username"/>
    </identifier>
    <metadata>
      <meta:role>
        <name>Guest</name>
      </meta:role>
      <meta:role>
        <name>Contractor</name>
      </meta:role>
    </metadata>
  </update>
</ifmap:publish>
```

3.8.1.2 delete

delete removes metadata from identifiers and links; identifiers and links are never explicitly deleted. A MAP Server MUST delete metadata specified by a valid delete message.

For example, when a Sensor no longer detects a vulnerability on the AR it may delete the associated metadata from the ip-address identifier.

```
<?xml version="1.0"?>
<ifmap:publish
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <delete
    filter='meta:event[publisher-id="222" and name="attack521"]'>
    <identifier>
      <ip-address value="192.0.2.11"/>
    </identifier>
  </delete>
</ifmap:publish>
```

3.8.1.3 Multiple PublishTypes in a single publish request

Publish requests which contain multiple update and/or delete requests have special behavior. MAP Servers MUST process all the changes specified inside a publish request before sending any notification as a result of the changes. To all MAP Clients, each publish MUST appear atomic with respect to any other publish request. In other words, the MAP Server must not notify subscribers of changes until all updates and deletes in the single publish request have been processed. Any concurrent search from a different client will observe the data in a state either before or after the entire publish. MAP Clients must never observe partial results of a publish request. Therefore, a MAP Client would never be able to tell the ordering of the operations on the MAP Server within a multi-part publish request.

Additionally, when a publish request contains multiple update and/or delete elements which operate on the same identifiers or links, the result of the publish request MUST be consistent with the update and delete elements having been applied to the IF-MAP database in the order in which they are specified by the client.

3.8.2 search

A search request retrieves metadata associated with an identifier instance and any linked identifiers.

3.8.2.1 The Retrieval Model

The retrieval model is that of an ever widening search of an arbitrarily connected graph by examination and rule matching of metadata on identifiers and links. The results are bounded by reachability by following valid link types, specified maximum result depth, and specified maximum result size.

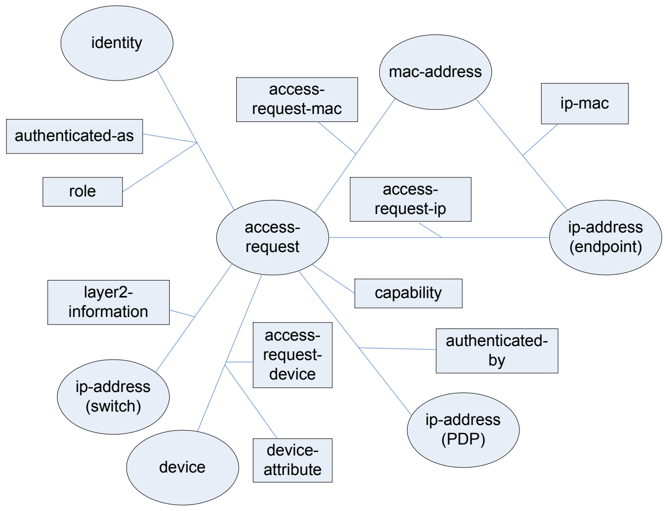


Figure 3

It may be helpful to consider the MAP Server state as a graph where searches find connected sub-graphs starting at a particular node. Figure 3 depicts identifiers as ovals, links as lines, and metadata as squares.

The following five values parameterize searches:

3.8.2.2 identifier

identifier specifies the starting place of the search.

3.8.2.3 match-links

match-links specifies the criteria for positive matching for including metadata from any link visited in the search. match-links also specifies the criteria for including linked identifiers in the search.

3.8.2.4 max-depth

Max-depth specifies the maximum distance of any included identifiers. Distance is measured by number of links away from the starting identifier.

3.8.2.5 max-size

Max-size specifies the maximum size of the results.

3.8.2.6 result-filter

The filter specifies any further rules for deleting data from the results. If there is no result-filter attribute, all metadata on all identifiers and links that match the search is returned to the client. If an empty filter is attribute is specified, the identifiers and links that match the search are returned to the client with no metadata.

3.8.2.7 The Search Algorithm

MAP Servers MUST provide results equivalent, with respect to the identifiers, links, and metadata, to a search which would result from the following algorithm.

Some terms and conditions:

A search is specified in IF-MAP syntax for SearchRequestType (see below).

The record contains the results and is returned to the MAP Client.

Current depth is defined as the number of links between the current identifier and the starting identifier.

A sub-result contains partial results.

Algorithm:

1. Start recursive search algorithm below with the search identifier, empty results, and depth zero.
2. Apply “result-filter” to the results, deleting any unmatched metadata from the results.
3. Give the results to the client indicating whether max-depth was reached or return an error (and no results) indicating max-size was reached.

Recursive Search Algorithm:

1. Start at the current identifier, with the current results, at the current depth.
2. If the current depth is greater than “max-depth”, return indicating the max-depth was reached.
3. Gather, as a sub-result, any metadata on the current identifier.
4. Append the sub-result to the current results.
5. For all links associated with the current identifier, gather, as a sub-result, all metadata on the links which matches the “match-links” filter.
6. Append the sub-result to the current results.
7. For all identifiers associated with the current identifier via links recorded in the log (i.e. which matched “match-links” in step 6), start the recursive search algorithm (step 1) with the current record and a depth equal to the current depth plus one.
8. Return the current results.

If a MAP Client does not specify max-depth, the MAP Server MUST process the search with a max-depth of zero. If a MAP Client specifies a max-depth less than zero, the MAP Server MAY process the search with an unbounded max-depth.

MAP Servers MUST support size constraints up to and including 100KB[1]. If a MAP Client does not specify max-size, the MAP Server MUST process the search with a max-size of 100KB. If a MAP Client specifies a max-size of -1, the MAP Server MAY process the search with an unbounded max-size. If a MAP Client specifies a max-size that exceeds what the MAP Server can support, the MAP Server MUST enforce its own maximum size constraints.

A MAP Server MUST return an error (and no partial results) if result exceeds max-size. In other words, the MAP Client will get back either full results or an error. The table below summarizes the possible combinations concerning max-size requests.

Client’s Requested “max-size” value	**Server’s maximum supported result size**	**Size of search result**	**Server’s response**
Unspecified	ANY (i.e. >= 100KB)	<=100KB	**Result**

[1]A limit is specified so that MAP Server implementations may protect themselves from resource exhaustion due to unreasonable searches. 100KB is large enough to return results for reasonable searches.

Unspecified	ANY (i.e. >= 100KB)	>100KB	**Error - result too large**
specified (i.e. > 0)	< client's requested "max-size"	<= server's maximum supported result size	**Result**
Specified (i.e > 0)	< client's requested "max-size"	> server's maximum supported result size	**Error – result too large**
specified (i.e. > 0)	>= client's requested "max-size"	<= client's requested "max-size"	**Result**
specified (i.e. > 0)	>= client's requested "max-size"	**>** client's requested "max-size"	**Error – result too large**
unbounded (i.e. -1)	>= 100KB and NOT unbounded	ANY	**Error – unsupported "max-size"**
unbounded (i.e. -1)	**unbounded**	**ANY**	**Result**

```
<xsd:complexType name="SearchRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType">
      <xsd:sequence>
        <xsd:element name="identifier"
          type="IdentifierType" minOccurs="1"
          maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="match-links" type="FilterType"/>
      <xsd:attribute name="max-depth" type="xsd:integer"/>
      <xsd:attribute name="max-size" type="xsd:integer"/>
      <xsd:attribute name="result-filter" type="FilterType"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

For example, when a Flow Controller detects a new flow from a previously unseen IP address, it searches a MAP Server for the list of capabilities assigned to the corresponding access-request to make enforcement decision about this flow:

```
<?xml version="1.0"?>
<ifmap:search
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1"
  match-links="meta:access-request-ip or meta:ip-mac or
meta:access-request-mac"
  max-depth="2" result-filter="meta:capability">
  <identifier>
    <ip-address value="192.0.2.11" type="IPv4"/>
  </identifier>
</ifmap:search>
```

3.8.3 Search Results

A successful search MUST result in a response message containing a searchResult element comprised of identifiers and links along with their associated metadata. The resulting XML document contains an identifierResult element for each node and a linkResult element for each edge in the metadata graph that matches the query. The XML structure itself does not directly reflect the structure of the metadata graph. Metadata appearing in a searchResult is filtered by the result-filter attribute in the search or subscription corresponding to the searchResult. If the result-filter filters out all metadata associated with an identifier or link, the identifier or link MUST still be included in the searchResult even though no metadata is associated with the identifier or link in the search result. If no result-filter is present in a search or subscription, ALL metadata that matches the search MUST be returned.

Since all identifiers for a given identifier type are always valid to search, the MAP Server MUST never return an “identifier not found” error when searching for an identifier. In this case, the MAP Server MUST return the identifier with no metadata or links attached to it.

Each search result may contain a name attribute. The name attribute is used for poll results (see section 3.8.5).

```
<xsd:complexType name="IdentifierResultType">
  <xsd:sequence>
    <xsd:element name="identifier" type="IdentifierType"/>
    <xsd:element name="metadata" type="MetadataListType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="LinkResultType">
  <xsd:sequence>
    <xsd:element name="link" type="LinkType"/>
    <xsd:element name="metadata" type="MetadataListType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SearchResultType">
  <xsd:sequence>
    <xsd:element name="identifierResult"
      type="IdentifierResultType" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="linkResult" type="LinkResultType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name"/>
</xsd:complexType>
```

For example, a MAP Server may respond to the “search” request from the example above with the following message:

```
<?xml version="1.0"?>
<ifmap:response
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <searchResult>
    <identifierResult>
      <identifier>
```

```
        <ip-address value="192.0.2.11" type="IPv4"/>
      </identifier>
    </identifierResult>
    <identifierResult>
      <identifier>
        <access-request name="123"/>
      </identifier>
      <metadata>
        <meta:capability>
          <name>finance-server-access</name>
        </meta:capability>
      </metadata>
    </identifierResult>
    <linkResult>
      <link>
        <identifier>
          <ip-address value="192.0.2.11" type="IPv4"/>
        </identifier>
        <identifier>
          <access-request name="123"/>
        </identifier>
      </link>
    </linkResult>
  </searchResult>
</ifmap:response>
```

3.8.4 subscribe

A MAP Client uses subscribe requests to manage its subscription to searches which may be polled on a MAP Server.

A subscription is a list of searchRequest items. The subscription list is consulted by the MAP Server when determining what clients need to be notified about the results of a publish request.

A MAP Server MUST maintain only one subscription list per connected MAP Client.

A subscribeRequest contains one or more searchRequest elements associated with a subscription list management function.

A MAP Server MUST respond to a valid subscribe message with a subscribeReceived message. If the subscribeRequest is not valid, the MAP Server MUST respond with an appropriate errorResult.

When a MAP Client initially connects to a MAP Server, the MAP Server MUST delete any previous subscriptions corresponding to the MAP Client. In other words, subscription lists are only valid for a single MAP Client session.

```
<xsd:complexType name="DeleteSearchRequestType">
  <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="SubscribeRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType">
      <xsd:sequence>
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
          <xsd:element name="update">
            <xsd:complexType>
              <xsd:complexContent>
```

```
                <xsd:extension base="SearchRequestType">
                  <xsd:attribute name="name"
                    type="xsd:string" use="required"/>
                </xsd:extension>
               </xsd:complexContent>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="delete"
            type="DeleteSearchRequestType"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

3.8.4.1 update

update adds or changes subscribed searches. Each subscribed search is identified by a name attribute. The value of the name attribute is generated and managed by the MAP Client. To add a new search request to a subscription, the MAP Client specifies a new name attribute on the update element. To modify an existing search request, the MAP Client specifies the name attribute of an existing search. The value of the name attribute can be any string up to 20 bytes in length, such as a unique integer value for each search the client subscribes to. The client MUST specify the name attribute in every update element of a subscriptionRequest.

For example, a Flow Controller subscribes to a MAP Server for changes in the list of roles assigned to an IP address via access-request and identity to make enforcement decisions about this flow:

```
<?xml version="1.0"?>
<ifmap:subscribe
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <update name="35" result-filter="meta:role"
    match-links="meta:access-request-ip or meta:access-request-
mac or meta:ip-mac meta:access-request-identity"
    max-depth="3">
    <identifier>
      <ip-address value="192.0.2.11" type="IPv4"/>
    </identifier>
  </update>
</ifmap:subscribe>
```

3.8.4.2 delete

delete removes the subscribed search associated with the specified name.

For example, a Flow Controller deletes its subscription to a MAP Server for changes in the list of roles assigned to an AR:

```
<?xml version="1.0"?>
<ifmap:subscribe
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <delete name="35"/>
</ifmap:subscribe>
```

3.8.5 poll

A poll request is sent by a MAP Client to a MAP Server to request notification of metadata publish requests based on the MAP Client's subscription.

