TCG Trusted Network Communications

TNC IF-MAP Binding for SOAP

Specification Version 2.2
Revision 10
26 March 2014

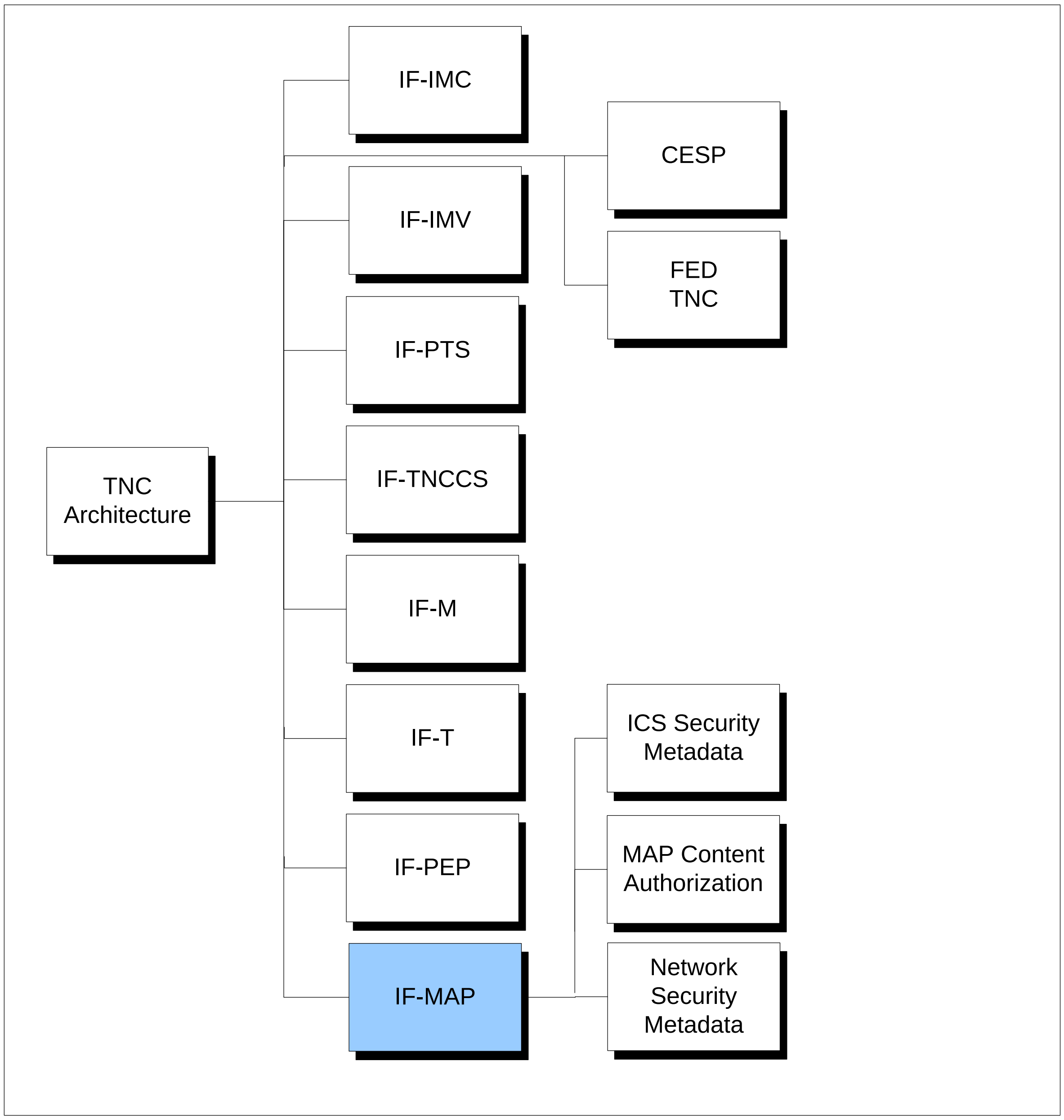
Contact:

admin@trustedcomputinggroup.org

TCG

TCG Published

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.

Maxime Olivier	**AMOSSYS**
Scott Kelly	**Aruba Networks**
Amit Agarwal	**Avaya**
Mahalingam Mani	**Avaya**
Craig Dupler	**Boeing Corporation**
Steve Hatch	**Boeing Corporation**
Richard Hill	**Boeing Corporation**
David Mattes	**Boeing Corporation**
Travis Reid	**Boeing Corporation**
Steven Venema (Co-Editor)	**Boeing Corporation**
Eric Byres (Invited Expert)	**Byres Security**
Nancy Cam-Winget	**Cisco Systems**
Allan Thomson	**Cisco Systems**
Mark Townsend	**Enterasys**
Michael McDaniels	**Extreme Networks**
Ingo Bente	**Fachhochschule Hannover**
Josef von Helden	**Fachhochschule Hannover**
Arne Welzel	**Fachhochschule Hannover**
Thomas Rossow	**Fachhochschule Hannover**
Henk Birkholz	**Fraunhofer SIT**
Nicolai Kuntze	**Fraunhofer SIT**
Hidenobu Ito	**Fujitsu Limited**
Seigo Kotani	**Fujitsu Limited**
Houcheng Lee	**Fujitsu Limited**
Sung Lee	**Fujitsu Limited**
Gerald Maunier	**Gemalto**
Graeme Proudler	**Hewlett-Packard**
Mauricio Sanchez	**Hewlett-Packard**
Andreas Steffen	**Hochschule für Technik Rapperswil**
Han Yin	**Huawei Technologies**
Diana Arroyo	**IBM**
Guha Prasad Venkataraman	**IBM**
Sean Convery	**Identity Engines**
Chris Hessing	**Identity Engines**
Morteza Ansari	**Infoblox**
Stuart Bailey (Co-Editor)	**Infoblox**
Andrew Benton	**Infoblox**
Navin Boddu	**Infoblox**
Tom Clark	**Infoblox**
Peter Lee	**Infoblox**
Rod Murchison	**Infoblox**
Ivan Pulleyn	**Infoblox**
James Tan	**Infoblox**
David Vigier (Co-Editor)	**Infoblox**
Rena Yang	**Infoblox**
Ravi Sahita	**Intel Corporation**

Ned Smith	**Intel Corporation**
Josh Howlett	**JANET(UK)**
Yan Avlasov	**Juniper Networks**
Roger Chickering (Co-Editor)	**Juniper Networks**
Charles Goldberg	**Juniper Networks**
Steve Hanna (TNC co-chair)	**Juniper Networks**
Clifford Kahn (Co-Editor)	**Juniper Networks**
PJ Kirner	**Juniper Networks**
Lisa Lorenzin (Co-Editor)	**Juniper Networks**
Yang Lee	**Juniper Networks**
John Jerrim	**Lancope**
Mark Labbancz	**Lumeta**
Matt Webster	**Lumeta**
Bill Nemec	**Lumeta**
Kent Landfield	**McAfee**
Eric Fitzgerald	**Microsoft**
Ryan Hurst	**Microsoft**
Sandilya Garimella	**Motorola**
Charles Schmidt	**MITRE**
Jens Lucius	**NCP Engineering**
Meenakshi Kaushik	**Nortel**
Carolin Latze	**SWISSCOM**
Paul Sangster (TNC co-chair)	**Symantec Corporation**
Ted Fornoles	**Trapeze Networks**
Matthew Gast	**Trapeze Networks**
Tim McCarthy	**Trapeze Networks**
Jeffrey Peden	**Trapeze Networks**
Brian Wangerian	**Trapeze Networks**
Brad Upson	**UNH InterOperability Lab**
Mike Boyle	**US National Security Agency**
Jessica Fitzgerald-McKay	**US National Security Agency**
Lauren Giroux	**US National Security Agency**
Chris Salter	**US National Security Agency**
Gloria Serrao	**US National Security Agency**
Thomas Hardjono	**Wave Systems**
Greg Kazmierczak	**Wave Systems**

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

David Mattes	Boeing
Steve Venema	Boeing
Peter Lee	Infoblox
James Tan	Infoblox
David Vigier	Infoblox
Steve Hanna	Juniper
Clifford Kahn	Juniper
Lisa Lorenzin	Juniper

Mark Labbancz	Lumeta
Matt Webster	Lumeta
Matthew Gast	Trapeze
Jeffrey Peden	Trapeze
Gloria Serrao	US National Security Agency
Mike Boyle	US National Security Agency

Table of Contents

1 Introduction

1.1 Scope and Audience

The Trusted Network Communications Work Group (TNC-WG) has defined an open solution architecture that enables network operators to control access to a network. 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 IETF 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 Overview of Changes from Version 2.1

IF-MAP 2.2 (referring to this document) includes several changes, as outlined in this section. No changes were made to the schema itself. IF-MAP 2.2 is fully backwards compatible with previous IF-MAP 2.x versions.

- Clarification of the nature of the IF-MAP data model (section 2.6)
- Clarification and grouping of normative language around how a MAP Server identifies a MAP Client (section 3.1)
- Normative requirements for how to normalize strings to lower case (section 3.2)
- Normative language around rejection of malformed identifiers (section 3.3)
- Normative requirements around validation of identifiers (3.3) and metadata (3.4)
- Clarification on the purpose of the administrative-domain attribute (section 3.3.1)
- Clarification on usage (section 3.3.1.2) and restoration (section 3.3.2.1) of administrative-domain on access-request identifiers
- Additional normative requirements on the name attribute of a device identifier (section 3.3.2.2)
- Guidance on how to express an administrative domain within the name attribute of a device identifier (section 3.3.2.2)
- Normative requirements on normalization of identity identifiers of type username (section 3.3.2.3)
- Normative requirements (strengthened from previous normative language) around X.500 distinguished name equivalence and conversion (section 3.3.2.3.1)
- Clarification of normative requirements on value form and type agreement for the ip-address identifier (section 3.3.2.4)
- New operational identifier types and definition of the ifmap-server identifier (section 3.3.4)
- New ifmap-timestamp-fraction metadata attribute (section 3.4 and 3.4.2)
- Clarification around ifmap-publisher-id (section 3.4.1) reflecting changes to section 3.1

- New operational metadata type server-capability (section 3.4.5.2)
- Clarification of namespace prefix declaration in filters (section 3.5) reflecting changes to section 3.9.1
- Normative requirements around schema compliance (section 3.7)
- Normative requirements around namespace prefix declaration (section 3.9.1)
- Additional terminal identifier types (section 3.9.3.2.6)
- Clarification around identity identifiers vs. extended identifiers in search results (section 3.9.3.5)
- Normative language around poll results that approach/exceed the size limit (section 3.9.5.2)
- Normative language and clarification around delta pollResults with no metadata (section3.9.5.3)
- Clarification on difference between search and poll (section 3.9.5.9)
- Guidance on handling many concurrent MAP Clients (section 4.8)
- Security considerations for Certification Authorities (sections 6.1.4, 6.2.4, and 6.3.4)
- Normative language around how a MAP Server authenticates a MAP Client and authorizes IF-MAP operations (section 6.3.1)
- Stronger normative language around use of TCG technology to verify MAP Client identity / integrity (section 6.3.2)
- New MAP Server schema (section 10.4)

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 endpoints, 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. All of this information can be stored in the MAP and made available to authorized parties.

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 MAP Server changes. A single MAP Client may publish, search, and subscribe; however, many MAP Clients are solely 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 that *nominally* reflects recent metadata values and relationships as reported by MAP Clients. A MAP Server cannot guarantee that the information it dispenses is accurate. MAP Clients control the accuracy of the data. 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 2.2 (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 MAP Client, such as a PDP, Sensor, or PEP, publishes metadata to a MAP Server.
- A MAP Client, such as a PDP or Flow Controller, searches a MAP Server for metadata associated with an endpoint.
- A MAP Client, such as a PDP or Flow Controller, subscribes to notifications from a MAP Server about changes in metadata for an endpoint.

2.4 Requirements

The following are the requirements that 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 frequent updates to metadata. While directory protocols like LDAP are optimized for infrequent updates and frequent reads, IF-MAP is required to have strong support for frequent updates and reads.

2. **Secure**

   - Communication between MAP Clients and Servers MUST be authenticated and integrity-protected against unauthorized modifications en route.
   - 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, users, etc.) for which metadata exists and which are interacting with the MAP Server. For example users, endpoints, 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 an IF-MAP service in a large organization in which thousands of updates occur per second. It is expected that within a few years, 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

As described in the TNC Architecture specification [1], the IF-MAP related components in the TNC architecture include:

- A MAP Client role which includes Flow Controllers and Sensors. A Flow Controller is an enforcer that is not required to communicate directly with a PDP. A Sensor shares metadata about the network.
- A Metadata Access Point role which coordinates metadata exchange.

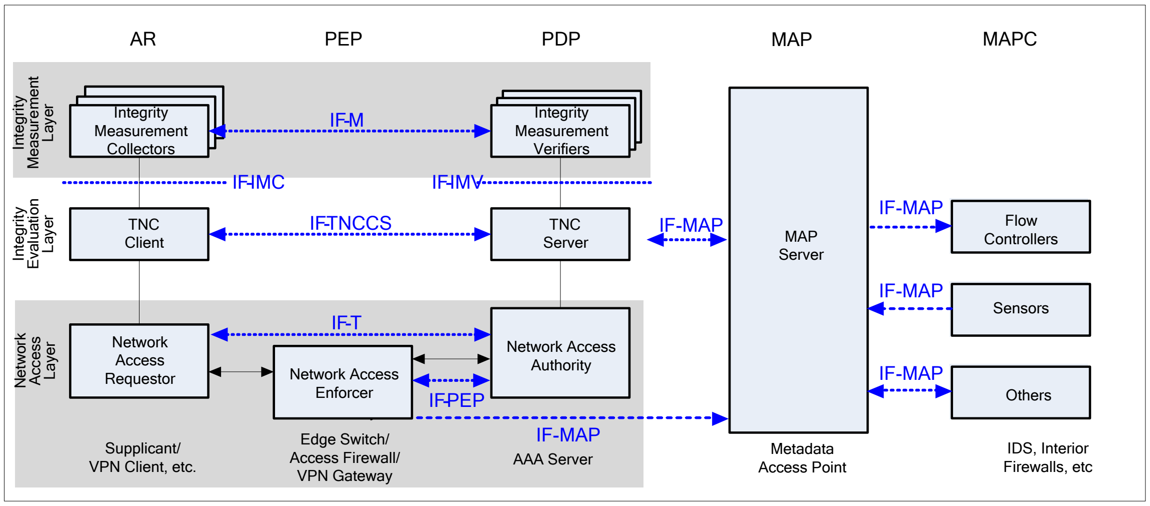


Figure 1: TNC Architecture

The Metadata Access Point role of the TNC architecture is performed by an element that has a MAP Server function. All network elements that use IF-MAP to access a MAP Server (PEPs, Flow Controllers, Sensors, PDPs, and any other elements) are MAP Clients. All MAP Clients must be authenticated and authorized to use the MAP Server.

2.6 Data Model

The IF-MAP data model is an undirected, labeled graph. For this reason, the collection of data stored on a MAP Server is sometimes referred to as the MAP Server's "MAP graph". The edges in this graph are called links, while the nodes are called identifiers. Metadata represents additional information which can be attached to identifiers or links. A single item of metadata can only be attached to a single entity (one identifier or one link), but a single entity may have multiple items of metadata attached to it. Identifiers are intended to be succinct labels, containing only enough information to identify some real-world entity (e.g., a device, an IP address, a user, an access request, etc.), while metadata is intended to be descriptive of either that entity or the relationship between two entities. All IF-MAP data types and operations are represented as XML documents.

MAP Clients modify the MAP graph by requesting IF-MAP operations to add, remove, or update metadata to specific identifiers or links in the graph. Links and identifiers are not explicitly created, but all possible links and identifiers are treated as always existing, regardless of whether they are currently in use. This allows a MAP Client to assign metadata to any valid identifier or link, regardless of whether it was explicitly part of a MAP Server's MAP graph. Links are only meaningful when there is metadata attached to them; identifiers are only meaningful if there is metadata attached to them or when they are one end of a meaningful link. Data is retrieved from the MAP graph (using the "search" operation) by identifying a starting identifier and, from there, traversing the graph according to client-specified rules. The result consists of a set of identifiers, links, and their attached metadata. Only meaningful identifiers and links are traversed when collecting data from the map graph. (A request could start on an identifier that was not meaningful, but the result set would only consist of that identifier since there would be no meaningful links to traverse.) As such, all search returns are themselves graphs representing some subset of the MAP Server's full MAP graph at a given point in time.

Figure 2 illustrates an example MAP graph containing metadata from the IF-MAP Metadata for Network Security specification [15]:

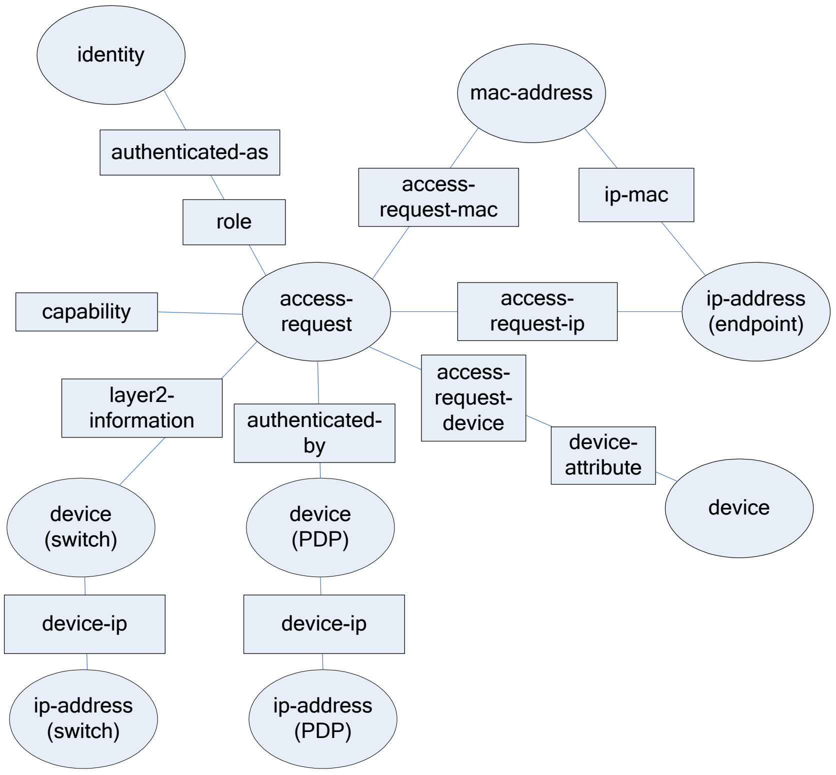


Figure 2: Example MAP Graph

In Figure 2, identifiers are represented by ovals, metadata is represented by rectangles, and links are represented by lines connecting identifiers.

