TCG Trusted Network Communications

TNC MAP Content Authorization

Specification Version 1.0
Revision 36
18 June 2014

Contact:

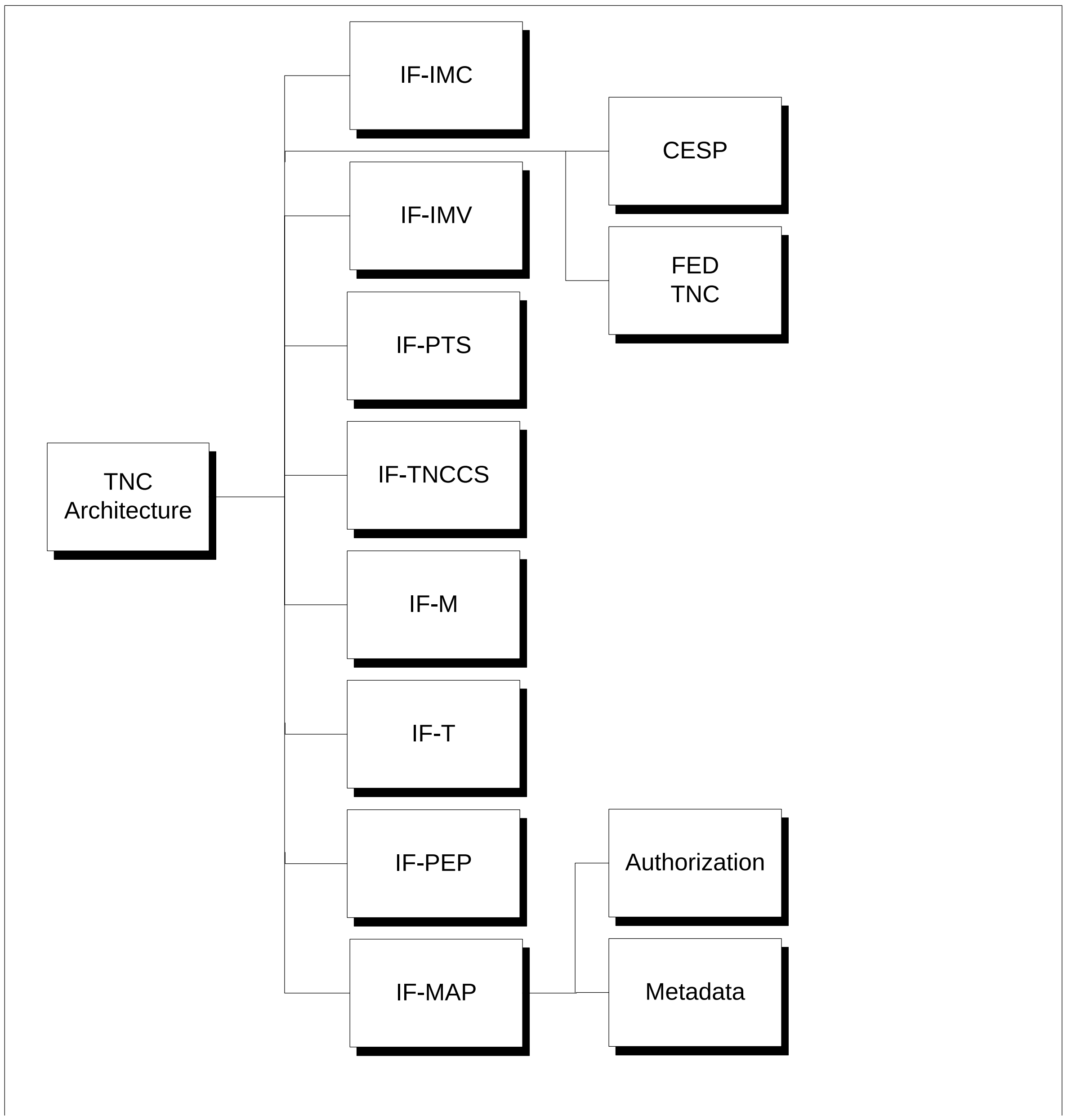
admin@trustedcomputinggroup.org

TCG

TCG Published

Disclaimers, Notices, and License Terms

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 worki groups in the TCG.

Frédéric Guihéry	Amossys
David Louin	Amossys
Maxime Olivier	Amossys
David Mattes (Invited Expert)	Asguard Security
Padma Krishnaswamy	Batelle Memorial Institute
Eric Fleischman	Boeing
Steven Venema	Boeing
Fong Pong	Broadcom
Eric Byres (Invited Expert)	Byres Security
Nancy Cam-Winget	Cisco Systems
John Schwaller	DRS Tactical Systems
Kurt Semba	Enterasys Networks
Josef von Helden	Fachhochschule Hannover
Ronald Marx	Fraunhofer SIT
Bobbi Michelle Wehrfritz	Fritz Technology
Gerald Maunier	Gemalto N.V.
Chris Daly	General Dynamics C4 Systems
Graeme Proudler	Hewlett-Packard
Andreas Steffen	HSR University of Applied Sciences Rapperswil
Guerney Hunt	IBM
Stu Bailey	Infoblox
Navin Boddu	Infoblox
Rick Kagan	Infoblox
James Tan	Infoblox
David Vigier	Infoblox
Ben Warren	Infoblox
Kent Landfield	Intel Corporation
Cliff Kahn (Co-Editor)	Juniper Networks
Lisa Lorenzin	Juniper Networks
Steve Hanna	Juniper Networks
Mark Labbancz	Lumeta Corporation
Atul Shah	Microsoft
Wolfgang Durr	mikado soft GmbH
Charles Schmidt	MITRE
Jim Banoczi	US National Security Agency
Chris Bean	US National Security Agency
Julie Haney	US National Security Agency
Jessica Fitzgerald-McKay	US National Security Agency
Gloria Serrao	US National Security Agency
Theresa Thomas	US National Security Agency
Josef Allen	Oak Ridge National Lab
Ira McDonald	Samsung
Carolin Latze	SwissCom
Paul Sangster	Symantec
Anne-Rose Gratadour	Thales

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

Richard Hill	Boeing
Steve Venema	Boeing
John Tolbert	Boeing
Nancy Cam-Winget	Cisco Systems
Arne Welzel	FHH
Josef von Helden	FHH
James Tan	Infoblox
David Vigier	Infoblox
Stu Bailey	Infoblox
Navin Boddu	Infoblox
Steve Hanna	Juniper
Clifford Kahn	Juniper (Editor)
Lisa Lorenzin	Juniper
Venkata Srikar Damaraju	Juniper
Atul Shah	Microsoft
Trevor Freeman	Microsoft
Charles Schmidt	The Mitre Corporation

Thanks also to ViewDS for providing XACML software used in the development of this specification.

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 and facilitates coordination between networking and security components. IF-MAP, one of the specifications that make up the TNC architecture, provides a standard interface between a Metadata Access Point and other elements of the TNC architecture. This specification, TNC MAP Content Authorization, provides a standard model for controlling what operations MAP Clients can execute upon the content of a MAP Server.

Architects, designers, developers and technologists who wish to implement, use, or understand TNC MAP Content Authorization should read this document carefully. Before reading this document any further, the reader should review and understand the TNC architecture as described in [1], the TNC IF-MAP Binding for SOAP[3], and one or both of the TNC IF-MAP Metadata for Network Security[10] and TNC IF-MAP Metadata for ICS Security[11]6.2.

1.2 Keywords

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

1.3 Glossary

Administrator	One who configures or monitors MAP Content Authorization, defines and/or assigns roles and tasks, defines and/or customizes policy, reviews logs, etc.
TNC MAP Content Authorization (MAP Content Authorization)	Authorization that governs each MAP client’s access to the content of a MAP Server
External authorization	Authorization that permits actions in the environment that the MAP content describes. External authorization is out of scope of this specification, but provides use cases
Access-request constellation	The metadata attached to and around an access-request identifier—all of the metadata depicted in [10], Figure 1
Metadata Item	In this specification, a metadata item is an XML document found in a MAP or in a request for storing in a MAP, plus the identifier or link to which the XML document is or will be attached.
Role	A purpose of a particular MAP Client, and a designation that lets the MAP Client do certain things. A role may be: • A role found in section 3.4 of the TNC Architecture[1] • A function found in section 3.6 of the TNC Architecture • Found in another specification, such as IF-MAP Metadata for ICS Security[11]6.2 • A custom role defined by the customer By contrast, [10] uses “role” as a property of an access-request, affecting an endpoint’s access.
Decision Request	A request by a XACML PEP to a XACML PDP to render an authorization decision [adapted from [4], section 1.1.1]

Authorization Decision	The result returned by the XACML PDP to the XACML PEP. “Permit”, “Deny”, “Indeterminate”, or “NotApplicable", plus optional parts [adapted from [4], section 1.1.1]
Content Authorization Policy	A XACML Policy Set that governs access to the MAP Server’s content – an input to the XACML Policy Decision Point
XACML Policy Decision Point (PDP)	A component that evaluates applicable policy and renders an Authorization Decision [adapted from [4], section 1.1.1]
XACML Policy Enforcement Point (PEP)	A component that performs access control, by making Decision Requests and enforcing Authorization Decisions (adapted from [4], section 1.1.1). In this specification, the MAP Server is the XACML PEP
XACML Policy Administration Point (PAP)	A component that creates a XACML policy or policy set [adapted from [4], section 1.1.1]
XACML Policy Information Point (PIP)	A component that supplies attribute values [adapted from [4], section 1.1.1]
XACML resource	“Data, service or system component” [adapted from [4], section 1.1.1]. Something to which access is controlled. In this specification, a metadata item or an identifier

1.4 Aligned OASIS Profile

This specification is aligned with OASIS XACML MAP Content Authorization Profile Version 1.0[21]. Compliance with this specification is intended to imply compliance with the profile. In case of differences, implementers would do well to comply with both documents.

2 Background

2.1 Role of TNC MAP Content Authorization

This specification defines an authorization model that restricts the operations each MAP Client can perform on MAP content.

A standard authorization model has value to implementers and end users, who are likely to be reluctant to deploy products which bind them to one particular vendor and/or product. For example, a vendor developing an administrative client for authorization purposes may be reluctant to have their implementation depend on one MAP Server vendor’s authorization system. And an end user deploying an IF-MAP enabled infrastructure which relies on authorization should not be locked into a single vendor's MAP Server implementation.

The relatively free-form, non-hierarchical character of the MAP data model makes access control challenging. A standard authorization model also allows metadata and suggested policy elements to be designed together.

TNC MAP Content Authorization is specified separately from the base IF-MAP operations for two reasons. First, because MAP authorization is a complex topic which benefits from concentrating use cases, requirements, and normative text into a single specification informed by other TNC work items and products that require authorization. Second, to enable MAP Server implementers to decide whether or not to implement authorization in their MAP Server and, if so, whether or not to pursue compliance with this specification.

2.2 IF-MAP

This specification builds on TNC IF-MAP Binding for SOAP[3]. Ideally, the reader should be thoroughly familiar with that specification; however, a non-normative summary of pertinent parts of that specification follows.

A MAP Server is a store of “state information about devices, users, and flows in a network”. ([3], section 2.1)

MAP Clients may publish information to a MAP Server, search the information in a MAP Server, and subscribe to notifications from a MAP Server when information stored in the server changes. ([3], section 2.1)

2.2.1 Data Model

For the definitive description of the IF-MAP data model and examples of its use, see [3], especially section 2.6.

2.2.1.1 Identifiers and Links

The IF-MAP data model is an undirected, labeled graph [12]. The edges are called *links*. The graph is infinite: all possible vertices and all possible links are considered to exist at all times.

Every possible *identifier* labels a vertex. An identifier is an XML element. Each identifier has an *identifier type*: either the namespace of the identifier’s XML schema plus the type name within the schema, or one of five original identifier types.

Each identifier may include an administrative-domain. The administrative-domain is a string whose format is organizationally defined, an optional qualifier used to prevent name conflicts. It provides a way of grouping identifiers; however, this group structure is not generally represented in the structure of the graph.

2.2.1.2 Metadata

Metadata items, the basic units of content, are attached to identifiers and links. A finite subset of the identifiers and links have one or more metadata items attached.

A metadata item is an XML document. When stored, a metadata item is given certain top-level XML attributes, called *operational attributes*, by the MAP Server. One of these is a time stamp; another is a publisher-id, indicating which client published the metadata. publisher-ids have a one-to-one correspondence with client identities (see section 2.2.2).

A metadata item has a *metadata type,* which is the XML name space of the XML schema plus the type name within the schema.

Each metadata item also contains an attribute indicating whether it is single- or multi-valued. If it is multi-valued, other metadata items of the same type may be attached to the same identifier or link, and the order is significant. If it is single-valued, then only one metadata item of that type may be attached to a specific identifier or link, and an update operation that attaches it automatically deletes any previous metadata item of the same metadata type.

2.2.1.3 Extensibility

A MAP Server is required to be able to process identifier types and metadata types unknown to the server's implementers. (See section 2.6.3 of [3].)

2.2.2 MAP Client

For this specification, the MAP Client is the subject – the actor executing operations.

A MAP client can be authenticated using HTTP basic authentication or HTTPS certificate-based authentication. Certificate authentication is preferred for security reasons.

A MAP Server distinguishes its MAP Clients mainly by some of their credentials. For basic authentication, the MAP Server uses the MAP Client's username as the identifying credential attribute. For certificate authentication, the MAP Server uses 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") as the identifying credential attributes.

In addition, a MAP Server may be configurable so that two MAP Clients with the same identifying credential attributes can be distinguished by their IP addresses. (See [3], section 3.3.1.)

2.2.3 IF-MAP Requests

A single IF-MAP request can access many metadata items. See [3], section 3.7.1, for details.

A publish request is a series of operations:

- update adds or replaces metadata attached to an identifier or a link.
- notify tells the MAP Server to notify subscribers of metadata that does not persist in the database.
- delete removes metadata from an identifier or a link; the operation includes a filter specifying which metadata to remove.