```
<xsd:complexType name="PollRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType"/>
  </xsd:complexContent>
</xsd:complexType>
```

A MAP Server MUST respond to a valid poll with a pollResult message. If the poll is not valid, the MAP Server MUST respond with an appropriate errorResult. A valid poll may result in an error (most likely SearchResultsTooBig) for one of the client's subscriptions. In this case the MAP Server MUST respond with a pollResult message containing an errorResult element identifying the specific subscription that caused the error.

In response to a poll request, the MAP Server checks to see if any search results are available for subscriptions the client has made. If any results are available, the MAP Server sends a pollResult response containing search results. If no results are available, the MAP Server does not immediately send a response. At some future time when IF-MAP publish requests have resulted in changes that match client subscriptions, the MAP Server sends a pollResult response containing search results.

The server returns a separate searchResult or errorResult element for each subscription that has changes. Each searchResult and errorResult element in a pollResult response has a name attribute which identifies the subscription corresponding to the searchResult or errorResult.

The server does not return searchResult elements for a subscription that has no changes associated with it.

```
<xsd:complexType name="PollResultType">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="searchResult"
        type="SearchResultType"/>
      <xsd:element name="errorResult"
        type="ErrorResultType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

The figure below illustrates subscription and polling between the MAP Client and MAP Server.

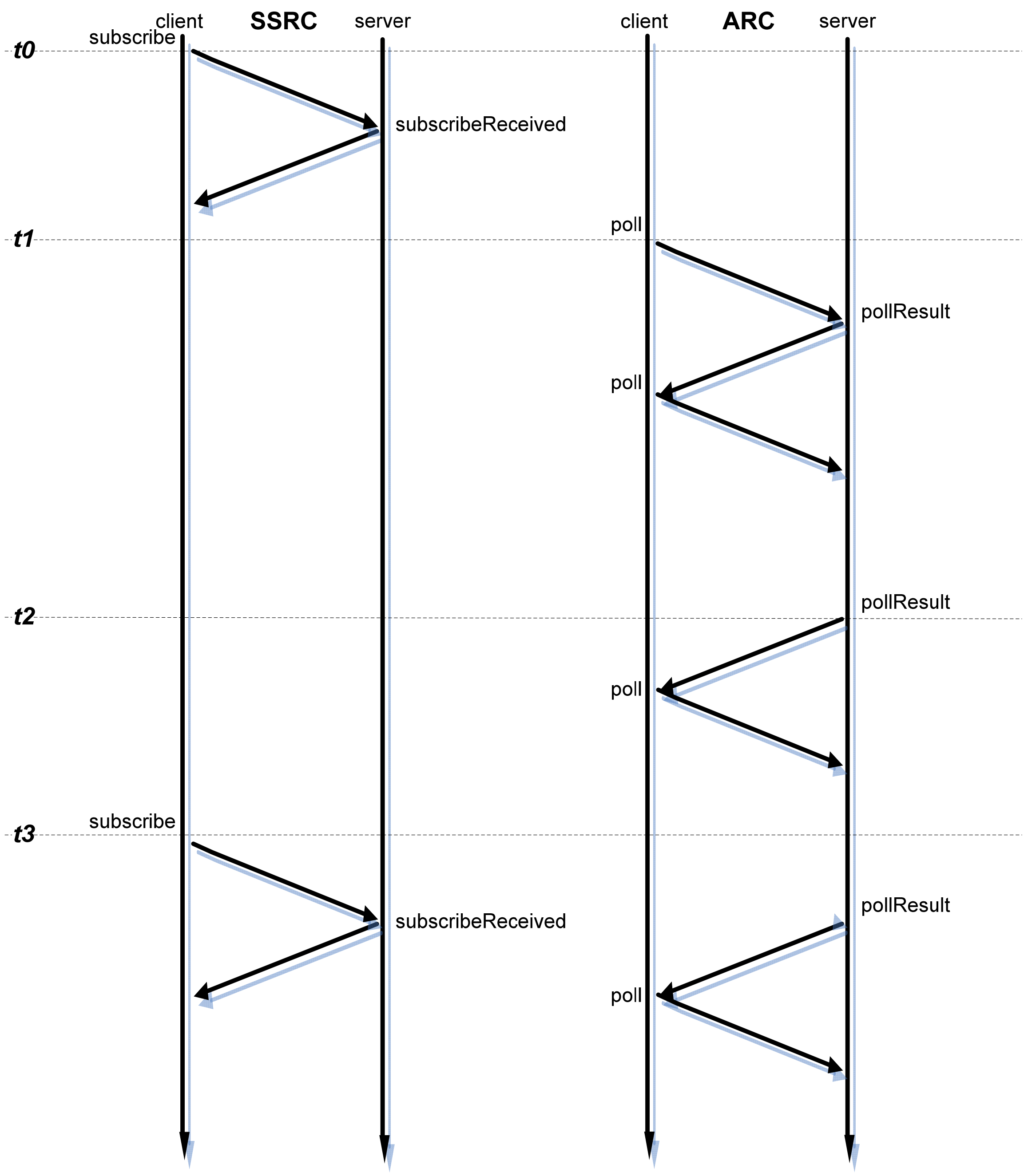


Figure 4

The figure shows two channels between the client and the server (see section 4 for a full explanation of SSRC and ARC). On the SSRC channel, the client sends subscribe requests. On the ARC channel, the client sends poll requests. In the figure, the vertical arrows represent elapsed time starting at the top. At t0, the client sends a subscribe request, and the server responds immediately with a subscribeReceived response. At t1, the client sends a poll request. Since the client has already subscribed and the subscription search matches some results, the server immediately replies with a pollResult response containing the search results. Upon receiving the pollResult response, the client issues another poll request. When the server receives the second poll request, the server does not respond right away because there are no changes to the search results since the last poll.

At t2, another MAP Client makes changes to the IF-MAP metadata that match the subscription. The server sends a pollResult response containing the new search results. This pollResult is in response to the last poll request the client issued. When the client receives the pollResult response it issues another poll request. Again, the server does not respond right away because there are no more changes to the search results.

At t3, the client issues a subscribe request on the SSRC channel that alters the search results for the subscription. On the SSRC channel, the server sends a subscribeReceived response. At the same time, the server sends a pollResult response on the ARC channel containing the new search results. When the client receives the pollResult response, it sends another poll request so the client can be notified of further changes.

3.8.6 purgePublisher

A purgePublisher request is sent by a MAP Client to ask that the MAP Server remove all metadata associated with a particular publisher. The purgePublisher request is typically used by a MAP Client to purge its own data after a power cycle or system reset if the client has no persistent knowledge of metadata it published prior to the reset. A MAP Server MAY forbid a MAP Client to use the purgePublisher request to remove data published by a different MAP Client, in which case the MAP Server MUST respond with an AccessDenied error.

```
<xsd:complexType name="PurgePublisherRequestType">
  <xsd:attribute name="publisher-id" type="xsd:string"/>
</xsd:complexType>
```

A MAP Server must respond to a purgePublisher request with either a purgePublisherReceived message or an errorResult message.

3.9 Operational Attributes

Operational attributes are not metadata. However, MetadataType defines operational attributes that MAP Servers add to stored metadata. MAP Clients MUST NOT specify operational attributes in publish requests (section 3.8.1). MAP Servers SHOULD include operational attributes in search responses (section 3.8.2).

```
<xsd:complexType name="MetadataType" abstract="true">
  <xsd:attribute name="publisher-id"/>
  <xsd:attribute name="timestamp" type="xsd:dateTime"/>
</xsd:complexType>
```

3.9.1 publisher-id

publisher-id is a unique identifier assigned by a MAP Server and associated with a specific MAP Client which performs a publish operation.

The publisher-id MUST be a function of the MAP Client's identity; however, this may be a one-way function. The publisher-id MUST NOT be a function of specific credentials. A MAP Client must be able to change credentials (e.g. new cert after expiration) and continue to be assigned the same publisher-id. The publisher-id MUST NOT be a function of time or of any connection specific information; it must remain consistent across time, and across multiple connections.

The MAP Server MAY include other server specific information when generating a publisher-id, which means the same MAP Client might be represented by two different publisher-ids on two different MAP Servers.

3.9.2 timestamp

timestamp is the time, as understood by the MAP Server, of the completion of an IF-MAP publish operation. The granularity of timestamp is one second. A timestamp without a timezone component MUST be interpreted as UTC time.

3.10 Schema Versioning

Schema version agreement between MAP Servers and MAP Clients is based on XML namespaces. XML namespaces included in IF-MAP message documents SHOULD be compared by the MAP Server to determine compatibility of MAP Client requests. A MAP Server MUST include the XML namespaces associated with the newest schema versions in any message documents. If a MAP Server is unable to process a request due to a schema version mismatch it SHOULD return an InvalidSchemaVersion error.

3.11 Vendor-specific Metadata

IF-MAP standard metadata schema should not be extended directly with vendor-specific extensions. Separate vendor-specific metadata schema MAY be defined. Vendor-specific metadata schema MAY rely on IF-MAP standard schemas. MAP Servers MUST support operations using vendor-specific metadata which is defined by an XML schema.

publisher-id and standard metadata names can be used to uniquely associate instances of vendor-specific metadata which add more detail to instances of standard metadata. For example, an IDS acting as a Sensor might update a MAP Server with an IF-MAP standard **event** and update the MAP Server with a vendor-specific event which contains the name of the IF-MAP standard **event** and more vendor-specific detail. A Flow Controller which understands the vendor-specific event can combine the standard event and vendor-specific event by matching on publisher-id and **event** name.

MAP Clients MUST ignore unrecognized vendor-specific metadata returned by searches and subscriptions. This enables MAP Clients that use vendor-specific metadata to coexist with MAP Clients that do not use vendor-specific metadata.

4 SOAP Binding

As a half-duplex web services transport protocol for XML payloads which is easy to set up for synchronous and asynchronous modes of operation, SOAP is a good match for the needs of IF-MAP.

All connections over which MAP Clients and Servers communicate are initiated by MAP Clients. All IF-MAP operations are initiated by the MAP Client. A MAP Server may respond to an operation from a client either synchronously or asynchronously.

4.1 Client-Server Communication Model

MAP Clients and Servers communicate within the context of a session.

4.1.1 Sessions

An IF-MAP session consists of one synchronous send-receive channel (SSRC) and no more than one optional asynchronous receive channel (ARC). Each session is uniquely identified by a session id and lasts as long as the TCP connection associated with the SSRC.

4.1.1.1 Synchronous Send-Receive Channel (SSRC)

The SSRC is a SOAP/HTTPS channel over which all the IF-MAP messages may be communicated: publish, search, subscribe, response. After opening a session, the MAP Client sends a series of requests on the SSRC, and the MAP Server synchronously responds to each request over the same channel.

The IF-MAP protocol is a lock-step (request-response) protocol. A MAP Client MUST wait for a valid response to each message before sending another request on the SSRC. Valid request - response pairs are:

- publish -> publishReceived | errorResult
- subscribe -> subscribeReceived | errorResult
- search -> searchResult | errorResult

A MAP Client SHOULD timeout if the MAP Server does not respond in a timely fashion to a request. A timeout SHOULD result in the termination of the transport connection associated with the given SSRC and therefore the termination of the HTTPS session.

4.1.1.2 Asynchronous Receive Channel (ARC)

The ARC is an optional channel intended to allow a MAP Client to asynchronously receive the results of its current subscription (i.e. poll). The following IF-MAP messages may be communicated over the ARC: poll and response. The response element sent in response to a poll request MUST contain an appropriate pollResult element corresponding to the MAP Client's subscription.

A MAP Client MUST have a valid SSRC for a MAP Server to open an ARC to the MAP Client. If an SSRC is closed for any reason, a MAP Server MUST close the associated ARC if it exists. A MAP Client is only allowed to open one ARC per SSRC (see section 0 for how this is enforced).

4.2 SOAP Transport

All implementations of the IF-MAP protocol MUST support Simple Object Access Protocol (SOAP) v. 1.1 as defined in [7]. All IF-MAP messages are encapsulated inside SOAP bodies which in turn are inside SOAP envelopes.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
```

```
xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1">
  <env:Header>
    <ifmap:session-id>128738734</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      ...
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

4.3 Session ID

All IF-MAP messages are associated with a session-id. The session-id is a value chosen by the MAP Server. The content of the session-id is not meaningful to a MAP Client, nor is the method used to derive a session-id defined by this specification. A session-id is a string that matches NMTOKEN, as defined in the XML 1.0 specification. A session-id may be up to 128 characters in length. The purpose of the session-id is to enable the server to share state across multiple incoming TCP connections (one each for SSRC and ARC) from a single client.