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

All identifiers in an identifier space are always legal for any operation within the limits of a MAP Server's authorization policy for the operating MAP Client. 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).

An identifier or set of identifiers is required for all IF-MAP metadata operations. The publish operation associates metadata with identifiers or links between identifiers. The search and subscribe operations use an identifier as the starting point for a query (see 3.9).

There are two classes of identifiers:

- Original Identifiers - see section 3.3.2
    - The schemas for all original types of identifiers are defined by this document.
    - There are 5 original identifier types and they provide network-oriented elements such as IP address.

- Extended Identifiers - see section 3.3.3
    - Extended identifier types are defined in external schemas.
    - They allow vendors and other standards to supplement the identifier space and develop elementary types for their applications.

2.6.2 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

For purposes of extensibility, the schema for standard metadata is defined in supporting specifications, such as TNC IF-MAP Metadata for Network Security[15]. Additionally, the TNC IF-MAP Binding for SOAP specification defines a set of Standard Metadata types called Operational Metadata in section 3.4.5. Most metadata is added to the MAP graph by MAP Clients; the exception is operational metadata, which may be added by MAP Clients or the MAP Server depending upon the metadata type.

Vendor-specific metadata is used to supplement the use cases defined by the IF-MAP Metadata specifications. 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 be capable of, and support as default behavior, storage and retrieval of vendor-specific metadata regardless of whether the MAP Server can validate the vendor-specific metadata. All MAP Server implementations MAY support XML Schema validation as described by [3] and [4]. All MAP Clients MAY likewise validate documents using the same mechanisms and MUST NOT send metadata that does not comply with the relevant schema. No provisions are made in this specification for uploading XML Schemas.

2.6.3 Link

IF-MAP specifies a **link** as an unnamed, bi-directional binding relationship between two **identifiers** indicated by metadata attached to the pair of identifiers. For example, a DHCP server might publish ip-mac metadata indicating a link between a mac-address identifier and an ip-address identifier. A link is identified in a MAP operation as a pair of identifiers.

3 IF-MAP Interface

3.1 Clients

It is important to know how a MAP Server differentiates one MAP Client from another, because (a) each MAP Client can have only one session at a time and (b) each MAP Client gets a different and permanent ifmap-publisher-id string (see section 3.4.1).

A MAP Server MUST distinguish MAP Clients by all of:

- The authentication method used (basic or certificate)
- The MAP Client's username if basic authentication is used
- The Subject field of the client certificate, and the Issuer field of the certificate at the top of the client certificate chain (the "trust anchor"), if certificate authentication is used
    - A MAP Server administrator should not configure two different trust anchors with same Issuer field, unless those trust anchors coordinate to ensure that the same subject name implies the same identity. Otherwise, overlapping credentials could cause security problems.
    - For the Subject and Issuer field, a MAP Server MUST consider two fields whose value is equivalent under the comparison rules specified in section 3.3.2.3.1 to be the same. The certificate chain can have length 1 or greater.
- The source IP address of the MAP Client's connection to the MAP Server, in the case where a list of permitted IP addresses or address ranges is provided for the sole use of that MAP Client by the administrator

The MAP Server MAY be configurable with a list of zero or more IP addresses or address ranges to which the MAP Client's source IP address may belong. Each list is intended to be a set of alternate addresses for a particular MAP Client (for example, a multi-homed MAP Client). If no IP addresses are specified, the MAP Server MUST NOT consider the source IP address of the MAP Client when mapping to the publisher-id.

A MAP Server MAY consider other, implementation-dependent factors when distinguishing MAP Clients. However, a MAP Server MUST NOT consider secret credentials, such as the password or private key. A MAP Client MUST be able to change credentials (e.g. new password or new cert after expiration) and still be seen by the MAP Server as the same MAP Client. A MAP Server MUST NOT consider time-dependent or connection-dependent variables.

3.2 String Encoding

MAP Clients and MAP Servers MUST exchange string values in UTF-8.

Whenever this specification calls for a MAP Client or MAP Server to normalize a string to lower case, the normalizing SHOULD be done according to the default toCasefold() function defined in Chapter 3 of The Unicode Standard[37]. (This "SHOULD" may become a "MUST" in a future version of this specification.)

3.3 Identifiers

Identifiers provide a unified namespace that can be used as attachment points for specific metadata instances. There are several types of identifiers, which fall into two categories: original identifiers, as defined in IF-MAP 1.0 (access-request, device, identity, ip-address, and mac-address); and extended identifiers, which enable extension of the identifier space (see section 3.3.3). 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 MAP Client that exceeds its maximum identifier length, the MAP Server SHOULD respond with an IdentifierTooLong error (see section 3.8.1). If a MAP Client receives an identifier from a MAP Server that exceeds its maximum identifier length, the MAP Client SHOULD treat the operation as having failed and log an administrator-viewable message.

The MAP Server MUST validate original identifiers, and SHOULD validate extended identifiers for which it has the schema, to ensure that they conform to the requirements of this section. In the case where an invalid identifier is detected, the MAP Server MUST reject it with an InvalidIdentifier errorResult. Examples of invalid identifiers include a mac-address identifier in which the value attribute contains uppercase letters, or an ip-address identifier with no type specified, or an identity identifier of type email-address in which the name attribute contains uppercase letters in the domain part of the email address.

3.3.1 Administrative Domains

The administrative-domain attribute, a string whose format is organizationally defined, is an optional qualifier used to differentiate multiple instances of the same identifier used in separate, discrete environments (e.g. the same IP address used in multiple routing domains, as often happens in environments utilizing NAT). For example, in an environment where the MAP Clients are configured to utilize multiple administrative domains, setting the administrative-domain attribute on ip-address and mac-address identifiers enables any MAP Client observing a packet on the network to figure out the appropriate ip-address and mac-address identifiers for the packet. This ensures that two MAP Clients observing the same packet come up with the same identifier, and that two addresses that look the same but denote different objects are represented by different identifiers.

3.3.1.1 Requirements

A MAP Server MUST process the administrative-domain attribute as CASE SENSITIVE. A MAP Server MUST process as equivalent (i.e. belonging to the same administrative domain) administrative-domain attributes which are specified as an empty string or unspecified.

The administrative-domain qualifier is “optional” in the sense that the administrator should have the option to configure it or not configure it as appropriate for their environment. For all identifiers on which the administrative-domain attribute is permitted by the schema, a MAP Client MUST allow the administrator either to apply no administrative-domain or configure an empty string for the administrative-domain, and MUST allow the administrator to manually configure the administrative-domain, overriding any default configuration of the administrative-domain.

A MAP Client MAY have a default configuration; the recommended default configuration is no administrative-domain qualifier.

A MAP Client that derives the content of the administrative-domain attribute from a case insensitive external source MUST normalize the administrative-domain attribute to lower case (see 3.2).

3.3.1.2 Recommended Usage

- For access-request identifiers (see section 3.3.2.1), the administrative domain may identify a group of cooperating PDPs.
- For device identifiers (see section 3.3.2.2), the administrative domain may identify a group of cooperating PDPs. For original identity identifiers (see section 3.2.2.3), the administrative domain is intended to represent the authentication authority from which the identity is drawn.
- For ip-address identifiers (see section 3.3.2.4), the administrative domain is intended to represent the routing domain in which the IP address occurs. Where two network interface cards (or virtual NICs) could use the same IP address without a conflict in the network, administrators should ensure that the corresponding ip-address identifiers are different. They should ensure this by configuring MAP Clients’ administrative domains. The general practice is one administrative domain for each IP routing domain.

- For mac-address identifiers (see section 3.3.2.5), the administrative domain is intended to represent the local network (VLAN / LAN segment) in which the MAC address occurs. Where two network interface cards (or virtual NICs) could use the same MAC address without a conflict in the network, administrators should ensure that the corresponding mac-address identifiers are different. They should ensure this by configuring MAP Clients' administrative domains. They should not rely on the uniqueness of manufacturer-supplied MAC addresses. The general practice is one administrative domain per local network (VLAN / LAN segment).

3.3.2 Original Identifiers

3.3.2.1 access-request

An access-request identifier represents a request for access to a network by a logical endpoint. Multiple access-request identifiers for the same endpoint may be stored in the MAP database, such as when a multi-homed endpoint requests access to multiple networks. Note: the use of administrative-domain attributes for access-request identifiers was deprecated in IF-MAP 2.1, but is restored in IF-MAP 2.2.

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 attributes in the two access-request identifiers are equivalent.

A MAP Client MUST specify a name attribute consisting of a non-empty string. MAP Clients MUST choose the value of the name attribute in such a way that the odds of having two logically different access-request identifiers with the same name are negligible. The following strategies all satisfy this requirement:

- The name attribute takes the form "***ifmap-publisher-id***:***UID***" where ***ifmap-publisher-id*** refers to a specific MAP Client and ***UID*** may be a simple ordinal value
- The name attribute is a UUID as described in IETF RFC 4122[16]
- The name attribute is based on a cryptographically strong random number of at least 128 bits

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

3.3.2.2 device

A device identifier represents a physical or virtual asset, such as an endpoint which is attempting to gain entry to or is present on a network, or a PDP or other authenticating element. However, device identifiers are not (in general) permanent or unique. There can be many device identifiers for one asset, and the set can change.

IF-MAP specifies a DeviceType consisting of either an aik-name string or a name string. Because aik-names are not guaranteed to be globally unique, that attribute is deprecated; see section 5.7 of [29]. MAP Clients MUST NOT refer to a device identifier with an aik-name. 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.

A MAP Client MUST specify a name attribute consisting of a non-empty string. The name attribute MUST satisfy the same uniqueness requirements as the name attribute of an access-request identifier and MUST NOT begin with the reserved prefix ifmap_:.

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>
```

Note that unlike other identifiers, for historical reasons, the device identifier uses XML elements, rather than XML attributes, to store its field values.

Also for historical reasons, a device identifier cannot have an administrative-domain attribute, the way other identifiers do. However, an administrative-domain attribute is useful for MAP Content Authorization, to control access to device identifiers by grouping, so that a MAP Server can enforce different authorization restrictions based on the administrative-domain attribute of the device identifier. The following notation provides a way to express an administrative domain within the name attribute:

> ifmap_:***DOMAIN***:***TAIL***, where ***DOMAIN*** is a string of one or more non-colon characters and ***TAIL*** is a string of one or more characters.

For maximum interoperability, it is advisable to keep the entire identifier under a size restriction of 1000 bytes, which is the minimum length MAP Servers are required to support per section 3.3.

MAP Clients using this notation MUST choose the value of the name attribute in such a way that the odds of having two logically different access-request identifiers with the same name are negligible. The following strategies all satisfy this requirement:

- ***TAIL*** takes the form "***ifmap-publisher-id***:***UID***" where ***ifmap-publisher-id*** refers to a specific MAP Client and ***UID*** may be a simple ordinal value
- ***TAIL*** is a UUID as described in IETF RFC 4122[16]
- ***TAIL*** is based on a cryptographically strong random number of at least 128 bits

3.3.2.3 identity

An original (non-extended) identity identifier (i.e., an identity identifier that is not an extended identifier as defined in section 3.3.3) represents an end-user, device, application or other logical or physical entity. A single organization may have several differentiated units, or administrative domains, in which identities are independently provisioned. A non-extended identity identifier MAY include an administrative-domain attribute in order to distinguish the identity. Identities may take several different syntactic forms, as described in this section.

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 identity identifiers as equivalent if and only if the values of ALL corresponding attributes in the two identity identifiers are equivalent based on a binary comparison, except where this specification requires special handling for a particular kind of identity identifier type (such as distinguished-name).

The type attribute determines the format of the value of the name attribute. (E.g., a dns-name type would indicate that the name attribute contained a DNS name.) For some of these types, the name is based on case-insensitive information (such as a DNS name). When constructing a name based on case insensitive material (such as a DNS name), MAP Clients MUST normalize to lower-case (see 3.2). When constructing an identity identifier of type username, a MAP Client MUST normalize case-insensitive characters to lower case and MUST preserve case for case-sensitive characters; for example, a Windows username is entirely case-insensitive and requires normalization, whereas a Unix username is case sensitive, so normalization is prohibited. This requires that the MAP Client be capable of determining whether the original username is case sensitive; the mechanism for doing so is implementation dependent. When constructing an identifier of type email-address, a MAP Client MUST normalize the domain part to lower case, and SHOULD normalize all case-insensitive characters of the local part to lower case, as specified in IETF RFC 5322 [23].

A MAP Client MUST specify a name attribute consisting of a non-empty string. A MAP Client MUST specify a type in order to indicate to the MAP Server how to parse the name attribute. A MAP Server MUST process two syntactically equal name attributes as distinct if they are associated with different types. For example, an identity with an unspecified administrative domain, type=username, and name="foo.bar" is distinct from an identity with an unspecified administrative domain, type=dns-name, and name="foo.bar", since the types are different.

3.3.2.3.1 Distinguished Names

For X.500 distinguished names, MAP Servers and MAP Clients MUST use the following equivalence rules, which are identical to the ones used by XACML 2.0[30]:

1. Normalize the two name attributes in the respective identity identifiers according to IETF RFC 2253 [31].
2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the AttributeValuePairs in that RDN in ascending order when compared as octet strings (described in ITU-T Rec. X.690 [32], Section 11.6 "Set-of components").
3. Compare RDNs using the rules in IETF RFC 3280 [33], Section 4.1.2.4 "Issuer".

When returning an identity identifier with a type of distinguished-name to a MAP Client (e.g. in a searchResult or a pollResult), a MAP Server MAY use any form of that distinguished name that is equivalent according to the algorithm specified above.

In order to ensure interoperability between MAP Servers and MAP Clients, X.500 distinguished names MUST be converted to UTF-8 string form before being used in the name attribute of an identity identifier, using the algorithm described in IETF RFC 2253 [31]. For AttributeTypes not listed in the table on page 4 of that specification, the dotted-decimal notation MUST be used.

3.3.2.3.2 Identity Type Table

The following table lists supported identity types, with the Usage column indicating requirements for the name attribute based on the value of the type attribute:

Identity Type	**Usage**
aik-name	TCG AIK Name [20]
distinguished-name	An X.500 Distinguished Name [21]
dns-name	A name from the Domain Name System [22]
email-address	An email address, e.g. sally@example.com [23]
kerberos-principal	A Kerberos unique identity [24]
username	A user's login name
sip-uri	A SIP identifier [25]
tel-uri	A telephone number URI [26]

hip-hit	Host Identity Tag (HIT) from the Host Identity Protocol. This is typically a 128-bit hash of a public key with some of the upper bits masked out for other purposes. [27] HIT MUST be expressed as 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 HIT. Examples: 2001:10:7654:3210:fedc:ba98:7654:3210 2001:10:0:0:8:800:200c:417a No leading zeros are allowed except that the number 0 is represented by a single 0 character.
other	If a MAP Client specifies identity type as “other”, the MAP Client MUST specify a non-empty string for the other-type-definition attribute. The other-type-definition attribute’s value MUST take one of two forms: 1. “Vendor-ID:Name”: A vendor-defined type. Vendor-ID is an SMI Private Enterprise Number [19] 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. This includes, but is not limited to, the TCG-defined type “extended”, specified for extended identifiers support (see section 3.3.3).

