TCG Compliance_TNC IF-MAP v2.0 Base Spec Compliance Test Plan

Version 1.0

Revision 28

8 August 2011

Published

Contact:

admin@trustedcomputinggroup.org

TCG PUBLISHED

Version 1.0

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide specific requirements for the compliance tests for IF-MAP v2.0. In particular, it defines and lists all the compliance test cases that must be passed to prove Compliance with respect to the IF-MAP v2.0 specification. This document does not contain any normative statements.

1.2 Scope and Audience

The intended audience for this document includes test designers and implementers, as well as product developers and customers who need to understand the IF-MAP v2.0 compliance tests. Readers should be familiar with the TNC Architecture[1], with the Compliance_TNC Compliance and Interoperability Principles specification [3] and with IF-MAP v2.0[2].

2 Specifications and Components

2.1 Specifications

This document is based on the IF-MAP v2.0 specification [2], IF-MAP Metadata for Network Security [6] and Interoperability Principles with IF-MAP document [3]. The IF-MAP v2.0 specification defines the IF-MAP interface. Metadata v1.0 defines the supported standard metadata definitions. The Compliance_TNC Compliance and Interoperability Principles document provides an overview of the Compliance_TNC testing.

2.2 Components

There are two sets of IF-MAP compliance tests that test the two prospective to the IF-MAP protocol: Server side and client side.

2.2.1 IF-MAP Server

The IF-MAP Compliance tests for MAP server tests that a MAP server properly implements IF-MAP. The Test Target for this test is a IF-MAP server.

To test a MAP server compliance with IF-MAP v2.0, a simulated client will send various requests to MAP server. After each request, the response packet will be examined to ensure that the criteria are met and the MAP has properly implemented the behaviors stated on the normative requirements.

2.2.2 IF-MAP Client

The IF-MAP Compliance tests for MAP client tests that a MAP client properly implements IF-MAP. The Test Target for this test is an IF-MAP client.

To test a MAP client compliance with IF-MAP v2.0, a MAP server will be used to monitor and examine the requests coming from the client. Since MAP server operates passively, the tester has to manually generate traffic from client side while the server monitors request packets to have been properly implemented based on the behaviors stated in the normative requirements.

3 Requirements and Recommendations

The IF-MAP v2.0 specification includes many requirements and recommendations for the server side and client side. This section lists only the mandatory requirements since the compliance tests for IF-MAP only test normative requirements (not recommendations).

This section has three subsections. The first section lists mandatory requirements from the server's perspective. The second section lists mandatory requirements from the client's perspective. The third section lists other requirements that will not be tested by this test plan.

As required by the TCG Compliance and Interoperability Guidelines[14], each requirement listed below has a unique name composed of the string "CTNC" (for Compliance_TNC), "IFMAP2.0" (indicating that these are requirements from IF-MAP v2.0), "SERVER" or "CLIENT" depending on which component the requirement applies to, a requirement number unique within the preceding prefix, "REQ" indicating it is a requirement, and a compliance classifier ("M" for MUST or MUST NOT). Usage classifiers are not included in requirement names at this time.

3.1 Requirements on IF-MAP Server

[CTNC-IFMAP2.0-SERVER-REQ-1-M] Communication between MAP Clients and Servers MUST be authenticated and integrity-protected against unauthorized modifications enroute. (Section 2.4.2, "Secure")

[CTNC-IFMAP2.0-SERVER-REQ-2-M] Communication between MAP Clients and Servers MUST provide confidentiality against unauthorized disclosure. (Section 2.4.2, "Secure")

[CTNC-IFMAP2.0-SERVER-REQ-3-M] Communication between MAP Clients and Servers MUST NOT be susceptible to replay attacks. (Section 2.4.2, "Secure")

[CTNC-IFMAP2.0-SERVER-REQ-4-M] All MAP Clients and Servers MUST support both standard and vendor-specific metadata. (Section 2.6.3, "Metadata")

[CTNC-IFMAP2.0-SERVER-REQ-5-M] A MAP Server MUST allow storage and retrieval of vendor-specific metadata. (Section 2.6.3, "Metadata")

[CTNC-IFMAP2.0-SERVER-REQ-6-M] MAP Clients and MAP Servers MUST exchange string fields in UTF-8. (Section 3.1, "string encoding")

[CTNC-IFMAP2.0-SERVER-REQ-7-M] All MAP Server and Client implementations MUST support the entire set of identifiers. (Section 3.2, "identifiers")

[CTNC-IFMAP2.0-SERVER-REQ-8-M] MAP Clients and Servers MUST support identifiers up to 1000 bytes in length. (Section 3.2, "identifiers")

[CTNC-IFMAP2.0-SERVER-REQ-9-M] A MAP Server MUST process two access-request identifiers as equivalent if and only if ALL corresponding fields in the two access-request identifiers are equivalent. (Section 3.2.1, "access-request identifier")

[CTNC-IFMAP2.0-SERVER-REQ-10-M] A MAP Server MUST process the administrative domain field as CASE SENSITIVE. (Section 3.2.1, "access-request identifier")

[CTNC-IFMAP2.0-SERVER-REQ-11-M] A MAP Server MUST process as equivalent (i.e. belonging to the same administrative domain) administrative domains which are specified as an empty string or unspecified. (Section 3.2.1, "access-request identifier")

[CTNC-IFMAP2.0-SERVER-REQ-12-M] A MAP Server MUST process two device identifiers as equivalent if and only if the corresponding names in the two device identifiers are equivalent in both type (i.e. name or aik-name) and value. (Section 3.2.2, “device identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-13-M] A MAP Server MUST process two identities as equivalent if and only if ALL corresponding fields in the two identities are equivalent. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-14-M] A MAP Server MUST process the administrative domain field as CASE SENSITIVE. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-15-M] A MAP Server MUST process as equivalent (i.e. belonging to the same administrative domain) administrative domains which are specified as an empty string or unspecified. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-16-M] A MAP Server MUST process the name field according to the syntax specified by the type enumeration. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-17-M] A MAP Server MUST process two syntactically equal name fields as distinct if they are associated with distinct types. (See identity type table in section 3.2.3 of IF-MAP 2.0 spec.) For example an identity with an unspecified administrative domain, type=username, and name=”foo.bar” MUST be considered distinct from an identity with an unspecified administrative domain, type=dns-name name=”foo.bar” since the types are distinct. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-18-M] HIT MUST be expressed as x:x:x:x:x:x:x:x, where the 'x's are the lowercase hexadecimal values of the eight 16-bit pieces of the HIT. No leading zeros are allowed except that the number 0 is represented by a single 0 character. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-19-M] MAP Servers MUST reject IP addresses which are not in canonical form. (Section 3.2.4, “ip-address identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-20-M] A MAP Server MUST process the administrative domain field as CASE SENSITIVE. (Section 3.2.4, “ip-address identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-21-M] A MAP Server MUST process as equivalent (i.e. belonging to the same administrative domain) administrative domains which are specified as an empty string or unspecified. (Section 3.2.4, “ip-address identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-22-M] MAP Clients and Servers MUST specify the MAC address as six groups of two lowercase hexadecimal digits, separated by colons (:) in transmission order, e.g. 01:23:45:67:89:ab. (Section 3.2.5, “mac-address identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-23-M] A MAP Server MUST process the administrative domain field as CASE SENSITIVE. (Section 3.2.5, “mac-address identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-24-M] A MAP Server MUST process as equivalent (i.e. belonging to the same administrative domain) administrative domains which are specified as an empty string or unspecified. (Section 3.2.5, “mac-address identifier”)