When a MAP Client first connects to a MAP Server, the MAP Client requests a new session-id by sending a SOAP request containing a "new-session" element in the SOAP body:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"

xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1">
<env:Body>
    <ifmap:new-session/>
  </env:Body></env:Envelope>
```

The MAP Server's response contains a "session-id" element that specifies the MAP Client's session-id, as well as a "publisher-id" element that the MAP Client can use to recognize metadata that it published by examining operational attributes:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"

xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1">
<env:Body>
  <ifmap:publisher-id>111</ifmap:publisher-id>
  <ifmap:session-id>222</ifmap:session-id>
 </env:Body>
</env:Envelope>
```

If a MAP Client sends more than one SOAP request containing a "new-session" element in the SOAP body, the MAP Server MUST respond each time with the same session-id.

A MAP Client associates an ARC channel with the same session-id as its SSRC channel. It does this by sending a SOAP request containing an "attach-session" element in the SOAP body:

```
<?xml version="1.0"?>
```

```
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"

xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1">
<env:Body>
    <ifmap:attach-session>222</ifmap:attach-session>
  </env:Body>
</env:Envelope>
```

The MAP Server's response is identical to the response to the request containing the "new-session" header.

Each subsequent request MUST contain a "session-id" element in its SOAP header, specifying the session-id assigned to the connection by the MAP Server:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    …
  </env:Body>
</env:Envelope>
```

Each MAP Server response MUST also contain a "session-id" element identifying the session in its SOAP header. The value of the "session-id" in the MAP Server's response MUST be the same as the value of the "session-id" specified by the MAP Client in its request. This is true even if the client specifies an invalid "session-id". If the MAP Client specifies an invalid session-id, the MAP Server MUST include that invalid session-id in the SOAP header of its response and indicate an InvalidSessionID errorResult in its reponse.

When terminating communication with a MAP Server, a MAP Client MUST close both the SSRC and the ARC. When a MAP Server detects that a MAP Client has closed one of the SSRC or the ARC, the MAP Server MAY close the other channel (possibly after a delay to recover from a short network outage as noted above).

A MAP Server MUST terminate the SSRC if it receives an invalid session-id in a message from a MAP Client.

If a MAP Server receives a message containing a SOAP header containing an attach-session element that specifies a session which already has an ARC, the MAP Server MUST close the older ARC.

5 Security Considerations

A MAP serves as a metadata clearing house for MAP Clients such as PEPs, PDPs, Flow Controllers, and Sensors, using a publish-subscribe-search model of information exchange and lookup. By increasing the ability of MAP Clients to learn about and respond to security-relevant events and data, IF-MAP can improve the timeliness and utility of the security system. However, this integrated security system can also be exploited by attackers if they can compromise it. Therefore, strong security protections for IF-MAP are essential.

This section provides a security analysis of the IF-MAP protocol and the architectural elements that employ it, specifically with respect to their use of this protocol. Three subsections define the trust model (which elements are trusted to do what), the threat model (attacks that may be mounted on the system), and the countermeasures (ways to address or mitigate the threats previously identified).

5.1 Trust Model

The first step in analyzing the security of the IF-MAP protocol is to describe the trust model, listing what each architectural element is trusted to do. The items listed here are assumptions, but provisions are made in the Threat Model and Countermeasures sections for elements that fail to perform as they were trusted to do.

5.1.1 MAP Server

The MAP Server is trusted to:

- Store data and protect the integrity of this data throughout its lifecycle
- Perform service requests in a timely and accurate manner
- Create and maintain accurate operational attributes
- Resist attacks (including denial of service and other attacks from MAP Clients)
- Only reveal data to and accept service requests from authorized parties

The MAP Server is not expected (trusted) to:

- Verify the truth (correctness) of data

The MAP Server MAY validate data against schema but is not required to do so.

5.1.2 MAP Clients

Authorized MAP Clients are trusted to:

- Preserve the confidentiality of sensitive data retrieved from the MAP Server
- Ensure the accuracy of data in the MAP Server database, by avoiding database corruption and inaccurate data
- Avoid placing too much data on the MAP Server
- Avoid creating too many links on the MAP Server
- Avoid creating too many subscriptions on the MAP Server
- Not delete valuable data from the MAP Server

5.1.3 Network

The network used to carry IF-MAP messages is trusted to:

- Perform best effort delivery of network traffic

The network used to carry IF-MAP messages is not expected (trusted) to:

- Provide confidentiality or integrity protection for messages sent over it
- Provide timely or reliable service

5.2 Threat Model

To secure the IF-MAP protocol and the architectural elements that implement it, this section identifies the attacks that can be mounted against the protocol and elements.

5.2.1 Network Attacks

A variety of attacks can be mounted using the network. For the purposes of this subsection the phrase “network traffic” should be taken to mean messages and/or parts of messages. Any of these attacks may be mounted by network elements, by parties who control network elements, and (in many cases) by parties who control network-attached devices.

- Network traffic may be passively monitored, gleaning information from any unencrypted traffic
- Even if all traffic is encrypted, valuable information can be gained by traffic analysis (volume, timing, source and destination addresses, etc.)
- Network traffic may be modified in transit
- Previously transmitted network traffic may be replayed
- New network traffic may be added
- Network traffic may be blocked, perhaps selectively
- A “Man In The Middle” (MITM) attack may be mounted where an attacker interposes itself between two communicating parties and poses as the other end to either party or impersonates the other end to either or both parties
- Undesired network traffic may be sent in an effort to overload an architectural component, thus mounting a denial of service attack

5.2.2 MAP Clients

An unauthorized MAP Client (one which is not recognized by the MAP Server or is recognized but not authorized to perform any actions) cannot mount any attacks other than those listed in the Network Attacks section above.

An authorized MAP Client, on the other hand, can mount many attacks. These attacks might occur because the MAP Client is controlled by a malicious, careless, or incompetent party (whether because its owner is malicious, careless, or incompetent or because the MAP Client has been compromised and is now controlled by a party other than its owner); because the MAP Client is running malicious software; because the MAP Client is running buggy software (which may fail in a state that floods the network with traffic); or because the MAP Client has been configured improperly. From a security standpoint, it generally makes no difference why an attack is initiated. The same countermeasures can be employed in any case.

Here is a list of attacks that may be mounted by an authorized MAP Client:

- Incorrectly create, delete, or modify metadata, perhaps causing network access to be incorrectly blocked or allowed
- Cause many false alarms or otherwise overload the MAP Server or other elements in the network security system (including human administrators) leading to a denial of service or disabling parts of the network security system
- Omit important actions (such as posting incriminating data), resulting in incorrect access

- Use confidential information obtained from the MAP Server to enable further attacks (such as using endpoint health check results to exploit vulnerable endpoints)
- Upload metadata crafted to exploit vulnerabilities in the MAP Server or in other MAP Clients, with a goal of compromising those systems
- Issue a search request or set up a subscription that matches an enormous result, leading to resource exhaustion on the MAP Server and/or the network
- Establish an ARC channel using another client's session-id

Dependencies of or vulnerabilities of authorized MAP Clients may be exploited to effect these attacks. Another way to effect these attacks is to gain the ability to impersonate a MAP Client (through theft of the MAP Client's identity credentials or through other means).

Even a clock skew between the MAP Client and MAP Server can cause problems if the MAP Client assumes that old metadata should be ignored.

5.2.3 MAP Servers

An unauthorized MAP Server (one which is not trusted by MAP Clients) cannot mount any attacks other than those listed in the Network Attacks section above.

An authorized MAP Server can mount many attacks. Similar to the MAP Client case described above, these attacks might occur because the MAP Server is controlled by a malicious, careless, or incompetent party (either a MAP Server administrator or an attacker who has seized control of the MAP Server). They might also occur because the MAP Server is running malicious software, because the MAP Server is running buggy software (which may fail in a state that corrupts data or floods the network with traffic), or because the MAP Server has been configured improperly.

All of the attacks listed for MAP Clients above can be mounted by the MAP Server. Detection of these attacks will be more difficult since the MAP Server can create false operational attributes and/or logs that imply some other party created any bad data.

Additional MAP Server attacks may include:

- Expose different database state to different MAP Clients to mislead investigators or cause inconsistent behavior
- Mount an even more effective denial of service attack than a single MAP Client could
- Send results to a MAP Client that claim to have been validated as schema compliant by the server but are not
- Leverage control of the MAP Server to attack other systems (e.g. attack other MAP Servers employed for availability that may be vulnerable to attacks from peer MAP Servers or use privilege escalation to gain control of the machine where the MAP Server is running)
- Obtain and cache MAP Client credentials so they can be used to impersonate MAP Clients even after a breach of the MAP Server is repaired
- Obtain and cache MAP Server administrator credentials so they can be used to regain control of the MAP Server after thea breach of the MAP Server is repaired

Dependencies of or vulnerabilities of the MAP Server may be exploited to obtain control of the MAP Server and effect these attacks.

5.3 Countermeasures

5.3.1 Securing the IF-MAP Protocol

To address network attacks, the IF-MAP binding for SOAP described in this document requires that the IF-MAP protocol MUST be carried over TLS as described in [11]. The MAP Client MUST verify the MAP Server's certificate and determine whether the MAP Server is trusted by this MAP Client before completing the TLS handshake. The MAP Server MUST authenticate the MAP Client either using mutual certificate-based authentication in the TLS handshake or using Basic Authentication as described in [12]. All MAP Servers and MAP Clients MUST implement both mutual certificate-based authentication and Basic Authentication. The selection of which client authentication technique to use in any particular deployment is left to the administrator. Since Basic Authentication has many security disadvantages (especially the transmission of reusable client passwords to the server), it SHOULD only be used when absolutely necessary. SOAP intermediaries MUST NOT be used.

Upon successful authentication, the trusted client entities MUST be verified for authorization to serve the MAP Client role. The means of authorization is out of scope of this specification.

These protocol security measures provide protection against all the network attacks listed in section 5.2.1 except denial of service attacks. If protection against these denial of service attacks is desired, ingress filtering [13], rate limiting per source IP address, and other denial of service mitigation measures [14] may be employed.

5.3.2 Securing MAP Clients

MAP Clients (such as branch office firewalls) may be deployed in locations that are susceptible to physical attacks[2]. Physical security measures may be taken to avoid compromise of MAP Clients, but these may not always be practical or completely effective. An alternative measure is to configure the MAP Server to provide read-only access for such systems. MAP Servers MUST allow the administrator to configure read-only access for MAP Clients. The MAP Server SHOULD also include a full authorization model so that individual clients may be configured to have only the privileges that they need. The MAP Server MAY provide functional templates so that the administrator can configure a specific client as a DHCP server and authorize only the operations and metadata types needed by a DHCP server to be permitted for that client. These techniques can reduce the negative impacts of a compromised MAP Client without diminishing the utility of the overall system.

To handle attacks within the bounds of this authorization model, the MAP Server MAY also include rate limits and alerts for unusual MAP Client behavior. MAP Servers SHOULD make it easy to revoke a MAP Client's authorization when necessary. Another way to detect attacks from MAP Clients is to create fake entries in the IF-MAP database (honeytokens) which normal MAP Clients will not attempt to access. The MAP Server SHOULD include auditable logs of client activities.

To avoid content-based attacks, the MAP Server MAY validate metadata posted by MAP Clients. However, MAP Servers and MAP Clients SHOULD also be robust against malformed data. This is especially important for vendor-specific metadata, which the MAP Server may not be able to validate.

To avoid compromise of MAP Clients, MAP Clients SHOULD be hardened against attack and minimized to reduce their attack surface. They MAY include a Trusted Platform Module (TPM) and go through a TNC handshake to verify the integrity of the MAP Client. They should be well managed to minimize vulnerabilities in the underlying platform and in systems upon which the MAP Client depends. Personnel with administrative access should be carefully screened and monitored to detect problems as soon as possible.

[2] Example is a WLAN access point that may have to be placed strategically for radio coverage but in physically ill-secured locations.

To detect clock skew relative to the MAP Server, MAP Clients MAY occasionally add a small amount of harmless metadata and check the operational attributes to see how the server's concept of time differs from the client's. MAP Clients and Servers SHOULD be configurable to synchronize their clocks with an NTP server.

5.3.3 Securing MAP Servers

Because of the serious consequences of MAP Server compromise, MAP Servers SHOULD be especially well hardened against attack and minimized to reduce their attack surface. They SHOULD include a Trusted Platform Module (TPM) and go through a regular TNC handshake to verify the integrity of the MAP Server. They should be well managed to minimize vulnerabilities in the underlying platform and in systems upon which the MAP Server depends. Network security measures such as firewalls or intrusion detection systems may be used to monitor and limit traffic to and from the MAP Server. Personnel with administrative access should be carefully screened and monitored to detect problems as soon as possible. Administrators should not use password-based authentication but should instead use non-reusable credentials and multi-factor authentication (where available). Physical security measures SHOULD be employed to prevent physical attacks on MAP Servers.

To ease detection of MAP Server compromise should it occur, MAP Server behavior should be monitored to detect unusual behavior (such as a reboot, a large increase in traffic, or different views of the database for different clients). MAP Clients should log and/or notify administrators when peculiar MAP Server behavior is detected. MAP Clients should also check data sent from the MAP Server carefully to detect malformed data. To aid forensic investigation, permanent read-only audit logs of security-relevant information (especially administrative actions) should be maintained. If MAP Server compromise is detected, a careful analysis should be performed of the impact of this compromise. Any reusable credentials that may have been compromised should be reissued.

5.3.3.1 Limit on search result size

A MAP Server MAY have a limit to the amount of data it is willing to return in search or subscription results (see section 3.8.2.7). This mitigates the threat of a MAP Client causing resource exhaustion by issuing a search or subscription that leads to an enormous result.

5.3.3.2 Cryptograhpically random session-id and authentication checks for ARC

A MAP Server SHOULD ensure that the client establishing an ARC is the same client as the client that established the corresponding SSRC. The MAP Server SHOULD employ both of the following strategies:

1. session-ids SHOULD be cryptographically random
2. The HTTPS transport for the SSRC and the ARC SHOULD be authenticated using the same credentials. SSL session resumption MAY be used to establish the ARC based on the SSRC SSL session.

5.4 Summary

IF-MAP's considerable value as a clearing-house for security-sensitive data exchange distribution also makes the protocol and the network security elements that implement it a target for attack. Therefore, strong security has been included as a basic design principle within the IF-MAP design process.

The IF-MAP protocol provides strong protection against a variety of different attacks. In the event that a MAP Client or MAP Server is compromised, the effects of this compromise have been reduced and limited with the recommended role-based authorization model and other provisions, and best practices for managing and protecting IF-MAP systems have been described. Taken together, these measures should provide protection commensurate with the threat to IF-MAP systems thus ensuring that they fulfill their promise as a network security clearing-house.

6 Privacy Considerations

MAP Clients may publish information about endpoint health, network access, events (which may include information about what services an endpoint is accessing), roles and capabilities, and the identity of the end user operating the endpoint. Any of this published information may be queried by other MAP Clients and could potentially be used to correlate network activity to a particular end user.

Dynamic and static information published to a MAP Server, ostensibly for purposes of correlation by Flow Controllers for intrusion detection, could be misused by a broader set of MAP Clients which hitherto have been performing specific roles with strict well-defined separation of duties.

Care should be taken by deployers of IF-MAP to ensure that the information published by MAP Clients does not violate agreements with end users or local and regional laws and regulations. This can be accomplished either by configuring MAP Clients to not publish certain information or by restricting access to sensitive data to trusted MAP Clients[3].

6.1 identity Identifier

The identity identifier may include specific information about an end user's identity, enabling MAP Clients to determine how a particular end user is accessing the network.

6.2 mac-address Identifier

It may be possible to determine the identity of an end user by correlation with the MAC address of an endpoint. For example, an employee may be issued a laptop and a company database may store the MAC address of the laptop along with information that identifies the employee. If an association between MAC address and end user is known, then a MAP Client could determine how a particular end user is accessing the network by querying for the known MAC address.

6.3 ip-address Identifier

If an endpoint has a static IP address or a dynamic IP address with a very long lease, it may be possible over time to make an association between a particular IP address and a particular end user. In this case, a MAP Client could determine how a particular end user is accessing the network by querying for the known IP address.

6.4 role and capability metadata

In some organizations it may be possible to identify a particular end user based on role and capability metadata published by MAP Clients. In this case, a MAP Client could determine how a particular end user is accessing the network by observing role and capability metadata associated with links and identifiers.

[3]A MAP Server implementation may provide an authorization model which protects data published by one MAP Client from being visible to another MAP Client. The specifics of such an authorization model are outside the scope of this specification.

7 References

[1] Trusted Computing Group, *TNC Architecture for Interoperability*, Revision 1.4, May 2009

[2] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, RFC 2119, Best Practices, March 1997, IETF

[3] W3C, *Extensible Markup Language (XML) 1.1 (Second Edition)*, September 2006

[4] W3C, *XML Schema Part 0: Primer Second Edition*, October 2007

[5] W3C, *XML Path Language (XPath) 2.0*, January 2007

[6] C. Rigney, S. Willens, A. Rubens, W. Simpson , *Remote Authentication Dial In User Service* (RADIUS), RFC2865, Standards Track, June 2000, IETF

[7] W3C*, SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, April 2007

[8] T. Dierks, C. Allen, *The TLS Protocol Version 1.0*, RFC 2246, Standards Track, January 1999, IETF

[9] R. Hinden, S. Deering, *IP Version 6 Addressing Architecture*, RFC 4291, Standards Track, February 2006, IETF

[10] T. Dierks, E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.1*, RFC 4346, Standards Track, April 2006, IETF

[11] E. Rescorla, *HTTP Over TLS*, RFC 2818, Informational, May 2000, IETF

[12] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, *HTTP Authentication: Basic and Digest Access Authentication*, RFC 2617, Standards Track, June 1999, IETF

[13] P. Ferguson, D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, RFC 2827, Best Current Practices, May 2000, IETF

[14] M. Handley, Ed., E. Rescorla, Ed., *Internet Denial-of-Service Considerations*, RFC 4732, Informational, November 2006, IETF

8 Examples

This section describes in detail an example flow of information between Policy Decision Points (PDPs), Policy Enforcement Points (PEPs), Flow Controllers, Sensors, and the MAP Server in a typical example.

8.1 Network Diagram

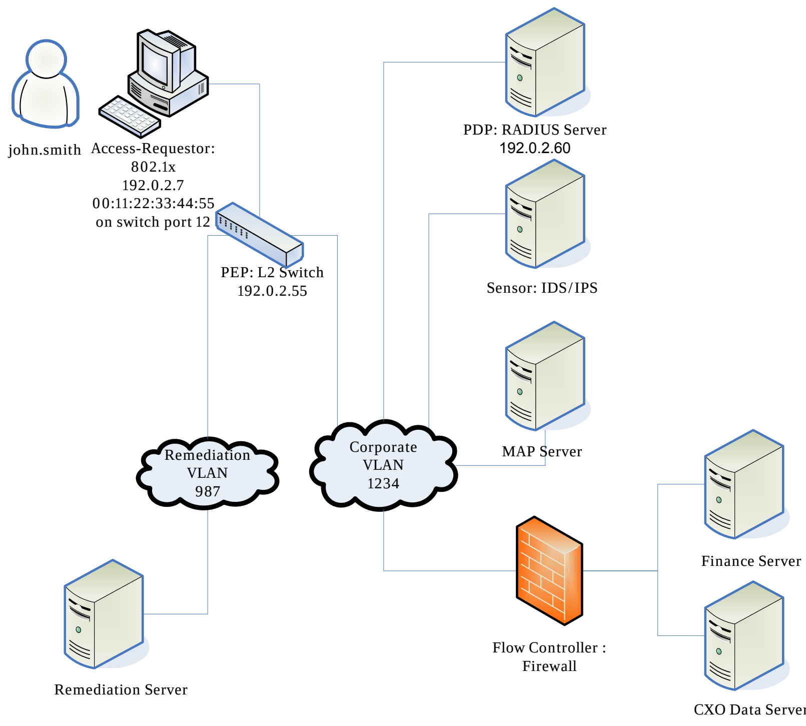

john.smith
Access-Requestor: 802.1x 192.0.2.7 00:11:22:33:44:55 on switch port 12
PEP: L2 Switch 192.0.2.55
PDP: RADIUS Server 192.0.2.60
Sensor: IDS/IPS
MAP Server
Remediation VLAN 987
Corporate VLAN 1234
Remediation Server
Flow Controller : Firewall
Finance Server
CXO Data Server

Figure 5

8.2 Example #1

Description: An 802.1X access requestor (AR) gains access through an L2 switch PEP to the corporate VLAN.

1. The PDP acting as a MAP Client connects to the MAP Server for the first time and asks the MAP Server to create a new session for the client:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:new-session/>
  </env:Header>
  <env:Body/>
</env:Envelope>
```