A search request retrieves metadata. A simple query language allows the MAP Client to specify the starting identifier, the radius of the search, what links to traverse (by metadata type and other criteria), and what identifier types to avoid traversing.

A subscribe request is like a search request, but creates a standing query. The MAP Server sends the initial search result promptly. Whenever the result subsequently changes, the MAP Server sends the change.

A poll request is part of the implementation of subscriptions. A MAP Client that has subscriptions makes a poll request to get the results. Poll is not used periodically, as might be expected from its name. Instead, the MAP Server replies to an outstanding poll request only when it has new subscription results for the MAP Client. Then the MAP Client sends a new poll request.

A purgePublisher request removes all metadata items that have a particular publisher-id – usually, the requester’s own publisher-id.

2.3 XACML

This specification builds on eXtensible Access Control Markup Language (XACML) Version 3.0 [4]. The reader should be familiar with at least Section 2 of that document.

XACML and the TNC Architecture [1] have two overlapping terms.

Term	TNC Definition	XACML Definition
PDP **(Policy Decision Point)**	Compares an Access Requestor’s “credentials (e.g. user certificates, password, etc) and information about its security posture against certain network access policies, and then decides whether network access should be granted to the AR.” (See [1], section 3.)	Evaluates applicable policy and renders an Authorization Decision (adapted from [4], section 1.1.1).
PEP **(Policy Enforcement Point)**	Receives a decision from a PDP and “grants or denies access (i.e. enforces access control).” (See [1], section 3.)	Performs access control, by making Decision Requests and enforcing Authorization Decisions (adapted from [4], section 1.1.1).

In this specification, the MAP Server is the XACML PEP. The figure illustrates the roles of the two kinds of PDPs.

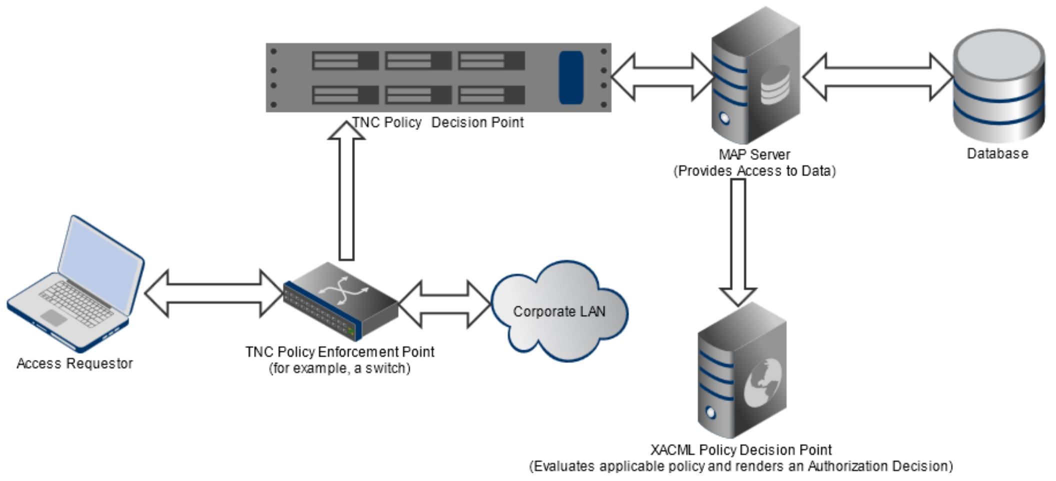

TNC Policy Decision Point
MAP Server
(Provides Access to Data)
Database
Access Requestor
TNC Policy Enforcement Point
(for example, a switch)
Corporate LAN
XACML Policy Decision Point
(Evaluates applicable policy and renders an Authorization Decision)

Figure 1 Two Kinds of PDPs

2.4 Use Cases

In every use case, a decision made by an authorization administrator ("Administrator") limits what MAP Clients can do, and so limits the harm a compromised client can do.

2.4.1 Administrator Specifies MAP Clients’ Roles and Tasks

An Administrator assigns roles and tasks to a single MAP Client or to a group of MAP Clients.

The TNC MAP Content Authorization specification provides a set of predefined roles that address common scenarios encountered by network or industrial control system administrators. Other roles can be added by implementers, Administrators, and other specifications. Tasks are more specialized than roles, and allow a MAP Client limited access.

This use case is addressed by the ifmap:client-role-administrator role (section 3.8.1) and by section 3.8.2.

2.4.1.1 New MAP Client Roles Are Defined

An implementer or Administrator defines MAP Client roles for purposes not covered in the predefined roles.

2.4.1.2 New MAP Client Tasks Are Defined for ICS Security

This is done by the overall Administrator, the ICS Administrator, and Overlay Manager roles.

For ICS security [11]:

- An ICS Administrator defines an Overlay Manager task for each overlay network.
- An ICS Administrator defines an overlay member task for each overlay network.

These definitions are straightforward to create; the task names simply have to follow patterns found in [19]. As a result, such task definitions don't require specialized administration expertise.

This use case is addressed by [19].

2.4.1.3 New MAP Client Roles Are Defined for Sensors of Specified Routing Domains

Consider an enterprise that has two or more IP routing domains, with security boundaries between them, most likely enforced by Flow Controllers. For example, the enterprise could be a public or private cloud provider that offers Infrastructure as a Service (IaaS) and leases VLANs to its cloud customers. In such a scenario, it is worthwhile to prevent a MAP Client from accessing ip-address identifiers of routing domains with which the MAP Client is not involved. Such limits can keep a rogue or malfunctioning Sensor from publishing false reports about IP addresses in another routing domain and thereby denying service.

A Sensor that monitors a routing domain can be assigned a Sensor role associated with that routing domain. Having the routing-domain-specific role will permit the Sensor to attach metadata around ip-address identifiers whose administrative-domain indicates the routing domain.

This use case is addressed by the XACML attribute identifier-attribute:ip-address:administrative-domain (section 3.4.3) and by [18].

This use case is a special case of 2.4.3.

2.4.2 Administrator Configures a MAP Client that Uses Metadata Types Unknown to the Authorization Policy

MAP Clients may seek to employ metadata types (such as vendor-specific metadata) for which no authorization policy is configured. An Administrator configures the authorization implementation to control whether MAP Clients may use metadata types unknown to the authorization policy.

This avoids the following undesirable scenario:

1. An Administrator turns on MAP content authorization.
2. All of the MAP Clients from vendor X stop working, because they use a metadata type unknown to the authorization policy.

This use case is addressed by normative text in section 3.9.10, Treatment of NotApplicable Decisions.

2.4.3 Administrator Limits Access to Some Administrative Domains

An Administrator specifies which roles can change metadata associated with a particular administrative domain.

Examples:

- An Administrator specifies that only MAP Clients with role abc may change metadata associated with IP addresses that have administrative-domain xyz.

  Such restrictions can help protect a multi-tenant infrastructure, especially one with leased VLANs. Isolation between tenants is paramount, so the fewer devices that can breach the isolation, the better.

  Such restrictions are also important for ICS security.
- A sensor that works with a subset of the IP routing domains needs only to publish to ip-address identifiers whose administrative-domain corresponds to one of those IP routing domains. A rogue sensor could do less harm if it were limited accordingly. (This assumes that IP administrative domains are paired with layer-3 routing domains, which may not be true for all environments.)
- A sensor that works with a subset of the layer-2 broadcast domains needs only to publish to mac-address identifiers whose administrative-domain corresponds to one of those layer-2 broadcast domains. (Similar assumption.)

To support this use case as conveniently as possible, the policy language should accept wildcards or regular expressions for administrative-domains. Fortunately, XACML does; string-regexp-match and string-starts-with are mandatory functions.

This use case is addressed by the administrative-domain XACML attributes (section 3.4.3). The administrator must create custom XACML rules to grant a custom role access to particular identifiers.

2.4.4 Implementer or Specification Offers Policy Recommendations

Whenever new metadata types or new use cases for existing metadata types are defined, whoever defines it – implementer, specification committee, or otherwise – should offer a threat analysis and recommend policy elements.

To do so requires some expertise in IF-MAP, XACML, threat analysis, and this TNC MAP Content Authorization specification.

This use case is addressed by section 3.10, Policy Recommendations.

2.4.5 Administrator Adds Custom Rule to Let Roleless Client Act

There are applications, including the ESUKOM project introducing IF-MAP-based security measures for mobile devices[20], in which large numbers of MAP Clients publish information about themselves. In this use case the policy administrator adds custom policy elements to enable such clients to work without needing a role.

This use case is enabled by self identifiers. A XACML Rule can enable these clients to attach metadata to their self identifiers, and to adjacent links.

2.4.6 Administrator Adds Suggested Policy Elements

An Administrator adds suggested policy elements that result from use case 2.4.4.

Adding these elements needs to be straightforward, and should *not* require expertise in XACML, threat analysis, or this specification. Policy elements from different sources should not conflict, as that would put the Administrator in an untenable position. (Therefore the overall policy needs to have a regular structure into which elements from different sources can be inserted.) Section 3.10 helps prevent conflicts among policy elements.

It is hoped that MAP-aware XACML policy authoring tools will appear and will make this administrative task easier and safer.

2.4.7 Administrator Customizes Policy in Prescribed Ways

A suggested policy element may come with prescriptions for customizing it. See the CUSTOMIZATION POINTs in [17], [18], and [19].

Such customizing should be straightforward, and should not require expertise in XACML, threat analysis, or this specification.

It is hoped that MAP-aware policy authoring tools will appear and will help make this administrative task easier and safer.

2.4.8 Administrator Customizes Policy in Novel Ways

An Administrator or a MAP-aware policy authoring tool customizes the XACML policy to handle use cases or security requirements not covered by the policy suggestions TCG provides or created by use case 2.4.4.

An example is defining roles that can grant only a subset of all possible roles. (See 3.8.2.)

The Administrator or PAP implementer needs expertise in XACML, in threat analysis, and in this specification.

A custom policy is expressed in the same policy language as the suggested policies, and receives the XACML attributes required to meet the other use cases.

2.4.9 Administrator Establishes Trust in a Certification Authority

A trust anchor is a CA certificate that can anchor a certificate chain used to authenticate a MAP Client.

In particular:

- An Administrator may establish trust in an enterprise's root CA. This enables certificate-based authentication of the enterprise's MAP Clients.
- An Administrator may establish trust in a manufacturer's CA. This enables use of factory-installed client certificates.
- An Administrator may establish trust in a cloud provider's CA. This enables some of the cloud provider's devices to be clients of the cloud customer's MAP Server.

Therefore it is desirable for a MAP Server to offer built-in support for multiple trust anchors.

This use case is addressed by section 3.9.6, Trusted Certification Authorities.

2.4.10 Administrator Reviews Access Attempts

An Administrator reviews authorization logs to determine the decision made about each IF-MAP Operation, and possibly the results of each XACML Decision Request. Accounting, also known as auditability, is essential to an authorization system. Auditability and authorization are complementary means of keeping MAP Clients from misbehaving.

Examples:

- A valid MAP Client is blocked from performing an operation it legitimately needs to execute; the Administrator discovers why it is blocked and unblocks it without opening up more access than is warranted.

- A rogue client probing for holes in the authorization policy, or a malfunctioning or misconfigured client, generates multiple access denials. An Administrator is able to identify the misbehaving client and address the underlying issue.

Figuring that out may be complex, offering much potential for implementations to add value in this area.

This use case is addressed by section 3.14, Logging.

2.4.11 Policy Stops MAP Client from Misplacing a Metadata Type

For example, [10] specifies:

Clients *MUST publish [access-request-device] only between: access-request and device*

A MAP Server is not required to enforce that rule. MAP Servers cannot be expected to know the rules for particular metadata.

An authorization policy, on the other hand, can enforce a rule like that.

Enforcement is desirable. Malformed data can be an attack vector and can cause problems even without an attack. A MAP Client might cause trouble by putting a metadata type on the wrong type of identifier.

This use case is addressed by the suggested policy elements. For a policy author who departs from those recommendations, the suggested policy elements show how to use the XACML attributes in section 3.4, Attributes of a XACML Decision Request, to enforce correct metadata placement.

2.5 Non-supported Use Cases

Several use cases—including, but not limited to, the following—are not covered by this version of TNC MAP Content Authorization, but may be addressed in a future version.

2.5.1 Block Acting at Inappropriate Times

A MAP Client may need to perform particular actions only at particular times of the day and week, or only during a particular range of dates. An attacker may tend to strike at off hours when less likely to be noticed. Many authorization systems include date-and-time restrictions.

This document does not explicitly address such a use case. Administrators may be able to create a custom policy to do so; an XACML PDP automatically supplies a time attribute and comes with support for time-based restrictions. See 3.6.3.1.

2.5.2 Block Weakly Authenticated Clients from Certain Operations

Some MAP Clients may use basic authentication, while others may use certificate authentication. Basic authentication is usually easier to defeat. Sensitive operations may deserve the extra security of certificate authentication.

2.5.3 Block Trespassing in an IP-Address Range

An Administrator may desire to limit which roles can publish which metadata types on ip-address identifiers with which combinations of administrative-domain and ip-address range, or on adjacent links. This document does not explicitly address such a use case; however, Administrators may be able to create a custom policy to do so.

2.5.4 Federated Authorization

An Administrator may desire to create policies that depend on attributes from multiple sources in the same decision, or the policy itself may need to be determined dynamically.

2.5.5 Compliance with Regulations

An Administrator may desire policies designed to ensure compliance with regulations such as Sarbanes-Oxley, FIPS, HIPPA, etc. This document does not explicitly address such a use case; however, Administrators may be able to create a custom policy to do so.

2.6 Requirements

Here are the requirements that TNC MAP Content Authorization must meet in order to successfully implement the use cases listed in section 2.4, above. These are stated as general requirements, with specific requirements called out as appropriate.

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

   The model must support all the functions and use cases of IF-MAP within the TNC architecture. It must allow MAP Clients to follow all use cases, examples, and usage guidelines from other current TNC IF-MAP specifications, while quietly blocking threats.

2. **Secure**

   TNC MAP Content Authorization must protect the *integrity* of the MAP content, by limiting what each MAP Client can publish and delete, and must protect the *confidentiality* of the MAP content, by limiting the information each MAP Client can retrieve.