```
<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="username"/>
        <xsd:enumeration value="sip-uri"/>
        <xsd:enumeration value="tel-uri"/>
        <xsd:enumeration value="hip-hit"/>
        <xsd:enumeration value="other"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="other-type-definition" type="xsd:string"/>
</xsd:complexType>
```

3.3.2.4 ip-address

An ip-address identifier represents a single IP address.

Since many networks are deployed using overlapping IP address spaces, when utilizing the IPAddressType defined in IF-MAP, a MAP Client MAY specify an administrative-domain attribute in order to uniquely identify the address.

IF-MAP specifies an IPAddressType consisting of administrative domain string, value string, and type enumeration. A MAP Server MUST process two ip-address identifiers as equivalent if and only if ALL corresponding attributes in the two ip-address identifiers are equivalent.

A MAP Client MUST specify a value attribute consisting of a non-empty string. A MAP Client MUST specify a type in order to indicate to MAP Clients how to parse the value attribute. 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”. The form of the value attribute MUST agree with the “type” attribute.

IP addresses MUST be canonicalized by MAP Clients.

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

For the purposes of this specification, 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

No leading zeros are allowed except that the number 0 is represented by a single 0 character.

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

3.3.2.5 mac-address

A mac-address identifier represents a single Ethernet MAC address.

Due to the prevalence of locally administered (“virtual”) MAC addresses, when utilizing the MACAddressType defined in IF-MAP, a MAP Client MAY specify an administrative-domain attribute in order to uniquely identify the address. In fact, including an administrative-domain attribute with a MACAddressType is a good idea in general since it reduces the impact of MAC address spoofing.

IF-MAP specifies a MACAddressType consisting of administrative-domain string and value string. A MAP Server MUST process two mac-address identifiers as equivalent if and only if ALL corresponding attributes in the two mac-address identifiers are equivalent.

A MAP Client MUST specify a value attribute consisting of a non-empty string. 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.

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

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

3.3.3 Extended Identifiers

This section describes a mechanism to extend the identifiers space with new identifier types. It explains the reason behind this extension and specifies the requirements that result from it.

All MAP Servers MUST support both original and extended identifiers as specified in this document.

3.3.3.1 Rationale

When IF-MAP was first developed, its purpose was to integrate with the TNC architecture and provide a shared repository for metadata in this context. The scope of the original identifiers as defined in previous IF-MAP specifications is driven by use cases such as network security and access control.

However, the simplicity, extensibility, and aggregation capability of IF-MAP make it attractive for other applications. As new use-cases appear, the need for new identifier types emerges. For example, an extended identifier representing a network enables a DHCP server acting as a MAP Client to publish links between ip-address identifiers and a network identifier to model the physical network in the MAP, allowing other MAP Clients to search for all IP addresses in that network. A configuration management database (CMDB) acting as a MAP Client could also use this network identifier to group assets it manages. For even broader flexibility, an extended identifier representing a group enables MAP Clients to express group membership by linking device and/or identity identifiers to the new identifier.

IF-MAP already supports identifier extension by the mean of the “other” identity type (see section 3.3.2.3). The “other” identify type has some limitations that extended identifiers are intended to address. Particularly, extended identifiers provide a way to define new structured identifier types not limited by a single “name” attribute. Section 3.3.3.2 defines a new extended identifier type; however, IF-MAP 2.0 (or earlier) MAP Clients will not recognize these new extended identifiers. For backwards compatibility, section 3.3.3.3 specifies a mandatory method of encapsulating an extended identifier within an identity identifier of type "other".

3.3.3.2 Extended Identifier Type Definition

Extended identifier types are declared in an XML schema, which defines a content model for the extended identifiers of the particular type.

An XSD document is assigned a unique namespace. An extended identifier type is identified by its element name within this namespace.

This specification provides the base type for extended identifiers, which has a required administrative-domain attribute.

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

The schema in which this base type is defined appears in section 10.3.

All extended identifier types MUST extend the IdentifierType complexType as defined by this specification. If no administrative domain applies, the administrative-domain attribute MUST be set to an empty string in an extended identifier.

Extended identifier types MAY have attributes and elements specified in a complex type as defined by the World Wide Web Consortium (W3C) in the XML Schema specifications [4].

Elements of an extended identifier type MAY themselves be specified as a complex type and have attributes as well as sub-elements.

All MAP Servers and Clients MAY optionally support XML Schema validation for extended identifiers, but MAP Servers MUST at least validate that the extended identifiers are well-formed XML documents.

The following restrictions apply to the declaration of the schema components of an extended identifier type:

- An attribute declaration MUST NOT specify a default value via a *default* attribute.
- An element declaration MUST NOT specify a default value via a *default* attribute.

These are restrictions on schema attributes that affect the bindings of an extended identifier and, as such, would require a MAP Server or Client to know the schema definition of such extended identifier type. Any other schema attributes only affect the content validation and are therefore allowed.

The following example defines an extended identifier type for network identifiers:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:base-id="http://www.trustedcomputinggroup.org/2012/IFMAP-
IDENTIFIER/1"
  xmlns="http://www.example.com/extended-identifiers"
  targetNamespace="http://www.example.com/extended-identifiers">

  <xsd:element name="network">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="base-id:IdentifierType">
          <xsd:attribute name="address" type="xsd:string"
use="required"/>
          <xsd:attribute name="netmask" type="xsd:string"
use="required"/>
          <xsd:attribute name="type" use="required">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="IPv4"/>
                <xsd:enumeration value="IPv6"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:attribute>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

This network identifier has 4 attributes of type string. For stronger validation, the schema may provide additional restrictions. For instance, the definition of the “address” attribute may include a *pattern* (not shown here) to match the string in the attribute value, to verify that it is a valid IPv4 or IPv6 address.

An instance of such a network identifier would look like:

```
<ns:network
   xmlns:ns="http://www.example.com/extended-identifiers"
   address="10.0.0.0"
   netmask="255.0.0.0"
   type="IPv4"
   administrative-domain=""/>
```

MAP Servers that implement XML Schema validation for a particular extended identifier type MUST return an InvalidIdentifier error result (see section 3.8.1) to any request containing an extended identifier of such type that does not pass the schema validation.

A MAP Server MUST process two extended identifiers as equivalent if and only if they compare equal from an XML point of view. The algorithm to compare two XML documents is outside the scope of this specification. However, given the normalization procedure that MAP Clients are required to implement (see 3.3.3.3), a MAP Server only needs to perform binary comparison of the canonical representation of extended identifiers to assert their equivalence. A more generic comparison algorithm would satisfy the following criteria, given here for clarification:

- The type name and namespace are the same
- The namespace prefix that qualifies the element name is not used for the comparison
- An optional attribute or element may be omitted in the extended identifier. If omitted, that identifier is considered different from an identifier with the same attribute or element with an empty value.

When designing a schema for an extended identifier, keep in mind that:

- Identifiers are key objects in the MAP data model. They should uniquely identify an entity. The attributes or elements of an extended identifier should be chosen such that they do not include data that varies over time, for instance.
- Keeping identifiers small in size is recommended for efficiency, since they are likely to be indexed in a MAP Server implementation.

If there is information associated with a resource that is large in size, the preferred way to handle this is to create a simple identifier that is small and unchanging, then create a piece of metadata that can be associated with that identifier, which can be used to store the large and dynamic information.

3.3.3.3 Extended Identifiers in IF-MAP Requests and Responses

IF-MAP 2.1 introduced the new TCG-defined type “extended” for identity of type “other”. In order to provide a definition of this feature that is backward compatible with earlier versions of IF-MAP, extended identifiers defined in the previous section MUST be encapsulated in an identity identifier of type “other”.

The mechanism described in this section is an interim definition to ensure interoperability between MAP Servers and Clients implementing IF-MAP 2.1 (or later) and earlier versions of the protocol, by using a new type of identity and passing the extended identifier in the “name” attribute of the identity identifier.

Although encapsulated extended identifiers have the syntactic form of identity identifiers in this revision of the IF-MAP specifications, any text in this or any TNC specification that mentions IF-MAP identity identifiers and does not specifically mention extended identifiers shall be understood to exclude extended identifiers.

In order to use extended identifiers in requests and responses, MAP Servers and Clients MUST convert the extended identifier to an encoded string as described below. They MUST create an identity identifier with type “other”, set its other-type-definition attribute to “extended”, and set its name attribute to the encoded extended identifier. The administrative-domain attribute of the identity identifier MUST NOT be specified.

The encoding of an extended identifier is defined as the canonical XML representation in which special characters are escaped. MAP Servers and Clients MUST convert extended identifier to a string, equivalent to what results from the following algorithm:

1. Produce the XML document representing the extended identifier.
2. If the extended identifier element is qualified by a namespace prefix, remove the prefix and set the extended identifier namespace as the default for the element. The intent of this step is to remove any reference to the namespace prefix, which by definition is local to a particular XML document to prevent any ambiguity.
3. Normalize this document using the canonicalization procedure defined by the W3C Canonical XML Version 1.1 [28].[1]
4. Escape special characters present in the canonical representation using the encoding defined in the XML specifications [3]. This consists of replacing the seven characters [<>&"‘] by their by XML entities.[2]

Steps 2 and 3 ensure that there is a one-to-one mapping between an identity identifier and an extended identifier; this process guarantees repeatability in generating an identity identifier from an extended identifier in such a way that two identity identifiers from a single extended identifier will always match. This allows the use of binary match to compare two extended identifiers, which is compatible with the IF-MAP 2.0 specification in regards with the identity identifiers comparison. Since extended identifiers are structured objects, MAP Clients MUST normalize attributes and elements that are case-insensitive to lower-case (see 3.2).

MAP Servers and Clients MUST NOT create multiple layers of nested escaped extended identifiers. For XML validation of extended identifiers, the MAP Server must perform one and only one round of un-escaping before checking against the schema.

For maximum interoperability, it is advisable to keep the entire identifier under a size restriction of 1000 bytes, which is the minimum length MAP Servers are required to support (per section 3.3). Note that the restriction applies to the enclosing identity identifier, which slightly reduces the limit for the extended identifier itself.

The execution of this algorithm on a network identifier (as defined in the previous section) would give the following output:

- Step 1:

```
<ns:network
   xmlns:ns="http://www.example.com/extended-identifiers"
   address="10.0.0.0"
   netmask="255.0.0.0"
   type="IPv4"
   administrative-domain=""/>
```

- Step 2:

```
<network
   xmlns="http://www.example.com/extended-identifiers"
   address="10.0.0.0"
```

[1] As of this writing, there is a more recent XML canonicalization procedure published by the W3C (http://www.w3.org/TR/xml-c14n2/), but it is a Working Group Note and is not in progress to become an official W3C Recommendation.

[2] Extended identifier values may already contain XML entities such a &lt;, in which case the & symbol will be escaped in this step. This results in a sequence of characters such as &amp;lt; for the example. This is expected and ensures that the decoding will reconstruct proper XML, containing the initial escape sequence &lt;

```
   netmask="255.0.0.0"
   type="IPv4"
   administrative-domain=""/>
```

- Step 3:

```
<network xmlns="http://www.example.com/extended-identifiers"
address="10.0.0.0" netmask="255.0.0.0" type="IPv4"
administrative-domain=""></network>
```

- Step 4:

```
&lt;network xmlns=&quot;http://www.example.com/extended-
identifiers&quot; address=&quot;10.0.0.0&quot;
netmask=&quot;255.0.0.0&quot; type=&quot;IPv4&quot;
administrative-domain=&quot;&quot;&gt;&lt;/network&gt;
```

Such an encoded identifier may be used in a publish request (see section 3.9.2) as follows:

```
<?xml version="1.0"?>
<ifmap:publish session-id="111"

xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
   xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
   <update>
      <identity name="&lt;network
xmlns=&quot;http://www.example.com/extended-identifiers&quot;
address=&quot;10.0.0.0&quot; netmask=&quot;255.0.0.0&quot;
type=&quot;IPv4&quot; administrative-
domain=&quot;&quot;&gt;&lt;/network&gt;" type="other" other-type-
definition="extended"/>
      <metadata>
         <meta:location ifmap-cardinality="singleValue">
            <name>HQ</name>
         </meta:location>
      </metadata>
   </update>
</ifmap:publish>
```

3.3.4 Operational Extended Identifiers

Operational extended identifiers are used by the MAP Server to enable MAP Clients and Servers to establish a context for IF-MAP operations. IF-MAP 2.2 specifies an operational extended identifier, ifmap-server, which is defined in a dedicated XML schema in section 10.4.

3.3.4.1 ifmap-server

The ifmap-server identifier is a well-known extended identifier that provides a dedicated identifier on which MAP Servers can create server-capability metadata to indicate supported capabilities (see section 3.4.5.2). The administrative-domain attribute of the ifmap-server extended identifier MUST be empty.

The schema definition of the ifmap-server identifier is:

```
<xsd:element name="ifmap-server">
  <xsd:complexType>
    <xsd:extension base="base-id:IdentifierType">
    </xsd:extension>
  </xsd:complexType>
</xsd:element>
```

The ifmap-server extended identifier can be encoded in an identity:other identifier name element as follows:

```
&lt;ifmap-server
xmlns=&quot;http://www.trustedcomputinggroup.org/2013/IFMAP-
SERVER/1&quot; administrative-domain=&quot;&quot;&gt;&lt;/ifmap-
server&gt;
```

3.4 Metadata

IF-MAP metadata takes the form of an XML element that is a child element of the <metadata> element. All IF-MAP metadata elements are is allowed to have the attributes in the metadataAttributes attributeGroup. Subsequent versions of this specification or other TNC specifications may define new XML attributes for IF-MAP metadata elements. To anticipate this, the schema places <anyAttribute/> on every top-level element's declaration.

```
<xsd:simpleType name="IfmapTimeStampFractionType">
  <xsd:restriction base="xsd:decimal">
      <xsd:minInclusive value="0"/>
      <xsd:maxExclusive value="1"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:attributeGroup name="metadataAttributes">
  <xsd:attribute name="ifmap-publisher-id"/>
  <xsd:attribute name="ifmap-timestamp" type="xsd:dateTime"/>
  <xsd:attribute name="ifmap-timestamp-fraction"
type="xsd:IfmapTimeStampFractionType"/>
  <xsd:anyAttribute/>
</xsd:attributeGroup>
```

MAP Clients SHOULD normalize to lower-case (see 3.2) any metadata for which the data is not case sensitive before sending the metadata to the MAP Server, and SHOULD do the same normalization on a local basis, for purposes of comparison with metadata received from the MAP Server.

The MAP Server SHOULD validate metadata for which it has the schema; in the case where invalid metadata is detected, the MAP Server MUST reject it with an InvalidMetadata errorResult (see section 3.8.1).

ifmap-publisher-id, ifmap-timestamp, and ifmap-timestamp-fraction are special attributes (known as “operational attributes”) that MAP Servers add to stored metadata. MAP Clients MUST NOT specify operational attributes in publish requests (section 3.9.2). MAP Servers MUST include ifmap-publisher-id and ifmap-timestamp in searchResults (section 3.9.3.5) and pollResults (section 3.9.5.2), and SHOULD include ifmap-timestamp-fraction. If a change occurs to an operational attribute on a metadata element, MAP Servers MUST notify subscribers even if no change is made to the metadata itself. For example, if a MAP Client publishes location metadata, and then five minutes later publishes identical location metadata again, both events will result in notification from the MAP Server to subscribing MAP Clients of the same metadata, the latter with an updated ifmap-timestamp and ifmap-timestamp-fraction.

The ifmap- prefix in metadata attribute names is reserved for use by this specification and its successors. Metadata elements MUST NOT include attributes that begin with the ifmap- prefix other than attributes specified in this document or its successors. Any unrecognized attributes beginning with the ifmap- prefix MUST be ignored by MAP Clients and MAP Servers.

3.4.1 ifmap-publisher-id

ifmap-publisher-id is a unique value assigned by a MAP Server and associated with a specific MAP Client which performs a publish operation. A MAP Client is informed of its ifmap-publisher-id by the MAP Server, in response to a newSession request (see section 4.3).

The ifmap-publisher-id is assigned by a MAP Server to identify a particular MAP Client. A MAP Server MUST NOT assign the same ifmap-publisher-id to multiple different MAP Clients and MUST consistently use the same ifmap-publisher-id for a particular MAP Client. For the definition of a MAP Client see section 3.1.

The way of deriving the ifmap-publisher-id is implementation dependent, which means the same MAP Client might be represented by two different ifmap-publisher-ids on two different MAP Servers.

A MAP Server might be able to derive the ifmap-publisher-id from the DN of the MAP Client and of the trust anchor; however, if that is not possible, these rules may require the MAP Server to maintain configuration information about each MAP Client that might publish data. In that case, the MAP Server MUST maintain a persistent mapping from the MAP Client’s identifying attributes to the publisher-id. On receiving a newSession request, the MAP Server MUST select the publisher-id according to the mapping.

3.4.2 ifmap-timestamp and ifmap-timestamp-fraction

ifmap-timestamp and ifmap-timestamp-fraction together are the time, as understood by the MAP Server, of the completion of an IF-MAP publish operation.

The granularity of ifmap-timestamp is one second, and MUST be rounded down to a one-second multiple. An ifmap-timestamp without a timezone component MUST be interpreted as UTC time.

ifmap-timestamp-fraction is optional, but if present MUST be a string of one or more decimal digits which represents the decimal fraction of a second. For example, if the precision were one microsecond, ifmap-timestamp-fraction would contain six digits. ifmap-timestamp-fraction MUST NOT include a sign (+ or -) or any separation at the thousand mark. There is no support for scientific notation.

To interpret the two attributes, a MAP Client removes the timezone expression from ifmap-timestamp if present; adds the value of the ifmap-timestamp-fraction attribute to value of the ifmap-timestamp attribute; and restores the trailing timezone expression if it was originally present. For example, if the value of ifmap-timestamp is 2011-10-27T23:51:42Z and the value of ifmap-timestamp-fraction is 0.236781, then 2011-10-27T23:51:42.236781Z represents the time at completion of the publish operation.

Ifmap-timestamp-fraction was introduced in IF-MAP 2.2.

3.4.3 ifmap-cardinality

A metadata type may be either singleValue or multiValue. Thus, IF-MAP metadata takes the form of an XML element that includes either the singleValueMetadataAttributes or multiValueMetadataAttributes attribute group (but not both). The schema dictates, and the MAP Server or Client discovers, whether a metadata type is singleValue or multiValue using the ifmap-cardinality attribute.

All metadata type schemas MUST include either the singleValueMetadataAttributes attributeGroup or the multiValueMetadataAttributes attributeGroup (per the box below). Inclusion of one or the other of these attributeGroups ensures that all instances of a particular metadata type will have consistent ifmap-cardinality.

Every metadata item MUST include a valid ifmap-cardinality attribute (per the box below). (“Metadata item” means a top-level child element of a <metadata> element.) If a metadata item in a request lacks a valid ifmap-cardinality attribute, the MAP Server MUST return an InvalidMetadata errorResult (see section 3.8.1). If a metadata item in an update request specifies different if-map cardinality from an instance of the same metadata type *already associated with the identifier or link specified in the update request* (in violation of the ifmap-cardinality requirement in the preceding paragraph), the MAP Server MUST return an InvalidMetadata errorResult. The MAP Server SHOULD NOT consider cardinality of instances of the same metadata type on identifiers or links other than the one specified in the update request.

Whenever there is an errorResult, the request MUST have no other effect. For the balance of this section, we assume that there is no errorResult.

If a metadata item in an update request specifies singleValue, the MAP Server MUST replace any previous instance of that metadata type associated with the same identifier or link with the new metadata.

If a metadata item in an update request specifies multiValue, the MAP Server MUST append the metadata to the list of existing metadata on the identifier or link even if it duplicates existing metadata on that identifier or link.

```
<xsd:attributeGroup name="singleValueMetadataAttributes">
  <xsd:attributeGroup ref="metadataAttributes"/>
  <xsd:attribute name="ifmap-cardinality" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="singleValue"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>

<xsd:attributeGroup name="multiValueMetadataAttributes">
  <xsd:attributeGroup ref="metadataAttributes"/>
  <xsd:attribute name="ifmap-cardinality" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="multiValue"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>
```

Note that singleValueMetadataAttributes and multiValueMetadataAttributes both include the metadataAttributes attributeGroup.

3.4.4 Metadata Size

There are no explicit limits on the size of metadata. MAP Clients and Servers MUST support metadata at least 100000 bytes in length, and SHOULD support longer metadata. Specifically, this is a measure of the size of an individual metadata element within an IF-MAP request in wire format. In other words, the length of a metadata element is determined by looking at the bytes used to represent that element on the wire in the publish request, starting with the open bracket of the metadata element and ending with its closing bracket.

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 3.8.1). 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.

3.4.5 Operational Metadata types

Operational metadata types (see section 2.6.2) enable MAP Clients and Servers to establish a context for IF-MAP operations. They are defined in a dedicated XML schema in section 10.2.

Operational metadata are created by MAP Clients and Servers as with other metadata. The reason they are designated "operational metadata" is that they are fundamental to the operation of the MAP Server.

For general guidance on usage of metadata, see section 3.1 of the IF-MAP Metadata for Network Security specification [15].

3.4.5.1 client-time

Recommended for: device

The client-time metadata is used to detect clock skew between MAP Server and MAP Client. It is a singleValue metadata published on a device identifier that is unique and represents the MAP Client. MAP Clients SHOULD publish client-time metadata and SHOULD specify a lifetime of session when doing so.

The client-time metadata has an attribute current-timestamp, which type is xsd:dateTime and which contains the current date and time of the MAP Client (including fractional seconds, if available) at the time of the request in UTC.

The schema definition of the client-time metadata is:

```
<xsd:element name="client-time">
  <xsd:complexType>
    <xsd:attribute name="current-time" type="xsd:dateTime"
use="required"/>
    <xsd:attributeGroup
ref="ifmap:singleValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

3.4.5.2 server-capability

MUST be attached to: ifmap-server identifier

server-capability metadata is used by the MAP Server to indicate optional capabilities supported by that MAP Server. It is a singleValue metadata attached by the MAP Server to the ifmap-server identifier (see section 3.3.4.1). When creating this metadata, the MAP Server MUST use a unique publisher-id not associated with any MAP Client.