2. The MAP Server response informs the PDP of the PDP's publisher-id and session-id:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"

xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1">
 <env:Header>
   <ifmap:publisher-id>111</ifmap:publisher-id>
   <ifmap:session-id>222</ifmap:session-id>
 </env:Header>
 <env:Body/>
</env:Envelope>
```

3. User John Smith initiates an 802.1X connection to an L2 switch PEP. The PEP is configured to communicate with the PDP using IF-PEP (RADIUS). The PEP communicates the identifying information for this specific access request to the PDP as RADIUS attributes:
    - NAS-IP-Address: The L2 Switch's IP address, in this example is 192.0.2.55
    - NAS-Port: The physical port number to which the AR is attached, in this example port 12
    - Calling-Station-Id: The MAC address of the access-request as seen by the switch, in this example 00:11:22:33:44:55
4. The PDP uses EAP to authenticate the user and perform a TNC Handshake on the AR. In this example the IP address of the PDP is 192.0.2.60
5. Based on the authentication identity, credentials, and endpoint integrity data, the PDP applies local policy to define the roles, capabilities, VLAN, and device-attributes.
6. The PDP generates a unique access-request ID and assigns the "access-finance-service-allowed" capability. The access-request ID is of the form "publisher-id:UID". The prefixing of the publisher-id guarantees that a PDP's access-request identifiers will not collide with access-request identifiers generated by other MAP Clients. The UID can be a simple ordinal value, even monotonically increasing starting at one. The PDP's mechanism for assigning this UID should be safe and consistent across crashes and reboots.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1" >
 <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
 </env:Header>
 <env:Body>
   <ifmap:publish>
      <update>
         <identifier>
            <access-request name="111:33"/>
         </identifier>
         <metadata>
           <meta:capability>
               <name>access-finance-service-allowed</name>
          </meta:capability>
         </metadata>
      </update>
   </ifmap:publish>
  </env:Body>
</env:Envelope>
```

7. The MAP Server responds with a success result. For brevity this is the only IF-MAP response to a publish request described in this example chapter.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
<env:Header>
  <ifmap:session-id>222</ifmap:session-id>
</env:Header>
<env:Body>
  <ifmap:response>
    <publishReceived/>
 </ifmap:response>
</env:Body>
</env:Envelope>
```

8. The PDP links the identity identifier and the access-request identifier using the roles “finance” and “employee” to the user “john.smith”

```
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <identity name="john.smith" type="username"/>
          </identifier>
        </link>
        <metadata>
          <meta:role>
            <name>finance</name>
          </meta:role>
          <meta:role>
            <name>employee</name>
          </meta:role>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

9. The PDP links the identity identifier and the access-request identifier using the authenticated-as link type.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <identity name="john.smith" type="username"/>
          </identifier>
        </link>
        <metadata>
          <meta:authenticated-as/>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The metadata updates in the previous steps result in a metadata graph that looks like:

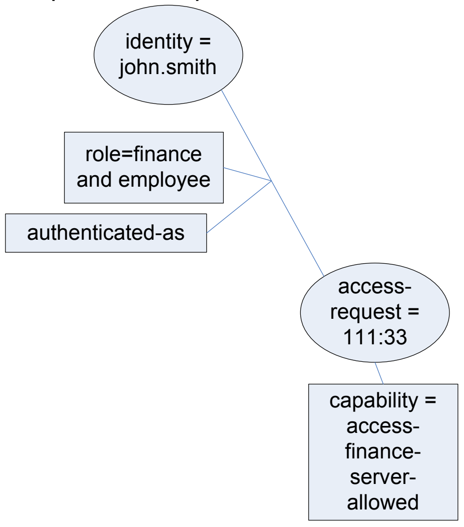

identity = john.smith
role=finance and employee
authenticated-as
access-request = 111:33
capability = access-finance-server-allowed

Figure 6

10. The PDP assigns the access requestor to VLAN 123 (the corporate VLAN). The PDP combines the information it received in the RADIUS request from the PEP with the VLAN assignment to create the link to the layer 2 PEP.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.55" type="IPv4"/>
          </identifier>
        </link>
        <metadata>
          <meta:layer2-information>
            <vlan>1234</vlan>
            <port>12</port>
          </meta:layer2-information>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The metadata graph now looks like:

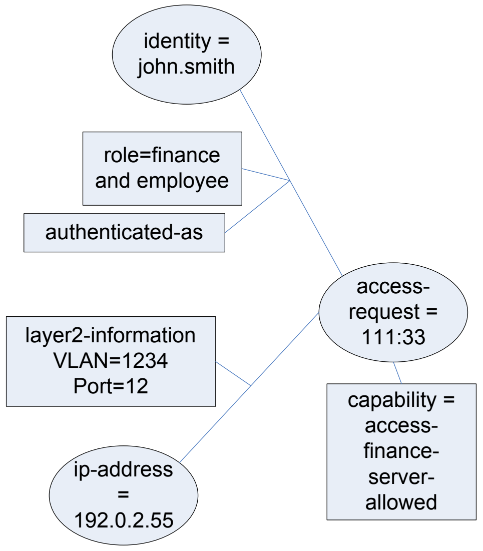

identity = john.smith
role=finance and employee
authenticated-as
access-request = 111:33
layer2-information VLAN=1234 Port=12
capability = access-finance-server-allowed
ip-address = 192.0.2.55