[CTNC-IFMAP2.0-SERVER-REQ-25-M] MAP Servers MUST include operational attributes in search responses (section 3.7.2). (Section 3.3, “metadata”)

[CTNC-IFMAP2.0-SERVER-REQ-26-M] Metadata elements MUST NOT include attributes that begin with the ifmap- prefix other than attributes specified in this document or its successors. (Section 3.3, “metadata”)

[CTNC-IFMAP2.0-SERVER-REQ-27-M] Any unrecognized attributes beginning with the ifmap-prefix MUST be ignored by MAP Clients and MAP Servers. (Section 3.3, “metadata”)

[CTNC-IFMAP2.0-SERVER-REQ-28-M] A MAP Server MUST NOT assign the same ifmap-publisher-id to multiple different MAP Clients. (Section 3.3.1, “ifmap-publisher-id”)

[CTNC-IFMAP2.0-SERVER-REQ-29-M] A MAP Server MUST consistently use the same ifmap-publisher-id for a particular MAP Client. (Section 3.3.1, “ifmap-publisher-id”)

[CTNC-IFMAP2.0-SERVER-REQ-30-M] The ifmap-publisher-id MUST NOT be a function of specific credentials. (Section 3.3.1, “ifmap-publisher-id”)

[CTNC-IFMAP2.0-SERVER-REQ-31-M] A MAP Client MUST be able to change credentials (e.g. new cert after expiration) and continue to be assigned the same ifmap-publisher-id. (Section 3.3.1, “ifmap-publisher-id”)

[CTNC-IFMAP2.0-SERVER-REQ-32-M] The ifmap-publisher-id MUST NOT be a function of time or of any connection specific information; it must remain consistent across time, and across multiple connections. (Section 3.3.1, “ifmap-publisher-id”)

[CTNC-IFMAP2.0-SERVER-REQ-33-M] A ifmap-timestamp without a timezone component MUST be interpreted as UTC time. (Section 3.3.2 “ifmap-timestamp”)

[CTNC-IFMAP2.0-SERVER-REQ-34-M] Every metadata item MUST include a valid ifmap-cardinality attribute (Section 3.3.3, “ifmap cardinality”)

[CTNC-IFMAP2.0-SERVER-REQ-35-M] If a metadata item in a request lacks a valid ifmap-cardinality attribute, the MAP Server MUST return an InvalidMetadata errorResult. (Section 3.3.3, “ifmap cardinality”)

[CTNC-IFMAP2.0-SERVER-REQ-36-M] If a metadata item in an update request specifies different if-map cardinality from an instance of the same metadata type already associated with the identifier or link, the MAP Server MUST return an InvalidMetadata errorResult. (Section 3.3.3, “ifmap cardinality”)

[CTNC-IFMAP2.0-SERVER-REQ-37-M] Whenever there is an errorResult, the request MUST have no other effect. (Section 3.3.3, “ifmap cardinality”)

[CTNC-IFMAP2.0-SERVER-REQ-38-M] If a metadata item in an update request specifies singleValue, the MAP Server MUST replace any previous instance of that metadata type associated with the same identifier or link with the new metadata. (Section 3.3.3, “ifmap cardinality”)

[CTNC-IFMAP2.0-SERVER-REQ-39-M] If a metadata item in an update request specifies multiValue, the MAP Server MUST append the metadata to any existing metadata on the identifier or link even if it duplicates existing metadata on that identifier or link. (Section 3.3.3, “ifmap cardinality”)

[CTNC-IFMAP2.0-SERVER-REQ-40-M] If an element was published with lifetime=”session” and the client session ends, either due to inactivity (see Section 4.1.1) or at the client’s request, the MAP server MUST delete the metadata. This deletion MUST be completed before the publishing client is allowed to create another session. (Section 3.3.5 “Lifetime of Metadata”)

[CTNC-IFMAP2.0-SERVER-REQ-41-M] The lifetime attribute is meaningful only in update requests. A client SHOULD not use it in a notify request and a MAP server MUST ignore it if it appears there. (Section 3.3.5 “Lifetime of Metadata”)

[CTNC-IFMAP2.0-SERVER-REQ-42-M] A MAP server MUST retain all metadata from an update request until (a) it is explicitly deleted by a client, via a delete or purgePublisher request, (b) it is deleted at session end, as discussed above, or (c) the server deletes all data that has the same ifmap-publisher-id. (Section 3.3.5 “Lifetime of Metadata”)

[CTNC-IFMAP2.0-SERVER-REQ-43-M] MAP Clients and Servers MUST support metadata at least 100000 bytes in length, and SHOULD support longer metadata. (Section 3.3.6, “Metadata Size”)

[CTNC-IFMAP2.0-SERVER-REQ-44-M] A MAP Server MUST consider a filter consisting of the empty string as a request to match nothing. (Section 3.4, “Filters”)

[CTNC-IFMAP2.0-SERVER-REQ-45-M] The namespace prefixes in a filter MUST be declared in the XML document containing the filter, and the declaration of the namespace prefix MUST apply to the scope of the filter. (Section 3.4, “Filters”)

[CTNC-IFMAP2.0-SERVER-REQ-46-M] A MAP Server MUST respond to every MAP Client request. (Section 3.6, “Responses”)

[CTNC-IFMAP2.0-SERVER-REQ-47-M] If a MAP Client request cannot be processed, the MAP Server MUST respond with an errorResult element and appropriate error code and message or with a lower-level error response (e.g. SOAP). (Section 3.6, “Responses”)

[CTNC-IFMAP2.0-SERVER-REQ-48-M] A successful metadata publish MUST result in a publishReceived message. Otherwise, the entire publish request MUST fail without effect and the response MUST contain an errorResult element with an errorCode attribute indicating the cause of the failure. (Section 3.7.1, “publish”)

[CTNC-IFMAP2.0-SERVER-REQ-49-M] A MAP Server MUST process valid publish messages which contain any combination of the three publish subtypes. (Section 3.7.1, “publish”)

[CTNC-IFMAP2.0-SERVER-REQ-50-M] An update request with ifmap-cardinality="singleValue" MUST replace the previous value for that metadata type if it exists on the MAP Server. (Section 3.7.1.1, “Publish update”)

[CTNC-IFMAP2.0-SERVER-REQ-51-M] An update request with ifmap-cardinality="multiValue” MUST append the new value to a list of values for that metadata type on the MAP Server. (Section 3.7.1.1, “Publish update”)

[CTNC-IFMAP2.0-SERVER-REQ-52-M] Metadata published with notify MUST be queued for delivery to all clients with subscriptions whose searches match the published data at the time the data is published. (Section 3.7.1.2, “Publish notify”)

[CTNC-IFMAP2.0-SERVER-REQ-53-M] Metadata published with notify MUST be delivered to clients inside notifyResult elements within pollResult elements (see section 3.7.5). (Section 3.7.1.2, “Publish notify”)

[CTNC-IFMAP2.0-SERVER-REQ-54-M] Metadata published with notify MUST NOT be delivered to clients in response to a search request. (Section 3.7.1.2, “Publish notify”)