3. **Extensible**

   TNC MAP Content Authorization needs to expand over time as new features and supported network, message, and authentication technologies are added to the TNC architecture. The authorization model must allow new features to be added easily, providing for a smooth transition and allowing newer and older architectural components to work together. The model must also be extensible by vendors. It must support IF-MAP extensibility by working with metadata types unknown to the authorization policy.

4. **Easy to use**

   TNC MAP Content Authorization must be reasonably easy to administer. The authorization model must operate on roles, each MAP Client having a set of roles. There should be a way to derive these roles from external authorities such as LDAP or certification authorities (CAs).

   Policy elements must be portable: they must be usable with different MAP Server implementations.

5. **Easy to implement**

   TNC MAP Content Authorization must be easy for MAP Server vendors to implement. Implementers must be able to use off-the-shelf software as a major ingredient.

6. **Unambiguous**

   TNC MAP Content Authorization must be clear and unambiguous about the policy language and its effects.

7. **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. Within a few years, MAP Servers may be required to serve millions of networking devices. Therefore, the model must not place implementation burdens on servers that would prevent scaling to meet the demands of a large organization.

8. **Standards-based**

   TNC MAP Content Authorization must leverage existing standards wherever possible, such as the widely used policy language XACML [4].

2.7 Non-Requirements

There are certain requirements that this version of TNC MAP Content Authorization explicitly is not required to meet. This list may not be exhaustive (complete).

- There is no new requirement to protect the *availability* of the MAP Server and its content.
- In particular, there is no requirement to limit which clients can make subscribe requests, though these could overload some MAP Servers.
- There is no requirement to specify the format or mechanism of the log.
- This specification does not deal with policy migration in case a metadata schema is revised. The new metadata schema would have a new name space. The authorization system would treat the new schema's metadata types as unrelated to their counterparts in the old metadata schema.

2.8 Assumptions

Here are the assumptions that TNC MAP Content Authorization makes about other components in the TNC architecture.

The TNC-specified uses of IF-MAP are found in:
- o TNC IF-MAP Metadata for Network Security, version 1.1[10]
- o IF-MAP Metadata for ICS Security[[11]

So, if MAP Content Authorization supports the functions and use cases of the above specifications, it meets Requirement 1, above. As these specifications are revised and updated, and as other TNC specifications define other ways of using IF-MAP, work will be needed on suggested policy elements and on this specification.

3 Authorization Mechanism

TNC MAP Content Authorization specifies a mechanism for authorizing MAP Client operations based on the following:

- The MAP Client's roles and tasks (a client can have zero or many)
- The metadata type (such as ip-mac)
- The identifier type (such as ip-address)
- Top-level attributes of the identifier, such as its administrative-domain
- Top-level attributes of the metadata item
- The action to be performed (read, write, finer-grained actions)

Each MAP Client is associated with a self identifier, to which it may have special rights.

purgePublisher is special: a MAP Client may be blocked from doing it altogether, allowed to purge its own metadata (typical), or allowed to purge other MAP Clients' metadata as well.

3.1 How to Read This Specification

Normative uses of "the MAP Server" or "a MAP Server" mean a MAP Server that complies with this specification. Not all MAP Servers need to comply with this specification.

3.2 Architectural Model

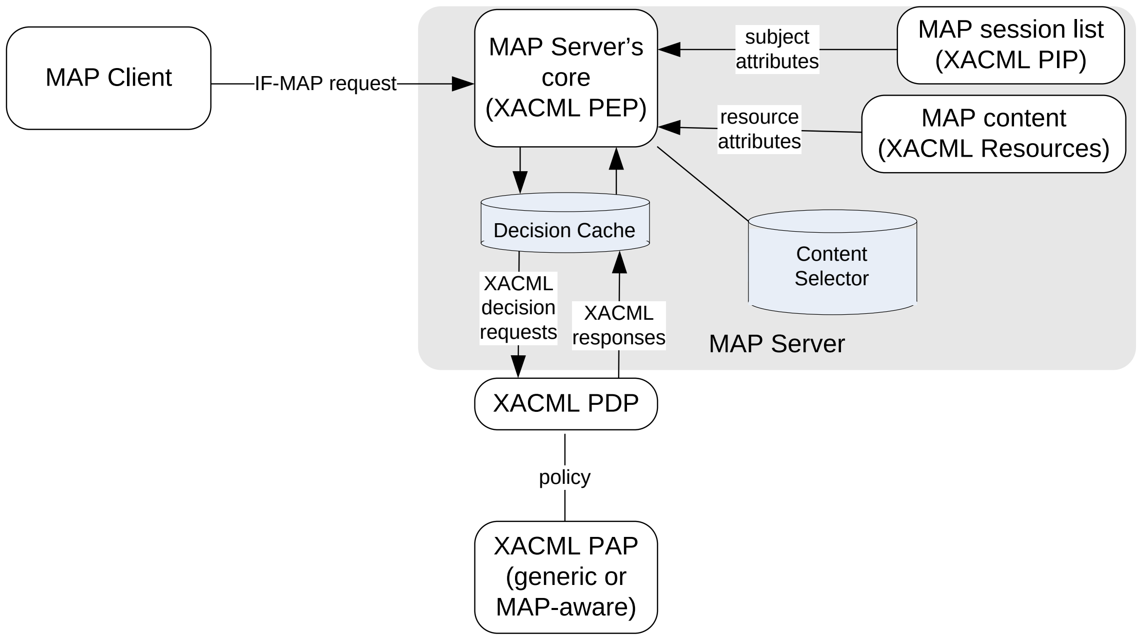


Figure 2 Architectural Model

The MAP Server authenticates each MAP Client (according to [3], Section 6.3.1) and determines its roles and tasks (see section 3.8). These roles and tasks are passed as attributes of each XACML Decision Request.

This specification obeys a locality rule. In preparing a XACML Decision Request, the MAP Server never needs to retrieve extra information from the MAP store. This rule helps MAP Servers achieve good performance.

3.3 XACML Decision Requests

In processing an IF-MAP request or a subscribed search, the MAP Server MUST send XACML Decision Requests and MUST obey the Authorization Decisions, as specified in this section, if authorization is enabled (see Section 3.9.1).

Further, if dry-run is enabled (section 3.9.2), the MAP Server MUST send Decision Requests as specified in this section – but with "true" as the value of the dry-run attribute. Therefore, if authorization and dry-run are both enabled, the MAP Server MUST send two sets of Decision Requests, identical except for the dry-run attribute. For a dry-run Decision Request, the Authorization Decision MUST NOT affect the MAP store or what is returned to the MAP Client. Rather, dry-run Authorization Decisions affect only logging (see section 3.9.2).

A search request, a subscription, or a publish request acts on a (possibly empty) set of metadata items. For each metadata item in that set the MAP Server MUST send a XACML Decision Request as detailed below.

3.3.1 Decision Request for a purgePublisher Request

For a purgePublisher request, the MAP Server MUST send one XACML Decision Request (see Section 3.4). To *obey* the Authorization Decision means: if the Authorization Decision is not "Permit", to deny the purgePublisher request and to send an AccessDenied errorResult (see [3], section 3.6.1).

3.3.2 Decision Requests for a Search Request or a Subscription

The set of metadata items acted on consists of the items that the search request or the subscribed search would *match,* or would *include in its result,* if authorized.

Each Decision Request MUST contain resource attributes derived, according to section 3.4, from the identifier(s) and metadata item.

To *obey* the Authorization Decision means: if the Authorization Decision is not "Permit", to censor the metadata item. (For more information about censoring, see [3], section 3.7.2.8, Search Algorithm.)

For example, consider a search whose starting ip-address is 192.168.0.1, and whose match-links filter includes "ip-mac". Assume ip-mac metadata is attached to a link from 192.168.0.1 to mac-address identifier aa:bb:cc:dd:ee:ff.

The search mechanism would generate a Decision Request with the following resource attributes:

Meaning	Value(s)	XACML Attribute ID
Metadata type	ip-mac	urn:oasis:names:tc:xacml:3.0:if-map:content: resource:**metadata-type**
Identifier type	ip-address mac-address	urn:oasis:names:tc:xacml:3.0:if-map:content: resource:**identifier-type**
Value attribute of ip-address identifier, if needed	192.168.0.1	urn:oasis:names:tc:xacml:3.0:if-map:content: resource:identifier-attribute:**ip-address:value**
Value attribute of mac-address identifier, if needed	aa:bb:cc:dd:ee:ff	urn:oasis:names:tc:xacml:3.0:if-map:content: resource:identifier-attribute:**mac-address:value**

Only if the decision was Permit would the ip-mac link be added to the widening search result. Otherwise, the search mechanism would behave as if the ip-mac metadata were not present.

If a metadata item matches a search's match-links filter (or there is no match-links filter), then the presence of that metadata item can bring in more search results. But if the metadata item is censored, it does not bring in any additional search results. Therefore, the search mechanism may need to generate Decision Requests for link metadata that matches the match-links filter as it goes, and refrain from following censored links.

In addition, the search mechanism needs to generate a XACML Decision Request for each other metadata item that would, if permitted, be part of the search result. These Decision Requests can be generated more lazily. The result-filter may be applied before or after these XACML Decision Requests are made.

Inevitably, a subscription triggers asynchronous XACML Decision Requests as necessary whenever the inputs that affect the query results change. These inputs include parts of the metadata graph, the XACML policy, and the MAP Client's roles. See section 3.6.1, Reevaluation of Subscriptions when Inputs Change.

3.3.3 Decision Requests for a Publish Request

The set of metadata items acted on consists of:

- Each metadata item found in an update or notify element
- Each metadata item found in the store that a delete element will delete, if authorized to
- Each single-valued metadata item found in the store that an update element will implicitly delete, if authorized to

Each Decision Request MUST contain resource attributes derived, according to Section 3.4, from the identifier(s) concerned and the metadata item.

If an update operation would replace existing (single-valued) metadata, the MAP Server MUST send an additional XACML request about the implicit deletion of the existing metadata item. This request MUST be the same one as for an explicit delete operation. For example, publish-request-subtype MUST be "delete" (see section 3.4.4).

To *obey* the Authorization Decisions means: if any Authorization Decision is not "Permit", to deny the publish request and to send an AccessDenied errorResult (see [3], section 3.6.1).

3.3.4 Decision Requests for a Poll Request

A MAP Server SHOULD NOT send XACML Decision Requests for an IF-MAP poll request.

Rather, the subscription triggers XACML Decision Requests. Those XACML decisions can cause the results to be censored. The poll request merely retrieves possibly censored subscription results.

3.3.5 Skippability

The MAP Server MAY skip sending one of the aforementioned XACML Decision Requests if the Authorization Decision would not affect the MAP database and would not affect the IF-MAP response sent to the MAP Client.

For example, during a search, if a client is not permitted to read a particular item of metadata attached to a link, this may disconnect part of the search-result subgraph. There is no need to make XACML Decision Requests about items the search cannot reach.

3.4 Attributes of a XACML Decision Request

This section specifies the XACML attributes that a MAP Server puts into a XACML Decision Request.

3.4.1 Subject Attributes

3.4.1.1 Role

The identifier type shall be designated with the attribute identifier urn:oasis:names:tc:xacml:3.0:if-map:content:subject:**role**. The values MUST have the type http://www.w3.org/2001/XMLSchema#string and MUST be the roles assigned to the MAP Client's session. If the session has no roles this attribute MUST be omitted altogether. (See section 3.8, Client Roles.)

3.4.1.2 Task

The identifier types shall be designated with attribute identifiers of the form urn:oasis:names:tc:xacml:3.0:if-map:content:subject:task:***RELATIONSHIP***:***IDENTIFIER-TYPE***. The values MUST have the type http://www.w3.org/2001/XMLSchema#string. These attributes and values MUST be the tasks assigned to the MAP Client. (See section 3.8, Client Roles.) ***RELATIONSHIP*** and ***IDENTIFIER-TYPE*** both MUST be URL-encoded[9].

For example, an attribute identifier can be urn:oasis:names:tc:xacml:3.0:if-map:content:subject:task:member-of:http%3A//www.trustedcomputinggroup.org/2010/IFMAP-ICS-METADATA/1#overlay-network-group, and the value can be Overlay_1.

%3A is the URL encoding of a colon.

3.4.2 Environmental Attributes

This identifier shall be designated with the attribute identifier urn:oasis:names:tc:xacml:3.0:if-map:content:environment:**dry-run**. The attribute MUST be a singleton (a bag of one). It MUST be present. The value MUST have type http://www.w3.org/2001/XMLSchema#boolean.

For real decisions, this MUST have the value “false”. For dry-run decisions (see section 3.9.2), this MUST have the value “true”.

NOTE TO POLICY AUTHORS: The dry-run XACML Policy Set should be applicable only if this value is true. The real XACML Policy Set should be applicable only if this value is false.

3.4.3 Resource Attributes

The MAP Server MUST disregard identifier and metadata schemas when deriving Resource Attributes. For example, if an identifier’s schema provides a default value for a particular attribute, and if an actual identifier in an IF-MAP Request or the MAP Store lacks the attribute, then for purposes of this specification the attribute is absent.

For an IF-MAP *purgePublisher* request, the decision request MUST NOT include attributes defined in this section. This requirement overrides any conflicting requirements in the following subsections.

3.4.3.1 metadata-type

The metadata type shall be designated with the attribute identifier urn:oasis:names:tc:xacml:3.0:if-map:content:resource:**metadata-type.** This attribute MUST be a singleton and MUST be present. The value MUST have type http://www.w3.org/2001/XMLSchema#string and MUST have the form ***NAMESPACE***#***TYPE***. ***TYPE*** MUST be replaced with part of the top-level tag of the metadata item, the part to the right of the colon. For example, if the tag is ifmap:access-request-ip, then ***TYPE*** is replaced with access-request-ip. ***NAMESPACE*** MUST be replaced with the URI that corresponds to the part of the tag to the left of the colon, according to XML namespace resolution rules. For example, a publish request might contain an attribute xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/IFMAP_METADATA/2". Example value:

http://www.trustedcomputinggroup.org/2010/IFMAP-METADATA/2#access-request-ip

3.4.3.2 identifier-type

The identifier type(s) shall be designated with the attribute identifier urn:oasis:names:tc:xacml:3.0:if-map:content:resource:**identifier-type**. This attribute MUST be present. If the Decision Request concerns an identifier, the XACML attribute MUST be a singleton and MUST denote the type of the identifier. If the Decision Request concerns a link, the XACML attribute MUST have two values denoting the types of the two identifiers. However, it MUST have one value if the link is between two identifiers of the same identifier type**.** The value(s) MUST have type http://www.w3.org/2001/XMLSchema#string.