MAP Servers SHOULD create this metadata. MAP Clients SHOULD subscribe to server-capability metadata, since server-capability metadata is subject to change; for example, MAP Content Authorization can be turned on and off at run time by the MAP Server administrator, resulting in the creation, modification, or removal of server-capability metadata to indicate the presence or absence of support for MAP Content Authorization. MAP Clients MUST NOT publish this metadata type and MUST NOT delete this metadata from an ifmap-server identifier.

The server-capability metadata has one or more capability elements, of type xsd:string, indicating that the MAP Server supports a particular capability. The value of the string MUST take one of two forms:

1. "Vendor-ID:Capability": A vendor-defined capability. Vendor-ID is an SMI Private Enterprise Number [19] owned by the vendor, and Capability is the capability name.
2. "Capability": A TCG-defined capability.

The following TCG-defined capability types are supported:

- fractional-timestamps: support for fractional timestamps (see section 3.4.2)

- extended-identifier-search-termination: the ability to terminate searches on extended identifiers (see section 3.9.3.2.6)
- map-content-authorization: support for TNC MAP Content Authorization[38]
- ifmap-base-version-2.2
- ifmap-base-version-2.1
- ifmap-base-version-2.0
- ifmap-base-version-1.1
- ifmap-base-version-1.0

A MAP Server creating server-capability metadata indicating support for a particular IF-MAP base version MUST also include capability elements indicating support for all versions with which that base version is backwards compatible. Thus:

- a MAP Server creating this metadata is required to include three capability elements: one with value ifmap-base-version-2.2, one with value ifmap-base-version-2.1, and one with value ifmap-base-version-2.0
- a MAP Server supporting both IF-MAP 2.2 and 1.1 is required to include all values representing base versions from 1.0 to 2.2 inclusive.

Additional TCG-defined capabilities may be specified in a future version of IF-MAP or in a supplement to IF-MAP.

The schema definition of the server-capability metadata is:

```
<xsd:element name="server-capability">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="capability" type="xsd:string"
        minOccurs="1" maxOccurs="unbounded">
    </xsd:sequence>
    <xsd:attributeGroup
ref="ifmap:singleValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

3.5 Lifetime of Metadata

Liveness is essential for much metadata: if metadata becomes stale it can create security exposures or other bad effects. Normally, the publishing MAP Client is responsible for keeping such metadata current. But if the MAP Client's session ends, the MAP Server can automatically delete such metadata.

A publisher may specify whether metadata should be deleted at session end by attaching the XML attribute lifetime to the update request. The values are lifetime=”session” and lifetime=”forever”. “session” is the default.

```
    <xsd:attribute name="lifetime" default="session">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="session"/>
          <xsd:enumeration value="forever"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
```

If an element was published with lifetime=”session” and the MAP Server determines that the MAP Client's session has ended, either due to inactivity (see section 4.1.1) or at the MAP Client’s request, the MAP Server MUST delete the metadata. This deletion MUST be completed before the publishing MAP Client is allowed to create another session.

The lifetime attribute is meaningful only in update requests. A MAP Client SHOULD not use it in a notify request and a MAP Server MUST ignore it if it appears there.

A MAP Server MUST retain all metadata from an update request until (a) it is explicitly deleted by a MAP Client, via a delete or purgePublisher request, (b) it is deleted at session end, as discussed above, or (c) the MAP Server deletes all data that has the same ifmap-publisher-id. This may happen due to hardware failure or administrator action, for example.

For uses of this attribute, see the IF-MAP Metadata for Network Security specification [15].

3.6 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>
```

All FilterType values MUST be constructed with the IF-MAP filter syntax, which 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/2010/IFMAP-METADATA/2”, here are some example IF-MAP filters:

- meta:role
- meta:vlan[administrative-domain=”Main Campus” and name>=30 and name<=152]
- meta:role[name=”sales”] or meta:vlan[name=42]
- meta:event[confidence > 50 and significance=”critical”]
- [@ifmap-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”]

In contexts where an IF-MAP filter is used, the filter is applied separately to each individual piece of metadata. Conceptually, an ElemExpr matches a top-level node in an XML document consisting of a single piece of metadata. The result of applying a filter to a piece of metadata is either “match” or “don’t match”.

Per Section 3.9.1, the namespace prefixes in a filter are required to be declared in the XML document containing the filter, and the declaration of the namespace prefix is required to apply to the scope of the filter. Bear in mind that implementations - either client or server - are able to specify arbitrary prefixes, so it is a bad idea to assume that (for example) meta: always refers to the IF-MAP Metadata for Network Security 1.0 schema. A QName in a filter with no namespace prefix refers to the default namespace of the XML document it is contained in.

3.7 XML Schema Compliance and Validation

Every identifier (including extended identifiers) and every metadata item in an IF-MAP request MUST comply with its respective schema. A MAP Server SHOULD, if possible, reject a request that contains schema-violating extended identifiers or metadata items. It SHOULD return an InvalidSchemaVersion, InvalidIdentifierType, InvalidIdentifier, or InvalidMetadata errorResult, as appropriate. (However, it cannot detect violations when it does not know the schema.)

MAP Servers and Clients have the ability to assert that IF-MAP XML message documents adhere to some or all of their specified XML schemas. A MAP Server MAY perform XML validation on non-metadata (i.e. operations and identifiers) and on metadata. If a MAP Server validates non-metadata and a request contains invalid non-metadata XML, the MAP Server MUST respond with an errorResult indicating InvalidIdentifier, InvalidIdentifierType, or IdentifierTooLong, as applicable, or Failure if there is another error. (If more than one of these errors is present, the MAP Server may choose which error code to report.) If a MAP Server detects invalid metadata in a request, it MUST respond with an InvalidMetadata errorResult. MAP Servers and Clients

SHOULD inform each other about whether transmitted XML has been validated. A MAP Server MUST NOT assert that metadata is valid unless it has validated *all* of the metadata it returns in a particular response.

Per the requirements of section 2.6.2, a MAP Server is required to accept all well-formed metadata, including vendor-specific metadata for which the MAP has no schema. In practice, if the MAP Server has one particular vendor-specific schema but not another, the MAP Server should allow all metadata through except metadata that has been identified as invalid. A MAP Server MAY provide an option to operate in a "locked-down" mode, in which it rejects all metadata for which it does not have the schema.

Even if a MAP Server does validate XML, any MAP Clients it communicates with MAY validate XML that they receive.

In order to allow MAP Clients and Servers to communicate about whether or not XML has been validated, IF-MAP requests and responses include the validationAttributes attributeGroup. BaseOnly indicates that the XML has been validated against the IF-MAP base schema (specified in section 10.1), but not any associated metadata schema(s); MetadataOnly indicates that the XML has been validated against the metadata schema(s), but not the base schema; None indicates that no validation has been performed; and All indicates that the XML has been validated against both the operations and the relevant metadata schema(s).

```
<xsd:attributeGroup name="validationAttributes">
  <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:attributeGroup>
```

3.8 MAP Server Responses

IF-MAP involves the MAP Client sending requests to the MAP Server, and the MAP Server sending responses back to the MAP 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, or with a lower-level error response (e.g. SOAP). Responses to search and poll requests include additional results which are described in detail in sections 3.9.3.5 and 3.9.5.2.

```
<xsd:complexType name="ResponseType">
  <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:element name="newSessionResult"
      type="SessionResultType"/>
    <xsd:element name="renewSessionResult"/>
    <xsd:element name="endSessionResult"/>
  </xsd:choice>
  <xsd:attributeGroup ref="validationAttributes"/>
```

```
</xsd:complexType>
```

3.8.1 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="InvalidSchemaVersion"/>
        <xsd:enumeration value="InvalidSessionID"/>
        <xsd:enumeration value="MetadataTooLong"/>
        <xsd:enumeration value="SearchResultsTooBig"/>
        <xsd:enumeration value="PollResultsTooBig"/>
        <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**
AccessDenied	The MAP Client issued an unauthorized request. Which requests are allowed from a particular MAP Client depends on configuration settings of the MAP Server, which are beyond the scope of this document. A MAP Server MUST NOT reject a search, poll, or subscribe request with an AccessDenied error as a result of an authorization decision. Rather, search results are to be censored (see section 3.9.3.3, The Search Algorithm). A MAP Server MUST reject any other unauthorized request with AccessDenied, unless another error takes precedence. Precedence of errors is implementation-dependent.
Failure	The MAP Server was unable to successfully process a request for an unspecified reason.
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 MAP Server.
InvalidMetadata	Invalid metadata value in the request. This can happen if the specified metadata does not match the schema associated with its type, or is not well-formed.
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 MAP Server.
SearchResultsTooBig	A search or subscribe command generates too much data.
PollResultsTooBig	Updates that match the MAP Client's subscription have generated too much data
SystemError	The MAP Server was unable to successfully process a request because of a system error on the MAP Server

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 3.9.5.2).

3.8.2 Error Strings in responses

The errorString in an errorResult element contains a human readable string further describing the error that occurred. MAP Servers SHOULD include a good errorString to aid in debugging.

If the errorCode is AccessDenied and the request was publish or notify, errorString SHOULD describe one metadata item to which access was denied: it SHOULD include the item's identifier or link and metadata type. MAP Servers that support MAP Content Authorization [38] SHOULD log a security alert if a publish fails with AccessDenied because of a metadata item that would have been censored had the MAP Client done a search request.

3.8.3 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 errorCode as well as the errorString if present.

3.9 Client Requests and Server Responses

MAP Clients interact with MAP Servers by sending requests. MAP Servers interact with MAP Clients by sending responses.

All MAP Client requests except for newSession MUST include a session-id attribute, which is specified in the IF-MAP schema using the sessionAttributes attributeGroup.

```
<xsd:attributeGroup name="sessionAttributes">
```

```
  <xsd:attribute name="session-id" type="xsd:string"
    use="required"/>
</xsd:attributeGroup>
```

3.9.1 Namespace Prefixes

The namespace prefixes in a MAP Client request MUST be declared in the XML document containing the request, and the declaration of the namespace prefix MUST apply to the scope of the request. MAP Clients and MAP Servers MAY specify arbitrary prefixes for standard schemas. This document uses the prefix ifmap: to refer to the IF-MAP Binding for SOAP 2.x schema and the prefix meta: to refer to the IF-MAP Metadata for Network Security 1.0 schema:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>

   […IF-MAP operations…]

  </env:Body>
</env:Envelope>
```

However, MAP Clients MUST accept any declared prefix that refers to those schemas.

3.9.2 publish

A publish request may create, modify, or delete metadata associated with one or more identifiers or links. A publish request may also be used to tell the MAP Server to notify subscribers about the existence of transitory, non-persistent metadata. Publish requests do not create or destroy identifiers or links; requests simply attach or remove metadata to or from the identifier or link. (Per section 2.6.1, identifiers and links are never explicitly created or destroyed; all possible links and identifiers are treated as always existing, regardless of whether they are in use.) Links between identifiers have no meaning without metadata attached to them, and identifiers are not of interest unless they have metadata or meaningful links. A successful metadata publish MUST result in a publishReceived message. Otherwise, the entire publish request MUST fail without effect and the response MUST contain an errorResult element with an errorCode attribute indicating the cause of the failure.

There are three subtypes of publish: update, notify and delete. A MAP Server MUST process valid publish messages which contain any combination of the three publish subtypes.

```
<xsd:complexType name="UpdateType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="2">
      <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:element name="metadata" type="MetadataListType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="lifetime" default="session">
    <xsd:simpleType>
```

```
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="session"/>
        <xsd:enumeration value="forever"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="DeleteType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="2">
      <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:attribute name="filter" type="FilterType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="PublishRequestType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="update" type="UpdateType"/>
      <xsd:element name="notify" type="UpdateType"/>
      <xsd:element name="delete" type="DeleteType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attributeGroup ref="sessionAttributes"/>
  <xsd:attributeGroup ref="validationAttributes"/>
</xsd:complexType>
```

3.9.2.1 update

update adds or replaces metadata associated with identifiers and links. For example, a PDP publishes a list of roles to a user:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:publish session-id="111">
      <update>
        <identity name="joe" type="username"/>
        <metadata>
          <meta:role ifmap-cardinality="multiValue">
            <name>Guest</name>
          </meta:role>
          <meta:role ifmap-cardinality="multiValue">
            <name>Contractor</name>
          </meta:role>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

An update request that affects metadata with ifmap-cardinality="singleValue" MUST replace the previous value for that metadata type if it exists for the given identifier or link on the MAP Server. An update request that affects metadata with ifmap-cardinality="multiValue" MUST append the new value to a list of values for that metadata type for the given identifier or link on the MAP Server.

3.9.2.2 notify

notify tells the MAP Server to notify subscribers with metadata that does not persist in the database. The format of a notify operation is the same as the format of an update operation. One common use of notify is with event metadata, to notify other MAP Clients about events without storing events in the MAP Server. Per section 3.5, the lifetime attribute is not appropriate for metadata published with notify.

Metadata published with notify MUST be queued for delivery to all MAP Clients with subscriptions whose searches match the published data at the time the data is published. Metadata published with notify MUST be delivered to MAP Clients inside notifyResult elements within pollResult elements (see section 3.9.5.2). Metadata published with notify MUST NOT be delivered to MAP Clients in response to a search request. See section 5 of [29] for recommendations on backwards-compatible handling of metadata published with notify between IF-MAP 2.x and IF-MAP 1.x implementations.

For example, a Sensor publishes event metadata using notify:

```
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:publish session-id="111">
      <notify>
        <ip-address value="192.0.2.11" type="IPv4"/>
        <metadata>
          <meta:event ifmap-cardinality="multiValue">
            <name>attack</name>
            <discovered-time>2009-05-30T13:10:11</discovered-
time>
            <discoverer-id>1273</discoverer-id>
            <magnitude>45</magnitude>
            <confidence>100</confidence>
            <significance>important</significance>
            <type>worm infection</type>
          </meta:event>
        </metadata>
      </notify>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

3.9.2.3 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.

If a delete request has no filter, a MAP Server MUST delete all metadata associated with the identifier or link. If a delete request has a filter that is the empty string, the MAP Server MUST delete nothing.

For example, when an IF-MAP enabled DHCP server revokes the lease on an IP address it deletes the associated metadata from the link between the ip-address identifier and the mac-address identifier.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:publish session-id="111">
      <delete
        filter='meta:ip-mac[@ifmap-publisher-id="222"]'>
        <ip-address value="192.0.2.11"/>
        <mac-address value="00:11:22:33:44:55"/>
      </delete>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

3.9.2.4 Multiple elements in a single publish request

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 MAP Client.

A MAP Client MUST combine multiple update and/or delete elements into a single publish request when the operations performed by those elements are atomically related. A MAP Client MUST refrain from combining these elements into a single publish request for any reason other than an achieving atomic database semantics; for example, but not limited to, any attempts to increase efficiency of communication with the MAP Server.

A MAP Client SHOULD refrain from publishing very frequent updates to the same metadata. If a MAP Client is observing rapidly changing behavior in the network that needs to be reflected in MAP, the MAP Client SHOULD throttle its publish requests for that metadata to a reasonable rate such as once per second.

See section 3.12 for further discussion of the atomicity of multiple operations in a single publish request.

3.9.3 search

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

3.9.3.1 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 links with specified metadata types attached, to a specified maximum result depth, to a specified maximum result size, and stopping at specified terminal identifier types.

Figure 3 illustrates an example MAP graph containing metadata from the IF-MAP Metadata for Network Security specification [15]:

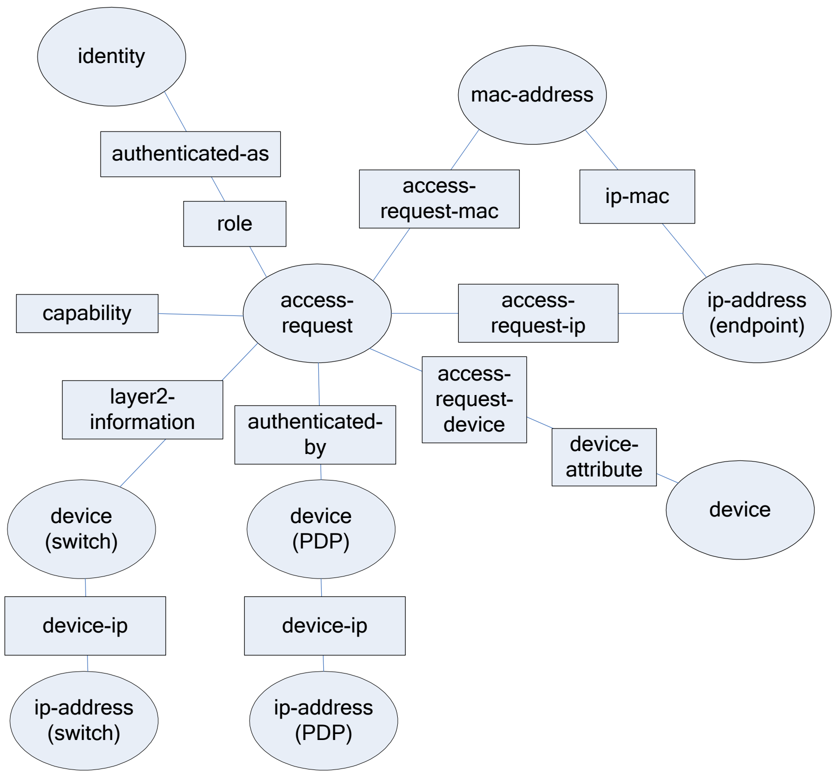


Figure 3: Example MAP Graph

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

3.9.3.2 Search Parameters

The following six values parameterize searches:

3.9.3.2.1 identifier

identifier specifies the starting place of the search.