[CTNC-IFMAP2.0-SERVER-REQ-55-M] A MAP Server MUST delete metadata specified by a valid delete message. (Section 3.7.1.3, “Publish delete”)

[CTNC-IFMAP2.0-SERVER-REQ-56-M] If a delete request has no filter, a MAP server MUST delete all metadata associated with the identifier or link. (Section 3.7.1.3, “Publish delete”)

[CTNC-IFMAP2.0-SERVER-REQ-57-M] If a delete request has a filter that is the empty string, the MAP server MUST delete nothing. (Section 3.7.1.3, “Publish delete”)

[CTNC-IFMAP2.0-SERVER-REQ-58-M] When a publish request contains multiple update and/or delete elements which operate on the same identifiers or links, the result of the publish request MUST be consistent with the update and delete elements having been applied to the IFMAP database in the order in which they are specified by the client. (Section 3.7.1.4, “Multiple Operations in a single publish request”)

[CTNC-IFMAP2.0-SERVER-REQ-59-M] MAP Servers MUST provide results equivalent, with respect to the identifiers, links, and metadata, to a search which would result from the following algorithm (See Section 3.7.2.8 for details on the algorithm). (Section 3.7.2.8, “Search Algorithm)

[CTNC-IFMAP2.0-SERVER-REQ-60-M] If a MAP Client does not specify max-depth, the MAP Server MUST process the search with a max-depth of zero.(Section 3.7.2.8, “Search Algorithm)

[CTNC-IFMAP2.0-SERVER-REQ-61-M] MAP Servers MUST support size constraints up to and including 100KB. (Section 3.7.2.8, “Search Algorithm)

[CTNC-IFMAP2.0-SERVER-REQ-62-M] If a MAP Client does not specify max-size, the MAP Server MUST process the search with a max-size of 100KB. (Section 3.7.2.8, "Search Algorithm)

[CTNC-IFMAP2.0-SERVER-REQ-63-M] If a MAP Client specifies a max-size that exceeds what the MAP Server can support, the MAP Server MUST enforce its own maximum size constraints. (Section 3.7.2.8, "Search Algorithm)

[CTNC-IFMAP2.0-SERVER-REQ-64-M] A MAP Server MUST return an error (and no partial results) if result exceeds max-size. (Section 3.7.2.8, "Search Algorithm)

[CTNC-IFMAP2.0-SERVER-REQ-65-M] A successful search MUST result in a response message containing a searchResult element comprised of identifiers and links along with their associated metadata. (Section 3.7.3, "Search Results")

[CTNC-IFMAP2.0-SERVER-REQ-66-M] If the result-filter filters out all metadata associated with an identifier or link, the identifier or link MUST still be included in the searchResult even though no metadata is associated with the identifier or link in the search result. (Section 3.7.3, "Search Results")

[CTNC-IFMAP2.0-SERVER-REQ-67-M] If no result-filter is present in a search or subscription, ALL metadata that matches the search MUST be returned. (Section 3.7.3, "Search Results")

[CTNC-IFMAP2.0-SERVER-REQ-68-M] Since all identifiers for a given identifier type are always valid to search, the MAP Server MUST never return an "identifier not found" error when searching for an identifier. In this case, the MAP Server MUST return the identifier with no metadata or links attached to it. (Section 3.7.3, "Search Results")

[CTNC-IFMAP2.0-SERVER-REQ-69-M] A MAP Server MUST maintain only one subscription list per connected MAP Client. (Section 3.7.4 "Subscribe")

[CTNC-IFMAP2.0-SERVER-REQ-70-M] A MAP Server MUST respond to a valid subscribe message with a subscribeReceived message. If the subscribeRequest is not valid, the MAP Server MUST respond with an appropriate errorResult. (Section 3.7.4 "Subscribe")

[CTNC-IFMAP2.0-SERVER-REQ-71-M] When a MAP Client initially connects to a MAP Server, the MAP Server MUST delete any previous subscriptions corresponding to the MAP Client. (Section 3.7.4 "Subscribe")

[CTNC-IFMAP2.0-SERVER-REQ-72-M] A MAP Server MUST respond to a valid poll with a pollResult, endSessionResult (see section 4), or errorResult message. (Section 3.7.5, "Poll")

[CTNC-IFMAP2.0-SERVER-REQ-73-M] If the poll is not valid, the MAP Server MUST respond with an appropriate errorResult. (Section 3.7.5, "Poll")

[CTNC-IFMAP2.0-SERVER-REQ-74-M] A valid poll may result in an error (most likely SearchResultsTooBig or PollResultsTooBig). In this case the MAP Server MUST respond with a pollResult message containing an errorResult. (Section 3.7.5, "Poll")

[CTNC-IFMAP2.0-SERVER-REQ-75-M] If a server responds to a poll with an errorResult, all of the client's subscriptions are automatically invalidated and MUST be removed by the server. (Section 3.7.5, "Poll")

[CTNC-IFMAP2.0-SERVER-REQ-76-M] In response to a poll request, the MAP Server checks to see if any search results are available for subscriptions the client has made. If any results are available, the MAP Server MUST send a pollResult response containing search results. (Section 3.7.5, "Poll")

[CTNC-IFMAP2.0-SERVER-REQ-77-M] [In response to a poll request,] If no results are available, including the case where there are no active subscriptions for this client, the MAP Server MUST NOT immediately send a response. At some future time when metadata changes occur that match client subscriptions, the MAP Server MUST send a pollResult response containing search results. (Section 3.7.5, "Poll")

[CTNC-IFMAP2.0-SERVER-REQ-78-M] The first time a pollResult contains search results for a new subscription, the search results MUST consist of the complete set of identifiers, links, and metadata for the subscription as specified in section 3.7.2.1. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-79-M] Subsequent pollResults MUST contain updates in updateResult elements as metadata is added and deletes in deleteResult elements as metadata is removed. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-80-M] The updateResult and deleteResult elements returned by the server MUST reflect ALL of the updates which have occurred since the last poll which affect the client’s subscriptions. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-81-M] The server MUST NOT compress search results by removing updateResult and deleteResult elements which cancel each other out. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-82-M] The server MUST return a separate searchResult, updateResult, deleteResult or errorResult element for each subscription that has changes. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-83-M] Each searchResult, updateResult, deleteResult and errorResult element in a pollResult response MUST include a name attribute which identifies the subscription corresponding to the result. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-84-M] The server MUST NOT return searchResult, updateResult or deleteResult elements for a subscription that has no changes associated with it. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-85-M] When a MAP client publishes using notify, the MAP server MUST add the published metadata to the poll results for each subscriber whose subscription matches the published metadata. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-86-M] For a client that does not have an active poll request, the MAP server MUST retain metadata published using notify until the client issues a poll request or the client’s session ends. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-87-M] notifyResult is only used to return metadata that was published using notify, even if other metadata matches the result-filter in the subscription. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-88-M] A MAP Server MUST buffer at least 5,000,000 bytes of poll results for each MAP Client (see section 4.3). (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-89-M] If the size of a MAP Client’s poll results exceeds the MAP Server’s limit, the MAP Server MUST indicate this to the MAP client by responding to a poll request with an errorResult containing an errorCode of PollResultsTooBig. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-SERVER-REQ-90-M] A MAP Server MAY forbid a MAP Client to use the purgePublisher request to remove data published by a different MAP Client, in which case the MAP Server MUST respond with an AccessDenied error. (Section 3.7.6, “purgePublisher”)