For a non-extended, original identifier type, the value MUST be the type string, such as ip-address or identity. For an extended identifier type, the value MUST have the form ***NAMESPACE***#***ELEMENT-NAME***. ***ELEMENT-NAME*** MUST be replaced with part of the top-level XML tag of the extended identifier – the part to the right of the colon. For example, if the top-level XML tag is ns:network, ***ELEMENT-NAME*** is replaced with network. ***NAMESPACE*** MUST be replaced with the URI that corresponds to the part of the tag to the left of the colon, according to XML namespace resolution rules. For example, the extended identifier might have an attribute xmlns:ns="http://www.example.com/extended-identifiers". Example value (corresponding to section 3.2.3.3 of [3]):

http://www.example.com/extended-identifiers#network

3.4.3.3 is-map-client-identifier

The attribute identifier shall be urn:oasis:names:tc:xacml:3.0:if-map:content:resource:**is-map-client-identifier**. It MUST be a singleton. The value MUST have type http://www.w3.org/2001/XMLSchema#boolean. It MUST have a value of true if and only if one or both of the identifiers the XACML requests concerns has the form of a MAP Client identifier, which means:

- The identifier is nonextended
- Its type is identity
- Its administrative-domain is ifmap:client. (See section 3.7.)

Otherwise it MUST be false or omitted altogether.

3.4.3.4 is-self-identifier

The attribute identifier shall be urn:oasis:names:tc:xacml:3.0:if-map:content:resource:**is-self-identifier.** The value MUST have type http://www.w3.org/2001/XMLSchema#boolean. It MUST have a value of *true* if and only if one or both of the identifiers the XACML request concerns is the subject MAP Client's self identifier (see section 3.7). Otherwise it MUST be false or omitted altogether.

3.4.3.5 on-link

The attribute identifier shall be urn:oasis:names:tc:xacml:3.0:if-map:content:resource:**on-link**. This attribute MUST be a singleton and MUST be present. The value MUST have type http://www.w3.org/2001/XMLSchema#boolean. MUST be true just if the metadata item is (or will be) attached to a link, false if to an identifier.

3.4.3.6 metadata-attribute

These attribute identifiers MUST have the form urn:oasis:names:tc:xacml:3.0:if-map:content:resource:**metadata-attribute:*ATTR***, where ***ATTR*** is the name of a top-level attribute of the metadata item. ***ATTR*** MUST be URL encoded[9]. An XACML attribute of this kind MUST be a singleton. The value MUST have type http://www.w3.org/2001/XMLSchema#string and MUST equal (byte for byte) the value of the metadata item's attribute.

This XACML attribute carries a top-level attribute of the metadata item. Therefore any top-level attribute of the metadata item can be a factor in Authorization Decisions.

A XACML attribute of this kind MUST NOT be present unless the metadata item has a top-level attribute named ***ATTR***.

If permitted by the foregoing paragraph, an attribute of this kind MUST be included if the Content Selector (section 3.9.3) selects it, and MAY be included otherwise. The XACML PDP MAY obtain these attributes from a XACML PIP. The PIP would have to obtain them from the MAP Server, by a mechanism not specified here.

Consider a metadata item:

<ics:overlay-policy name="Overlay_A" state="allow"/>

Here is a corresponding XACML attribute ID:

urn:oasis:names:tc:xacml:3.0:if-map:content:resource:metadata-attribute:**name**

Here is the corresponding XACML attribute value:

Overlay_A

3.4.3.7 identifier-attribute

These attribute identifiers MUST have the form urn:oasis:names:tc:xacml:3.0:if-map:content:resource:identifier-attribute:***IDENTIFIER-TYPE*:*ATTR***, where ***IDENTIFIER-TYPE*** is the type string (as defined in 3.4.3.2) of one of the identifiers the Decision Request concerns and ***ATTR*** is the name of a top-level attribute of that identifier. ***IDENTIFIER-TYPE*** and ***ATTR*** MUST be URL encoded[9].

The value of this XACML attribute MUST equal the value of the corresponding top-level attribute of the IF-MAP identifier. Therefore any top-level attribute of the identifier can be a factor in Authorization Decisions. Usually this XACML attribute of the Decision Request will be a singleton. However, if the Decision Request concerns a link between two identifiers of the same type and the identifiers both have the attribute ***ATTR***:

- If the values for ***ATTR*** are not equal, the XACML attribute of the Decision Request MUST have two values.
- If the values for ***ATTR*** are equal, the XACML attribute MUST have one value.

If the identifier's type is **ip-address** and ***ATTR*** is **value**, the XACML attribute's value MUST have type urn:oasis:names:tc:xacml:2.0:data-type:**ipAddress** and MUST be equivalent to the value of the ip-address identifier's value attribute. It MUST NOT include a network mask or portrange. The XACML attribute's value MUST be transformed, if necessary, to comply with the syntax of [4], section A.2.

If the identifier's type is identity, the subtype is distinguished-name, and ***ATTR*** is **name**, the XACML attribute's value MUST have type urn:oasis:names:tc:xacml:1.0:data-type:**x500Name** and MUST be equivalent, according to 3.2.2.3.1 of [3], to the value of the identity identifier's name attribute. Of course, the XACML attribute's value MUST comply with the syntax of [4], section A.2.

If the identifier's type is identity, the subtype is dns-name, and ***ATTR*** is **name**, the XACML attribute's value MUST have type urn:oasis:names:tc:xacml:2.0:data-type:**dnsName** and MUST equal (byte for byte) the value of the identity identifier's name attribute. It MUST NOT include a portrange or a wildcard.

In all other cases, the value MUST have type http://www.w3.org/2001/XMLSchema#string and MUST equal (byte for byte) the value of the identifier's attribute.

A XACML attribute of this kind MUST NOT be present unless the respective identifier has a top-level attribute named ***ATTR***, or ***ATTR*** is administrative-domain. If ***ATTR*** is administrative-domain and the identifier has no administrative-domain attribute, the value of the XACML attribute MUST be the empty string.

If permitted by the foregoing paragraph, an attribute of this kind MUST be included if the Content Selector (section 3.9.3) selects it, and MAY be included otherwise. The XACML PDP MAY obtain these attributes from a XACML PIP. The PIP would have to obtain them from the MAP Server, by a mechanism not specified here.

Consider an ip-address identifier:

<ip-address value="192.0.2.11" type="IPv4"/>

Here is a XACML attribute ID:

urn:oasis:names:tc:xacml:3.0:if-map:content:resource:attribute-identifier:**ip-address**:**value**

Here is the attribute value:

192.0.2.11

3.4.4 Action Attributes

- urn:oasis:names:tc:xacml:1.0:action:**action-id**. Indicates whether the access is a read or a write. This attribute MUST be a singleton, and MUST be present. If the IF-MAP request is search or subscribe, this attribute's value MUST be "read". Otherwise this attribute's value MUST be "write".

- urn:oasis:names:tc:xacml:3.0:if-map:content:action:**request-type**. This attribute MUST be a singleton and MUST be present. The value MUST have type http://www.w3.org/2001/XMLSchema#string and MUST be "publish", "subscribe", "search", or "purgePublisher", according to the request type of the IF-MAP request.

The following attribute MUST be present only if the IF-MAP request type is purgePublisher:

- urn:oasis:names:tc:xacml:3.0:if-map:content:action:**purge-own-metadata**. This attribute MUST be a singleton. The value MUST have type http://www.w3.org/2001/XMLSchema#boolean and MUST be true if and only if the publisher-id specified belongs to the subject client.

  Otherwise the value MUST be false or omitted altogether.

The following attribute MUST be present if and only if the IF-MAP request type is publish:

- urn:oasis:names:tc:xacml:3.0:if-map:content:action:**publish-request-subtype**. This attribute MUST be a singleton. The value MUST have type http://www.w3.org/2001/XMLSchema#string and MUST be "update", "notify", or "delete", according to the publish request subtype.

3.4.5 Additional Attributes

A MAP Server MAY include additional attributes that do not have urn:oasis:names:tc:xacml:3.0:if-map:content: as the attribute prefix, following XACML extensibility principles (see [4]).

3.5 XACML Response

If the XACML PDP's decision is Permit, the response MAY contain, and the MAP Server MUST be able to handle, one or more <Obligation> elements that have:

- an ObligationId of "urn:oasis:names:tc:xacml:3.0:if-map:content:**obligation:caching**"

- exactly one urn:oasis:names:tc:xacml:3.0:if-map:content:obligation:**maximum-policy-lag** attribute, with type http://www.w3.org/2001/XMLSchema#integer and with a nonnegative value (otherwise the MAP Server MUST treat the decision as Deny)

- no other attributes (otherwise the MAP Server MUST treat the decision as Deny)

If such an obligation is present, the MAP Server MUST NOT cache the decision longer than the number of seconds specified by the attribute. If there are two or more such obligations, the shortest value prevails. If the configured maximum-policy-lag is shorter, that prevails. If the result is zero, the decision MUST be used only once and not cached at all. See 3.6.3.

The MAP Server SHOULD NOT understand any other XACML obligations, and (per [4]) if an Authorization Decision contains any obligations the MAP Server does not understand, the MAP Server MUST treat the Authorization Decision as a denial. The XACML policy SHOULD NOT send it any other obligations.

3.6 Interaction with XACML PDP

This section specifies some lower-level mechanics of the interaction between the MAP Server and the XACML PDP.

3.6.1 Reevaluation of Subscriptions when Inputs Change

When a change in inputs causes the results of an existing subscribed search to change, the subscription MUST be reevaluated. How long the reevaluation takes is implementation and load dependent. These inputs MUST include:

- the XACML policy elements
- the MAP Client's roles and tasks
- whether MAP Content Authorization is enabled (see section 3.9.1, Authorization Enabled)

The new inputs may allow a given MAP Client to read some additional metadata items, and/or may revoke its permission to read some. These changes may affect subscribed searches. For each subscribed search whose results the change affects, the MAP Server MUST send an updateResult and/or a deleteResult (section 3.7.5 of [3]) as if a single atomic operation had added all of the newly readable metadata and deleted the newly unreadable metadata.

However, if changes to the mentioned inputs occur in rapid succession, the MAP Server MAY skip evaluating some or all subscribed searches against interim sets of inputs.

3.6.2 Atomicity of Policy Changes

An IF-MAP request or searchResult calculation can be in progress when the inputs mentioned in section 3.6.1 change, and some Authorization Decisions might be made against the old inputs and some against the new. Therefore the outcome of the IF-MAP request or searchResult calculation could be inconsistent. This would violate the spirit of the atomicity principles in [3]. Such inconsistencies could cause MAP Clients to behave unpredictably after a change.

To prevent this, the MAP Server MUST ensure that throughout processing of an IF-MAP request, and throughout calculation of a searchResult:

- Every XACML response whose <PolicyIdentifierList> mentions a particular PolicyId or PolicySetId gives the same Version value for the Id.
- The other inputs mentioned in 3.6.1 are also constant.

The MAP Server MUST specify ReturnPolicyIdList="true" in all XACML Decision Requests.

If the XACML PDP does not implement the <PolicyIdentifierList> feature, the <PolicyIdentifierList> requirement is waived.

IF-MAP needs to remain stable over a change of the inputs mentioned in 3.6.1. Therefore, a change of these inputs MUST NOT cause a MAP Server to send any error results except for (a) denials of access that are consistent with either the old or the new set of inputs, and (b) error results that this specification or [3] explicitly calls for. For example, if a MAP Server starts processing an IF-MAP request and then notices a policy change, it may not simply reject the request. If an input changes in the middle, the MAP Server MUST repeat Decision Requests that employed a different value of the input, until a consistent set of decisions is achieved. Nevertheless, in case a XACML PDP does not conform to [16], the MAP Server MAY protect itself from an infinite loop or other untoward results by permitting unequal Version values.

Administrators are advised to change policies that affect IF-MAP at quiet times.

3.6.3 Decision Cache and Policy Lag

The MAP Server MAY maintain a Decision Cache, to reduce the number of XACML requests it sends to the XACML PDP. Whenever this specification says that the MAP Server "MUST" or "SHALL" send a Decision Request, the MAP Server MAY avoid that by getting from its Decision Cache an Authorization Decision that the XACML PDP rendered previously in response to an identical Decision Request.

However, if there has been a non-recent change of policy elements on the XACML PDP that would lead to a different Authorization Decision, the MAP Server MUST evict the cached decision (if any) and make an actual Decision Request. “Non-recent” means longer ago than a maximum-policy-lag specified by the policy (see section 3.5) or by the configuration (see section 3.9.4).

A simple way to satisfy the foregoing paragraph is to discard each Authorization Decision from the Decision Cache after its maximum-policy-lag.

From a policy change until maximum-policy-lag later, interleaving of outcomes based on the old policy elements and outcomes based on the new is a possibility and could cause many problems. To minimize these problems, a MAP Server MUST associate each cached decision with the <PolicyIdentifierList> element returned by the XACML PDP for the decision. Whenever the XACML PDP returns a <PolicyIdentifierList> containing a Version value different from that found in a cached <PolicyIdentiferList> element, the MAP Server MUST discard the cached decision. If the XACML PDP does not implement the <PolicyIdentifierList> feature, this requirement is waived.

The format of the information in the Decision Cache, and between the MAP Server’s core and the Decision Cache, is an implementation matter and is not required to be XACML compliant.