Figure 7

11. The PDP also received information from the layer 2 switch about the MAC address of the access requestor. The PDP uses this information to create a link between the access-request and the mac-address identifiers.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <mac-address value="00:11:22:33:44:55"/>
          </identifier>
        </link>
        <metadata>
          <meta:access-request-mac/>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The metadata graph now looks like:

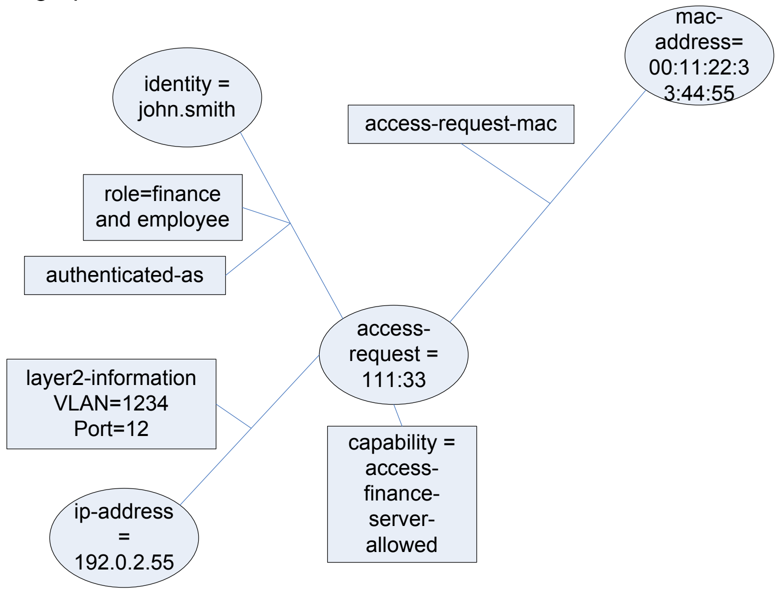

mac-address= 00:11:22:33:44:55
identity = john.smith
access-request-mac
role=finance and employee
authenticated-as
access-request = 111:33
layer2-information VLAN=1234 Port=12
capability = access-finance-server-allowed
ip-address = 192.0.2.55

Figure 8

12. The PDP links information about itself to the access-request.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.60" type="IPv4"/>
          </identifier>
        </link>
        <metadata>
          <meta:authenticated-by/>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The metadata graph now looks like:

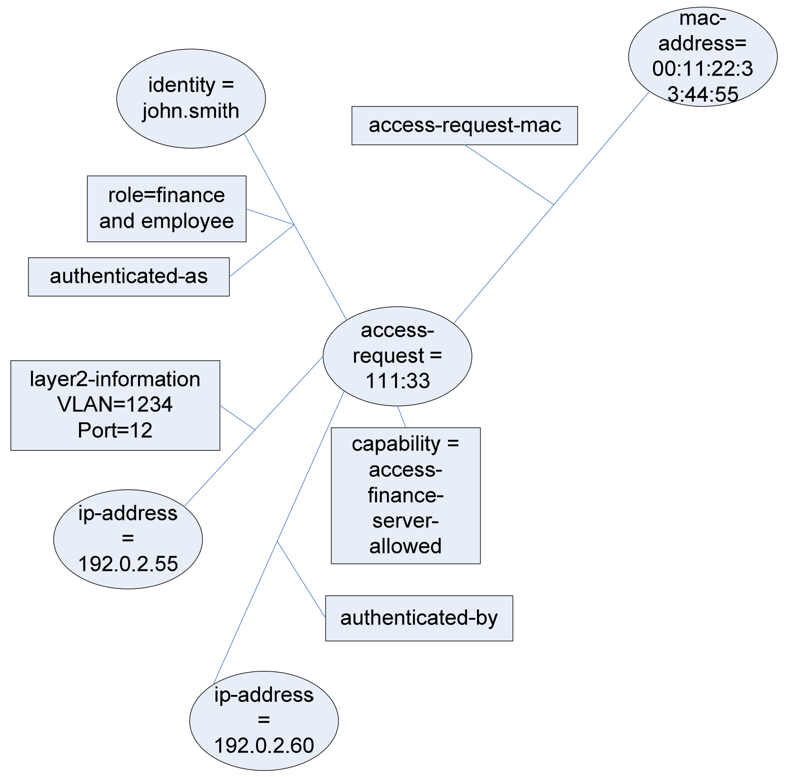


Figure 9

13. The PDP determines the device identifier name. Normally a PDP cannot determine that two access-requests are from the same device so in general the PDP creates a logical device identifier name of the form “publisher-id:UID” (similar in form to an access-request name). However, when the PDP is provided the AIK name as part of a TNC handshake the PDP should create a device identifier using the AIK. In this way two access-requests from the same device will automatically be linked in the MAP Server with no additional operations necessary. In addition to defining the device name, the PDP also attaches the access-request-device metadata. The PDP may also attach optional device-attributes, and in this example the device-attribute ‘anti-virus-running’ is attached.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <device>
```

```
              <name>111:55</name>
            </device>
          </identifier>
        </link>
        <metadata>
          <meta:access-request-device/>
          <meta:device-attribute>
            <name>anti-virus-running</name>
          </meta:device-attribute>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The metadata graph now looks like:

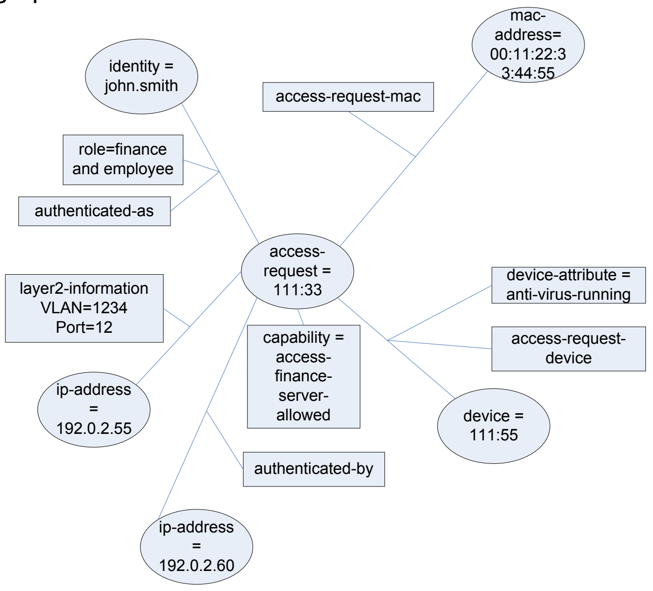


Figure 10

14. After the initial setup of the SSRC, the PDP sets up an ARC for receiving poll results in the following way:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:attach-session>222</ifmap:attach-session>
  </env:Header>
  <env:Body/>
</env:Envelope>
```

15. A poll request is issued on the ARC. The poll result doesn’t return until there is a change to a link or identifier in the IF-MAP database that matches a subscription for the session.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:poll/>
  </env:Body>
</env:Envelope>
```

16. The PDP wishes to learn about any events that a Sensor might attach to the IP address identifier. The PDP issues a subscription request of the following form to be notified of those events.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:subscribe>
      <update name="1"
        match-links="meta:access-request-mac or meta:ip-mac"
        max-depth="2" result-filter="meta:event">
        <identifier>
          <access-request name="111:33"/>
        </identifier>
      </update>
    </ifmap:subscribe>
  </env:Body>
</env:Envelope>
```

17. The MAP Server acknowledges it received the subscription request. Additional subscription requests will generate responses from the MAP Server, but for brevity those are left out in this example.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:response>
```

```
      <subscribeReceived/>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

18. The PDP wishes to learn about additional device-attributes that might be correlated with this device and assigned by other PDPs. The PDP issues a subscription request of the following form to be notified of changes to device-attributes.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:subscribe>
      <update name="2" match-links="meta:access-request-device"
        max-depth="2"
        result-filter="meta:device-attribute">
        <identifier>
          <access-request name="111:33"/>
        </identifier>
      </update>
    </ifmap:subscribe>
  </env:Body>
</env:Envelope>
```

The two subscribe requests may be combined into one subscribe request containing two update elements. The advantage of issuing both updates in the same subscribe request is that it enables the server to return searchResult elements for both subscriptions in a single pollResult response.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:subscribe>
      <update name="1"
        match-links="meta:access-request-mac or meta:ip-mac"
        max-depth="2" result-filter="meta:event">
        <identifier>
          <access-request name="111:33"/>
        </identifier>
      </update>
      <update name="2" match-links="meta:access-request-device"
```

```
        max-depth="2"
        result-filter="meta:device-attribute">
        <identifier>
          <access-request name="111:33"/>
        </identifier>
      </update>
    </ifmap:subscribe>
  </env:Body>
</env:Envelope>
```

19. The PDP immediately gets a poll response for this subscription on the ARC channel. The poll response contains the current device-attributes that the PDP just published to the MAP Server. The PDP will compare this value with the values it expects for that access-request, and on comparing these values and seeing that they are the same will take no action. If additional device-attributes are been added by other MAP Clients that are interacting with the same device, the PDP will get additional poll responses associated with this subscription, and based on that information may choose to take action.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:response>
      <pollResult>
        <searchResult name="2">
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
            <metadata/>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
              <identifier>
                <device>
                  <name>111:55</name>
                </device>
              </identifier>
            </link>
            <metadata>
              <meta:device-attribute>
                <name>anti-virus-running</name>
              </meta:device-attribute>
            </metadata>
          </linkResult>
        </searchResult>
      </pollResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

8.3 Example #2

Description: The IP address of an L2 access requestor is discovered.

1. The next step in the process is for the association between the access-request and the IP address of the access requestor to be discovered. There are three possibilities
   a. The AR leases an IP address from a DHCP server
   b. An agent on the client (e.g. an IMC) indicates to the PDP that an IP address has been assigned
   c. A device in the network observes IP packets originating from the access requestor

   In case (a) the following publish request comes from an IF-MAP-enabled DHCP server. In case (b) the publish request comes from the PDP, and in case (c) the publish request comes from an IF-MAP-enabled Sensor. In all cases the publish request is the same. In this example the access requestor IP address is 192.0.2.7.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <mac-address value="00:11:22:33:44:55"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.7" type="IPv4"/>
          </identifier>
        </link>
        <metadata>
          <meta:ip-mac/>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

At this point the metadata graph looks like this:

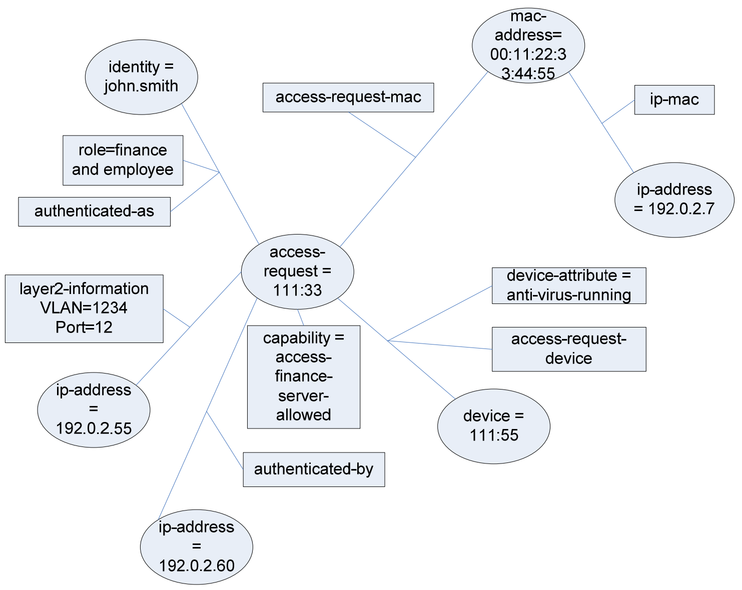


Figure 11

8.4 Example #3

Description: A Flow Controller (firewall) sees a network session from an endpoint that it has not seen before and needs to determine whether to allow or deny access.

1. The Flow Controller establishes SSRC and ARC connections to the MAP Server.
2. A network connection is attempted through a Flow Controller (such as a firewall). The Flow Controller uses the source IP address in the connection initiation packet (in this case 192.0.2.7) to subscribe to an ip-address identifier. The Flow Controller fetches the capabilities and events using one subscription, device-attributes using a second subscription, and roles using a third subscription.

   Getting from the ip-address to the access-request can take two possible paths through the metadata graph: a two step path through ip-mac and access-request-mac for layer 2 (802.1x) based authentications, or a single step path through access-request-ip for layer 3 (VPN) based authentications.

   Since capability metadata is attached to access-request identifiers and event metadata is typically attached to ip-address identifiers, the depth for the capabilities and events subscription is 2.

   Role metadata is attached to links between access-request identifiers and identity identifiers. In order to receive notification about changes to roles for all access-request identifiers for the identity associated with an ip-address, the search depth is 4: ip-address → mac-address → access-request → identifier → other access-requests.

   Similarly, since device-attribute metadata appears on links from access-request identifiers to device identifiers, in order to receive notification about changes to device-attributes for all access-request identifiers for the device associated with an ip-address, the search depth is 4: ip-address → mac-address → access-request → device → other access-requests.

For a more detailed description of the metadata associated with a multi-homed device see section 8.12.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>223</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:subscribe>
      <update name="1"
        match-links="meta:ip-mac or meta:access-request-mac or
meta:access-request-ip"
        max-depth="2"
        result-filter="meta:capabilities or meta:event">
        <identifier>
          <ip-address value="192.0.2.7" type="IPv4"/>
        </identifier>
      </update>
      <update name="2"
        match-links="meta:ip-mac or meta:access-request-mac or
meta:access-request-ip or role"
        max-depth="4"
        result-filter="meta:role">
        <identifier>
          <ip-address value="192.0.2.7" type="IPv4"/>
        </identifier>
      </update>
      <update name="3"
        match-links="meta:ip-mac or meta:access-request-mac or
meta:access-request-ip or access-request-device"
        max-depth="4"
        result-filter="meta:device-attribute">
        <identifier>
          <ip-address value="192.0.2.7" type="IPv4"/>
        </identifier>
      </update>    </ifmap:subscribe>
  </env:Body>
</env:Envelope>
```