[CTNC-IFMAP2.0-SERVER-REQ-91-M] A MAP Server MUST either respond to a purgePublisher request with a purgePublisherReceived message and remove all data from with that publisher ID, or return an errorResult message and not delete any data. (Section 3.7.6, “purgePublisher”)

[CTNC-IFMAP2.0-SERVER-REQ-92-M] A MAP Server MUST include the XML namespaces associated with the newest schema versions in any message documents. (Section 3.8, “Schema Versioning)

[CTNC-IFMAP2.0-SERVER-REQ-93-M] MAP Servers MUST support operations using vendor-specific metadata which is defined by an XML schema. (Section 3.9, "Vendor-specific Metadata)

[CTNC-IFMAP2.0-SERVER-REQ-94-M] Publish requests that contain multiple update, notify and/or delete requests have special behavior. The Server MUST ensure that they appear atomic. (Section 3.10, "Atomicity")

[CTNC-IFMAP2.0-SERVER-REQ-95-M] The Server MUST ensure that purgePublisher requests appear atomic. (Section 3.10, "Atomicity")

[CTNC-IFMAP2.0-SERVER-REQ-96-M] When (per Section 3.3.5) the Server deletes all metadata that came from a particular client and had lifetime="session", the Server MUST ensure that this bulk deletion appears atomic. (Section 3.10, "Atomicity")

[CTNC-IFMAP2.0-SERVER-REQ-97-M] This requirement for special handling of atomic changes supersedes the requirement in section 3.7.5 that "[t]he updateResult and deleteResult elements returned by the server MUST reflect ALL of the updates which have occurred since the last poll which affect the client's subscriptions."

[CTNC-IFMAP2.0-SERVER-REQ-98-M] However, if a MAP Client sends a single publish request containing both a delete operation and an update operation, polling MAP Clients should see only the end result due to the atomicity of the publish request; the MAP Server MUST send a pollResult containing the end state after the delete and update operations in the single publish request. (Section 3.10, "Atomicity")

[CTNC-IFMAP2.0-SERVER-REQ-99-M] If a MAP client has an existing SSRC connection and sends an SSRC request over a different connection, the MAP server MUST associate the new connection with the SSRC and MUST close the old connection (if possible) or respond to new requests on the old connection with an AccessDenied errorResult. (Section 4.1.1, "Sessions")

[CTNC-IFMAP2.0-SERVER-REQ-100-M] The SSRC MUST NOT be used for poll requests. (Section 4.1.1.1, "SSRC")

[CTNC-IFMAP2.0-SERVER-REQ-101-M] The following IFMAP messages MAY be communicated over the ARC: poll and response. Other messages MUST NOT be communicated over the ARC. (Section 4.1.1.2, "ARC")

[CTNC-IFMAP2.0-SERVER-REQ-102-M] The response element sent in response to a poll request MUST contain either an appropriate pollResult element corresponding to the MAP Client's subscription, an errorResult element indicating the cause of an error, or an endSessionResult indicating that the session ended. (Section 4.1.1.2, "ARC")

[CTNC-IFMAP2.0-SERVER-REQ-103-M] All implementations of the IFMAP protocol MUST support Simple Object Access Protocol (SOAP) v. 1.2 as defined in[5]. (Section 4.2, "SOAP Transport")

[CTNC-IFMAP2.0-SERVER-REQ-104-M] All IFMAP messages are encapsulated inside SOAP bodies which in turn are inside SOAP envelopes. HTTP compression for responses, if used, MUST be negotiated according to HTTP 1.1 [7]. (Section 4.2, "SOAP Transport")

[CTNC-IFMAP2.0-SERVER-REQ-105-M] MAP servers MUST accept requests that have been compressed using gzip and identity transformations. (Section 4.2, "SOAP Transport")

[CTNC-IFMAP2.0-SERVER-REQ-106-M] A MAP Server MUST support buffer sizes of at least 5,000,000 bytes, and MAY support larger sizes. (Section 4.3, "Session ID")

[CTNC-IFMAP2.0-SERVER-REQ-107-M] If the MAP Server can create a session for the client, the MAP Server's response MUST contain a "newSessionResult" element that has a "session-id" attribute that specifies the MAP Client's session-id along with a "ifmap-publisher-id" attribute that the MAP Client can use to recognize metadata that it published by examining operational attributes. (Section 4.3, "Session ID")

[CTNC-IFMAP2.0-SERVER-REQ-108-M] If the MAP Client included a max-poll-result-size attribute in its newSession request, the MAP Server MUST include a max-poll-result-size in its response indicating the actual amount of buffer space available for poll results for the client. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-109-M] If a MAP Client sends more than one SOAP request containing a “newSession” element in the SOAP body, the MAP Server MUST respond by ending the previous session and starting a new session. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-110-M] The server’s response MUST contain a “newSessionResult” element, and any state associated with the old session MUST be discarded. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-111-M] If an ARC is associated with the old session, the server MUST send an endSessionResult on the ARC. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-112-M] Each subsequent request (including a “poll” request on an ARC) MUST contain a “session-id” attribute in the top level element of the SOAP body, specifying the session-id assigned to the connection by the MAP Server. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-113-M] If the MAP Client specifies an invalid session-id, the MAP Server MUST indicate an InvalidSessionID errorResult in its response. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-114-M] A MAP Server MUST NOT accept a request from a MAP Client unless the session-id is valid for that client. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-115-M] A MAP Server MUST NOT permit one MAP Client to use a session that is created by another MAP Client. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-116-M] To explicitly terminate a session with a MAP server, a MAP Client MAY send an endSession request. If the session is valid, The MAP Server MUST respond with an endSessionResult response. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-117-M] When a session ends for any reason, and there is an outstanding poll request on the ARC, the MAP Server MUST send an endSessionResult to the MAP Client on the ARC. (Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-118-M] If a MAP Server receives a message containing a SOAP body containing a poll element that specifies a session which already has an ARC with an outstanding poll request, the MAP Server MUST:

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

(Section 4.3, “Session ID”)

[CTNC-IFMAP2.0-SERVER-REQ-119-M] If the session-id is valid, the server MUST respond with a renewSessionResult element. Otherwise, the server MUST respond with an errorResult element, specifying an InvalidSessionID errorCode. (Section 4.4, “Session Renewal”)

[CTNC-IFMAP2.0-SERVER-REQ-120-M] If a MAP Server or MAP Client detects an error at the TCP or SSL layer of an ARC, the MAP Server or Client MUST end the session. (Section 4.5, “ARC error handling”)

[CTNC-IFMAP2.0-SERVER-REQ-121-M] The IF MAP binding for SOAP described in this document requires that the IF MAP protocol MUST be carried over TLS ([8], [9], or [12]) as described in [10]. (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-SERVER-REQ-122-M] The MAP Server MUST authenticate the MAP Client either using mutual certificate-based authentication in the TLS handshake or using Basic Authentication as described in [11]. (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-SERVER-REQ-123-M] All MAP Servers and MAP Clients MUST implement both mutual certificate-based authentication and Basic Authentication. (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-SERVER-REQ-124-M] Upon successful authentication, the trusted client entities MUST be verified for authorization to serve the MAP Client role. (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-SERVER-REQ-125-M] MAP Servers MUST allow the administrator to configure read-only access for MAP Clients. (Section 6.3.2, “Securing MAP Clients”)

[CTNC-IFMAP2.0-SERVER-REQ-126-M] When generating delta pollResults, a MAP server MUST ensure that metadata elements are delivered in the same order they are received.