3.9.3.2.2 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. If there is no match-links attribute, all links that have metadata attached are visited (subject to depth and other search criteria). If an empty match-links attribute is specified, no links are visited in the search.

3.9.3.2.3 max-depth

max-depth specifies the maximum distance of any included identifiers. Distance is measured by number of links away from the starting identifier. If there is no max-depth attribute, or an empty max-depth attribute is specified, search depth is interpreted as 0 (zero).

3.9.3.2.4 max-size

max-size specifies the maximum size, in bytes, of the results. Specifically, this is a measure of the size of the searchResult element or pollResult element in its entirety, in wire format. This excludes the SOAP envelope and result wrapper; effectively, it includes everything under the ifmap:response element.

3.9.3.2.5 result-filter

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

3.9.3.2.6 terminal-identifier-type

The terminal-identifier-type specifies identifier types that when encountered terminate a search. This is used when the depth of a search might otherwise cause the search to gather metadata associated with more links or identifiers than desired. Metadata associated with a terminal identifier type is included in search results if it matches result-filter. Syntactically, terminal-identifier-type's value is a comma-separated list of one or more of the following strings:

- access-request
- identity
- ip-address
- mac-address
- device
- identity:x, where x is one of the identity types listed in the table in section 3.3.2.3
- identity:other:y, where y is an other-type-definition value as defined in section 3.3.2.3
- identity:other:extended, to terminate on any extended identifier type
- identity:nonextended
- ***NAMESPACE#TYPE***, for an extended identifier type; where ***NAMESPACE*** is the URI of an XML schema, and ***TYPE*** is a type name defined by that XML schema, e.g. "http://www.example.com/extended-identifiers#network"

If there is no terminal-identifier-type attribute, or an empty terminal-identifier-type is specified, no identifier is considered terminal. If a MAP Server receives a terminal-identifier-type that doesn't match one of the indicated strings above, it MUST return an InvalidIdentifierType errorResult.

3.9.3.3 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 on the shortest path that can be traversed to reach the current identifier, starting with the starting identifier.
- A sub-result contains partial results.

Algorithm:

1. Run recursive search subroutine below with the "current identifier" set to the identifier given in the search request, and "current depth" equal to 0.

2. If there is a “result-filter,” apply it to the results, deleting any unmatched metadata from the results.
3. Give the results to the MAP Client or return an error (and no results) indicating max-size was reached.

Recursive Search Subroutine:

1. Start at the current identifier, with empty current results, at the current depth.
2. Add any metadata on the current identifier to current results.
3. If the current identifier’s type is contained within “terminal-identifier-type”, return current results.
4. If the current depth is greater than or equal to “max-depth”, return current results.
5. For all links associated with the current identifier, add all metadata on the links that match the “match-links” filter to current results. If there is no “match-links” filter, add to current results all metadata on all links associated with the current identifier. Remember the set of links added in this step for use in step #6 below.
6. For all identifiers associated with the current identifier via links remembered in step 5, start the recursive search subroutine (step 1) with a depth equal to the current depth plus one. Add the return value of each recursive search subroutine to current results.
7. Return current results.

If the MAP contains any metadata that the MAP Client is not authorized to read, the MAP Server MUST process the search as if that metadata were not present. That is, the MAP Server must behave as if it took an atomic snapshot of the MAP store, censored all metadata that the MAP Client was not authorized to read, and applied the algorithm to the censored MAP store.

If a MAP Client does not specify max-depth, the MAP Server MUST process the search with a max-depth of zero.

3.9.3.4 Search Requests

A search request specifies the parameters of the requested search:

```
<xsd:complexType name="SearchType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <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:attribute name="match-links" type="FilterType"/>
  <xsd:attribute name="max-depth" type="xsd:unsignedInt"/>
  <xsd:attribute name="max-size" type="xsd:unsignedInt"/>
  <xsd:attribute name="result-filter" type="FilterType"/>
  <xsd:attribute name="terminal-identifier-type"
    type="xsd:string"/>
</xsd:complexType>
```

SearchType is used by SearchRequestType (as well as SubscribeRequestType - see section 3.9.4.1).

```
<xsd:complexType name="SearchRequestType">
  <xsd:complexContent>
```

```
    <xsd:extension base="SearchType">
      <xsd:attributeGroup ref="sessionAttributes"/>
      <xsd:attributeGroup ref="validationAttributes"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

For example, when a Flow Controller detects a new flow from a previously unseen IP address, it could search 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"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:search session-id="123"
      match-links="meta:access-request-ip or meta:ip-mac or
    meta:access-request-mac"
      max-depth="3" result-filter="meta:capability or
    meta:device-attribute or meta:roles"
      terminal-identifier-type="identity,device">
      <ip-address value="192.0.2.11" type="IPv4"/>
    </ifmap:search>
  </env:Body>
</end:Envelope>
```

3.9.3.5 Search Responses

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 a resultItem element for each node and each edge in the metadata graph that matches the query. The XML structure itself does not directly reflect the structure of the metadata graph, although the subgraph can be reconstructed from the result data. 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.

All identifiers for a given identifier type are always valid identifier parameters for a search, regardless of whether they have any metadata attached or meaningful links (i.e., links with metadata attached to them. In the case where a search's starting identifier has no metadata or meaningful links, the MAP Server MUST return the identifier with no metadata or links attached to it.

A searchResult consists of one or more resultItems, since all searches will, at a minimum, return an item for the starting identifier (possibly without any metadata). Each resultItem contains a) either an identifier, or a link (denoted by two identifiers), and b) zero or more items of metadata that were found attached to the identifier or link.

Each search result may contain a name attribute. The name attribute is used for poll results (see section 3.9.5.2).

```
<xsd:complexType name="ResultItemType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="2">
      <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:element name="metadata" type="MetadataListType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SearchResultType">
  <xsd:sequence>
    <xsd:element name="resultItem" type="ResultItemType"
      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"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:response>
      <searchResult>
        <resultItem>
          <ip-address value="192.0.2.11" type="IPv4"/>
        </resultItem>
        <resultItem>
          <access-request name="123"/>
          <metadata>
            <meta:capability ifmap-cardinality="multiValue">
              <name>finance-server-access</name>
            </meta:capability>
          </metadata>
        </resultItem>>
        <resultItem>
          <ip-address value="192.0.2.11" type="IPv4"/>
          <access-request name="123"/>
        </resultItem>
      </searchResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

If a MAP Client looks for identity identifiers in search results, it should not be confused by extended identifiers. Superficially, an extended identifier is an identity identifier, but it does not (in general) denote an identity.

MAP Servers MUST support result sizes up to and including 100KB[3]. Some use cases may demand larger size constraints. 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 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
Unspecified	ANY (i.e. >= 100KB)	>100KB	Error - result too large
Specified (i.e. > 0)	< MAP Client's requested "max-size"	<= MAP Server's maximum supported result size	Result
Specified (i.e. > 0)	< MAP Client's requested "max-size"	> MAP Server's maximum supported result size	Error – result too large
Specified (i.e. > 0)	>= MAP Client's requested "max-size"	<= MAP Client's requested "max-size"	Result
Specified (i.e. > 0)	>= MAP Client's requested "max-size"	**>** MAP Client's requested "max-size"	Error – result too large

3.9.4 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 SearchType items. The subscription list is consulted by the MAP Server when determining what MAP 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 update or delete elements used to manage the subscription list.

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" use="required"/>
```

[3]A limit is specified so that MAP Server implementations may protect themselves from resource exhaustion due to unreasonable searches.

```
</xsd:complexType>

<xsd:complexType name="SubscribeRequestType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="update">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="SearchType">
              <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:attributeGroup ref="sessionAttributes"/>
  <xsd:attributeGroup ref="validationAttributes"/>
</xsd:complexType>
```

3.9.4.1 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 replace an existing search request, the MAP Client specifies the name attribute of an existing search. The value of the name attribute MUST be a string between 1 and 20 characters in length. For instance, a MAP Client specifying a unique integer value as the name of each search might use “0” for the first search in an IF-MAP session and increment by one for each successive search it makes in the same session. The MAP Client MUST specify the name attribute in every update element of a subscribeRequest.

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"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:subscribe session-id="111">
      <update name="35" result-filter="meta:role"
        match-links="meta:access-request-ip or
      meta:access-request-mac or meta:ip-mac or
      meta:access-request-identity"
        max-depth="3">
          <ip-address value="192.0.2.11" type="IPv4"/>
      </update>
    </ifmap:subscribe>
  </env:Body>
</env:Envelope>
```

3.9.4.2 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"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:subscribe session-id="111">
      <delete name="35"/>
    </ifmap:subscribe>
  </env:Body>
</env:Envelope>
```

3.9.5 poll

Polling is a way for a MAP Client to asynchronously receive updates from a MAP Server in real-time when a portion of the MAP graph changes. A MAP Client sends a poll request to a MAP Server based on the MAP Client's subscription; the MAP Server then waits until it detects a change in a given portion of the MAP graph, at which point it sends a poll response. While the MAP Server is technically responding to the poll request, in effect the MAP Server is spontaneously pushing data to the MAP Client upon detecting a change. (The poll request message is an expedient required by the underlying SOAP transport mechanism, since the MAP Server cannot initiate a new connection to the MAP Client.)

3.9.5.1 Poll Requests

A poll request is sent by a MAP Client to a MAP Server to request notification of metadata changes based on the MAP Client’s subscription. Metadata may be changed by publish requests (section 3.9.2), purgePublisher requests (section 3.9.6), or automatic deletion of metadata by the MAP Server (section 3.5).

```
<xsd:complexType name="PollRequestType">
  <xsd:attributeGroup ref="validationAttributes"/>
  <xsd:attributeGroup ref="sessionAttributes"/>
</xsd:complexType>
```

A MAP Client that issues subscribe requests MUST create an ARC and issue poll requests. After receiving each pollResult response from a MAP Server, a MAP Client SHOULD immediately send another poll request in order to minimize the amount of poll result data the MAP Server accumulates on the MAP Client’s behalf.

3.9.5.2 Poll Responses

A MAP Server MUST respond to a poll request with a pollResult message, an endSessionResult message, or an errorResult message (see section 3.8.1).

In response to a poll request, the MAP Server checks to see if any search results are available for subscriptions the MAP Client has made. The first time a pollResult contains search results for a new subscription, the search results MUST consist of the complete set of identifiers, links, and metadata for the subscription's search criteria as specified in section 3.9.3. If any results are available, the MAP Server MUST send a pollResult response containing the complete search results in a searchResult element. If no results are available, including the case where there are no active subscriptions for this MAP Client, the MAP Server MUST NOT send a response to that poll at that time. At some future time when metadata changes occur that match MAP Client subscriptions, the MAP Server MUST send a pollResult response containing search results.

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

If poll results in aggregate are too big (but no subscription result is too big individually), or any other error occurs that is not caused by one individual subscription, the MAP Server MUST respond with an errorResult message and MUST invalidate all of the MAP Client's subscriptions. After receiving an errorResult message in response to a poll, a MAP Client will need to issue new subscribe requests in order to receive more poll results.

If no such general error occurs, but a particular subscription's results are too big, or any other error occurs that affects only a particular subscription, the MAP Server MUST respond with a pollResult message containing an errorResult element, in which the name attribute of the errorResult element is set to the name of the subscription that caused the error, and the MAP Server MUST invalidate and remove only the subscription that caused the error, and return other subscription results as usual.

Size limits on results for an individual subscription are governed by the same limits as search requests. pollResults themselves have a much larger size limit to accommodate multiple subscription results. A MAP Server MUST buffer at least 5,000,000 bytes of poll results for each MAP Client (see section 4.3). If the size of a MAP Client's poll results exceeds the MAP Server's poll buffer limit or the client-specified limit, the MAP Server MUST indicate this to the MAP Client by responding to a poll request with an errorResult message (not a pollResult message containing an errorResult element) containing an errorCode of PollResultsTooBig. In this situation, the MAP Server MAY opt to first return a pollResult containing a subset of the accumulated result elements (e.g. notifyResult, searchResult, updateResult, etc.) before responding to the next poll request with an errorResult response. (Note that while partial SearchResultTypes are not permitted, per section 3.9.3.5, partial PollResultTypes are permitted and may be better than no results at all.)

If poll results approach or exceed the size limit, the MAP Server SHOULD caution its administrator; the mechanism of doing so is implementation dependent.

3.9.5.3 Delta pollResults

After the first searchResult for a new subscription (containing the complete set of search results), subsequent pollResults (known as "delta pollResults") MUST contain updates in updateResult elements as metadata is added and deletes in deleteResult elements as metadata is removed. The updateResult and deleteResult elements returned by the MAP Server MUST reflect ALL of the updates which have occurred since the last poll which affect the MAP Client's subscriptions. The MAP Server MUST NOT compress search results by removing updateResult and deleteResult elements which cancel each other out. The MAP Server MAY return the accumulated result elements in multiple pollResult messages. The MAP Server accomplishes this by returning a subset of the accumulated results to the MAP Client in response to a single poll request. In response to the next poll request the MAP Server returns another subset, continuing in this fashion until all pollResults have been sent. Note that atomic changes require special handling; see section 3.12 for details.

The MAP Server SHOULD NOT send delta pollResults with no metadata. For example, if a MAP Client publishes two identical updates to the same single-valued metadata within one second, a MAP Server that does not support ifmap-timestamp-fraction should not send delta pollResults for the second update, since none of the metadata has changed. Or, a MAP Client may request a subscription with result filters that strip out all metadata on the results; the initial pollResult will contain identifier and link results with no metadata associated with them, and the MAP Server should not send delta pollResults for subsequent updates that change metadata on those identifiers and links.

3.9.5.4 Changes to Subscriptions

A metadata change may cause a subscription to traverse a new link, or to stop traversing a link. When a subscription result changes after an initial searchResult message has been sent, the MAP Server informs the MAP Client about the metadata that was added or removed to the subscription result, using updateResult and deleteResult elements. For each such MAP Client that has an active poll request, the MAP Server SHOULD return the accumulated results to the MAP Client as soon as possible. For a MAP Client that does not have an active poll request, the MAP Server MUST retain metadata changes until the MAP Client issues a poll request or the MAP Server determines that the MAP Client's session has ended.

The MAP Server MUST return separate searchResult, updateResult, deleteResult or errorResult elements for each subscription that has changes. Each searchResult, updateResult, deleteResult and errorResult element in a pollResult response MUST include a name attribute which identifies the subscription corresponding to the result.

After the first pollResult, the MAP Server MUST NOT return searchResult, updateResult or deleteResult elements for a subscription that has no changes associated with it.

3.9.5.5 pollResults with notify

pollResult is also used to send metadata to MAP Clients when a MAP Client publishes using the notify publish subtype. When a MAP Client publishes using notify, the MAP Server MUST add the published metadata to the poll results for each subscriber whose subscription matches the published metadata. For each such MAP Client that has an active poll request, the MAP Server SHOULD return the accumulated search results to the MAP Client as soon as possible. For a MAP Client that does not have an active poll request, the MAP Server MUST retain metadata published using notify until the MAP Client issues a poll request or the MAP Server determines that the MAP Client’s session has ended. Metadata published using notify MUST be sent to a subscribing MAP Client in a notifyResult element.

notifyResult is only used to return metadata that was published using notify, even if other metadata matches the result-filter in the subscription. A MAP Client receiving a notifyResult MUST NOT assume that metadata missing from a notifyResult has been deleted.

3.9.5.6 Processing pollResults

When processing pollResults, a MAP Client SHOULD consider a result as the combination of metadata, identifier, result type (search, notify, update, delete, or error), and subscription name, but not the manner in which the results are encoded in the XML. When generating delta pollResults, a MAP Server MUST ensure that metadata elements are delivered in the same order they are received, and a MAP Client SHOULD maintain the order of metadata elements as delivered in the XML when processing results.