3. Since the MAP Server already contains metadata for the IP address 192.0.2.7, the subscription generates an immediate search result containing the metadata requested in the subscription. The reason the Flow Controller issues a subscription request instead of a search in this example is that the subscription will continue to deliver updated responses as the data in the MAP database changes without any further requests from the MAP Client.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>223</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:response>
      <pollResult>
        <searchResult name="1">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <mac-address value="00:11:22:33:44:55"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
            <metadata>
              <meta:capability>
                <name>access-finance-service-allowed</name>
              </meta:capability>
            </metadata>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
            </link>
          </linkResult>
        </searchResult>
        <searchResult name="2">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <identifierResult>
```

```
            <identifier>
              <mac-address value="00:11:22:33:44:55"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <identity name="john.smith" type="username"/>
            </identifier>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
              <identifier>
                <identity name="john.smith" type="username"/>
              </identifier>
            </link>
            <metadata>
              <meta:role>
                <name>finance</name>
              </meta:role>
              <meta:role>
                <name>employee</name>
              </meta:role>
            </metadata>
          </linkResult>
        </searchResult>
        <searchResult name="3">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <identifierResult>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
            <metadata/>
          </identifierResult>
          <identifierResult>
              <identifier>
                <device>
                  <name>111:55</name>
                </device>
              </identifier>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
              <identifier>
                <device>
                  <name>111:55</name>
                </device>
              </identifier>
            </link>
            <metadata>
              <meta:device-attribute>
                <name>anti-virus-running</name>
              </meta:device-attribute>
            </metadata>
          </linkResult>
        </searchResult>
      </pollResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

4. The Flow Controller uses the capabilities, roles, and device-attributes as well as local policy to determine the access privileges allowed for this endpoint. For example, john.smith might be allowed to access the finance server but not allowed access to the CXO data server.

Change of capabilities, roles or device-attribute is described in Example #4. A PDP de-authorizing a user is described in Example #7.

It is possible for the lease (ip-mac link metadata) to be deleted, or a new lease (ip-mac link) to be created; in this case the Flow Controller would get the new information for the IP address and modify access accordingly.

When all connections from this IP address have closed and some period of time has gone by, the Flow Controller may unsubscribe for this IP address. This implementation optimization allows the Flow Controller to receive fewer poll results from the MAP Server.

8.5 Example #4

Description: A TNC handshake occurs that causes device-attributes and capabilities to change.

1. A PDP completes a TNC handshake with the access requestor. The output from the IMV indicates that the antivirus signatures are out of date. The PDP's local policy requires that access to the finance server is not allowed in this case.
2. The PDP removes the "access-finance-server-allowed" capability.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete filter="meta:capability[name='access-finance-
service-allowed' and @publisher-id='111']">
        <identifier>
          <access-request name="111:33"/>
        </identifier>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

3. The PDP also changes the device-attribute from "anti-virus-running" to "av-signatures-out-of-date". Because device-attribute is a multi-valued metadata type, a single update

operation would add to the list of device-attribute metadata elements. This is not what is desired. To accomplish a modify operation on a multi-valued metadata type, a single IF-MAP publish operation combines a delete followed by an update. Because these are combined into a single IF-MAP publish operation they are treated atomically by the MAP Server.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete filter="meta:capability[name='anti-virus-running'
and @publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <device>
              <name>111:55</name>
            </device>
          </identifier>
        </link>
      </delete>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <device>
              <name>111:55</name>
            </device>
          </identifier>
        </link>
        <metadata>
          <meta:device-attribute>
            <name>av-signatures-out-of-date </name>
          </meta:device-attribute>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The metadata graph now looks like:

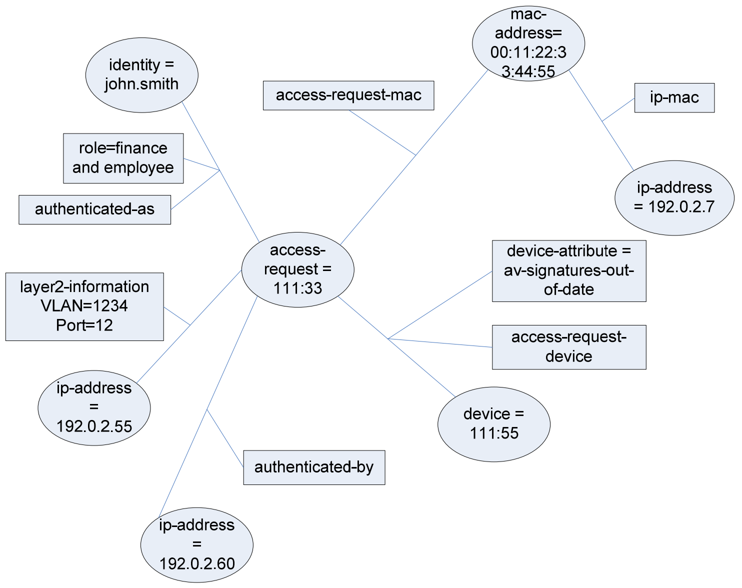


Figure 12

4. A Flow Controller subscribed to the ip-address identifier of the access requestor is notified that the capabilities and device attributes have changed and re-evaluates access as appropriate. The lack of the "access-finance-service-allowed" capability causes the Flow Controller to immediately shut down access to the finance server for this endpoint.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>223</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:response>
      <pollResult>
        <searchResult name="1">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <mac-address value="00:11:22:33:44:55"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
          </identifierResult>
          <linkResult>
```

```
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
            </link>
          </linkResult>
        </searchResult>
        <searchResult name="3">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <identifierResult>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
            <metadata/>
          </identifierResult>
          <identifierResult>
              <identifier>
                <device>
                  <name>111:55</name>
                </device>
              </identifier>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
```

```
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
              <identifier>
                <device>
                  <name>111:55</name>
                </device>
              </identifier>
            </link>
            <metadata>
              <meta:device-attribute>
                <name>av-signatures-out-of-date</name>
              </meta:device-attribute>
            </metadata>
          </linkResult>
        </searchResult>
      </pollResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

Since roles did not change, no searchResults are returned for subscription 2.

8.6 Example #5

Description: A TNC handshake occurs that causes the PDP to change the VLAN of an access requestor.

1. In addition to the action taken in Example #4, the PDP, based on local policy, also chooses to assign the access-request to the remediation VLAN. The PDP uses the following publish request to update the MAP database with the new VLAN. Because layer2-information is a single-valued metadata type, a single update operation updates the value.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
```

```
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.55" type="IPv4"/>
          </identifier>
        </link>
        <metadata>
          <meta:layer2-information>
            <vlan>978</vlan>
            <port>12</port>
          </meta:layer2-information>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

2. The PDP communicates the change of VLAN via RADIUS, and the L2 switch places the access-request on the remediation VLAN.
3. The IF-MAP enabled DHCP server updates the IF-MAP database with an additional ip-mac link representing the DHCP lease on the remediation VLAN. The metadata graph with all of the updates looks like:

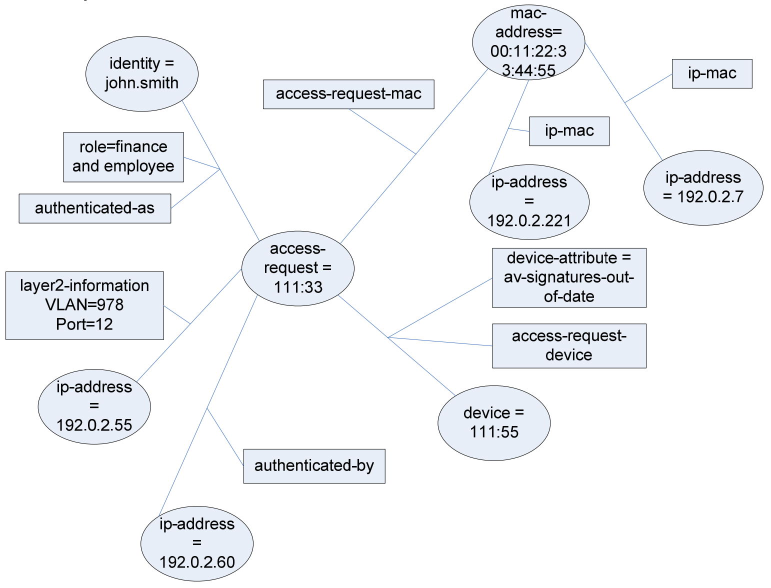


Figure 13

8.7 Example #6

Description: A Sensor detects a vulnerability on the access requestor device. Both the Flow Controller and the PDP discover this and cut off or limit access.

1. The Sensor establishes an SSRC connection to the MAP Server. The Sensor does not issue any subscription requests and therefore does not need to establish an ARC connection to the MAP Server.
2. The Sensor detects a vulnerability associated with the access requestor's IP Address and publishes an event to the IF-MAP database.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>224</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <identifier>
          <ip-address value=" 192.0.2.7" type="IPv4"/>
        </identifier>
        <metadata>
          <meta:event>
            <name>Kazaa In use</name>
            <event-recorded-time>2007-10-30T13:10:11</event-
recorded-time>
            <magnitude>45</magnitude>
            <confidence>100</confidence>
            <significance>important</significance>
            <type>policy violation</type>
            <information>Possible MP3 file sharing
            violation</information>
          </meta:event>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

3. The Flow Controller subscribed directly on the ip-address identifier gets notification of this event and re-evaluates access privileges accordingly.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>223</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:response>
      <pollResult>
        <searchResult name="1">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
            <metadata>
              <meta:event>
                <name>Kazaa In use</name>
                <event-recorded-time>2007-10-30T13:10:11</event-
recorded-time>
                <magnitude>45</magnitude>
                <confidence>100</confidence>
                <significance>important</significance>
                <type>policy violation</type>
                <information>Possible MP3 file sharing
violation</information>
              </meta:event>
            </metadata>
          </identifierResult>
          <identifierResult>
            <identifier>
              <mac-address value="00:11:22:33:44:55"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
            </link>
          </linkResult>
        </searchResult>
      </pollResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

4. The PDP subscribed on the access-request that is linked to this ip-address identifier (directly or through an intermediate node) also gets notification. The PDP may choose to take action by re-evaluating access privileges in a similar way to what occurred during a TNC handshake in Example #5.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:response>
      <pollResult>
        <searchResult name="1">
          <identifierResult>
            <identifier>
              <access-request name="111:33"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <mac-address value="00:11:22:33:44:55"/>
            </identifier>
          </identifierResult>
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
            <metadata>
              <meta:event>
                <name>Kazaa In use</name>
                <event-recorded-time>2007-10-30T13:10:11</event-
recorded-time>
                <magnitude>45</magnitude>
                <confidence>100</confidence>
                <significance>important</significance>
                <type>policy violation</type>
                <information>Possible MP3 file sharing
violation</information>
              </meta:event>
            </metadata>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <access-request name="111:33"/>
              </identifier>
            </link>
          </linkResult>
          <linkResult>
            <link>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
            </link>
          </linkResult>
        </searchResult>
      </pollResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

It would be reasonable for the Flow Controller to take no action in response to events. A Flow Controller could choose to not subscribe to events at all, instead relying on the PDP to change capabilities and device-attributes in response to events. The Flow Controller would then take action based on capabilities and device-attributes changes.