3.2 Requirements on IF-MAP Client

[CTNC-IFMAP2.0-CLIENT-REQ-1-M] Communication between MAP Clients and Servers MUST be authenticated and integrity-protected against unauthorized modifications enroute. (Section 2.4.2, “Secure”)

[CTNC-IFMAP2.0-CLIENT-REQ-2-M] Communication between MAP Clients and Servers MUST provide confidentiality against unauthorized disclosure. (Section 2.4.2, “Secure”)

[CTNC-IFMAP2.0-CLIENT-REQ-3-M] Communication between MAP Clients and Servers MUST NOT be susceptible to replay attacks. (Section 2.4.2, “Secure”)

[CTNC-IFMAP2.0-CLIENT-REQ-4-M] All MAP Clients and Servers MUST support both standard and vendor-specific metadata. (Section 2.6.3, “Metadata”)

[CTNC-IFMAP2.0-CLIENT-REQ-5-M] A MAP Client MUST ignore vendor-specific metadata that it does not understand. (Section 2.6.3, “Metadata”)

[CTNC-IFMAP2.0-CLIENT-REQ-6-M] All MAP Clients MAY likewise validate documents using the same mechanisms and MUST NOT send metadata that does not comply with the relevant schema. (Section 2.6.3, “Metadata”)

[CTNC-IFMAP2.0-CLIENT-REQ-7-M] MAP Clients and MAP Servers MUST exchange string fields in UTF-8. (Section 3.1, “string encoding”)

[CTNC-IFMAP2.0-CLIENT-REQ-8-M] All MAP Server and Client implementations MUST support the entire set of identifiers. (Section 3.2, “identifiers”)

[CTNC-IFMAP2.0-CLIENT-REQ-9-M] When utilizing the AccessRequestType specified in IFMAP, a MAP Client MUST NOT specify an administrative domain. (Section 3.2.1, “access-request identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-10-M] MAP Clients MUST choose the value of the name attribute in such a way that the odds of having two logically different access-request identifiers with the same name are negligible. (Section 3.2.1, “access-request identifier)

[CTNC-IFMAP2.0-CLIENT-REQ-11-M] MAP Clients MUST NOT create a device identifier with an aik-name. (Section 3.2.2, “device identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-12-M] The name attribute MUST satisfy the same uniqueness requirements as the name attribute of an access-request identifier. (Section 3.2.2, “device identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-13-M] A MAP Client MUST specify a name field consisting of a non-empty string. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-14-M] A MAP Client MUST specify a type in order to provide a hint to MAP Clients how to parse the name field. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-15-M] HIT MUST be expressed as x:x:x:x:x:x:x:x, where the 'x's are the lowercase hexadecimal values of the eight 16-bit pieces of the HIT. No leading zeros are allowed except that the number 0 is represented by a single 0 character. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-16-M] If a MAP Client specifies identity type as “other”, the client MUST specify a non-empty string for the other-type-definition field. The other-type-definition field’s value MUST take one of two forms:

- “Vendor-ID:Name”: A vendor-defined type. Vendor-ID is an SMI Private Enterprise Number [13] owned by the vendor, and Name is the type name. (Section 3.2.3, “identity identifier”)
- “Name”: A TCG-defined type. A TCG-defined type may be specified in a future version of IF MAP or in a supplement to IF MAP. (Section 3.2.3, “identity identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-17-M] IP addresses MUST be canonicalized by MAP Clients. (Section 3.2.4, “ip-address identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-18-M] MAP Clients and Servers MUST specify the MAC address as six groups of two lowercase hexadecimal digits, separated by colons (:) in transmission order, e.g. 01:23:45:67:89:ab. (Section 3.2.5, “mac-address identifier”)

[CTNC-IFMAP2.0-CLIENT-REQ-19-M] MAP Clients MUST NOT specify operational attributes in publish requests (section 3.7.1). (Section 3.3, “metadata”)

[CTNC-IFMAP2.0-CLIENT-REQ-20-M] Metadata elements MUST NOT include attributes that begin with the ifmap- prefix other than attributes specified in this document or its successors. (Section 3.3, “metadata”)

[CTNC-IFMAP2.0-CLIENT-REQ-21-M] Any unrecognized attributes beginning with the ifmap-prefix MUST be ignored by MAP Clients and MAP Servers. (Section 3.3, “metadata”)

[CTNC-IFMAP2.0-CLIENT-REQ-22-M] Every metadata item MUST include a valid ifmap-cardinality attribute (Section 3.3.3 “ifmap-cardinality”)

[CTNC-IFMAP2.0-CLIENT-REQ-23-M] MAP Clients and Servers MUST support metadata at least 100000 bytes in length, and SHOULD support longer metadata. (Section 3.3.6, “Metadata Size”)

[CTNC-IFMAP2.0-CLIENT-REQ-24-M] All FilterType values MUST be constructed with the IF MAP filter syntax (See section 3.4 for details). (Section 3.4, “Filters”)

[CTNC-IFMAP2.0-CLIENT-REQ-25-M] The namespace prefixes in a filter MUST be declared in the XML document containing the filter, and the declaration of the namespace prefix MUST apply to the scope of the filter. (Section 3.4, “Filters”)

[CTNC-IFMAP2.0-CLIENT-REQ-26-M] All requests except for newSession MUST include a session-id attribute, which is specified in the IF-MAP schema using the sessionAttributes attributeGroup. (Section 3.7, “Requests”)

[CTNC-IFMAP2.0-CLIENT-REQ-27-M] The value of the name attribute MUST be a string between 1 and 20 characters in length, such as a unique integer value for each search the client subscribes to. (Section 3.7.4.1, “subscribe update”)

[CTNC-IFMAP2.0-CLIENT-REQ-28-M] The client MUST specify the name attribute in every update element of a subscriptionRequest. (Section 3.7.4.1, “subscribe update”)

[CTNC-IFMAP2.0-CLIENT-REQ-29-M] A MAP Client receiving a notifyResult MUST NOT assume that metadata missing from a notifyResult has been deleted. (Section 3.7.5, “Poll”)

~~**[CTNC-IFMAP2.0-CLIENT-REQ-30-M]** When generating delta pollResults, a MAP server MUST ensure that metadata elements are delivered in the same order they are received~~ Not a CLIENT requirement, covered in [CTNC-IFMAP2.0-SERVER-REQ-126-M].