Because the MAP Server might choose to group the results differently, it SHOULD NOT be assumed that there is a 1:1 relationship between the number of notify requests and actual notifyResult responses. For example, the following two results are functionally equal:

```
<ifmap:response>
      <pollResult>
        <notifyResult name="sub1">
          <resultItem>
            <identity type="username" name="id1"></identity>
            <metadata>
              <meta:event ifmap-cardinality="multiValue"
                ifmap-publisher-id="x"
                ifmap-timestamp="xxx"
                ifmap-timestamp-fraction="yyy"
              >...event1...</meta:event>
            </metadata>
          </resultItem>
        </notifyResult>
        <notifyResult name="sub1">
          <resultItem>
            <identity type="username" name="id1"></identity>
            <metadata>
              <meta:event ifmap-cardinality="multiValue"
                ifmap-publisher-id="x"
                ifmap-timestamp="xxx"
                ifmap-timestamp-fraction="yyy"
              >...event2...</meta:event>
            </metadata>
          </resultItem>
        </notifyResult>
      </pollResult>
   </ifmap:response>
```

```
<ifmap:response>
      <pollResult>
        <notifyResult name="sub1">
          <resultItem>
            <identity type="username" name="id1"></identity>
            <metadata>
              <meta:event ifmap-cardinality="multiValue"
                ifmap-publisher-id="x"
                ifmap-timestamp="xxx"
                ifmap-timestamp-fraction="yyy"
              >...event1...</meta:event>
              <meta:event ifmap-cardinality="multiValue"
                ifmap-publisher-id="x"
                ifmap-timestamp="xxx"
                ifmap-timestamp-fraction="yyy"
              >...event2...</meta:event>
            </metadata>
          </resultItem>
        </notifyResult>
      </pollResult>
   </ifmap:response>
```

3.9.5.7 Subscription and Polling

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

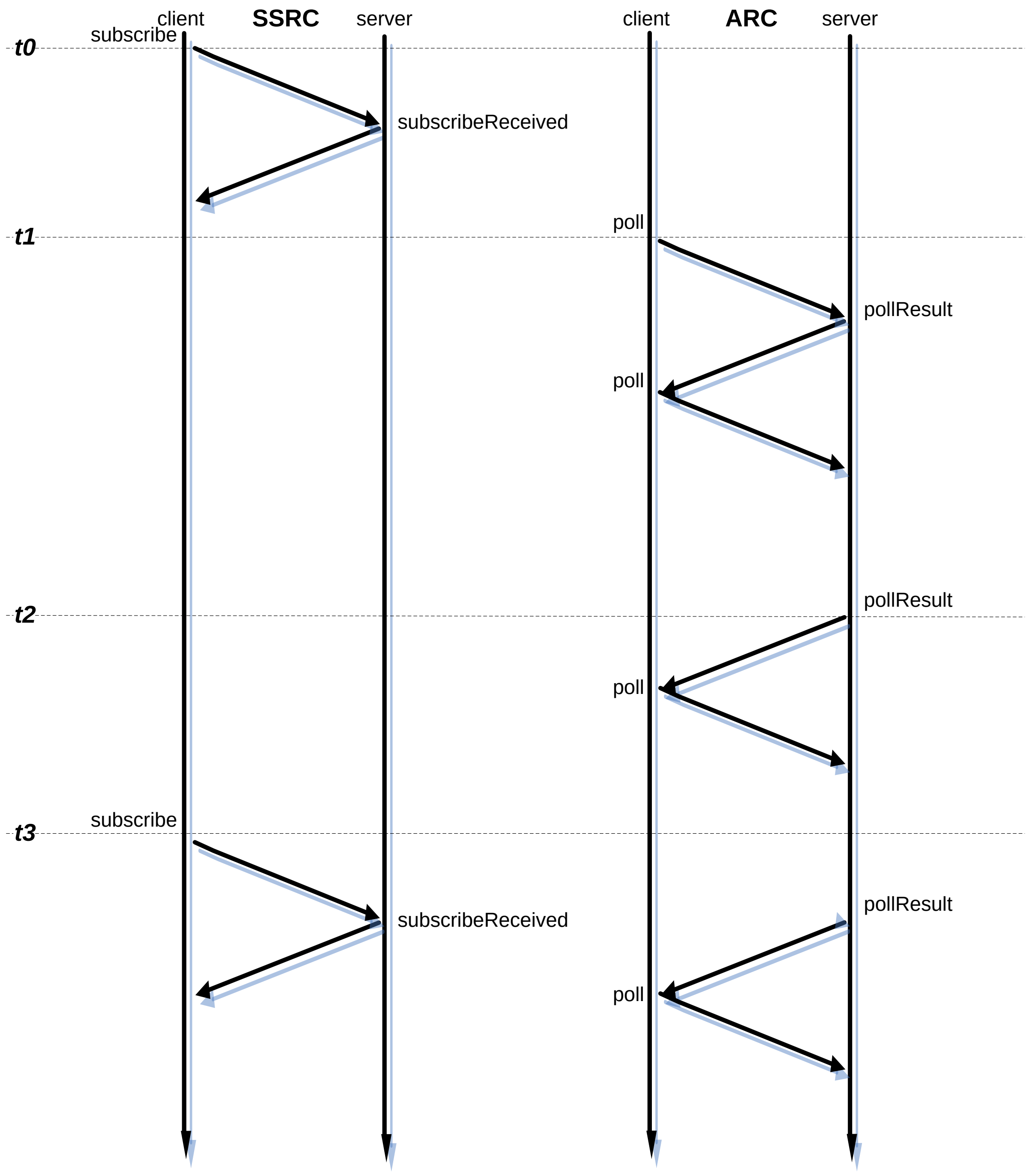


Figure 4

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

At t3, the MAP Client issues a subscribe request on the SSRC channel that alters the search results for the subscription. On the SSRC channel, the MAP Server sends a subscribeReceived response. At the same time, the MAP Server sends a pollResult response on the ARC channel containing the new search results; since altered subscriptions are effectively new subscriptions, the first poll response would contain a searchResult rather than a delta pollResult. When the MAP Client receives the pollResult response, it sends another poll request so the MAP Client can be notified of further changes.

3.9.5.8 Subscription-related Race Condition

A race condition is inherent in updating a subscribed search. There is no mechanism to synchronize management of a subscribed search with delivery of results for that subscription. A MAP Client may receive results that pertain to the subscription as it existed prior to an update after having received a subscribedReceived confirmation from the MAP Server for the update in question. This can happen because poll results are delivered to the MAP Client on a different channel than the one used for subscription management requests.

To avoid this race condition, a MAP Client may delete the old subscription and create a new subscription with a new name. That way, the MAP Client can distinguish between results that pertain to the old subscription and results that pertain to the new subscription.

A MAP Client that issues subscribe requests MUST create an ARC and issue poll requests. After receiving each pollResult response from a MAP Server, a MAP Client SHOULD immediately send another poll request in order to minimize the amount of poll result data the MAP Server accumulates on the MAP Client’s behalf.

3.9.5.9 Poll vs. Search

Polling shares many similarities with searches in IF-MAP, but there are some important differences.

Both use the same search parameters and algorithm (as described in section 3.9.3) to identify relevant portions of the MAP graph. However, a search sends these parameters in a search message (SearchRequestType), while a subscription request (SubscriptionRequestType) is used to convey search parameters to the MAP Server where they are used for subsequent poll requests.

A search result contains only the information associated with a single search request. By contrast, a poll result may contain information that responds to multiple subscriptions. In the latter case, each individual poll result item indicates which subscription it is responding to by using a "name" attribute.

Both a search result and the result for an individual subscription have a maximum size that is the lesser of the size specified by the MAP Client in the search parameters and the MAP Server's maximum supported result size (or 100KB if unspecified). However, as noted above, a poll result may contain information associated with multiple subscriptions. For this reason, a poll response has a separate size limit equal to the lesser of the MAP Server's maximum supported poll response size (or 5,000,000 bytes if unspecified) or the size specified by the MAP Client in the newSession message it used to initiate the session with the MAP Server. The MAP Server has different error responses for when the size of an individual subscription response exceeds its permitted limits and for when the size of the aggregate poll response message exceeds its permitted limits, as discussed in section 3.9.5.2.

Search results indicate a snapshot of a region of the MAP graph at a given point in time. Poll responses provide an initial snapshot, but thereafter only report differences observed within that reported region of the MAP graph.

Search results are snapshots, while poll results represent a sequence of changes over time. While MAP is not allowed to return a partial search result, it is possible to return a partial sequence of changes to a poll request, because each result that the MAP Server can fit in the partial poll response is correct, complete, and usable. Specifically, if a particular reported event cannot be conveyed (e.g., it exceeds the size limits of the subscription response), the Server MAY still report all preceding events before indicating the error condition.

3.9.6 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 MAP 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="ifmap-publisher-id" type="xsd:string"/>
  <xsd:attributeGroup ref="sessionAttributes"/>
</xsd:complexType>
```

A MAP Server MUST either respond to a purgePublisher request with a purgePublisherReceived message and remove all data with that publisher ID, or return an errorResult message and not delete any data.

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.

3.11 Vendor-specific Metadata

Per the requirements of section 2.6.2, MAP Clients are prohibited from extending the standard metadata schema 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.

ifmap-publisher-id and metadata name attributes corresponding to the standard metadata name attributes 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 ifmap-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.

3.12 Atomicity

In this section, we consider three cases (publish, purgePublisher, and automatic metadata deletion), each of which constitutes a “change”:

- Publish requests that contain multiple update, notify and/or delete requests have special behavior. The MAP Server MUST ensure that they appear atomic.
- The MAP Server MUST ensure that purgePublisher requests appear atomic.
- When (per section **Error! Reference source not found.**) the MAP Server deletes all metadata that came from a particular client and had lifetime=”session”, the MAP Server MUST ensure that this bulk deletion appears atomic.

Where “atomic” means:

- No interim results are visible to MAP Clients. In particular, a pollResult or searchResult will reflect the data in a state either before or after the entire change. A MAP Client would never be able to tell the order of the operations within a multi-part publish request.
- Changes are all-or-nothing; this holds even in case of power loss or other failure. Either a change fails and leaves the data unchanged, or the change succeeds in its entirety..

This requirement for special handling of atomic changes supersedes the requirement in section 3.9.5.3 that “[t]he updateResult and deleteResult elements returned by the MAP Server MUST reflect ALL of the updates which have occurred since the last poll which affect the MAP Client’s subscriptions.”

Per section 3.9.5, if a MAP Client sends a publish request containing a delete operation, followed by another publish request containing an update operation, polling MAP Clients should see the results of both operations.

However, if a MAP Client sends a single publish request containing both a delete operation and an update operation, polling MAP Clients should see only the end result due to the atomicity of the publish request; the MAP Server MUST send a pollResult containing the end state after the delete and update operations in the single publish request.

4 SOAP Binding and Session Management

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 MAP Client either synchronously or asynchronously.

A MAP Client initiates a connection using an https URI. A MAP Client MUST be capable of communicating with a MAP Server using any valid URI as specified in [11] and [17]. In particular, a MAP Client MUST be capable of communicating with a MAP Server using a tcp port specified in an https URI. A MAP Server MUST respond to a URI path of /ifmap; MAP Servers MAY support other paths. All paths on a single MAP Server MUST have exactly equivalent functionality.

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); a MAP Client MUST open exactly one SSRC and no more than one ARC. Each session is uniquely identified by a session id. A session lasts as long as the MAP Client actively uses it and doesn't attempt to establish a new session. To keep a session alive, a MAP Client is required to send a renewSession request to the MAP Server (see section 4.4). This ensures that if a MAP Client opens a session and then loses communication with the MAP Server, both the MAP Client and MAP Server will be aware of the loss, which would not necessarily happen if the MAP Client maintained only an ARC polling.

If a MAP Client has an existing SSRC connection and sends an SSRC request over a different connection, the MAP Server MUST associate the new connection with the SSRC and MUST close both the SSRC and the ARC for the old connection (if possible) or respond to new requests on the old connection with an AccessDenied errorResult. Since the MAP Client's old session has been terminated, the MAP server MUST delete all metadata published by that MAP Client with lifetime of session (see section 3.5).

A MAP Server MUST keep a session alive if the SSRC is active (TCP connection is open or MAP Client is sending renewSession requests). A MAP Server SHOULD end a session after a period of MAP Client inactivity which MUST be no shorter than three minutes. (Thus, if a MAP Client drops off due to a network connectivity issue, the MAP Client has at least three minutes to re-establish a connection to save all the session metadata it has published in the MAP.) To detect inactivity on the TCP connection associated with an SSRC, a MAP Server SHOULD send Keep-Alive probes (see IETC RFC 1122 [34] and section 4.2.3.6, activated by the socket option SO_KEEPALIVE). The length of the period of MAP Client inactivity that results in session termination MAY be configurable on a MAP Server.

A MAP Client may open a TCP connection, publish metadata, and close the connection rather than maintaining a TCP connection. In this case, the MAP Client SHOULD specify a metadata lifetime of "forever" on the metadata it publishes (see section **Error! Reference source not found.**); otherwise the metadata will be purged when the MAP Server ends the session (possibly as few as three minutes later). If the MAP Client opens a new TCP connection after the MAP Server has ended the previous session and attempts to publish metadata using the previous session-id (see section 4.3), the MAP Server will return an "Invalid Session ID" error. In this case, the MAP Client MUST respond by creating a new session.

4.1.1.1 Synchronous Send-Receive Channel (SSRC)

The SSRC is a SOAP/HTTPS channel over which the following IF-MAP messages MAY be communicated: newSession, renewSession, endSession, publish, search, subscribe, purgePublisher, response. The SSRC MUST NOT be used for poll requests. 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.

Valid request -response pairs are:

- newSession -> newSessionResult | errorResult
- publish -> publishReceived | errorResult
- subscribe -> subscribeReceived | errorResult
- search -> searchResult | errorResult
- purgePublisher -> purgePublisherReceived | errorResult
- renewSession -> renewSessionResult | errorResult
- endSession -> endSessionResult | errorResult

A MAP Client SHOULD timeout if the MAP Server does not respond in a timely fashion to a request.

If after sending a publish request the MAP Client does not get a response from the MAP Server for any reason (timeout, connection closed, client shutting down, etc) the MAP Client should generally reconnect when it is able to and should generally repeat the publish. However, repeating may result in the publish happening twice. If the publish includes an update of a multi-valued piece of metadata, this would result in a duplicate value. A MAP Client SHOULD prevent this duplication from happening. To prevent it, when republishing the MAP Client may prepend a delete request that would delete the identical piece of metadata with the identical ifmap-publisher-id. For example:

```
<ifmap:publish session-id="222">
  <delete
    filter="meta:capability[@ifmap-publisher-id=’111’]">
    <access-request name="111:42"/>
  </delete>
  <update>
    <access-request name="111:42"/>
    <metadata>
      <meta:capability ifmap-cardinality="multiValue">
        <name>unrestricted-access</name>
      </meta:capability>
    </metadata>
  </update>
</ifmap:publish>
```

4.1.1.2 Asynchronous Receive Channel (ARC)

A MAP Client MAY establish a secondary channel to asynchronously receive the results of its current subscription, by using the poll request. The IF-MAP messages for poll requests and poll responses MAY be communicated over the ARC (see sections 3.9.5.1 and 3.9.5.2). Other messages (including searchResult) MUST NOT be communicated over the ARC. The response element sent in response to a poll request MUST contain either an appropriate pollResult element corresponding to the MAP Client’s subscription, an errorResult element indicating the cause of an error, or an endSessionResult indicating that the session ended.

A MAP Client MUST have a valid session with a MAP Server before opening an ARC. A MAP Client MUST NOT send any IF-MAP request other than “poll” on an ARC. A “poll” request MUST contain a session-id attribute referring to a valid session. A MAP Client MUST have no more than one ARC active at a time, meaning that a MAP Client MUST NOT have more than one outstanding “poll” request for a particular session.

4.2 SOAP Transport

All implementations of the IF-MAP protocol MUST support Simple Object Access Protocol (SOAP) v. 1.2 as defined in [7]. All IF-MAP messages are encapsulated inside SOAP bodies which in turn are inside SOAP envelopes. HTTP compression for responses, if used, MUST be negotiated according to HTTP 1.1 [17]. A MAP Client MAY compress requests according to HTTP 1.1, where the compression algorithm used is indicated using the Content-Encoding HTTP header. MAP Servers MUST accept requests that have been compressed using gzip and identity transformations.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:publish session-id="128738734">
      ...
    </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 MAP Server to share state across multiple incoming TCP connections from a single MAP 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 “newSession” 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/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:newSession max-poll-result-size="10000000"/>
  </env:Body>
</env:Envelope>
```

max-poll-result-size is an optional attribute that indicates to the MAP Server the amount of buffer space the MAP Client would like to have allocated to hold poll results. A MAP Server MUST support buffer sizes of at least 5,000,000 bytes, and MAY support larger sizes.

If the MAP Server can create a session for the MAP Client, the MAP Server’s response MUST contain a “newSessionResult” element that has a “session-id” attribute that specifies the MAP Client’s session-id along with an “ifmap-publisher-id” attribute that the MAP Client can use to recognize metadata that it published by examining operational attributes. If the MAP Client included a max-poll-result-size attribute in its newSession request, the MAP Server MUST include a max-poll-result-size in its response indicating the actual amount of buffer space available for poll results for 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/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:response>
      <newSessionResult session-id="222" ifmap-publisher-id="111"
        max-poll-result-size="7500000"/>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

If a MAP Client sends more than one SOAP request containing a “newSession” element in the SOAP body, the MAP Server MUST respond by ending the previous session and starting a new session. The MAP Server’s response MUST contain a “newSessionResult” element, and any state associated with the old session (including metadata published with a lifetime of "session") MUST be discarded. The new session MAY use the same session-id or allocate a new one. If an ARC is associated with the old session, the MAP Server MUST send an endSessionResult on the ARC.

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 a “poll” element in the SOAP body.

Each subsequent request (including a “poll” request on an ARC) MUST contain a “session-id” attribute in the top level element of the SOAP body, 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/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:search session-id="222">
     …
    </ifmap:search>
  </env:Body>
</env:Envelope>
```

If the MAP Client specifies an invalid session-id, the MAP Server MUST indicate an InvalidSessionID errorResult in its response. A MAP Server MUST NOT accept a request from a MAP Client unless the session-id is valid for that MAP Client. A MAP Server MUST NOT permit one MAP Client to use a session that is assigned to another MAP Client.

To explicitly terminate a session with a MAP Server, a MAP Client MAY send an endSession request:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
```

```
    <ifmap:endSession session-id="222"/>
  </env:Body>
</env:Envelope>
```

If the session is valid, The MAP Server MUST respond with an endSessionResult response.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:response>
      <endSessionResult/>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

When a session ends for any reason, and there is an outstanding poll request on the ARC, the MAP Server MUST send an endSessionResult to the MAP Client on the ARC.

If a MAP Server receives a message containing a SOAP body containing a poll element that specifies a session which already has an ARC with an outstanding poll request, the MAP Server MUST:

- end the session
- respond to the poll request on the older ARC with an endSessionResult
- respond to the poll request on the newer ARC with an errorResult response with an errorCode of InvalidSessionID

4.4 Session Renewal

In order to keep an IF-MAP session from timing out, a MAP Client MUST send periodic renewSession requests to the MAP Server. The MAP Client should send renewSession requests somewhat more frequently than the session timeout on the MAP Server. Since the minimum session timeout is 180 seconds, a MAP Client MUST support sending renewSession requests every 150 seconds by default and MAY also be configurable otherwise..

Like other IF-MAP requests, renewSession request MUST specify a valid session-id.

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:renewSession session-id="222"/>
  </env:Body>
</env:Envelope>
```

If the session-id is valid, the MAP Server MUST respond with a renewSessionResult element. Otherwise, the MAP Server MUST respond with an errorResult element, specifying an InvalidSessionID errorCode.

```
<?xml version="1.0"?>
<env:Envelope
```

```
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP-
METADATA/2">
  <env:Body>
    <ifmap:response>
      <renewSessionResult/>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

4.5 ARC Error Handling

If a MAP Server or MAP Client detects an error at the TCP or SSL layer of an ARC, the MAP Server or Client MUST end the session. The reason is that after a transport error on the ARC the MAP Server does not know whether the MAP Client received the last pollResult sent by the MAP Server, and the MAP Client does not know whether it missed any pollResult data.

After the MAP Client realizes that the session has ended, either because the MAP Client ended it explicitly or because an IF-MAP request resulted in an errorResult with an InvalidSessionID errorCode, the MAP Client SHOULD establish a new session and reestablish any subscriptions it is interested in. This enables the MAP Client and MAP Server to resynchronize pollResults.