The Sensor is responsible for removing the event when the event should no longer be used for consideration in access decisions.

8.8 Example #7

Description: An access requestor disconnects from the network.

1. When the PDP detects that an access requestor has disconnected from the network, it first removes the subscriptions it requested for this access-request. There is no need for the PDP to get change notifications for things it will be deleting in subsequent steps.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:subscribe>
      <delete name="1"/>
      <delete name="2"/>
      <delete name="3"/>
    </ifmap:subscribe>
  </env:Body>
</env:Envelope>
```

2. The PDP deletes the capability it added.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
```

```
        filter="meta:capability[name='access-finance-service-
allowed' and @publisher-id='111']">
        <identifier>
          <access-request name="111:33"/>
        </identifier>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

3. The PDP deletes the roles it added.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
        filter="meta:role[@publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <identity name="john.smith" type="username"/>
          </identifier>
        </link>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

4. The PDP removes the authenticated-as link between the access-request and the identity.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
         filter="meta:authenticated-as[@publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <identity name="john.smith" type="username"/>
          </identifier>
        </link>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

5. The PDP removes the layer2-information link between the access-request and the IP address of the PEP.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
        filter="meta:layer2-information[@publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.55" type="IPv4"/>
          </identifier>
        </link>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

6. The PDP removes the access-request-mac link between the access-request and the mac-address.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
       filter="meta:access-request-mac[@publisher-id='111']">
        <link>
```

```
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <mac-address value="00:11:22:33:44:55"/>
          </identifier>
        </link>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

7. The PDP removes the authenticated-by link between the access-request and the IP address of the PDP.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
       filter="meta:authenticated-by[@publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.60" type="IPv4"/>
          </identifier>
        </link>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

8. The PDP removes the access-request-device and device-attributes metadata from the link between the access-request and the device.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
```

```
        filter="meta:device-attribute or meta:access-request-
device">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <device>
              <name>111:55</name>
            </device>
          </identifier>
        </link>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The PDP should combine all of these delete operations into a single publish request. This way the MAP Server will treat this as a single operation. Only a single poll response will go out to each client that has subscriptions to identifiers and links that have changed. This reduces the bandwith requirements between clients and servers as well as reduces the work the clients need to do in response to incremental changes. Batching requests of this type is preferred for any operation where it is logically possible.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <delete
        filter="meta:capability[name='access-finance-service-
allowed' and @publisher-id=111]">
        <identifier>
          <access-request name="111:33"/>
        </identifier>
      </delete>
      <delete filter="meta:role[@publisher-id='111'] or
meta:authenticated-as[@publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <identity name="john.smith" type="username"/>
          </identifier>
        </link>
      </delete>
      <delete
       filter="meta:layer2-information[@publisher-id='111']">
        <link>
```

```
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.55" type="IPv4"/>
          </identifier>
        </link>
      </delete>
      <delete
       filter="meta:access-request-mac[@publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <mac-address value="00:11:22:33:44:55"/>
          </identifier>
        </link>
      </delete>
      <delete
       filter="meta:authenticated-by[@publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.60" type="IPv4"/>
          </identifier>
        </link>
      </delete>
      <delete filter="meta:device-attribute[publisher-id='111']
or meta:access-request-device[publisher-id='111']">
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <device>
              <name>111:55</name>
            </device>
          </identifier>
        </link>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

9. At this point a Flow Controller that has subscribed on the ip-address identifier will get notification of the change.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
```

```
  <env:Header>
    <ifmap:session-id>223</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:response>
      <pollResult>
        <searchResult name="1">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
        </searchResult>
        <searchResult name="2">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
        </searchResult>
        <searchResult name="3">
          <identifierResult>
            <identifier>
              <ip-address value="192.0.2.7" type="IPv4"/>
            </identifier>
          </identifierResult>
          <linkResult>
            <link>
              <identifier>
                <ip-address value="192.0.2.7" type="IPv4"/>
              </identifier>
              <identifier>
                <mac-address value="00:11:22:33:44:55"/>
              </identifier>
            </link>
          </linkResult>
        </searchResult>
      </pollResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

The Flow Controller gave access to this endpoint based on capabilities, roles, and device-attributes. On receiving the poll result containing none of those metadata elements, the Flow Controller shuts off access.

8.9 Example #8

Description: An access requestor requests access through a PDP using a layer 3 access protocol (e.g. VPN)

1. Provisioning access at layer 3 is different from provisioning access at layer 2:
   a. The PDP does not attach any type of layer2-information metadata
   b. The PDP does not link the access-request to the mac-address. Instead, the PDP links the access-request directly to the ip-address of the access requestor through an access-request-ip link

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:publish>
      <update>
        <link>
          <identifier>
            <access-request name="111:33"/>
          </identifier>
          <identifier>
            <ip-address value="192.0.2.7" type="IPv4"/>
          </identifier>
        </link>
        <metadata>
          <meta:access-request-mac/>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

Assuming the PDP creates the authenticated-as, authenticated-by, access-request-device, and device-attribute links as in Example #1, the metadata graph looks like:

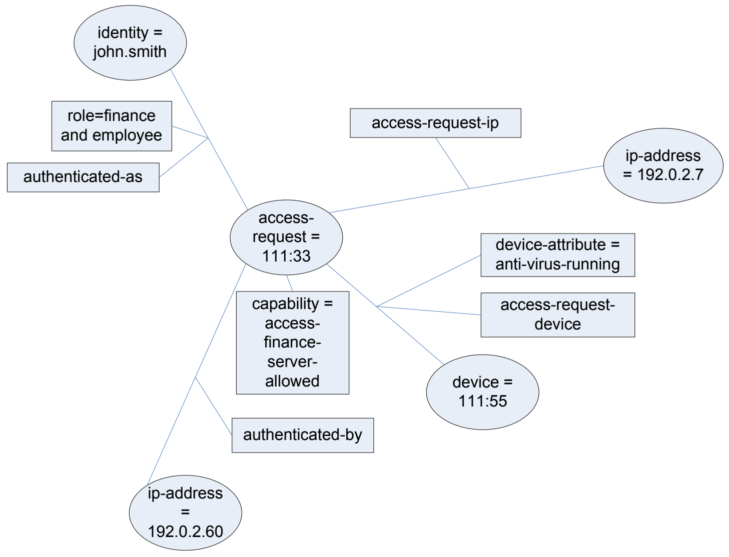


Figure 14

2. An IF-MAP enabled DHCP server could apply the ip-mac link as additional information. But in the layer 3 case that link would be purely informational, and does not play a role in any of the processing of these examples.

3. The Flow Controller subscription in Example #3 already contains support for handling the layer 2 or layer 3 cases by allowing the search to follow the access-request-ip link or the ip-mac and access-request-mac links. The Flow Controller needs to be able to parse either type of poll result set.

8.10 Example #9

Description: A network element (eg. PDP or Sensor) crashes and is rebooted.

Any MAP Client SHOULD maintain a persistent local store of information about the metadata it published to the IF-MAP database. For a Sensor publishing events attached to IP addresses, the Sensor could keep a list of IP address for which it has published event metadata. When the Sensor is rebooted, it checks to see if the metadata is still valid, and cleans it up where appropriate. In a similar way a PDP keeps track of the access-request elements that it has added, and cleans up the associated links after a reboot.

8.11 Example #10

Description: A network element (e.g. PDP or Sensor) crashes in a way where it cannot be rebooted (disk crash, burst into flames etc).

After being repaired, the network element connects to the MAP Server and issues a purgePublisher request specifying its own publisher-id.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns:meta="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">
  <env:Header>
    <ifmap:session-id>222</ifmap:session-id>
  </env:Header>
  <env:Body>
    <ifmap:purgePublisher publisher-id="345"/>
  </env:Body>
</env:Envelope>
```

This removes all of the network element's data from the MAP Server.

8.12 Example #11

Description: A user with a single identity using a single multi-homed device authenticates to PDP1 using a layer 2 access protocol (e.g. 802.1X) and authenticates to PDP2 using a layer 3 access protocol (e.g. VPN).

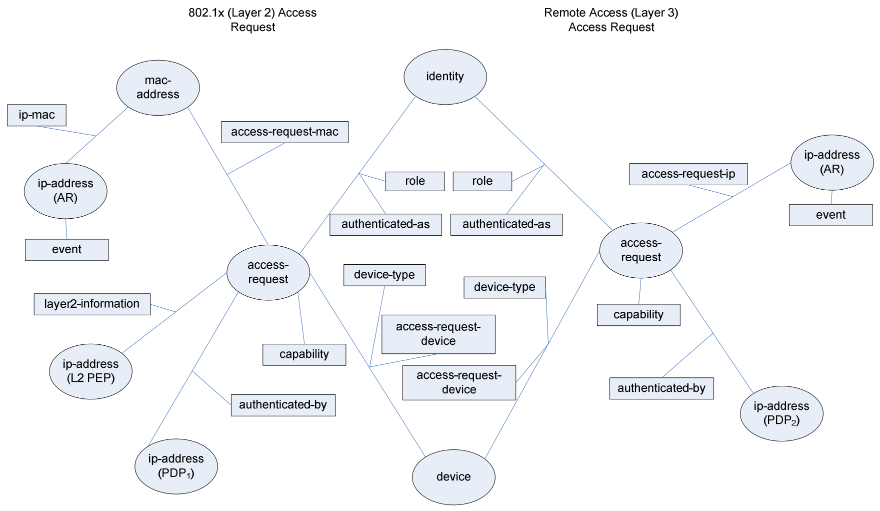


Figure 15

There are some additional considerations that must be handled to correctly address this case

1. PDPs or Flow Controllers that are looking at results posted by other PDPs must consider that the same device-attribute string can be applied multiple times on links to the same device and the same role string can be applied multiple times on links to the same identity. In the above example both PDP$_1$ and PDP$_2$ can assign the “employee” role, and multiple instances of the same role name must be ignored when doing evaluation.
2. The subscriptions in the above examples are constructed with the appropriate depth parameters that will return poll results that contain all of the device-attributes and roles. A Flow Controller or PDP should use all of the device-attributes and roles on any of the links when doing access calculations. How they are combined would depend on local policy.
3. In order to get roles from all links, device-attributes from all links, but capabilities from only the access-request of interest, it is necessary to use multiple subscriptions as shown in the preceding examples.

9 IF-MAP Schema

9.1 Identifier Types, Requests and Responses

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.trustedcomputinggroup.org/2006/IFMAP/1"

targetNamespace="http://www.trustedcomputinggroup.org/2006/IFMAP/
1">

  <!-- AccessRequestType Identifier represents an endpoint
       which is attempting to gain entry to the network-->
  <xsd:complexType name="AccessRequestType">
    <xsd:attribute name="administrative-domain"
      type="xsd:string"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
  </xsd:complexType>

  <!-- DeviceType Identifier represents a physical asset
       which is attempting to gain entry to the network -->
  <xsd:complexType name="DeviceType">
    <xsd:choice>
      <xsd:element name="aik-name" type="xsd:string"/>
      <xsd:element name="name" type="xsd:string"/>
    </xsd:choice>
  </xsd:complexType>

  <!-- IdentityType Identifier represents an end-user -->
  <xsd:complexType name="IdentityType">
    <xsd:attribute name="administrative-domain"
      type="xsd:string"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="type" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="aik-name"/>
          <xsd:enumeration value="distinguished-name"/>
          <xsd:enumeration value="dns-name"/>
          <xsd:enumeration value="email-address"/>
          <xsd:enumeration value="kerberos-principal"/>
          <xsd:enumeration value="trusted-platform-module"/>
          <xsd:enumeration value="username"/>
          <xsd:enumeration value="sip-uri"/>
          <xsd:enumeration value="tel-uri"/>
          <xsd:enumeration value="other"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="other-type-definition"
      type="xsd:string"/>
  </xsd:complexType>

  <!-- IPAddressType Identifier represents a single IP address --
>
  <xsd:complexType name="IPAddressType">
```

```
    <xsd:attribute name="administrative-domain"
      type="xsd:string"/>
    <xsd:attribute name="value" type="xsd:string"
      use="required"/>
    <xsd:attribute name="type">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="IPv4"/>
          <xsd:enumeration value="IPv6"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <!-- MACAddressType Identifier represents an Ethernet MAC
address -->
  <xsd:complexType name="MACAddressType">
    <xsd:attribute name="administrative-domain"
      type="xsd:string"/>
    <xsd:attribute name="value" type="xsd:string"
      use="required"/>
  </xsd:complexType>