3.6.3.1 Time-Sensitive Policies

Policy authors need to be aware of some peculiarities that can happen with XACML policies that are sensitive to the current date or time. Such policies typically mention one or more of these XACML attributes:

- urn:oasis:names:tc:xacml:1.0:environment:current-time
- urn:oasis:names:tc:xacml:1.0:environment:current-date
- urn:oasis:names:tc:xacml:1.0:environment:current-dateTime

If the XACML PDP contains a policy element that applies only at certain dates and times, that is in many ways equivalent to having a new policy installed at each time boundary. However, the MAP Server cannot be expected to detect the effective policy change. Therefore:

- Between the time boundary and max-policy-lag later, the MAP Server’s decision cache may contain some decisions made before the time boundary and others made after the time boundary. Results for a particular IF-MAP request or searchResult calculation may not be consistent (as defined above).
- Subscribed searches will not generally be reevaluated at the policy’s time boundaries.

It is a good idea to have the time boundaries occur at quiet times, if possible.

It may be helpful for time-sensitive Rules to specify a short maximum-policy-lag (see 3.5).

3.6.4 PolicyIds, PolicySetIds, and Versions

The consistency method described above depends on disciplined use of PolicyIds/PolicySetIds and Version values. On a given XACML PDP, a PolicyId-Version combination or a PolicySetId-Version combination MUST NOT be reused. In other words, if two Policies/PolicySets differ (outside of comments, spacing, and other cosmetic differences) the two MUST have different PolicyIds/PolicySetIds and/or different Version values. The XACML PDP and/or the XACML administrator are responsible for complying with this rule. Mechanisms for ensuring compliance are outside the scope of this specification.

TCG-suggested policy elements MAY be used exactly. However, if they are changed at all by a party other than TCG, the party MUST assign a new PolicyId/PolicySetId. Similarly, an original Policy/PolicyId MUST have a new PolicyId/PolicySetId.

The new ID must be a URI that the policy author is entitled to use.

However, not every policy author owns a portion of the URI space. For convenience and to encourage disciplined policy identification, a PolicyId/PolicySetId intended for use with MAP Content Authorization MAY have the following form:

urn:oasis:names:tc:xacml:3.0:if-map:content:other-policy:ENTERPRISE-ID:TAIL

ENTERPRISE-ID MUST be replaced with an SMI Private Enterprise Number (see [8]) owned by, or rightfully usable by, the policy author. TAIL MAY be replaced with any string, so long as the resulting ID complies with XACML requirements.

3.6.5 Content Selector

The configurable Content Selector governs which parts of a metadata item or an identifier are turned into XACML attributes and sent to the XACML PDP. See section 3.9.3, Content Selector, for details.

To ease configuration, the MAP Server MAY turn other parts of metadata items and identifiers into XACML attributes, as if the Content Selector specified them. In particular, the MAP Server SHOULD do so in reaction to <MissingAttributeDetail> elements received from the XACML PDP. (See [4], section 7.19.3.)

3.6.6 XACML Multiple Decision Profile

The MAP Server MAY take advantage of the XACML Multiple Decision Profile [5] if the XACML PDP implements it. The MAP Server SHOULD NOT require a XACML PDP that implements this profile. (See 3.9.11, Whether to Use XACML Multiple Decision Profile.)

3.6.7 Communication with XACML PDP

The communication between the MAP Server and the XACML PDP is implementation dependent. For example, the XACML PDP MAY be a library that the MAP Server calls. Or the XACML PDP MAY be a separate network node.

The MAP Server MAY:

- Send and receive Decision Requests and Authorization Decisions using Security Assertion Markup Language (SAML), specifically using XACMLAuthzDecisionQuery and XACMLAuthzDecision, in conformance with section 4 of [13] (or a superseding version of the same specification).
- Exchange these SAML elements over SOAP, in conformance with the SAML SOAP binding (section 3.2 of [14]).
- Send and receive the SOAP messages over HTTPS[15].

The MAP Server MUST ensure that the XACML PDP is authentic and is an authorized and trusted XACML PDP for this MAP Server. The MAP Server MUST also enable the XACML PDP to ensure that the MAP Server is authentic and trusted. For XACML requests and responses, the MAP Server MUST be able to enforce integrity protection, confidentiality protection, and replay protection. The MAP Server SHOULD enforce all of these by default. Certificate-based HTTPS can provide all of these protections.

3.6.8 Behavior when XACML PDP Is Unavailable

The XACML PDP may sometimes be unavailable. It may be unreachable, unresponsive, or unacceptably slow; authentication between it and the MAP Server may fail; there may be other causes of unavailability.

A MAP Server MAY be configurable to fail over one or a sequence of alternate XACML PDPs in this event. For best results, administrators should put identical policy elements into the alternates. However, if the MAP Server fails over to a XACML PDP that has policy differences, the MAP Server MUST treat this as a policy change (see 3.6.3).

When no configured XACML PDP is available and authorization is enabled (see 3.9.1), the MAP Server MUST reject all purgePublisher, publish, search requests, and new and pending poll requests with a SystemError errorResult.

3.7 The Self Identifier

For every MAP Client, a certain identifier is considered the *self identifier*. This identifier is an identity identifier and has administrative-domain “ifmap:client”. The administrative-domain “ifmap:client”, when applied to identity identifiers, is reserved for self identifiers and MUST NOT be used on other identity identifiers.

If the MAP Client was authenticated using certificate-based authentication, the self identifier has IdentityType distinguished-name, and the identifier’s name attribute is equivalent (according to section 3.2.2.3.1, Distinguished Names, of [3]) to the client’s certificate’s Subject field. If the MAP Client was authenticated using basic authentication, the self identifier has IdentityType username, and the identifier’s name attribute equals (byte for byte) the client’s userid.

The client has enough information to determine its self identifier. Administrative clients also can determine them.

3.8 Client Roles and Tasks

A MAP Server MUST associate a set of roles and a set of tasks with each MAP Client session.

By convention, roles are broader. A role lets a MAP Client perform particular actions on particular metadata types when they are attached to particular identifier and link types. The types are the key. But the *attributes* of the metadata and identifiers do not generally enter into it.

Tasks are more focused. A task gives a MAP Client particular access to a particular subset of the MAP graph. Attributes of the metadata and/or identifiers, such as their administrative-domain attributes, generally *do* enter into it.

3.8.1 Predefined Roles

The real and dry-run policies SHOULD give the specified access to each of the following administrative roles.

ifmap:all-reader	The MAP Client may read any metadata. The MAP Client may be a sensor that watches for misbehavior by other MAP Clients.
ifmap:all-writer	The MAP Client may read and write any metadata.
ifmap:client-role-administrator	The MAP Client assigns roles to MAP Clients.

Other predefined roles are found in [18] and [19].

Role names that begin with “ifmap:” or “tcg:” are reserved and MUST NOT be used except according to TCG specifications.

A URN MAY be used as a role name. Vendors and other specifications defining general-purpose role names SHOULD use Universal Resource Names (URNs) they own (see [6]), to avoid conflicts.

Administrators are advised to create role names that either are URNs they own or do not begin with “urn:” or with the reserved beginnings.

3.8.2 How Roles Are Granted

For an enterprise with many MAP Clients, roles need to be granted with labor efficiency. At the same time, roles must be granted accurately, so that MAP Clients can function and security can be maintained. Several methods are available. This section provides an overview; following sections provide normative details.

1. Some clients may not need roles. A policy author may permit even a role-less client to do certain things. (See 2.4.5.)
2. An IF-MAP administrator may grant roles using the MAP Server's implementation dependent administrative interface.
3. A MAP Client may have roles by virtue of its certificate chain. If an enterprise enrolls devices into its Public Key Infrastructure (PKI), many MAP Clients may get their roles without further administrative steps.(See 3.8.5.)
4. A MAP Client may get roles because its identity belongs to an LDAP group. If an enterprise puts devices into LDAP groups anyway, a MAP Client's roles may come along for free. (See 3.8.4.)
5. A MAP Client may publish a description of itself or another MAP Client. An Administrative MAP Client may respond by granting a role, partly or completely automatically. (See 3.8.3.3.)

Policies govern role granting by Administrative MAP Clients. Policies can specify:

- which roles may grant roles
- which roles each role may grant
- to which clients each role may grant roles

Under the suggested policies, a client with role ifmap:client-role-administrator can grant any role to any MAP Client. (See [17].) A client with the role tcg:ics:administrator can grant only Industrial Control System (ICS) roles. A client that manages a particular ICS overlay network can grant only the overlay network's membership role. (See [19].)

3.8.3 Roles Derived from Metadata

3.8.3.1 ifmap-client-role Identifier Type

ifmap-client-role is an extended identifier type, containing a single name attribute. The attribute's value is the name of a role.

The schema for this identifier type follows:

```
   <xsd:element name="ifmap-client-role">
     <xsd:complexType>
       <xsd:complexContent>
         <xsd:extension base="base-id:IdentifierType">
            <xsd:attribute name="name" type="xsd:string" use="required"/>
         </xsd:extension>
       </xsd:complexContent>
     </xsd:complexType>
   </xsd:element>
```

In a publish request, the MAP Client MUST set the administrative-domain of an ifmap-client-role identifier to the empty string. (A non-empty administrative-domain is reserved for possible future use.) The name MUST be a nonempty string.

Because name values that begin with "tcg:" or "ifmap:" are reserved (see section 3.8.1), a MAP Client MUST NOT mention such a name in a publish request unless the name is defined by this or another TCG specification.

3.8.3.2 ifmap-client-has-role Metadata Type

ifmap-client-has-role is a simple, single-valued metadata type, which MAP Clients MUST attach only to links between non-extended identity identifiers with administrative-domain ifmap:client and ifmap-client-role identifiers. The presence of the link means that sessions of the specified MAP Client are to receive the specified role. By updating these links, an administrative MAP Client grants and revokes roles of MAP Clients.

The schema for this metadata type follows:

```
    <xsd:element name="ifmap-client-has-role">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="SingleValueMetadataType”>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>
```

This type is meant to be attached by authorized Administrative Clients, and to be read by the affected client, by authorized Administrative Clients, and by other MAP Clients with need to know.

The roles of the MAP Client’s session MUST correspond to the links that have ifmap-client-has-role metadata attached and go between the MAP Client’s self identifier (see section 3.7) and ifmap-client-role identifiers. If those links change during the session, the session’s roles MUST change as well.

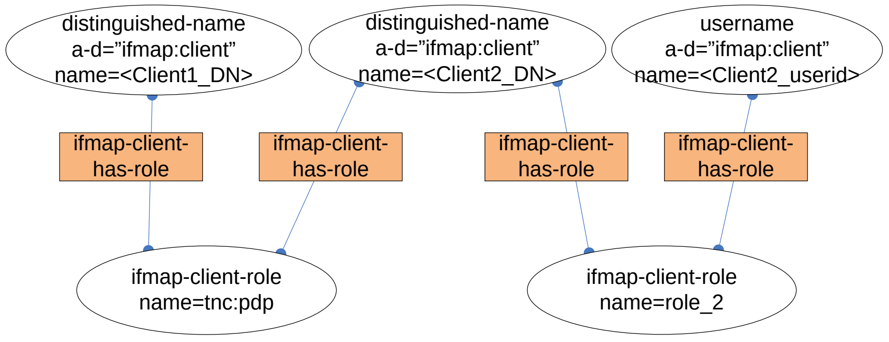


Figure 3: Example using ifmap-client-has-role metadata type

A role assigned to a MAP Client’s session MUST be a string, and MUST equal the value of the name attribute of one of these linked ifmap-client-role identifiers.

The policy in force SHOULD permit a MAP Client to read ifmap-client-has-role metadata attached to links adjacent to the MAP Client’s self identifier. This gives the client a way to learn most of its roles.

In case many thousands of MAP Clients have a particular role, an administrative MAP Client that queries ifmap-client-has-role metadata SHOULD perform searches and subscriptions with a max-size larger than the default.

The MAP Server SHOULD provide an Administrator an additional way to view and modify ifmap-client-has-role metadata, other than IF-MAP requests. This alternative is valuable for configuring an initial MAP Client that will manage this metadata, and in case the MAP Clients that manage this metadata lock themselves out by mistake.

3.8.3.3 ifmap-client-has-task Metadata Type

ifmap-client-has-task is a multi-valued metadata type. MAP Clients MUST attach it only at a valid attachment point, which is a link between:

a) A non-extended identity identifier with administrative-domain ifmap:client and
b) Another identifier that has:
    a. A nonempty name attribute

b. An empty or no administrative-domain attribute
c. No other attributes
d. No child elements

The schema for this metadata type follows:

```
    <xsd:element name="ifmap-client-has-task">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="MultiValueMetadataType”>
            <xsd:sequence>
              <xsd:element name="relationship" type="xsd:string"
                minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>
```

This type is meant to be attached by authorized Administrative Clients, and to be read by the affected client, by authorized Administrative Clients, and by other MAP Clients with need to know.

A MAP Server MUST associate a task with a MAP Client for each valid ifmap-client-has-task metadata item attached to a link associated with the MAP Client’s self identifier. Each task has three notional parts:

- *relationship*, which equals the value of the relationship attribute of the ifmap-client-has-task metadata item
- *identifier type*, which is the type string for the non-self identifier, in the same notation as in section 3.4.3.2
- *name*, which equals the value of the name attribute of the non-self identifier

When expressing the task as a XACML attribute, the MAP Server MUST apply the notation in section 3.4.1.2.

If the ifmap-client-has-task links change during the MAP Client’s session, subsequent decision requests MUST change accordingly.

(The example that follows and Figure 4 draw from [19].) The resulting XACML attribute ID would be:

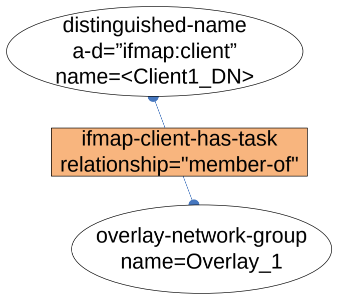


Figure 4 Example using ifmap-client-has-task metadata type

urn:oasis:names:tc:xacml:3.0:if-map:content:subject:task:**member-of**:
http%3A//www.trustedcomputinggroup.org/2010/IFMAP-ICS-METADATA/1#**overlay-network-group**

The XACML attribute would have the value “Overlay_1”.