[CTNC-IFMAP2.0-CLIENT-REQ-31-M] A MAP Client that issues subscribe requests MUST create an ARC and issue poll requests. (Section 3.7.5, “Poll”)

[CTNC-IFMAP2.0-CLIENT-REQ-32-M] MAP Clients MUST ignore unrecognized vendor-specific metadata returned by searches and subscriptions. (Section 3.9, “Vendor-specific Metadata)

[CTNC-IFMAP2.0-CLIENT-REQ-33-M] A MAP Client MUST be capable of communicating with a MAP Server using any valid URI as specified in[10]and [7]. In particular, a MAP Client MUST be capable of communicating with a MAP Server using a tcp port specified in an https URI. (Section 4, “SOAP Binding”)

[CTNC-IFMAP2.0-CLIENT-REQ-34-M] An IFMAP session consists of one synchronous send-receive channel (SSRC) and no more than one optional asynchronous receive channel (ARC); a MAP Client MUST open exactly one SSRC and no more than one ARC. (Section 4.1.1, “Sessions”)

[CTNC-IFMAP2.0-CLIENT-REQ-35-M] The SSRC MUST NOT be used for poll requests. (Section 4.1.1.1, “SSRC”)

[CTNC-IFMAP2.0-CLIENT-REQ-36-M] The following IFMAP messages MAY be communicated over the ARC: poll and response. Other messages MUST NOT be communicated over the ARC. (Section 4.1.1.2, “ARC”)

[CTNC-IFMAP2.0-CLIENT-REQ-37-M] A MAP Client MUST have a valid session with a MAP Server before opening an ARC (Section 4.1.1.2, “ARC”)

[CTNC-IFMAP2.0-CLIENT-REQ-38-M] A MAP Client MUST NOT send any IF-MAP request other than “poll” on an ARC. (Section 4.1.1.2, “ARC”)

[CTNC-IFMAP2.0-CLIENT-REQ-39-M] A “poll” request MUST contain a session-id attribute referring to a valid session. (Section 4.1.1.2, “ARC”)

[CTNC-IFMAP2.0-CLIENT-REQ-40-M] A MAP Client MUST have no more than one ARC active at a time, meaning that a MAP Client MUST NOT have more than one outstanding “poll” request for a particular session. (Section 4.1.1.2, “ARC”)

[CTNC-IFMAP2.0-CLIENT-REQ-41-M] All implementations of the IFMAP protocol MUST support Simple Object Access Protocol (SOAP) v. 1.2 as defined in[5] (Section 4.2, “SOAP Transport”)

[CTNC-IFMAP2.0-CLIENT-REQ-42-M] Like other IF-MAP requests, renewSession request MUST specify a valid session-id. (Section 4.4, “Session Renewal”)

[CTNC-IFMAP2.0-CLIENT-REQ-43-M] If a MAP Server or MAP Client detects an error at the TCP or SSL layer of an ARC, the MAP Server or Client MUST end the session. (Section 4.5, “ARC error handling”)

[CTNC-IFMAP2.0-CLIENT-REQ-44-M] The IF MAP binding for SOAP described in this document requires that the IF MAP protocol MUST be carried over TLS ([8], [9], or [12]) as described in [10]. (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-CLIENT-REQ-45-M] The MAP Client MUST verify the MAP Server’s certificate and determine whether the MAP Server is trusted by this MAP Client before completing the TLS handshake. (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-CLIENT-REQ-46-M] All MAP Servers and MAP Clients MUST implement both mutual certificate-based authentication and Basic Authentication. (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-CLIENT-REQ-47-M] Per the HTTP specification[7], when basic authentication is in use, a MAP server MAY respond to any request that lacks credentials with HTTP code 401. A client MAY avoid this code by submitting basic auth credentials with every request. If it does not do so, a client MUST respond to this code by resubmitting the same request with credentials (unless the client is shutting down). (Section 6.3.1, “Securing the IF-MAP Protocol”)

[CTNC-IFMAP2.0-CLIENT-REQ-48-M] MAP clients MUST NOT include publisher-id or timestamp in published metadata. (Section 10.1, “Identifier Types, Requests and Responses”)

3.3 Other Requirements

Requirements listed in this section are not the requirements for the server or client. They are listed here for completeness. However, they are out of scope and we will not provide test cases.

[CTNC-IFMAP2.0-OTHERS-REQ-1-M] All metadata type schemas MUST include either the singleValueMetadataAttributes attributeGroup or the multiValueMetadataAttributes attributeGroup. (Section 3.3.3 “ifmap-cardinality”)

[CTNC-IFMAP2.0-OTHERS-REQ-2-M] SOAP intermediaries MUST NOT be used. (Section 6.3.1, “Securing the IF-MAP Protocol”)

3.4 Un-testable

Requirements listed in this section are not testable for the provided reasons:

[CTNC-IFMAP2.0-NOTEST-REQ-1-M] In order to keep an IF-MAP session from timing out, a MAP Client MUST either keep the underlying TCP connection associated with the SSRC open, or send periodic renewSession requests to the MAP Server. (Section 4.4, “Session Renewal”)

Reason: It is not possible to know if the client is letting the session timeout on purpose when inactive.

[CTNC-IFMAP2.0-NOTEST-REQ-2-M] An IF-MAP Client MUST publish multiple operations in a single publish request when the operations are atomically related. (Section 3.7.1.4, “Multiple Operations in a single publish request”)

Reason: Per discussion with map_sg, Clifford responded: It's true that a client that wants multiple operations to be treated atomically had better publish them in a single publish request. But that is a consequence of the MUST on the server. It's only guidance for the client's author. It does not seem normative at all.

[CTNC-IFMAP2.0-NOTEST-REQ-3-M] An IF-MAP Client MUST refrain from publishing multiple operations in a single publish request for all reasons other than an achieving atomic database semantics, for example, but not limited to, any attempts to increase efficiency of communication with the IF-MAP Server. (Section 3.7.1.4, “Multiple Operations in a single publish request”)

Reason: Per discussion with map_sg, Clifford responded:The directive to refrain from publishing multiple operations in a single publish request for other reasons does seem normative, but perhaps it should have been a SHOULD, since it is not testable.

[CTNC-IFMAP2.0-NOTEST-REQ-4-M] When processing pollResults, a MAP Client MUST consider a result as the combination of metadata, identifier, result type (notify, update, or delete), and subscription name, but not the manner in which the results are encoded in the XML. (Section 3.7.5, “Poll”)

Reason: Requirements as to what the client should do in processing and interpreting the sequence order of data received are not verifiable on most clients.

[CTNC-IFMAP2.0-NOTEST-REQ-5-M] A MAP Client MUST NOT rely on the ordering of elements within the XML when processing results, and MUST look at the timestamp information to determine the actual order of operation. (Section 3.7.5, “Poll”)

Reason: Requirements as to what the client should do in processing and interpreting the sequence order of data received are not verifiable on most clients.

4 Configurations and Topology

4.1 IF-MAP Server Testing Common Setup

This common setup is for IF-MAP Server testing. For Client testing, please see section 4.2

<u>IF-MAP Server</u>

The IF-MAP Server is the passive device on the topology. The MAP Server is the device under test (DUT); no functional changes should be made to the DUT during the course of the testing.

The IF-MAP Server must be configured with a valid URI to communicate with the IF-MAP Client.

The IF-MAP Server must be configured to authenticate IF-MAP Client credentials for two clients that must have full privilege to all MAP operations, identifiers, and metadata (will be referred to as ‘client’ and ‘client2’ in this test plan).

The IF-MAP Server must be also beconfigured to authenticate IF-MAP Client credentials for one client that must only have read-only permission (will be referred to as ‘client-readonly’ in this test plan).

The IF-MAP Server must be provisioned with appropriate certificate(s) required for authentication by IF-MAP Clients.