4.6 Time Synchronization

Clock skew between the MAP Client and the MAP Server can cause interoperability issues, such as unexpected results in actions taken based on timestamped metadata. For example, a PDP responding to event messages could be configured to ignore messages older than a certain time delta; if the Sensor publishing those event messages has clock skew relative to the IF-MAP ecosystem in which it operates, the desired actions to be taken based on those events may not occur.

To avoid such interoperability issues, MAP Server time is considered to be the reference time in an IF-MAP ecosystem. Any timestamp present in the MAP is required to be accurate relative to the MAP Server time. This requirement applies to the following use cases:

- a MAP Client publishing metadata that include a timestamp,
- a MAP Client receiving such metadata in a search or poll result,
- or a MAP Client searching or subscribing to metadata matching a timestamp-based filter.

All MAP Clients and MAP Servers SHOULD use some automated time synchronization method such as NTP in order to avoid time differences between the components of the IF-MAP ecosystem.

MAP Clients MUST detect clock skew relative to the MAP Server immediately after starting a new session and SHOULD also do that occasionally during the session in order to detect time drift.

To achieve clock skew detection, MAP Clients MUST add a small amount of harmless metadata and check the operational attributes to see how the MAP Server's concept of time differs from the MAP Client's.

When a MAP Client detects a clock skew relative to the MAP Server, the MAP Client MUST log the clock skew in an administrator-accessible log. MAP Clients MUST create timestamps that are consistent with the MAP Server time by adjusting for the clock skew. In other words, a MAP Client MUST adjust the timestamps in metadata it publishes to match the MAP Server time, and MUST adjust the timestamps on metadata received from the MAP Server to match its local time.

4.6.1 Clock Skew Detection

In order to detect clock skew relative to the MAP Server, MAP Clients MAY implement the procedure described below:

1. Compose a unique device identifier as described in section 3.3.2.2. This identifier is used to represent the MAP Client itself in the MAP graph.
2. Compose a client-time metadata with the current-timestamp attribute set to the MAP Client's current time in UTC. client-time is an Operational Metadata described in section 3.4.5.1.
3. Send a publish request to update the client-time metadata on the device identifier. For example:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:op-meta="http://www.trustedcomputinggroup.org/2012/IFMAP-
OPERATIONAL-METADATA/1">
  <env:Body>
    <ifmap:publish session-id="222">
      <update lifetime="session">
         <device>
           <name>111:33</name>
         </device>
         <metadata>
           <op-meta:client-time ifmap-cardinality="singleValue"
             current-timestamp="2011-10-27T23:49:05Z"/>
         </metadata>
      </update>
   </ifmap:publish>
  </env:Body>
</env:Envelope>
```

4. Send a search request starting at the device identifier, with a maximum depth of 0 and a result-filter that matches the client-time metadata published previously:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:op-meta="http://www.trustedcomputinggroup.org/2012/IFMAP-
OPERATIONAL-METADATA/1">
  <env:Body>
    <ifmap:search session-id="222" max-depth="0"
     result-filter="op-meta:client-time[@ifmap-publisher-
id='111']">
      <device>
        <name>111:33</name>
      </device>
    </ifmap:search>
  </env:Body>
</env:Envelope>
```

The MAP Server will return a result:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:op-meta="http://www.trustedcomputinggroup.org/2012/IFMAP-
OPERATIONAL-METADATA/1">
  <env:Body>
    <ifmap:response>
      <searchResult>
        <resultItem>
          <device>
            <name>111:33</name>
          </device>
          <metadata>
            <op-meta:client-time ifmap-cardinality="singleValue"
             ifmap-publisher-id="111"
             ifmap-timestamp="2011-10-27T23:51:42Z"
             ifmap-timestamp-fraction="3729871"
             current-timestamp="2011-10-27T23:49:05Z"/>
          </metadata>
        </resultItem>
      </searchResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

5. Compute the difference between the value of the current-timestamp attribute and the value of the ifmap-timestamp and ifmap-timestamp-fraction attributes. If the precision of the MAP Client time (current-timestamp) and MAP Server time (ifmap-timestamp + ifmap-timestamp-fraction) differ, truncate the more precise timestamp to match the precision of the less-precise timestamp. In the above example, MAP Client and MAP Server have a difference of 2 minutes and 37 seconds.
6. From that point, the MAP Client is required to either adjust its local clock or compensate for the time difference when dealing with timestamps in IF-MAP requests and responses, so that timestamps in metadata are accurate relative to MAP Server local time.

4.7 WSDL and Example Code

An IF-MAP WSDL file and code examples utilizing various SOAP toolchains can be found on the TCG web site: http://trustedcomputinggroup.org/ by searching for IF-MAP WSDL.

4.8 Handling Many Clients at Once

If a MAP Server listens on one TCP port on a single IP address, it can accept only approximately 60,000 concurrent connections. (The total port count is 2^16-1; however, with system reserved ports and other practical limitations in different operating systems, in practice no more than approximately 60,000 ports are available to be used simultaneously.) A MAP Client may maintain two TCP connections, so approximately 30,000 MAP Clients can use up all of these connections.

To support more MAP Clients at once, a MAP Server MAY listen to a particular port number on multiple IP addresses (such as multiple virtual IP addresses on the same physical interface). DNS can be configured to resolve the MAP Server’s fully qualified domain name (FQDN) to one of the IP addresses, round robin.

5 Recommendations for Backward Compatibility

New features and requirements in IF-MAP 2.2 are backwards compatible with IF-MAP 2.0 [29] and 2.1 [35], and are intended to improve interoperability between IF-MAP 2.2 clients and servers while preserving interoperability between IF-MAP 2.2 and earlier IF-MAP 2.x clients and servers.

5.1 Extended Identifiers

A MAP Client MAY make use of extended identifiers (see section 3.3.3) even if the IF-MAP protocol used is IF-MAP 2.0 or 1.1.

Future specifications may modify the protocol schema in order to use extended identifiers directly in IF-MAP requests. However, even if the schema is so modified, the definition and behavior of extended identifiers are unlikely to change. Only the transport mechanism for the extended identifiers, by encapsulation in an identity identifier (for backwards compatibility with IF-MAP 1.x and 2.x), may be removed. It is therefore recommended to implement the encoding and encapsulation of extended identifiers in a middle layer, so that function can be easily discarded - while the application logic stays unmodified - after an upgrade to future IF-MAP versions.

A MAP Client that acts on identity identifiers found in search results will inevitably need to distinguish non-extended identity identifiers from extended identifiers. Also, implementers developing a MAP Client that attaches a link to an extended identifier are encouraged to consider what effect that link will have in an environment containing earlier MAP Clients that do not support extended identifiers.

5.2 DN Comparison

IF-MAP 2.1 introduced new rules for comparison of distinguished-name identity identifiers (see section 3.3.2.3.1). This ensures interoperability between IF-MAP 2.1 (and later) clients using DN identifiers. Previous versions of the IF-MAP protocol did not specify how to compare DN identifiers; as a result, interoperability was not guaranteed. IF-MAP 2.2, similarly, does not guarantee interoperability between IF-MAP 2.1 (and later) clients and MAP Clients using an older version of the protocol which may implement a different mechanism for comparison.

5.3 Maintaining an IF-MAP Session

Since IF-MAP 2.1, IF-MAP requires MAP Clients to periodically send renewSession requests (see section 4.1.1); however, IF-MAP 2.0 clients might not do so. Therefore, MAP Servers should not assume that all IF-MAP 2.x clients will send renewSession requests, and should still utilize TCP keepalives to detect inactivity on the TCP connection as recommended in section 4.1.1.

5.4 Operational Metadata for Time Check

The time check procedure specified in section 4.6 does not have any requirements for MAP Servers. This has the following benefits:

- IF-MAP 2.1 (and later) clients can verify time synchronization even when working with older IF-MAP 2.0 servers.
- IF-MAP 2.0 and 1.1 clients can implement the time check procedure using IF-MAP 2.1 operational metadata, even with IF-MAP 2.0 or IF-MAP 1.1 servers, without needing to upgrade to IF-MAP 2.1 or beyond (e.g. using their respective older versions of the protocol).

5.5 Backwards Compatibility of IF-MAP 2.2 with IF-MAP 1.1

Backwards compatibility with IF-MAP 1.1 is unchanged from IF-MAP 2.0. For specific details, see section 5 of the TNC IF-MAP Binding for SOAP, Revision 2.0 [29].

6 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).

6.1 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.

6.1.1 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

6.1.2 MAP Clients

Authorized MAP Clients are trusted to:

- Preserve the confidentiality of sensitive data retrieved from the MAP Server
- Protect the accuracy of data in the MAP Server database, by avoiding intentional 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

6.1.3 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.

6.1.4 Certification Authority

The Certification Authority (CA) that issues certificates for the MAP Server and/or MAP Clients (or each CA, if there are several) is trusted to:

- Protect the confidentiality of the CA's private key
- Ensure that only proper certificates are issued and that all certificates are issued in accordance with the CA's policies
- Revoke certificates previously issued when necessary
- Regularly and securely distribute certificate revocation information
- Promptly detect and report any violations of this trust so that they can be handled

The CA is not expected (trusted) to:

- Refrain from issuing certificates that go beyond name constraints or other constraints imposed by a relying party or a cross-certificate

6.2 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.

6.2.1 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 to glean 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

6.2.2 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). They might also occur 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 MAP 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.

6.2.3 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 MAP 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 the 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.

6.2.4 Certification Authority

A Certification Authority trusted to issue certificates for the MAP Server and/or MAP Clients can mount several attacks:

- Issue certificates for unauthorized parties, enabling them to impersonate authorized parties such as the MAP Server or a MAP Client. This can lead to all the threats that can be mounted by the certificate's subject.
- Issue certificates without following all of the CA's policies. Because this can result in issuing certificates that may be used to impersonate authorized parties, this can lead to all the threats that can be mounted by the certificate's subject.
- Fail to revoke previously issued certificates that need to be revoked. This can lead to undetected impersonation of the certificate's subject or failure to revoke authorization of the subject, and therefore can lead to all of the threats that can be mounted by that subject.
- Fail to regularly and securely distribute certificate revocation information. This may cause a relying party to accept a revoked certificate, leading to undetected impersonation of the certificate's subject or failure to revoke authorization of the subject, and therefore can lead to all of the threats that can be mounted by that subject. It can also cause a relying party to refuse to proceed with a transaction because timely revocation information is not available, even though the transaction should be permitted to proceed.
- Allow the CA's private key to be revealed to an unauthorized party. This can lead to all the threats above. Even worse, the actions taken with the private key will not be known to the CA.
- Fail to promptly detect and report errors and violations of trust so that relying parties can be promptly notified. This can cause the threats listed earlier in this section to persist longer than necessary, leading to many knock-on effects.

6.3 Countermeasures

6.3.1 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 ([8], [10], or [18]) as described in IETF RFC 2818 [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 IETF RFC 2617 [12]. MAP Clients and MAP Servers using mutual certificate-based authentication SHOULD each verify the revocation status of the other party.

All MAP Servers and MAP Clients MUST implement both mutual certificate-based authentication and Basic Authentication. The selection of which MAP Client authentication technique to use in any particular deployment is left to the administrator. A MAP Server SHOULD support Basic Authentication against a RADIUS [6] server and against an LDAP [36] server. A MAP Server MAY also support a local, configurable set of Basic Authentication userid-password pairs. If so, it is implementation dependent whether a MAP Server ends a session when an administrator changes the configured password. Since Basic Authentication has many security disadvantages (especially the transmission of reusable MAP Client passwords to the MAP Server), it SHOULD only be used when absolutely necessary. SOAP intermediaries MUST NOT be used.

Per the HTTP specification [17], when basic authentication is in use, a MAP Server MAY respond to any request that lacks credentials with HTTP code 401. A MAP Client SHOULD avoid this code by submitting basic auth credentials with every request when basic authentication is in use. If it does not do so, a MAP Client MUST respond to this code by resubmitting the same request with credentials (unless the MAP Client is shutting down).

Upon successful authentication, the trusted client entities MUST be verified for authorization to serve the MAP Client role. TNC MAP Content Authorization [38] SHOULD be used to authorize individual operations requested by MAP Clients.

These protocol security measures provide protection against all the network attacks listed in section 6.2.1 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. In addition, a MAP Server MAY automatically disable a misbehaving MAP Client.

6.3.2 Securing MAP Clients

MAP Clients (such as branch office firewalls) may be deployed in locations that are susceptible to physical attacks[4]. 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 MAP 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 MAP Client as a DHCP server and authorize only the operations and metadata types needed by a DHCP server to be permitted for that MAP 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 MAP Client activities.

To avoid content-based attacks, the MAP Server MAY validate the XML of the 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 SHOULD go through a TNC handshake to verify the integrity of the MAP Client, and SHOULD, if feasible, utilize a Trusted Platform Module (TPM) for identity and/or integrity measurements of the MAP Client within a TNC handshake. They

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

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.

6.3.3 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 go through a regular TNC handshake to verify the integrity of the MAP Server, and SHOULD utilize a Trusted Platform Module (TPM) for identity and/or integrity measurements of the MAP Client within a TNC handshake. 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 MAP 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.

6.3.3.1 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 3.9.3.5). This mitigates the threat of a MAP Client causing resource exhaustion by issuing a search or subscription that leads to an enormous result.

6.3.3.2 Cryptographically random session-id and authentication checks for ARC

A MAP Server SHOULD ensure that the MAP Client establishing an ARC is the same MAP Client as the MAP 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.

6.3.4 Securing the Certification Authority

As noted above, compromise of a Certification Authority (CA) trusted to issue certificates for the MAP Server and/or MAP Clients is a major security breach. Many guidelines for proper CA security have been developed: the CA/Browser Forum's Baseline Requirements, the AICPA/CICA Trust Service Principles, etc. The CA operator and relying parties should agree on an appropriately rigorous security practices to be used.

Even with the most rigorous security practices, a CA may be compromised. If this compromise is detected quickly, relying parties can remove the CA from their list of trusted CAs, and other CAs can revoke any certificates issued to the CA. However, CA compromise may go undetected for some time, and there's always the possibility that a CA is being operated improperly or in a manner that is not in the interests of the relying parties. For this reason, relying parties may wish to "pin" a small number of particularly critical certificates (such as the certificate for the MAP

Server). Once a certificate has been pinned, the relying party will not accept another certificate in its place unless the Administrator explicitly commands it to do so. This does not mean that the relying party will not check the revocation status of pinned certificates. However, the Administrator may still be consulted if a pinned certificate is revoked, since the CA and revocation process are not completely trusted.

6.4 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.

7 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[5].

Three identifier types, in particular, are attachment points for metadata with potentially privacy-sensitive implications: identity, mac-address, and ip-address.

7.1 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.

7.2 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.

7.3 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.

[5]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.

8 References

[1] Trusted Computing Group, *TNC Architecture for Interoperability*, Revision 1.5, May 2012

[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

[15] Trusted Computing Group, *TNC IF-MAP Metadata for Network Security*, Revision 1.1, May 2012

[16] P. Leach, M. Mealling, R. Salz, *A Universally Unique IDentifier (UUID) URN Namespace*, RFC 4122, July 2005, IETF

[17] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *Hypertext Transport Protocol – HTTP/1.1*, RFC 2616, June 1999, IETF

[18] T. Dierks, E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, August 2006, IETF

[19] K. McCloghrie, D. Perkins, J. Schoenwaelder, *Structure of Management Information Version 2 (SMIv2)*, April 1999, IETF

[20] Trusted Computing Group, *TPM Main Specification Level 2 Version 1.2*, Revision 103, June 2007

[21] ITU-T, *X.500 Directory Specification*, August 2005

[22] P. Mockapetris, *Domain names - Implementation and Specification*, RFC 1035, November 1987, IETF

[23] P. Resnick, *Internet Message Format*, RFC 5322, October 2008, IETF

[24] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, *The Kerberos Network Authentication Service (V5)*, RFC 4120, July 2005, IETF

[25] M. J. Handley, H. Schulzrinne, E. M. Schooler, J. Rosenberg, *SIP: Session Initiation Protocol*, RFC 3261, March 1999, IETF

[26] H. Schulzrinne, *The tel URI for Telephone Numbers*, RFC 3966, December 2004, IETF

[27] R. Moskowitz, P. Nikander, *Host Identity Protocol (HIP) Architecture,* RFC 4423, May 2006, IETF

[28] W3C, Canonical XML Version 1.1, May 2008

[29] Trusted Computing Group, *TNC IF-MAP Binding for SOAP*, Revision 2.0, November 2011

[30] T. Moses, *eXtensible Access Control Markup Language (XACML),* Version 2.0, February 2005, OASIS

[31] M. Wahl, S. Kille, T. Howes, *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*, RFC 2253, December 1997, IETF

[32] ITU-T, *Recommendation X.690*, December 1997

[33] R. Housley, W. Polk, W. Ford, D. Solo, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3280, April 2002, IETF

[34] R. Braden, *Requirements for Internet Hosts -- Communication Layers*, RFC 1122, October 1989, IETF

[35] Trusted Computing Group, *TNC IF-MAP Binding for SOAP*, Revision 2.1, October 2013

[36] M. Wahl, T. Howes, S. Kille, *Lightweight Directory Access Protocol (v3)*, RFC 2251, Standards Track, December 1997, IETF

[37] The Unicode Consortium. *The Unicode Standard, Version 6.2.0*, (Mountain View, CA: The Unicode Consortium, 2012. ISBN 978-1-936213-07-8) http://www.unicode.org/versions/Unicode6.2.0/

[38] Trusted Computing Group, *TNC MAP Content Authorization*, Revision 1.0, May 2014

9 Basic Example

The following examples are intended to demonstrate simple publish and search operations of IF-MAP, basic coordination between MAP Clients, the use of vendor specific metadata, and the use of extended identifiers. This example is not intended to be comprehensive; it is designed to be simple and suggestive of some uses to which IF-MAP might be applied. In a commercial implementation, the metadata schema and utilization will likely be more sophisticated than shown here. For detailed examples and additional real-world use-cases one should refer to a use-case driven IF-MAP metadata standard such as [15].