The policy in force SHOULD permit a MAP Client to read ifmap-client-has-task metadata attached to links adjacent to the MAP Client’s self identifier. This gives the client a way to learn its tasks.

An administrative MAP Client that queries ifmap-client-has-task metadata SHOULD perform searches and subscriptions with a max-size larger than the default.

3.8.3.4 IF-MAP Client Catalog

ifmap-client-catalog is an extended identifier type and has only one instance.

A MAP Client SHOULD publish device-characteristic[10] metadata describing itself. Self-describing device-characteristic metadata MUST be attached only to a link between a MAP Client's self identifier and the ifmap-client-catalog identifier.

An authorized MAP Client MAY attach device-characteristic to *any* link between an identity identifier with administrative-domain ifmap:client and the ifmap-client-catalog identifier. For example, a Sensor could detect a new MAP Client, recognize its make and model, and publish the device-characteristic metadata. An Administrator would need to create a custom XACML rule to permit this.

On finding new device-characteristic metadata attached to a link adjacent to the ifmap-client-catalog identifier, an Administrative client SHOULD log the occurrence. Administrative MAP Clients MAY take this metadata as a hint that a client exists and what roles and tasks the client can perform. The calculation of roles and tasks is implementation dependent.

For the device-type attribute of device-characteristic metadata, in addition to the values found in [10], the following are hereby defined:

- network-sensor
- tcg-policy-decision-point
- tcg-policy-enforcement-point
- tcg-flow-controller
- ip-mac-linker
- tcg-ics-backhaul-interface
- tcg-ics-overlay-manager
- tcg-ics-administrator

Regarding the discoverer-id attribute of device-characteristic metadata, [10] states:

> A MAP Client SHOULD use its ifmap-publisher-id as its discoverer-id if it is only publishing metadata that it generated itself.

This pertains when a MAP Client publishes self-describing device-characteristic metadata.

For the discovery-method attribute of device-characteristic metadata, self-describing device-characteristic metadata MUST have the value "self-reported".

A MAP Client MAY publish device-characteristic metadata about itself unconditionally each time it establishes a session with the MAP Server. A MAP Client SHOULD observe the following:

> [When] publishing new multi-valued metadata, the MAP Client SHOULD, in the same publish request, remove any metadata with its ifmap-publisher-id that has gone stale or is no longer relevant.[10]

The policy in force SHOULD permit every MAP Client to read and write self-describing device-characteristic metadata. Nevertheless, when publishing self-describing device-characteristic metadata, a MAP Client MUST ignore an AccessDenied errorResult.

The schema for the identifier type follows:

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

In a publish request, the MAP Client MUST set the administrative-domain of an ifmap-client-catalog identifier to the empty string.

Because the identifier type has no child elements, no attributes except administrative-domain, and administrative-domain fixed as the empty string, there can be only one instance of the ifmap-client-catalog identifier.

3.8.4 Roles and Tasks Derived from LDAP Groups

An Administrative MAP Client known as an *LDAP relay* MAY cause ifmap-client-has-role links around a certain ifmap-client-role identifier, or ifmap-client-has-task links around a certain identifier, to reflect membership in an LDAP group. An LDAP relay would allow role and task assignments to be managed through LDAP.

This client MAY have the ifmap:client-role-administrator role. Alternately, this client MAY be allowed to assign only particular roles. (For a policy that lets a role assign only particular roles, see [19].)

For MAP Clients that use basic authentication, the LDAP relay MAY construct a DN from the basic authentication user ID and a configurable DN template. For example, a template could be:

C=US, O=Xyz Corp, OU=ifmap-clients, CN=%USERID%

When the LDAP relay found an LDAP object that matched the template, it would determine what groups the LDAP object belonged to and adjust the role links adjacent to the corresponding self identifier.

3.8.5 Roles Derived from Certificates

If the client is authenticated using certificate-based authentication, the MAP Client session can get additional role attributes based on the Subject DN of a certificate in the certificate chain. The MAP Server MUST be configurable with a mapping from a Subject DN to an enumerated set of role strings. If a MAP Client uses certificate-based authentication and the Subject DN of one of the certificates in the certificate chain matches a DN in the configured mapping, the MAP Server MUST assign the corresponding roles to the MAP Client's session. If more than one Subject DN matches, the MAP Server MUST use only the one closest to the leaf for this purpose.

Upon a change in the mapping, the roles of existing MAP Client sessions MUST be updated accordingly.

The MAP Server MUST compare DNs using the equivalence rules in section 3.2.2.3.1, Distinguished Names, of [3].

For example, if a manufacturer Abc includes a factory certificate in each of its MAP Clients, and an Administrator wants all MAP Clients by that manufacturer to have the tcg:sensor role, the Administrator might configure this:

C=US O=Abc CN=Root CA → { **tcg:sensor** }

3.9 Configuration Model

This section lists configuration choices that this specification suggests or mandates.

The method of applying these configuration choices (GUI, CLI, etc.) is implementation-dependent.

3.9.1 Authorization Enabled

A MAP Server MUST be configurable either to enforce authorization or not to do so. The default MAY be either. It SHOULD be able to switch authorization on or off without disruption (e.g., without a restart).

3.9.2 Dry-Run Policy Enabled

The MAP Server SHOULD be configurable to use a dry-run policy set. If so configured:

- The MAP Server MUST send XACML Decision Requests with the “true” as the value of the dry-run attribute (see section 3.3).
- If authorization is also enabled (section 3.9.1), the MAP Server MUST log the differences between what the result of the IF-MAP request would be with the dry-run policy set and the result with the real policy set.
- If authorization is disabled, the MAP Server MUST log what the result of the IF-MAP request would be if the dry-run policy set were in force.

In this section, “result” means whether the IF-MAP request is permitted or denied. For a search request or a subscribed search, “result” also means whether any metadata is censored or not.

3.9.3 Content Selector

The Content Selector governs what parts of an identifier or metadata item are turned into XACML resource attributes.

The MAP Server MUST provide a configurable Content Selector that pairs metadata type strings and identifier type strings with XML attribute names (see 3.4.3.6 and 3.4.3.7).

A MAP Server implementer may wish to include a default Content Selector that specifies all of the attributes needed by TCG-suggested policies.

The Administrator may not need to configure the Content Selector, because the MAP Server may be able to discover automatically which attributes the XACML policy elements require. (See 3.6.5, Content Selector.)

3.9.4 Maximum Policy Lag

The maximum policy lag SHOULD be configurable unless the implementation guarantees the maximum lag is always zero. The maximum policy lag MUST NOT exceed 15 minutes by default. (See section 3.6.3, Decision Cache.)

3.9.5 Decision Cache Clear

If the MAP Server maintains a Decision Cache, an Administrator MUST be able to clear it at will. The Administrator may wish to clear it after a policy change on the XACML PDP.

3.9.6 Trusted Certification Authorities

Per section 6.3.1 of [3], a MAP Server is required to be able to perform certificate-based authentication of MAP Clients. Inevitably, it must be configurable with a certification authority (CA) certificate that anchors the certificate-based authentication.

A MAP Server SHOULD be configurable with several such certification authorities.

A manufacturer’s CA or one of its sub-CAs might issue a certificate with a Subject DN that would grant roles and tasks that the MAP Server Administrator did not want the manufacturer’s CA to be able to grant, and/or cause the is-self-identifier attribute to be true when it should be false.

Suppose the CA of Widgetco creates device certificates with a Distinguished Name of the form

C=GB O=Widget Corporation OU=Products DC=xxxxxx

If this CA is compromised, it could issue a certificate whose Distinguished Name looks like those that the customer's CAs would issue for MAP Clients. As a result, the compromised CA could empower a hostile device within the customer's enterprise to be a MAP Client, obtain powerful roles, and do a lot of damage.

To prevent this, MAP Servers MUST allow trusted CAs (trust anchors) to be configured with name constraints (thus restricting the set of names that the trust anchor may certify) and MUST properly process name constraints when encountered in a CA certificate, as described in RFC 5280 [7]. Administrators are advised to use permitted name constraints (which state exactly which set of Subject DNs should be permitted) for this purpose instead of excluded name constraints, since excluded name constraints are easily evaded.

3.9.7 Trusting Certain Certificates Although a CA Has Been Revoked

If compromise of a certification authority is suspected, good practice is to revoke the CA's certificate using standard mechanisms. Doing so revokes every certificate the CA has issued, whether directly or indirectly. However, this can be highly disruptive; for example, in an ICS environment, many ICS devices may be disabled.

If the date and time of the compromise are known, then the MAP Server may safely continue using certificates that it first encountered before the compromise.

The MAP Server MAY be configurable to continue, despite revocation, to accept certificates that are signed by a particular CA certificate and that the MAP Server first encountered before a particular date and time. The MAP Server MUST NOT rely on the issue dates within certificates for this purpose.

3.9.8 Management of MAP Client Role Metadata

An Administrator SHOULD be able to manage client role metadata in some way other than IF-MAP requests. See section 3.8.2.

3.9.9 Assignment of Roles Based on Distinguished Names of Certifiers

See section 3.8.5, Roles Derived from Certificates.

3.9.10 Treatment of NotApplicable Decisions

MAP Clients may seek to employ metadata types (such as vendor-specific metadata) for which no authorization policy is configured. Though there is some security exposure in allowing unsupervised use of such metadata types, it will sometimes be necessary as a practical matter.

If the recommendations in 3.10 are followed, an attempt to access such a metadata type will yield an XACML decision of NotApplicable.

The MAP Server MUST be configurable to either treat NotApplicable as equivalent to Permit or treat NotApplicable as equivalent to Deny. Treating it as Deny MUST be the out-of-the-box default.

3.9.11 Whether to Use XACML Multiple Decision Profile

A MAP Server that supports the XACML Multiple Decision Profile SHOULD be configurable to either use it or not use it.

3.10 Policy Recommendations

The following Venn diagram is a simplified (and non-normative) illustration of the possible access requests and the resulting decisions, if the policy recommendations in this section are followed.

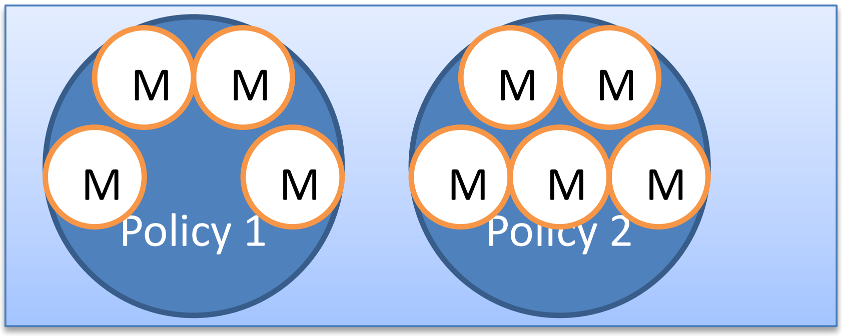


Figure 5 Access Requests and Decisions

The white areas (small circles) represent permitted access. Each Rj denotes a XACML Rule. Each Mk denotes a metadata type. Each of the white circles denotes the set of access requests that one Rule permits. So, Rule M1-R1 permits some actions on metadata type M1, and Rule M1-R2 permits other actions on M1.

The dark blue areas (large circles) represent prohibited access. Access to the metadata types defined by a particular schema is controlled by one Policy. Each blue circle denotes a Policy. One or more blanket rules in the Policy (not shown) deny all access requests that are not explicitly permitted.

The light blue areas (those outside of all circles) represent access to metadata types that the Policies do not mention. By default these actions are denied, but the MAP Server's administrator can configure it to permit these actions.

3.10.1.1 Permit Rules

If a Permit rule concerns particular metadata types, it SHOULD concern only one metadata type. In other words, have separate rules for each metadata type.

Within a policy, Permit rules concerning metadata types SHOULD be grouped by metadata type. Alphabetical order by metadata type is preferred.

3.10.1.2 Blanket Deny Rules

If any policy element references a particular metadata or identifier schema, there SHOULD be a blanket Deny rule for the schema's namespace. A blanket Deny rule ensures that an attempt to access a metadata type under the schema's namespace is denied unless a Permit rule permits it. The blanket Deny rule SHOULD have this form:

```
<Rule
  RuleId="YOUR-RULE-ID"
  Effect="Deny">

 <Target>
  <AnyOf>
   <AllOf>
    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-starts-with">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
      >NAMESPACE-URI#</AttributeValue>
     <AttributeDesignator
      MustBePresent="false"
      Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
      AttributeId="urn:oasis:names:tc:xacml:3.0:if-map:content:resource:metadata-type"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Match>
```

```
          </AllOf>
        </AnyOf>
      </Target>
    </Rule>
```

NAMESPACE-URI is replaced with the URI of the XML schema. The # to its right is literal.

3.10.1.3 Combining Algorithms

Permit Overrides is the RECOMMENDED Rule Combining and Policy Combining algorithm. That way, the XACML PDP returns Permit just if a Permit rule applies. Each Permit rule is self-contained: the effect of having or not having the rule should be evident from the rule alone.

A benefit of this structure is that two policies may permit use of the same metadata type. For example, two policies may permit use of the ifmap-client-has-role metadata type. TNC intends that different use cases and specifications can appropriately use the same metadata types. Therefore two policies sometimes need to permit use of the same metadata type.

Policy elements SHOULD be written so that if a Decision Request concerns a metadata type defined in a schema no policy element mentions, the XACML PDP returns an Authorization Decision of NotApplicable. Therefore Deny rules SHOULD be applicable only to metadata from specific schemas. See 3.9.10.