  <!-- MetadataListType is a container for metadata within
       other elements -->
  <xsd:complexType name="MetadataListType">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- all possible Identifiers -->
  <xsd:complexType name="IdentifierType">
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="access-request"
          type="AccessRequestType"/>
        <xsd:element name="identity" type="IdentityType"/>
        <xsd:element name="ip-address" type="IPAddressType"/>
        <xsd:element name="mac-address"
          type="MACAddressType"/>
        <xsd:element name="device" type="DeviceType"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>

  <!-- LinkType is for association between two identifiers -->
  <xsd:complexType name="LinkType">
    <xsd:sequence>
      <xsd:element name="identifier" type="IdentifierType"
        minOccurs="2" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- FilterType is a subset of XPath -->
  <xsd:simpleType name="FilterType">
    <xsd:restriction base="xsd:string"/>
```

```
</xsd:simpleType>

<!-- ValidationType is a base type that defines the
     validation attribute that requests and responses
     use to indicate what parts of the contained
     XML have been validated -->
<xsd:complexType name="ValidationType" abstract="true">
  <xsd:attribute name="validation" use="optional">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="None"/>
        <xsd:enumeration value="BaseOnly"/>
        <xsd:enumeration value="MetadataOnly"/>
        <xsd:enumeration value="All"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

<!-- PublishType is a base type for requests that add or
     remove metadata -->
<xsd:complexType name="PublishType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="identifier" type="IdentifierType"/>
      <xsd:element name="link" type="LinkType"/>
    </xsd:choice>
    <xsd:element name="metadata" type="MetadataListType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- DeleteType is the type for the delete element of
     a publish request, and specifies which metadata
     to delete. -->
<xsd:complexType name="DeleteType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="identifier" type="IdentifierType"/>
      <xsd:element name="link" type="LinkType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="filter" type="FilterType"
    use="optional"/>
</xsd:complexType>

<!-- PublishRequestType updates or deletes metadata -->
<xsd:complexType name="PublishRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType">
      <xsd:sequence>
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
          <xsd:element name="update" type="PublishType"/>
          <xsd:element name="delete" type="DeleteType"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- SearchRequestType queries the server for matching
     metadata -->
<xsd:complexType name="SearchRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType">
      <xsd:sequence>
        <xsd:element name="identifier"
          type="IdentifierType" minOccurs="1"
          maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="match-links" type="FilterType"/>
      <xsd:attribute name="max-depth" type="xsd:integer"/>
      <xsd:attribute name="max-size" type="xsd:integer"/>
      <xsd:attribute name="result-filter"
        type="FilterType"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- DeleteSearchRequestType is for removing subscriptions -->
<xsd:complexType name="DeleteSearchRequestType">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- SubscribeRequestType is for managing subscriptions -->
<xsd:complexType name="SubscribeRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType">
      <xsd:sequence>
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
          <xsd:element name="update">
            <xsd:complexType>
              <xsd:complexContent>
                <xsd:extension base="SearchRequestType">
                  <xsd:attribute name="name"
                    type="xsd:string" use="required"/>
                </xsd:extension>
               </xsd:complexContent>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="delete"
            type="DeleteSearchRequestType"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- PollRequestType is for polling for notification of
     metadata changes that match subscriptions -->
<xsd:complexType name="PollRequestType">
  <xsd:complexContent>
    <xsd:extension base="ValidationType"/>
  </xsd:complexContent>
```

```
</xsd:complexType>

<!-- PurgePublisherRequestType is for removing all metadata
     published by a particular publisher -->
<xsd:complexType name="PurgePublisherRequestType">
  <xsd:attribute name="publisher-id" type="xsd:string"/>
</xsd:complexType>

<!-- IdentiferResultType is for search or poll results showing
     metadata attached to identifiers -->
<xsd:complexType name="IdentifierResultType">
  <xsd:sequence>
    <xsd:element name="identifier" type="IdentifierType"/>
    <xsd:element name="metadata" type="MetadataListType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- LinkResultType is for search or poll results showing
     metadata attached to links -->
<xsd:complexType name="LinkResultType">
  <xsd:sequence>
    <xsd:element name="link" type="LinkType"/>
    <xsd:element name="metadata" type="MetadataListType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- SearchResultType contains the identifiers and links
     along with associated metadata -->
<xsd:complexType name="SearchResultType">
  <xsd:sequence>
    <xsd:element name="identifierResult"
      type="IdentifierResultType" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="linkResult" type="LinkResultType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name"/>
</xsd:complexType>

<!-- PollResultType contains a searchResult for each
     subscription that had changes since the last poll -->
<xsd:complexType name="PollResultType">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="searchResult"
        type="SearchResultType"/>
      <xsd:element name="errorResult"
        type="ErrorResultType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<!-- ErrorResultType indicates the cause of an error -->
<xsd:complexType name="ErrorResultType">
  <xsd:sequence>
```

```
      <xsd:element name="errorString" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="errorCode" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="AccessDenied"/>
          <xsd:enumeration value="Failure"/>
          <xsd:enumeration value="InvalidIdentifier"/>
          <xsd:enumeration value="InvalidIdentifierType"/>
          <xsd:enumeration value="IdentifierTooLong"/>
          <xsd:enumeration value="InvalidMetadata"/>
          <xsd:enumeration value="InvalidMetadataListType"/>
          <xsd:enumeration value="InvalidSchemaVersion"/>
          <xsd:enumeration value="InvalidSessionID"/>
          <xsd:enumeration value="MetadataTooLong"/>
          <xsd:enumeration value="SearchResultsTooBig"/>
          <xsd:enumeration value="SystemError"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="name"/>
  </xsd:complexType>

  <!-- ResponseType encapsulates results from all the different
       requests -->
  <xsd:complexType name="ResponseType">
    <xsd:complexContent>
      <xsd:extension base="ValidationType">
        <xsd:choice>
          <xsd:element name="errorResult"
            type="ErrorResultType"/>
          <xsd:element name="pollResult"
            type="PollResultType"/>
          <xsd:element name="searchResult"
            type="SearchResultType"/>
          <xsd:element name="subscribeReceived"/>
          <xsd:element name="publishReceived"/>
          <xsd:element name="purgePublisherReceived"/>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- top-level elements represent all the possible
       requests and responses -->
  <xsd:element name="publish" type="PublishRequestType"/>
  <xsd:element name="search" type="SearchRequestType"/>
  <xsd:element name="subscribe" type="SubscribeRequestType"/>
  <xsd:element name="poll" type="PollRequestType"/>
  <xsd:element name="purgePublisher"
    type="PurgePublisherRequestType"/>
  <xsd:element name="response" type="ResponseType"/>

  <!-- new-session, attach-session, session-id and publisher-id
       are elements used in SOAP envelope headers for session
       and publisher ID management -->
  <xsd:element name="new-session" type="xsd:string"/>
  <xsd:element name="attach-session" type="xsd:string"/>
  <xsd:element name="session-id" type="xsd:string"/>
  <xsd:element name="publisher-id" type="xsd:string"/>
</xsd:schema>
```

9.2 Standard Metadata Types

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2006/IFMAP/1"
  xmlns="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1"

targetNamespace="http://www.trustedcomputinggroup.org/2006/IFMAP-
METADATA/1">

  <!-- Schema for IF-MAP Standard Metadata -->

  <!-- MetadataType is the base type for IF-MAP metadata.
       MetadataType defines the publisher-id and timestamp
       attributes that are maintained by the MAP Server -->
  <xsd:complexType name="MetadataType" abstract="true">
    <xsd:attribute name="publisher-id"/>
    <xsd:attribute name="timestamp" type="xsd:dateTime"/>
  </xsd:complexType>

  <!-- SingleValueMetadataType is the base type for
       metadata that can have at most one value
       for a particular identifier or link -->
  <xsd:complexType name="SingleValueMetadataType"
    abstract="true">
    <xsd:complexContent>
      <xsd:extension base="MetadataType">
        <xsd:attribute name="cardinality"
          default="singleValue">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="singleValue"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- MultiValueMetadataType is the base type for
       metadata that may have multiple values for
       a particular identifier or link -->
  <xsd:complexType name="MultiValueMetadataType"
    abstract="true">
    <xsd:complexContent>
      <xsd:extension base="MetadataType">
        <xsd:attribute name="cardinality"
          default="multiValue">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="multiValue"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- access-request-device is link metadata that
       associates an access-request identifier with
       a device identifier -->
  <xsd:element name="access-request-device">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SingleValueMetadataType"/>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- access-request-ip is link metadata that
       associates an access-request identifier with
       an ip-address identifier -->
  <xsd:element name="access-request-ip">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SingleValueMetadataType"/>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- access-request-mac is link metadata that
       associates an access-request identifier with
       a mac-address identifier -->
  <xsd:element name="access-request-mac">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SingleValueMetadataType"/>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- authenticated-as is link metadata that
       associates an access-request identifier with
       an identity identifier -->
  <xsd:element name="authenticated-as">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SingleValueMetadataType"/>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- authenticated-by is link metadata that
       associates an access-request identifier with
       the ip-address identifier of the PDP that
       authenticated the access-request -->
  <xsd:element name="authenticated-by">
    <xsd:complexType>
```

```
      <xsd:complexContent>
        <xsd:extension base="SingleValueMetadataType"/>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- capability is access-request metadata that names
       a collection of privileges assigned to an endpoint -->
  <xsd:element name="capability">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="MultiValueMetadataType">
          <xsd:sequence>
            <xsd:element name="name" type="xsd:string"
              minOccurs="1" maxOccurs="1"/>
            <xsd:element name="administrative-domain"
              type="xsd:string" minOccurs="0" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- device-attribute is link metadata that associates
       an access-request identifier with a device identifier
       and which includes information about the device such
       as its health -->
  <xsd:element name="device-attribute">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="MultiValueMetadataType">
          <xsd:sequence>
            <xsd:element name="name" type="xsd:string"
              minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- event is access-request, identity, ip-address, or
       mac-address metadata that describes activity of
       interest detected on the network -->
  <xsd:element name="event">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="MultiValueMetadataType">
          <xsd:sequence>
            <xsd:element name="name" type="xsd:string"
              minOccurs="1" maxOccurs="1"/>
            <xsd:element name="event-recorded-time"
              type="xsd:dateTime" minOccurs="1"
              maxOccurs="1"/>
            <xsd:element name="magnitude" minOccurs="1"
              maxOccurs="1">
              <xsd:simpleType>
                <xsd:restriction base="xsd:integer">
                  <xsd:minInclusive value="0"/>
                  <xsd:maxInclusive value="100"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="confidence" minOccurs="1"
              maxOccurs="1">
              <xsd:simpleType>
                <xsd:restriction base="xsd:integer">
                  <xsd:minInclusive value="0"/>
                  <xsd:maxInclusive value="100"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="significance" minOccurs="1"
              maxOccurs="1">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="critical"/>
                  <xsd:enumeration value="important"/>
                  <xsd:enumeration value="informational"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="type" minOccurs="0"
              maxOccurs="1">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="p2p"/>
                  <xsd:enumeration value="cve"/>
                  <xsd:enumeration value="botnet infection"/>
                  <xsd:enumeration value="worm infection"/>
                  <xsd:enumeration value="excessive flows"/>
                  <xsd:enumeration value="behavioral change"/>
                  <xsd:enumeration value="policy violation"/>
                  <xsd:enumeration value="other"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="other-type-definition"
              type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="information"
              type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="vulnerability-uri"
              type="xsd:anyURI" minOccurs="0" maxOccurs="1"
            />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- layer2-information is link metadata that
       associates an access-request identifier with
       the ip-address identifier of the PEP through
       which the endpoint is accessing the network -->
  <xsd:element name="layer2-information">
```

```
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SingleValueMetadataType">
          <xsd:sequence>
            <xsd:element name="vlan" type="xsd:integer"
              minOccurs="1" maxOccurs="1"/>
            <xsd:element name="port" type="xsd:integer"
              minOccurs="1" maxOccurs="1"/>
            <xsd:element name="administrative-domain"
              type="xsd:string" minOccurs="0" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- ip-mac is link metadata that associates an
       ip-address identifier with a mac-address identifier
       and which includes optional DHCP lease information -->
  <xsd:element name="ip-mac">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SingleValueMetadataType">
          <xsd:sequence>
            <xsd:element name="start-time"
              type="xsd:dateTime" minOccurs="0"
              maxOccurs="1"/>
            <xsd:element name="end-time" type="xsd:dateTime"
              minOccurs="0" maxOccurs="1"/>
            <xsd:element name="dhcp-server"
              type="xsd:string" minOccurs="0" maxOccurs="1"
            />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- role is link metadata that associates an
       access-request identifier with an identity
       identifier and which names collections of
       privileges associated with the end-user -->
  <xsd:element name="role">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="MultiValueMetadataType">
          <xsd:sequence>
            <xsd:element name="administrative-domain"
              type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="name" type="xsd:string"
              minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

```
</xsd:schema>
```