9.1 Webcam Conferencing

A video conferencing application uses IF-MAP to store and retrieve information about the webcam capabilities of computers. The application publishes metadata defined by the following XML schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns="urn:example.com:webcam"
  targetNamespace="urn:example.com:webcam">

  <!-- webcam-capabilities is attached to a device identifier and
       describes the capabilities of a computer’s webcam -->
  <xsd:element name="webcam-capabilities">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="enabled" type="xsd:int"/>
        <xsd:element name="video-format" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attributeGroup
        ref="ifmap:singleValueMetadataAttributes"/>
    </xsd:complexType>
  </xsd:element>

  <!-- webcam-user is attached to the link between a device
       identifier and an identity identifier, and is used to
       associate a user with a computer that has a webcam -->
  <xsd:element name="webcam-user">
    <xsd:complexType>
      <xsd:attributeGroup
        ref="ifmap:singleValueMetadataAttributes"/>
    </xsd:complexType>
  </xsd:element>

  <!-- webcam-ip is attached to the link between a device
       identifier and an ip-address identifier, and is used to
       associate an IP address with a computer that has a
       webcam -->
  <xsd:element name="webcam-ip">
    <xsd:complexType>
      <xsd:attributeGroup
        ref="ifmap:singleValueMetadataAttributes"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

When the user Joe starts the video conferencing application, the application publishes information about Joe, the webcam, and the IP address of Joe's computer into IF-MAP. In order to do this, the application generates a device name to be used in a device identifier to represent Joe's computer. Once the application has a device identifier for Joe's computer, it publishes the following metadata:

- webcam-capabilities attached to the device identifier
- webcam-user attached to the link between the device identifier and Joe's identity identifier
- webcam-ip attached to the link between the device identifier and the ip-address identifier that represents the IP address of Joe's computer.

The publish request looks like this:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:wc="urn:example.com:webcam">
  <env:Body>
    <ifmap:publish session-id="222">
      <update>
        <device><name>222:1234</name></device>
        <metadata>
          <wc:webcam-capabilities
            ifmap-cardinality="singleValue">
            <enabled>1</enabled>
            <video-format>VGA</video-format>
          </wc:webcam-capabilities>
        </metadata>
      </update>
      <update>
        <device><name>222:1234</name></device>
        <identity name="Joe" type="username"/
        <metadata>
          <wc:webcam-user ifmap-cardinality="singleValue"/>
        </metadata>
      </update>
      <update>
        <device><name>222:1234</name></device>
        <ip-address value="192.0.2.11" type="IPv4"/>
        <metadata>
          <wc:webcam-ip ifmap-cardinality="singleValue"/>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The MAP Server responds with a publishReceived message:

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

xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2">
  <env:Body>
```

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

Sally is also running the video conferencing application, and decides to place a video call to Joe. Sally enters the user name "Joe" into the application. The application performs an IF-MAP search to determine the IP address and webcam capabilities of Joe's computer. The search starts with the identity identifier for Joe, and then follows the webcam-user link to find Joe's device. The search continues by following the webcam-ip link to find the IP address of Joe's computer. Along the way, the search picks up the webcam-capabilities metadata.

The search request looks like this:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:wc="urn:example.com:webcam">
  <env:Body>
    <ifmap:search session-id="223">
      match-links="wc:webcam-user or wc:webcam-ip"
      max-depth="2" result-filter="wc:webcam-capabilities"
      <identity name="Joe" type="username"/>
    </ifmap:search>
  </env:Body>
</env:Envelope>
```

The MAP Server responds with a searchResult message:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:wc="urn:example.com:webcam">
  <env:Body>
    <ifmap:response>
      <searchResult>
        <resultItem>
          <device><name>222:1234</name></device>
          <metadata>
            <wc:webcam-capabilities
              ifmap-cardinality="singleValue"
              ifmap-timestamp="xxx"
              ifmap-timestamp-fraction="yyy"
              ifmap-publisher-id="XYZ">
              <enabled>1</enabled>
              <video-format>VGA</video-format>
            </wc:webcam-capabilities>
          </metadata>
        </resultItem>
        <resultItem>
          <device><name>222:1234</name></device>
          <identity name="Joe" type="username"/>
        </resultItem>
        <resultItem>
```

```
          <device><name>222:1234</name></device>
          <ip-address value="192.0.2.11" type="IPv4"/
        </resultItem>
      </searchResult>
    </ifmap:response>
  </env:Body>
</env:Envelope>
```

The video conferencing application on Sally's computer determines from the webcam-capabilities that a video call is possible. The application uses the IP address returned in the search to contact Joe's computer and start the video call.

9.2 Webcam Conferencing with Extended Identifier

A new version of the video conferencing application implements an extended identifier. Its schema is modified to define a “webcam” identifier as well as one new metadata type:

```
<xsd:element name="webcam">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="base-id:IdentifierType">
        <xsd:attribute name="vendor-id" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[A-F0-9]+"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="serial-number" type="xsd:string"
use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- webcam-connected is attached to a link between a webcam
     identifier and a device identifier. It specifies which
     webcam is currently connected to the computer -->
<xsd:element name="webcam-connected">
  <xsd:complexType>
   <xsd:attribute name="status" use="required">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:enumeration value="idle"/>
      <xsd:enumeration value="in-use"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
   <xsd:attributeGroup ref="ifmap:singleValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

When the user Joe starts the new release of the video conferencing application, the application detects multiple webcams connected to Joe’s computer. It lets Joe select which webcam he wants to use and publishes this information with the “webcam-connected” metadata on a link between each webcam identifier and the device identifier representing Joe’s computer:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:wc="urn:example.com:webcam">
  <env:Body>
    <ifmap:publish session-id="222">
        <device><name>222:1234</name></device>

      <update>
        <identity type="other" other-type-definition="extended"
name="&lt;webcam xmlns=&quot;urn:example.com:webcam&quot; vendor-
id=&quot;AABBCC&quot; serial-
number=&quot;123&quot;&gt;&lt;/webcam&gt;"/>
        <metadata>
          <wc:webcam-connected ifmap-cardinality="singleValue"
status="in-use"/>
        </metadata>
      </update>
      <update>
        <device><name>222:1234</name></device>
        <identity type="other" other-type-definition="extended"
name="&lt;webcam xmlns=&quot;urn:example.com:webcam&quot; vendor-
id=&quot;DDEEFF&quot; serial-
number=&quot;456&quot;&gt;&lt;/webcam&gt;"/>
        <metadata>
          <wc:webcam-connected ifmap-cardinality="singleValue"
status="idle"/>
        </metadata>
      </update>
      <update>
        <device><name>222:1234</name></device>
        <identity type="other" other-type-definition="extended"
name="&lt;webcam xmlns=&quot;urn:example.com:webcam&quot; vendor-
id=&quot;001122&quot; serial-
number=&quot;789&quot;&gt;&lt;/webcam&gt;"/>
        <metadata>
          <wc:webcam-connected ifmap-cardinality="singleValue"
status="idle"/>
        </metadata>
      </update>
    </ifmap:publish>
  </env:Body>
</env:Envelope>
```

The MAP Server responds with a publishReceived message:

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

xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2">
  <env:Body>
    <ifmap:response>
      <publishReceived/>
    </ifmap:response>
  </env:Body>
```

```
</env:Envelope>
```

The IT administrator of Joe’s company would like to know if the webcam with vendor id “DDEEFF” and serial number “456” is available. He searches the IF-MAP repository and finds that it is indeed available and connected to Joe’s computer.

The search request looks like this:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:wc="urn:example.com:webcam">
  <env:Body>
    <ifmap:search session-id="223">
      match-links="wc:webcam-connected[@status=’idle’] or
wc:webcam-user"
      max-depth="2" result-filter="wc:webcam-connected"
        <identity type="other" other-type-definition="extended"
name="&lt;webcam xmlns=&quot;urn:example.com:webcam&quot; vendor-
id=&quot;DDEEFF&quot; serial-
number=&quot;456&quot;&gt;&lt;/webcam&gt;"/>
    </ifmap:search>
  </env:Body>
</env:Envelope>
```

The MAP Server responds with a searchResult message:

```
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
  xmlns:wc="urn:example.com:webcam">
  <env:Body>
    <ifmap:response>
      <searchResult>
        <resultItem>
          <device><name>222:1234</name></device>
          <identity type="other" other-type-definition="extended"
name="&lt;webcam xmlns=&quot;urn:example.com:webcam&quot; vendor-
id=&quot;DDEEFF&quot; serial-
number=&quot;456&quot;&gt;&lt;/webcam&gt;"/>
          <metadata>
            <wc:webcam-connected status="idle"
              ifmap-cardinality="singleValue"
              ifmap-timestamp="123"
              ifmap-timestamp-fraction="456"
              ifmap-publisher-id="XYZ"/>
          </metadata>
        </resultItem>
        <resultItem>
          <device><name>222:1234</name></device>
          <identity name="Joe" type="username"/>
        </resultItem>
      </searchResult>
    </ifmap:response>
  </env:Body>
```

```
</env:Envelope>
```

10 IF-MAP Schema

10.1 Identifier Types, Requests and Responses

IF-MAP requests and responses MUST comply with the following schema.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.trustedcomputinggroup.org/2010/IFMAP/2"

targetNamespace="http://www.trustedcomputinggroup.org/2010/IFMAP/
2">

  <!-- 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="newSession" type="NewSessionRequestType"/>
  <xsd:element name="renewSession" type="SessionRequestType"/>
  <xsd:element name="endSession" type="SessionRequestType"/>
  <xsd:element name="response" type="ResponseType"/>

  <!-- 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="hip-hit"/>
          <xsd:enumeration value="kerberos-principal"/>
          <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>

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

  <xsd:complexType name="NewSessionRequestType">
    <xsd:attribute name="max-poll-result-size" type="xsd:integer"
use="optional"/>
  </xsd:complexType>

  <xsd:attributeGroup name="validationAttributes">
    <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:attributeGroup>

  <xsd:attributeGroup name="sessionAttributes">
    <xsd:attribute name="session-id" type="xsd:string"
use="required"/>
  </xsd:attributeGroup>

  <!-- SessionRequestType is stateful session handling -->
  <xsd:complexType name="SessionRequestType">
    <xsd:attributeGroup ref="sessionAttributes"/>
  </xsd:complexType>

  <!-- UpdateType is the type for requests that update
       metadata -->
  <xsd:complexType name="UpdateType">
    <xsd:sequence>
      <xsd:choice minOccurs="1" maxOccurs="2">
        <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:element name="metadata" type="MetadataListType"
minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="lifetime" default="session">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="session"/>
          <xsd:enumeration value="forever"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </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="2">
        <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:attribute name="filter" type="FilterType"
use="optional"/>
  </xsd:complexType>

  <!-- PublishRequestType updates or deletes metadata -->
  <xsd:complexType name="PublishRequestType">
    <xsd:sequence>
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="update" type="UpdateType"/>
        <xsd:element name="notify" type="UpdateType"/>
        <xsd:element name="delete" type="DeleteType"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attributeGroup ref="sessionAttributes"/>
    <xsd:attributeGroup ref="validationAttributes"/>
  </xsd:complexType>

  <!-- SearchType specifies the parameters for a search, and is
       used for the search element as well as the update
       sub-element of a subscribe element -->
  <xsd:complexType name="SearchType">
    <xsd:sequence>
      <xsd:choice minOccurs="1" maxOccurs="1">
        <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:attribute name="match-links" type="FilterType"/>
    <xsd:attribute name="max-depth" type="xsd:unsignedInt"/>
    <xsd:attribute name="terminal-identifier-type"
type="xsd:string"/>
    <xsd:attribute name="max-size" type="xsd:unsignedInt"/>
    <xsd:attribute name="result-filter" type="FilterType"/>
  </xsd:complexType>

  <!-- SearchRequestType queries the server for matching
       metadata -->
  <xsd:complexType name="SearchRequestType">
    <xsd:complexContent>
      <xsd:extension base="SearchType">
        <xsd:attributeGroup ref="sessionAttributes"/>
        <xsd:attributeGroup ref="validationAttributes"/>
      </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:sequence>
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="update">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="SearchType">
                <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:attributeGroup ref="sessionAttributes"/>
    <xsd:attributeGroup ref="validationAttributes"/>
  </xsd:complexType>

  <!-- PollRequestType is for polling for notification of
       metadata changes that match subscriptions -->
  <xsd:complexType name="PollRequestType">
    <xsd:attributeGroup ref="validationAttributes"/>
    <xsd:attributeGroup ref="sessionAttributes"/>
  </xsd:complexType>

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


  <!-- ResultItemType is for search or poll results showing
       metadata attached to identifiers and links-->
  <xsd:complexType name="ResultItemType">
    <xsd:sequence>
      <xsd:choice minOccurs="1" maxOccurs="2">
        <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: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="resultItem" type="ResultItemType"
minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name"/>
  </xsd:complexType>

  <!-- PollResultType contains information for each
       subscription that had changes since the last poll:
       an optional searchResult, zero or more updateResults,
       zero or more deleteResults, zero or more notifyResults -->
  <xsd:complexType name="PollResultType">
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="searchResult"
type="SearchResultType"/>
        <xsd:element name="updateResult"
type="SearchResultType"/>
        <xsd:element name="deleteResult"
type="SearchResultType"/>
        <xsd:element name="notifyResult"
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="InvalidSchemaVersion"/>
          <xsd:enumeration value="InvalidSessionID"/>
          <xsd:enumeration value="MetadataTooLong"/>
          <xsd:enumeration value="SearchResultsTooBig"/>
          <xsd:enumeration value="PollResultsTooBig"/>
          <xsd:enumeration value="SystemError"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="name"/>
  </xsd:complexType>

  <!-- SessionResultType is for stateful session handling -->
  <xsd:complexType name="SessionResultType">
```

```
    <xsd:attribute name="session-id" type="xsd:string"
use="required"/>
    <xsd:attribute name="ifmap-publisher-id" type="xsd:string"
use="required"/>
  </xsd:complexType>

  <xsd:complexType name="NewSessionResultType">
    <xsd:attribute name="session-id" type="xsd:string"
use="required"/>
    <xsd:attribute name="ifmap-publisher-id" type="xsd:string"
use="required"/>
    <xsd:attribute name="max-poll-result-size" type="xsd:integer"
use="optional"/>
  </xsd:complexType>

  <!-- ResponseType encapsulates results from all the different
       requests -->
  <xsd:complexType name="ResponseType">
    <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:element name="newSessionResult"
type="NewSessionResultType"/>
      <xsd:element name="renewSessionResult"/>
      <xsd:element name="endSessionResult"/>
    </xsd:choice>
    <xsd:attributeGroup ref="validationAttributes"/>
  </xsd:complexType>

  <!-- IfmapTimeStampFractionType constrains the ifmap-timestamp-
       fraction content →
  <xsd:simpleType name="IfmapTimeStampFractionType">
    <xsd:restriction base="xsd:decimal">
        <xsd:minInclusive value="0"/>
        <xsd:maxExclusive value="1"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- metadataAttributes specifies attributes on metadata which
       are used by MAP servers.

       A MAP Server MUST behave as if it adds ifmap-publisher-id,
       ifmap-timestamp, and ifmap-timestamp-fraction to all
       metadata before storage in the database.

       MAP Clients MUST NOT include these or any attributes that
       begin with ifmap- in published metadata.

       cardinality is used by the MAP Client to indicate to the
       server whether the metadata can have multiple values.

       anyAttribute enables metadata elements which include the
       ifmap:metadataAttributes attributeGroup to add new
       attributes for use with future versions of IF-MAP. -->
  <xsd:attributeGroup name="metadataAttributes">
    <xsd:attribute name="ifmap-publisher-id"/>
    <xsd:attribute name="ifmap-timestamp" type="xsd:dateTime"/>
    <xsd:attribute name="ifmap-timestamp-fraction"
type="xsd:IfmapTimestampFractionType"/>
    <xsd:anyAttribute/>
  </xsd:attributeGroup>

  <xsd:attributeGroup name="singleValueMetadataAttributes">
    <xsd:attributeGroup ref="metadataAttributes"/>
    <xsd:attribute name="ifmap-cardinality" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="singleValue"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:attributeGroup>

  <xsd:attributeGroup name="multiValueMetadataAttributes">
    <xsd:attributeGroup ref="metadataAttributes"/>
    <xsd:attribute name="ifmap-cardinality" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="multiValue"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:attributeGroup>

</xsd:schema>
```

10.2 Operational Metadata

This schema contains metadata for use in IF-MAP operations.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://www.trustedcomputinggroup.org/2012/IFMAP-
OPERATIONAL-METADATA/1"
xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
targetNamespace="http://www.trustedcomputinggroup.org/2012/IFMAP-
OPERATIONAL-METADATA/1">

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

  <!-- client-time is a device metadata used
       by MAP Clients to detect clock skew with
       MAP Server -->
  <xsd:element name="client-time">
    <xsd:complexType>
      <xsd:attribute name="current-time" type="xsd:dateTime"
use="required"/>
      <xsd:attributeGroup
ref="ifmap:singleValueMetadataAttributes"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

10.3 Base Identifier Type

This schema defines the base type for extended identifiers.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://www.trustedcomputinggroup.org/2012/IFMAP-
IDENTIFIER/1"

targetNamespace="http://www.trustedcomputinggroup.org/2012/IFMAP-
IDENTIFIER/1">

  <!-- Schema for IF-MAP extended identifiers base type -->
  <xsd:complexType name="IdentifierType">
    <xsd:attribute name="administrative-domain" type="xsd:string"
use="required"/>
  </xsd:complexType>

</xsd:schema>
```

10.4 MAP Server Schema

This schema contains extended identifiers and metadata used by the MAP Server.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
 xmlns:base-id="http://www.trustedcomputinggroup.org/2012/IFMAP-
IDENTIFIER/1"
 xmlns="http://www.trustedcomputinggroup.org/2013/IFMAP-SERVER/1"

targetNamespace="http://www.trustedcomputinggroup.org/2013/IFMAP-
SERVER/1">

  <!-- Schema for extended identifiers and metadata used
       by the MAP Server -->

  <!-- ifmap-server is an extended identifier used
       by MAP Servers as a target on which to
       attach server-capability metadata -->
  <xsd:element name="ifmap-server">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="base-id:IdentifierType">
        </xsd:extension>
      <xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <!-- server-capability is operational metadata used
       by MAP Servers to indicate support for optional
       capabilities -->
  <xsd:element name="server-capability">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="capability" type="xsd:string"
minOccurs="1" maxOccurs="unbounded">
        </xsd:element>
      </xsd:sequence>
      <xsd:attributeGroup
ref="ifmap:singleValueMetadataAttributes"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```