The method of making Permit Overrides be the Policy Combining Algorithm is somewhat implementation dependent. Methods include:

- The administrator does not load the Policies into the XACML PDP separately; rather, the administrator combines all of the Policies into an XML document whose root element is <PolicySet>. This PolicySet specifies Permit Overrides. The administrator loads the PolicySet into the XACML PDP; it is the sole top-level policy element. This method is thought to work with all XACML PDPs, but it may be inconvenient.
- The administrator loads each of the Policies separately into the XACML PDP, and configures the XACML PDP to use Permit Overrides as the Policy Combining Algorithm of the implicit root PolicySet. (Deny Overrides is more likely to be the default.) Not all XACML PDPs offer this configuration option, however.
- The administrator loads each of the Policies separately into the XACML PDP. The administrator also creates and loads a PolicySet document that:
  - Specifies PermitOverrides
  - Refers to each of the Policies using a <PolicyIdReference> element

  The administrator configures the XACML PDP to use this PolicySet as its root, its entry point. Not all XACML PDPs offer this configuration option, however.

3.10.1.4 XACML Decisions

Policies SHOULD be written so that every decision will be one of:

- Permit
- Deny
- Indeterminate if the MAP Server needs to send an additional attribute (see section 3.6.5)
- NotApplicable if the metadata type is not mentioned by any policy element

An Indeterminate decision that contains no <MissingAttributeDetail> element or one that asks for an unavailable attribute SHOULD be logged by the MAP Server for troubleshooting purposes.

A Target or Condition that mentions a metadata attribute (see 3.4.3.6):

- SHOULD specify MustBePresent="true" in the AttributeDesignator of the metadata attribute

- SHOULD also explicitly require the corresponding metadata type

A Target or Condition that mentions an identifier attribute (see 3.4.3.7)

- SHOULD specify MustBePresent="true" in the AttributeDesignator of the identifier attribute
- SHOULD also explicitly require the corresponding identifier type.

Other Attribute Designators that mention XACML attributes found in this specification SHOULD specify MustBePresent="false".

For example, a Rule that requires the attribute urn:oasis:names:tc:xacml:3.0:if-map:content:resource:identifier-attribute:**ip-address**:**administrative-domain** should specify MustBePresent="true" and should explicitly require that urn:oasis:names:tc:xacml:3.0:if-map:content:resource:**identifier-type** have the value ip-address.

Benefits of these practices are:

- If the Content Selector fails to select an attribute the policy needs, an Indeterminate decision will result. This aids in troubleshooting the Content Selector. Better, it can enable the MAP Server to automatically learn what XACML attributes the policy needs. (See section 3.6.5.)
- Other indeterminate decisions are avoided. Indeterminate decisions are not a disaster; if the final decision is Indeterminate the MAP Server will deny access. But, as noted above, best practice is to write policies in such a way that Indeterminate decisions are avoided.

3.11 Suggested Policies/PolicySets

The Policies and PolicySets in [17], [18], and [19] are intended to be suitable for use with no changes except the optional changes indicated by "CUSTOMIZATION POINT".

If any MAP Clients of the MAP Server implement a given specification, the policy administrator is advised to include the corresponding suggested Policy, or a custom version of it.

When a party other than TCG modifies one of the TCG-suggested Policies and PolicySets, that party MUST give the modified Policy/PolicySet a new URN that they may rightfully use.

3.12 Security Threats Addressed by the Suggested Policies

The primary goal of TNC MAP Content Authorization is to counter security threats involving rogue MAP Clients. This subsection discusses which specific threats TNC MAP Content Authorization addresses, assuming the suggested policy elements are used.

The analysis here does not apply only to the suggested policies. If the suggested policies counter these threats, then other policies also can.

3.12.1 MAP Client Reads or Writes Metadata Types It Should Not

Threat: A rogue MAP Client could do much damage by pretending to be a type of MAP Client it is not. For example, a sensor could act like a PDP, publish an access-request constellation, and give itself access to protected resources. Or it could act like a backhaul interface and compromise an overlay network.

Countermeasure: A MAP Client can be given one or more roles and tasks. Each role or task can be given read access to specified metadata types and write access to specified metadata types.

Predefined roles and task templates that correspond to TNC architecture elements and to ICS elements, together with the suggested policies, simplify this process. The Administrator only has to indicate which role(s) and task(s) each MAP Client has.

3.12.2 MAP Client Puts Metadata Types on the Wrong Kinds of Identifiers

Threat: if a client places metadata on the wrong kind of identifier or link, in violation of language in the specification that defines the metadata type, this may lead to unforeseen security compromises.

Countermeasure: The suggested policies block such misplacements.

3.12.3 MAP Clients Unrelated to an ICS Overlay Network Compromise It

Threat: Elements outside of an ICS overlay network may compromise its operation. In particular, MAP Clients not involved in an ICS overlay network may compromise its operation.

Countermeasure: Metadata belonging to an overlay network can be read and changed only by MAP Clients belonging to the overlay network, and by other authorized MAP Clients responsible for managing the overlay network.

3.12.4 MAP Clients that Are Members of an Overlay Network Overwrite Each Others' Metadata

Threat: An element of an overlay network could be malicious or could malfunction. These devices are widely distributed in lights-out environments. Limiting the damage to the rest of the overlay network is important.

Countermeasure: Limit each MAP Client in an overlay network to publishing metadata about itself, and prevent it from publishing metadata about other members. "Metadata about itself" means a well-defined set of identifiers and metadata types that are associated with that client. See [19].

3.12.5 Overlay Manager MAP Clients Affect Overlay Networks They Should Not

Threat: A rogue or malfunctioning Overlay Manager MAP client can damage or halt many overlay networks.

Countermeasure: Only designated MAP Clients can act as Overlay Manager for each overlay network. A rogue Overlay Manager can damage only overlay networks for which it is a designated Overlay Manager. See [19].

3.12.6 MAP Client Accesses Metadata in an administrative-domain It Should Not Touch

Threat: A rogue or malfunctioning MAP Client can read or write metadata in administrative-domains to which it has no legitimate access.

Countermeasure: See Section 2.4.3, Administrator Limits Access to Some Administrative Domains**.**

3.13 Security Threats Not Addressed by the Suggested Policies

A threat analysis should clearly assess threats not addressed.

3.13.1 Not Addressed: Rogue TNC PDP

Threat: A rogue or malfunctioning TNC PDP grants access that it has no right to grant, though another PDP may have the right.

Countermeasure: None in this version.

If any PDP can grant particular access, then a rogue PDP can, in general, grant that access too. For example, a PDP can attach any capabilities and roles it wants to an access-request constellation it publishes. TNC PEPs will trust these.

A PDP can also tamper with an access-request constellation another PDP has published, for example attaching capabilities or an access-request-ip link.

To be clear, there are constraints even on a rogue PDP: it can still only read and write the metadata types PDPs read and write. Therefore, for example, ICS overlay networks can be protected against rogue PDPs unrelated to them.

3.13.2 Not Addressed: MAP Client Probes Content of MAP Store

Threat: Metadata is not fully hidden. If a MAP Client has no permission to access a metadata item, it can attempt to delete it. The result will be AccessDenied if the metadata item exists. A malicious MAP Client might be able to use this to learn something about activities and vulnerabilities.

Countermeasure: None in this version. Trying to delete a metadata item that isn't there isn't an error. The MAP Server would have to pretend to succeed when actually the metadata wasn't gone. But this would violate the atomicity of publish requests, leading to inconsistent MAP content.

3.14 Logging

The formal policy cannot enforce everything. Formal policies never do. A rogue or malfunctioning MAP Client may misbehave without violating the formal policy.

Logging and review of logs deter, and enable detection and remediation of, unauthorized behavior.

The MAP Server MUST be able to log its decisions and MUST, by default, log denied IF-MAP requests. It MAY offer an option to disable logging of them. Whether to log allowed IF-MAP requests SHOULD be configurable.

For each denial logged, the MAP Server MUST be able to include in its log the content of the XACML Decision Request that was denied and, if present, the content of the <PolicyIdentifierList> element of the XACML Authorization Decision. Whether to include this information MAY be configurable.

The MAP Server MAY be configurable with other selection criteria that control what requests get logged.

The MAP Server SHOULD be configurable to log every XACML Decision Request and its Response.

(See also section 3.9.2, Dry-Run Policy Enabled.).

4 Security Considerations

While this entire document is concerned with security since it is concerned with authorization, a single section that describes the overall security considerations for TNC MAP Content Authorization is worthwhile. This section does so by defining the trust model for TNC MAP Content Authorization (which elements are trusted to do what), enumerating the threat model (attacks that may be mounted on the system), and suggesting countermeasures (ways to address or mitigate the threats previously identified).

The IF-MAP protocol specification[3] includes a strong Security Considerations section. Readers should review that section before reading this one. While all of the issues listed in that section apply here and are not duplicated in this document, TNC MAP Content Authorization introduces a new role into the TNC architecture: the XACML PDP. Therefore, additional security issues are raised.

This section complements section 3.12, Security Threats Addressed by the Suggested Policies. That section is about the *benefits* of TNC MAP Content Authorization. This section is about the *risks* from TNC MAP Content Authorization, and how to minimize them.

4.1 Trust Model

The first step in analyzing the security of TNC MAP Content Authorization is to articulate the trust model, listing what each architectural element is trusted (i.e. expected) to do and further what each element is not trusted (expected) to do.

The items listed here are explicit expectations, but elements sometimes fail to perform as expected (due to bugs, compromise, or other reasons). Therefore, provisions are made in the Threat Model and Countermeasures sections to handle elements that fail to perform as they were trusted to do.

Note that all of the elements of the trust model in the IF-MAP protocol specification[3] apply here also. Only the elements that are added or altered by this specification are described here.

4.1.1 Network

The network used to carry communications between the MAP Server and the XACML PDP (and the XACML PAP, if one is used) is trusted to:

- Perform best-effort delivery of network traffic

This network is not expected (trusted) to:

- Provide confidentiality or integrity protection for messages sent over it
- Provide timely or reliable service (but see section 3.6.8, Behavior when XACML PDP Is Unavailable)

4.1.2 MAP Clients

The primary purpose of this specification is to reduce the level of trust that must be placed in authorized MAP clients, by providing a standard way for Administrators to establish policies restricting the operations that MAP clients can perform. Yet within the scope established by the XACML policies established by the Administrator, the MAP client is still trusted as described in the MAP Client subsection of the Trust Model in the IF-MAP protocol specification[3].

4.1.3 MAP Server

In addition to the items listed in the IF-MAP protocol specification[3], the MAP Server is trusted to:

- Send proper Decision Requests to the XACML PDP
- Properly enforce the responses received from the XACML PDP
- Use secure communications protocols when communicating with the XACML PDP (ensuring that the confidentiality and integrity of Decision Requests and responses is maintained)

4.1.4 XACML PDP

The XACML PDP consulted by the MAP Server is trusted to:

- Obtain the proper XACML policies (from an XACML PAP, configuration, or another source)
- Render proper Authorization Decisions in response to Decision Requests received from the MAP Server
- Preserve the confidentiality of Decision Requests, Authorization Decisions, and policies
- Preserve the confidentiality of the XACML PDP's private key and promptly report its compromise to the appropriate parties (e.g. all CAs that have certified that key)
- Use secure communications protocols when communicating with the MAP Server and XACML PAP (ensuring that the confidentiality and integrity of Decision Requests and responses is maintained)
- Provide timely and reliable service

The XACML PDP consulted by the MAP Server is not expected (trusted) to:

- Verify the veracity of information received from the MAP Server
- Verify that any policies received are accurate and proper

4.1.5 XACML PAP

The XACML PAP used to supply policies to the XACML PDP is trusted to:

- Provide the proper XACML policies to the XACML PDP and update those policies as needed
- Preserve the confidentiality and integrity of XACML policies

The XACML PAP is not expected (trusted) to:

- Verify the veracity of information received from the MAP Server
- Verify that any policies provided are accurate and proper

4.1.6 XACML Policy Administrator

The XACML Policy Administrator (the person who configures the XACML policies) is trusted to:

- Carefully select or design the configured policies, perhaps using the suggested policies [17], [18], and [19]

4.1.7 Certification Authority

The Certification Authority (CA) that issues certificates for the MAP Server, the XACML PDP, and the MAP Clients (or the CAs if there are several) is trusted, as discussed in section 6.1.4 of [3].

4.2 Threat Model

To secure TNC MAP Content Authorization and the architectural elements that implement it, this section identifies the attacks that can be mounted against the protocol and elements. This section lists only attacks that have not already been identified in the IF-MAP protocol specification.

4.2.1 Network Attacks

All of the attacks listed as Network Attacks in the IF-MAP protocol specification can also be mounted with TNC MAP Content Authorization. Since the only new network communications introduced by this specification are those between the MAP Server and the XACML PDP, those are the only new network communications that may be affected by these attacks.

To avoid duplication of the IF-MAP protocol specification, and to avoid potential inconsistencies as the two specifications evolve, specific network attacks are not enumerated here. Consult the IF-MAP protocol specification to find the complete list of network attacks.

4.2.2 MAP Clients

The threats described in the IF-MAP protocol specification for MAP Clients still apply. However, they are diminished by the TNC MAP Content Authorization policies established by the Administrator. Still, a new set of threats is enabled by this specification. A MAP Client may cause a threat by performing one of these actions:

- Send peculiar IF-MAP requests to the MAP Server that cause the MAP Server to send XACML Decision Requests to the XACML PDP that could compromise the XACML PDP. Compromise of the XACML PDP can lead to the threats listed below for a compromised XACML PDP and even to other threats if the XACML PDP is used by other XACML PEPs such as a file server or database. These peculiar IF-MAP requests may be the result of a compromised or malicious MAP Client, but they may also be due to a bug in the MAP Client or simply a peculiarity in the way the MAP Client has been implemented (e.g. sending special characters or very long strings for one field).
- Send a large number of IF-MAP requests, enough to cause the XACML PDP to become overloaded and unresponsive. This can lead to a denial of service on systems that depend on the XACML PDP or even to a “fail-open” condition if some of those systems are configured to permit all access when the XACML PDP cannot be reached or does not respond within a given period of time.
- Assign or unassign roles and tasks according to the letter of the XACML policy, but to the wrong MAP Clients. Victim MAP Clients can lose access that they require. Compromised MAP Clients can get too much access. Therefore, a MAP Client authorized to assign and unassign roles and tasks should be carefully managed and protected.