<u>IF-MAP Client</u>

A device acting as an IF-MAP client will be needed to send requests to the IF-MAP server. In IF-MAP Server testing, the MAP client will be simulated by the test program to send various controlled requests while monitoring the response from the MAP server, hence the MAP client and/or test program needs to have the following capabilities:

- Must be implemented using SOAP v1.2 [5]
- Capable of capture and validation of SOAP request and response contents.
- In some scenarios, multiple clients need to be simultaneously connected with an IF-MAP Server.Therefore, the MAP client program needs to also have the capability to create multiple instances of connection using different client credentials.
- Able to set parameters (e.g. gzip compression) in HTTP header in request packet, as well as validate HTTP header in response packet [7].
- Able to send incorrect or corrupted SSL or TCP layer in a packet.
- Able to send SOAP packets via both secured (https) and non-secured (http) connection.
- Able to authenticate via either certificate-based authentication in the TLS handshake or Basic Authentication[11].

IF-MAP Clients must be configured with a valid URI to communicate with the IF-MAP Server.

IF-MAP Clients must be configured to authenticate with the IF-MAP Server via certificate-based authentication.

4.2 IF-MAP Client Testing Common Setup

This common setup is for IF-MAP Client testing. For Server testing, please see section 4.1

IF-MAP Client

In IF-MAP Client testing, only one client is required; this client is the device under test (DUT). No functional changes should be made to the DUT during the course of the testing.

The IF-MAP Client must be configured with a valid URI to communicate with the IF-MAP Server.

The IF-MAP Client must be configured to authenticate with the IF-MAP Server via certificate-based authentication.

IF-MAP Server

The IF-MAP Server will be monitoredby the test program for the requests coming from the IF-MAP Client DUT, hence the IF-MAP Server needs to be integrated with the test program and have the ability to send controlled packets. The IF-MAP Server needs to have the following capabilities:

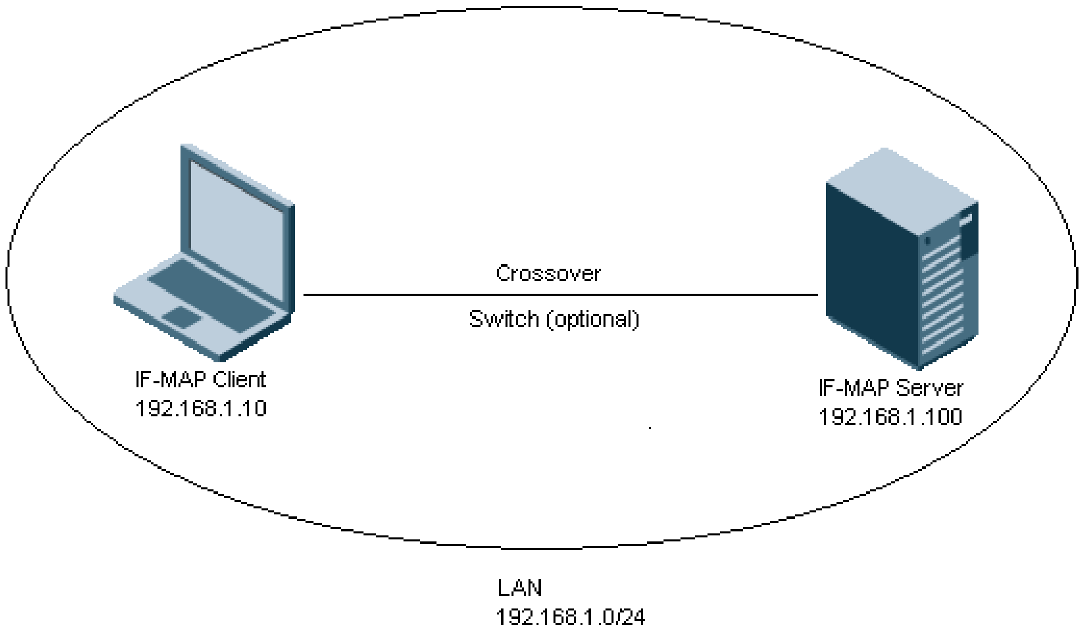
- Must be implemented using SOAP v1.2 [5]
- Capable of capture and validation of SOAP request and response contents.
- Capable of changing the listening port and absolute path. (e.g.: https://<host>:port/abs_path)
- Capable of responding with controlled http code 401 to the MAP client.
- Able to send incorrect or corrupted SSL or TCP layer in a packet.
- Able to send SOAP packets via both secured (https) and non-secured (http) connection.
- Able to generate and send controlled SOAP packet and envelopes.
- Able to authenticate the MAP client via either certificate-based authentication in the TLS handshake or Basic Authentication[11].

The IF-MAP Server must be configured to authenticate one IF-MAP Client credentials. This client must have full privilege to all MAP operations, identifiers, and metadata.

The IF-MAP Server must be provisioned with appropriate certificate(s) required for authenticationby the IF-MAP Client.

4.3 Common Setup Topology

In both IF-MAP Server testing and Client testing, the test topology is the same. However, for IF-MAP Server testing, the SOAP content will be monitored from the IF-MAP Client side. For IF-MAP Client testing, the SOAP content will be monitored from the IF-MAP Server side.



The following address assignments are used:

- 192.168.1.100 – IF-MAP Server
- 192.168.1.10 – IF-MAP Client

All devices MUST have consistent time and date. The test topology MUST be reset to default configuration at the start of every test case.

4.4 Validate Common Setup

Before running any tests, validate the test environment as follows:

Test Steps:

Validate IF-MAP Server/Client configuration:

1. Disconnect IF-MAP Clients from IF-MAP Server if connected.
    a. Maintain disconnect state until MAP Server's httpd times out the previous session (this time may vary depends on the MAP Server configuration).
2. From IF-MAP Server side, configure IF-MAP serving URI (if configurable).
3. From IF-MAP Server side, configure proper authentication for IF-MAP client.
4. From IF-MAP Client side, configure valid IF-MAP server URI.
5. From IF-MAP Client side, configure matching authentication to be used.

6. From IF-MAP Client side, initiate a newSession request, and validate that session_id and publisher_id are returned.

Expected Outcomes:

- Network traffic flows as expected.
- IF-MAP Client is able to communicate and authenticate with IF-MAP server.
- IF-MAP server recognizes the IF-MAP client and responds with publisher_id.

5 Test Cases

In test cases where there are multiple expected outcomes listed, all of the expected outcomes must be met in order to pass the test. In test cases where there are multiple anticipated failures listed, any single failure results in failing the test.

5.1 IF-MAP Compliance Test Cases for MAP Server

In the IF-MAP Compliance Test Cases for MAP Servers, the Device Under Test (DUT) is the IF-MAP Server. To verify that the MAP server correctly implements the IF-MAP 2.0 specification, the test program will send various requests and examine the respond packets to ensure anticipated response is received.

When the test is initiated, the following questionnaire needs to be answered by the test executor to allow the test program to validate the correct server implementation.

1.What is the max server limitation on search's "max-size"?

2. What is the default server session timeout?

5.1.1 Always Available

Following are "Always Available" (AA) test cases.This behaviour must be monitored throughout the test, and any mismatching behaviour must immediately be flagged as a failure.

As such, there are no specific test steps associated with these test cases; the expected outcomes should be observed in the results of all subsequent test cases.

5.1.1.1 Unrecognized attributes with ifmap- prefix

CTNC-IFMAP2.0-SERVER-AA-1

Purpose: To verify that metadata elements do not include attributes that begin with the *ifmap-* prefix other than attributes specified in this document or its successors.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-26-M]

Preconditions:

- o None.