4.2.3 MAP Server

The set of threats posed by a MAP Server changes little when TNC MAP Content Authorization is employed. A MAP Server may simply ignore the policies established by this specification. However, a few new threats are made possible. The MAP Server can:

- Send specially crafted XACML Decision Requests to the XACML PDP that may lead to compromise of the XACML PDP. Compromise of the XACML PDP can lead to the threats listed below for a compromised XACML PDP and even to other threats if the XACML PDP is used by other XACML PEPs such as a file server or database. These Decision Requests could arise because the MAP Server is compromised or malicious, because of a bug in the MAP Server, or because a MAP Client sent a particular kind of MAP request (e.g. a request with a particularly long string in it).
- Overload or disable the XACML PDP. This can lead to a denial of service on systems that depend on the XACML PDP or even to a “fail-open” condition if some of those systems are configured to permit all access when the XACML PDP cannot be reached or does not respond within a given period of time.

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

4.2.4 XACML PDP

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

An authorized XACML PDP can mount several attacks:

- Return improper Authorization Decisions, thus causing a MAP Server to grant or deny access when it should not. If the attacker also has compromised a MAP Client, this can disable the protections provided by TNC MAP Content Authorization. Alternatively, it can be used to deny requests coming from a legitimate MAP Client. For example, a sensor's request to publish an event could be denied, and thus an attacker's malicious activities might remain unknown.

- Return specially crafted XACML Authorization Decisions to the MAP Server that may lead to compromise of or slow performance in the MAP Server. These Authorization Decisions could arise because the XACML PDP is compromised or malicious, or because of a bug in the XACML PDP.
- Return XACML Authorization Decisions with strange Obligations. Since Obligations can be used to trigger arbitrary actions by the PEP (the MAP Server), they can be dangerous unless the set of actions is carefully limited.
- Respond slowly to Decision Requests, causing the performance of the MAP Server to suffer.
- Reveal XACML policies to attackers, helping them to evade these policies.
- Reveal Decision Requests or Authorization Decisions to attackers, allowing them to see what requests are being sent to the MAP Server and thus to monitor security measures.

If an XACML PDP is dependent on a service, compromise of that service may have significant negative effects on the security and reliability of the XACML PDP, starting with unreliability or poor performance and going as far as possible compromise of the XACML PDP. Therefore, the services supporting the XACML PDP (e.g. a PIP) should be carefully managed and protected.

4.2.5 XACML PAP

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

An authorized XACML PAP can mount several attacks:

- Provide incorrect policies to the XACML PDP, causing the XACML PDP to make incorrect Authorization Decisions. This can cause a MAP Server to grant or deny access when it should not. If the attacker also has compromised a MAP Client, this threat can disable the protections provided by TNC MAP Content Authorization. Alternatively, it can be used to deny requests coming from a legitimate MAP Client. For example, a Sensor's request to publish an event could be denied, and thus an attacker's malicious activities might remain unknown.
- Provide maliciously crafted policies to the XACML PDP (e.g. with strange Obligations), causing the XACML PDP to return Authorization Decisions that may lead to compromise of the MAP Server, slow performance in the MAP Server, or strange side effects due to Obligations.
- Reveal XACML policies to attackers, helping them to evade these policies.

4.2.6 XACML Policy Administrator

An authorized XACML Policy Administrator (one who is able to change policies on the XACML PAP) can mount the same attacks as the XACML PAP, except that the system (probably the PAP) may prevent the XACML Policy Administrator from loading invalid policies.

However, it's worth noting that an inattentive XACML Policy Administrator can cause a large number of security and functional problems by improperly configuring policies. Malicious intent is not required.

4.2.7 Certification Authority

A Certification Authority trusted to issue certificates for the MAP Server, the XACML PDP, and the MAP Clients can mount several attacks, as discussed in section 6.2.4 of [3].

4.2.8 LDAP

If an LDAP relay – as suggested in section 3.8.4 – is in use, the LDAP relay and the LDAP service can assign the relayed roles and tasks to the wrong MAP Clients. Victim MAP Clients can lose access that they require. Compromised MAP Clients can get too much access. Therefore, the LDAP relay and LDAP service should be carefully managed and protected.

4.3 Countermeasures

The threats in the previous section may be worrying. Fortunately, this section describes countermeasures for them. And it is worth noting that this specification does not increase the risks of running a MAP Server. Rather, it decreases them by reducing the privileges of MAP Clients, which are numerous and generally geographically distributed. Adding the XACML PDP and PAP adds two highly trusted parties, but they may be integrated into the MAP Server if desired, resulting in a net reduction in privilege.

4.3.1 Securing the IF-MAP Protocol

By securing the IF-MAP protocol as described in the IF-MAP protocol specification[3], most network attacks can be defeated. Also, attacks by unauthorized MAP Clients and MAP Servers can be eliminated. Therefore, the security measures described in the IF-MAP protocol specification should be implemented. As section 6.3.1 of [3] states, MAP Clients and MAP Servers SHOULD each verify the revocation status of the other party.

4.3.2 Securing MAP Clients

The recommendations for securing MAP Clients included in the IF-MAP protocol specification[3] are helpful and should be followed. This specification fleshes out the “full authorization model” mentioned in the IF-MAP protocol specification, thus reducing the threat posed by a rogue MAP Client.

4.3.3 Securing MAP Servers

The recommendations for securing MAP Servers included in the IF-MAP protocol specification[3] are helpful and should be followed.

To reduce the threat of a MAP Client sending peculiar IF-MAP requests to the MAP Server that cause the MAP Server to send XACML Decision Requests to the XACML PDP that could compromise the XACML PDP, the MAP Server SHOULD perform careful validation of MAP Client requests.

To reduce the threat of a MAP Client sending a large number of IF-MAP requests, enough to cause the XACML PDP to become overloaded and unresponsive, the MAP Server MAY implement rate limits for MAP Clients. The Decision Cache may also help with this.

4.3.4 Securing XACML PDPs

Because of the serious consequences of XACML PDP compromise, XACML PDPs 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 XACML PDP, and SHOULD utilize a Trusted Platform Module (TPM) for identity and/or integrity measurements of the XACML PDP within a TNC handshake. They should be well managed to minimize vulnerabilities in the underlying platform and in systems upon which the XACML PDP depends. Network security measures such as firewalls or intrusion detection systems may be used to monitor and limit traffic to and from the XACML PDP. 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 XACML PDPs.

To ease detection of XACML PDP compromise should it occur, XACML PDP behavior should be monitored to detect unusual behavior (such as a reboot, a large increase in traffic, or traffic to unexpected destinations). MAP Servers should log and/or notify Administrators when peculiar XACML PDP behavior is detected. MAP Servers should also check data received from the XACML PDP carefully to detect malformed data. MAP Servers should be especially wary of Obligations returned in an Authorization Decision. To aid forensic investigation, permanent read-only audit logs of security-relevant information (especially administrative actions) should be maintained. If XACML

PDP 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.

A MAP Server SHOULD NOT accept XACML obligations, except according to section 3.5.

4.3.5 Securing XACML PAPs

As described above, compromise of an XACML PAP can be serious since it can cause the XACML PAP to send incorrect policies to the XACML PDP. The XACML PAP should be protected using similar techniques to those recommended for in the previous section for securing the XACML PDP.

Because changes in policy are relatively rare, it may be desirable for the XACML PDP to cache the most recent policies and request confirmation from the Administrator whenever a revised policy is received from the XACML PAP. If this confirmation is carefully designed (recognizing that policy changes often come in a group), the Administrator should see this as a benefit.

4.3.6 Securing the XACML Policy Administrator

While it may seem odd to think about securing an Administrator, personnel security is critical for a security-sensitive role like the XACML Policy Administrator. XACML Policy Administrators should be carefully screened and monitored to detect problems as soon as possible. XACML Policy Administrators should not use password-based authentication but should instead use non-reusable credentials and multi-factor authentication (where available).

Compromise of the XACML Policy Administrator’s workstation can be devastating. Even if the XACML Policy Administrator’s credentials cannot be stolen (e.g. a private key stored on the TPM), malware on the workstation can infect the policy administration tool and cause a modified version of the policy to be uploaded while keeping all of this hidden from the XACML Policy Administrator. Therefore, the XACML Policy Administrator’s workstation should be carefully secured and regularly screened with TNC handshakes, and should utilize a Trusted Platform Module (TPM) for identity and/or integrity measurements of the workstation within a TNC handshake.

To detect improper XACML Policy Administrator actions or compromise of the XACML Policy Administrator’s workstation, policies on the PAP should be inspected periodically by other parties using other workstations to make sure that they are OK. One easy way to do this is to have the XACML PDP administrator inspect the policies using the XACML PDP’s administrative interface. This can also detect compromise of the XACML PAP.

The XACML Policy Administrator can also cause exposures through mistakes. Ideally, MAP-aware policy authoring tools will emerge that will help the Policy Administrator to adhere to the policy recommendations (except for intentional departures) and to avoid mistakes.

4.3.7 Securing the Certification Authority

Compromise of a Certification Authority (CA) trusted to issue certificates for the MAP Server, the XACML PDP, or MAP Clients is a major security breach. The countermeasures found in section 6.3.4 of [3] should be applied.

4.4 Summary

TNC MAP Content Authorization provides a net improvement in security, since it allows Administrators to easily restrict the privileges of MAP Clients, which are numerous and generally geographically distributed, and therefore at risk for compromise.

The TNC MAP Content Authorization specification does add two highly trusted elements: the XACML PDP and the XACML PAP. However, these elements are less trusted than the MAP Server so the impact of adding these elements is lessened. Also, they may be integrated into the MAP Server if desired, avoiding the need to add more trusted elements.

5 Privacy Considerations

The MAP Server contains privacy-sensitive information, such as user identity, roles, and capabilities as well as information about the user’s endpoint and a variety of events, which may include information about what services an endpoint is accessing. For this reason restricting access to MAP Server contents is essential.

This specification enables the Administrator to create policies about which MAP Clients can access which data on the MAP Server. By defining a standard format for such policies and a comprehensive set of suggested policies that provide appropriate restrictions on MAP Clients, this specification helps Administrators to install appropriate access controls on privacy-sensitive information. The suggested Policies allow each role and task to read only the metadata types it needs to read. An Administrator can go further: read access to metadata in a particular administrative-domain can be limited to particular Administrator-defined roles.

6 References

6.1 Normative References

[1] Trusted Computing Group, *TNC Architecture for Interoperability*, Specification Version 1.5, Revision 3, May 2012

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", Internet Engineering Task Force RFC 2119, March 1997.

[3] Trusted Computing Group, *TNC IF-MAP Binding for SOAP*, Specification Version 2.2, March 2014

[4] *eXtensible Access Control Markup Language (XACML)* Version 3.0, OASIS Standard, 22 January 2013. Editable source (Authoritative): http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.dochttp://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os01-en.html.

[5] *OASIS XACML v3.0 Multiple Decision Profile* Version 1.0. 10 August 2010. OASIS Committee Specification. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cs-01-en.pdf

[6] RFC-2141, *URN Syntax*, IETF

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280, May 2008, IETF

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

[9] World Wide Web Consortium, *URL Encoding Reference*, http://www.w3schools.com/tags/ref_urlencode.asp

[10] Trusted Computing Group, *TNC IF-MAP Metadata for Network Security*, Specification Version 1.1, May 2012

6.2 Informative References

[11] Trusted Computing Group, *IF-MAP Metadata for ICS Security*, Version 1.0r44 (public review draft)

[12] Graph (mathematics), https://en.wikipedia.org/wiki/Graph_(mathematics)

[13] *SAML 2.0 Profile of XACML* v2.0, OASIS Standard, 1 February 2005

[14] *Bindings for the OASIS Security Assertion Markup Language (SAML)* V2.0, OASIS Standard, 15 March 2005

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

[16] Hal Lockhart, *RE: [xacml] PolicySetIdReference Questions*, January 10, 2013, https://lists.oasis-open.org/archives/xacml/201301/msg00006.html

[17] Trusted Computing Group, *Suggested Authorization Policy for General Metadata*, Suggested-Authzn-Policy-for-General-Metadata-1.0.xml, June 2014

[18] Trusted Computing Group, *Suggested Authorization Policy for Metadata for Network Security*, Suggested-Authzn-Policy-for-Network-Security-10.xml, June 2014

[19] Trusted Computing Group, *Suggested Authorization Policy for Metadata for ICS Security*, Suggested-Authzn-Policy-for-ICS-Security-1.0.xml, June 2014

[20] http://www.esukom.de/

[21] *XACML MAP Authorization Profile Version 1.0*. 07 April 2014. OASIS Committee Specification 01 http://docs.oasis-open.org/xacml/xacml-map-authz/v1.0/cs01/xacml-map-authz-v1.0-cs01.pdf

7 Metadata Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<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/2011/IFMAP-IDENTIFIER/1"
 xmlns="http://www.trustedcomputinggroup.org/2013/MAP-CONTENT-AUTHORIZATION/1"
 targetNamespace="http://www.trustedcomputinggroup.org/2013/MAP-CONTENT-
AUTHORIZATION/1"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">

<!-- Importing from IFMAP-2.1r14 namespace for schema validation -->
  <xsd:import
   namespace="http://www.trustedcomputinggroup.org/2010/IFMAP/2"
   schemaLocation="IF-MAP-2.1r14-schema.xsd"/>

<!-- Importing from IFMAP-2.1r14-ExtendedIdentifier namespace for schema validation -->
  <xsd:import
   namespace="http://www.trustedcomputinggroup.org/2011/IFMAP-IDENTIFIER/1"
   schemaLocation="IF-MAP-2.1r14-ExtIdentifier-schema.xsd"/>

<!-- Schema for IF-MAP Metadata for TNC MAP Content Authorization v1.0 -->

    <xsd:element name="ifmap-client-role">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="base-id:IdentifierType">
             <xsd:attribute name="name" type="xsd:string" use="required"/>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="ifmap-client-has-role">
      <xsd:complexType>
          <xsd:extension base="SingleValueMetadataType">
          </xsd:extension>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="ifmap-client-has-task">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="MultiValueMetadataType">
            <xsd:sequence>
              <xsd:element name="relationship" type="xsd:string"
                minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>

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

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