Expected Outcomes:

- Monitor all metadata from searchResult and pollResult, MUST NOT contain metadata attributes other than 'ifmap-timestamp', 'ifmap-publisher-id', 'ifmap-cardinality'

Anticipated Failures:

- If a metadata contains an unknown attribute with prefix of *ifmap-*

5.1.1.2 XML namespaces from MAP Server

CTNC-IFMAP2.0-SERVER-AA-2

Purpose: To verify that the MAP Server includes the XML namespaces associated with the newest schema versions in any message documents

This test case is for the following requirements:[CTNC-IFMAP2.0-SERVER-REQ-92-M]

Preconditions:

- None.

Expected Outcomes:

- All response SOAP packet MUST always include, at a minimum, the following XML namespace in SOAP envelope:

  xmlns:ifmap=”http://www.trustedcomputinggroup.org/2010/IFMAP/2

Anticipated Failures:

- Response packet omits or includes incorrect XML namespace in SOAP envelope.

5.1.1.3 IFMAP protocol on SOAP v1.2

CTNC-IFMAP2.0-SERVER-AA-3

Purpose: To verify that all implementations of the IF-MAP interface support Simple Object Access Protocol (SOAP) v. 1.2.as defined in[5].

This test case is for the following requirements:[CTNC-IFMAP2.0-SERVER-REQ-103-M]

Preconditions:

- None

Expected Outcomes:

- SOAP Compliance test will be done using a 3rd party vendor SOAP compliance suite.

Anticipated Failures:

- N/A

5.1.1.4 MAP Server responds to every MAP Client request

CTNC-IFMAP2.0-SERVER-AA-4

Purpose: To verify that the MAP Server responds to every MAP Client request.If a MAP Client request cannot be processed, the MAP Server MUST respond with an errorResult element and appropriate error code and message or with a lower-level error response. Including a MAP Server

MUST respond to a valid poll with a pollResult, endSessionResult, or errorResult message. The response element sent in response to a poll request MUST contain either an appropriate pollResult element corresponding to the MAP Client's subscription, an errorResult element indicating the cause of an error, or an endSessionResult indicating that the session ended.If the session-id is valid, the server MUST respond with a renewSessionResult element.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-46-M] and [CTNC-IFMAP2.0-SERVER-REQ-47-M] and [CTNC-IFMAP2.0-SERVER-REQ-72-M] and [CTNC-IFMAP2.0-SERVER-REQ-102-M] and [CTNC-IFMAP2.0-SERVER-REQ-119-M]

Preconditions:

- None.

Expected Outcomes:

1. For each of the following requests sent to the MAP Server, the MAP Server MUST always return one of the corresponding responses:

Request	**Response**
newSession	newSessionResult or errorResult
Publish	publishReceived or errorResult
Subscribe	subscribeReceived or errorResult
Search	searchResult or errorResult
purgePublisher	purgePublisherReceived or errorResult
renewSession	renewSessionResult or errorResult
endSession	endSessionResult or errorResult
Poll	pollResult or errorResult

Anticipated Failures:

- The MAP Server returns no response or an unexpected response.

5.1.2 Test Case

5.1.2.1 Security between MAP Server and Client

CTNC-IFMAP2.0-SERVER-TC-1.

Purpose: To verify that communications between MAP Clients and Servers are authenticated and integrity-protected against unauthorized modifications enroute, provides confidentiality against unauthorized disclosure, and are not susceptible to replay attacks.The IF MAP binding for SOAP described in this document requires that the IF MAP protocol MUST be carried over TLS. The MAP Server MUST authenticate the MAP Client either using mutual certificate-based authentication in the TLS handshake or using Basic Authentication. All MAP Servers and MAP Clients MUST implement both mutual certificate-based authentication and Basic Authentication.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-1-M] and [CTNC-IFMAP2.0-SERVER-REQ-2-M] and [CTNC-IFMAP2.0-SERVER-REQ-3-M] and[CTNC-IFMAP2.0-SERVER-REQ-121-M] and[CTNC-IFMAP2.0-SERVER-REQ-122-M] and [CTNC-IFMAP2.0-SERVER-REQ-123-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server using SOAP over HTTPS with configured authentication method. Client must receive server SSL certificate.
3. Disconnect client
4. Client requests new session from server using SOAP over HTTPS without providing any authentication.
5. Client requests new session from server using SOAP over non-secured HTTP.

Expected Outcomes:

- MAP Server MUSTprovide SSL certificate for client to validate its authenticity.
- Client MUST NOTbe given session-id and publisher-id if authentication information is not provided upon request new session.
- Client MUST NOTbe able to connect using non-secured HTTP.

Anticipated Failures:

- If client is able to receive session-id even without providing authentication.
- If client is able to establish connection with non-secured HTTP.

5.1.2.2 Supports both standard and vendor-specific metadata

CTNC-IFMAP2.0-SERVER-TC-2.

Purpose: To verify that the MAP server accepts both standard and vendor-specific metadata.A MAP Server MUST allows storage and retrieval of vendor-specific metadata and MUST support operations using vendor-specific metadata which is defined by an XML schema.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-4-M] and [CTNC-IFMAP2.0-SERVER-REQ-5-M] and [CTNC-IFMAP2.0-SERVER-REQ-93-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IF-MAP publish update request to IF-MAP server with the following multiValue standard metadata in an identifier:

Identifier	Identifier: access-request Name: id1
Metadata	metadata: meta:capability name: metadata1

4. Send an IF-MAP search request to IF-MAP server to search for the standard metadata published in step#3.

Identifier	Identifier: access-request Name: id1

5. Send an IF-MAP publish update request to IF-MAP server with the following singleValue standard metadata in a link:

Link	Identifier1	Identifier: access-request Name: id2
	Identifier2	Identifier: device Name: id2
Metadata		metadata: meta:access-request-device

6. Send an IF-MAP search request to IF-MAP server to search for the standard metadata published in step#4.
7. Send an IF-MAP publish update request to IF-MAP server with the following singleValue vendor-specific metadata in an identifier:

Identifier	Identifier: identity Name: id3 type: username
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

8. Send an IF-MAP search request to IF-MAP server to search for the vendor-specific metadata published in step#7.
9. Send an IF-MAP publish update request to IF-MAP server with the following multiValue vendor-specific metadata in a link:

Link	Identifier1	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
	Identifier2	Identifier: device Name: id4
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy1

10. Send an IF-MAP search request to IF-MAP server to search for the vendor-specific metadata published in step#9.
11. Disconnect client

Expected Outcomes:

- Publish standard metadata in step#3 and 5 MUST respond in “publishReceived”
- Search for standard metadata in step#4 and 6 MUST respond with the identical metadata as published.
- Publish vendor-specific metadata in step#7 and 9 MUST respond in “publishReceived”
- Search for vendor-specific metadata in step#8 and 10 MUST respond with the identical metadata as published.

Anticipated Failures:

- If server is responding the metadata in different namespace than it was published.

5.1.2.3 Server supported string fields in UTF8

CTNC-IFMAP2.0-SERVER-TC-3.

Purpose: To verify that the MAP server properly stores and responds with string fields in UTF8 format.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-6-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request(s) to IF-MAP server with following UTF8 characters:

identifier	Identifier: identity Name: 中文 type: username
Metadata	Xmlns: español=http://ns1.example.com metadata: español: ελληνικά Cardinality: singleValue Element name: עברית Element value: العربية

4. Send IF-MAP subscribe request(s) to IF-MAP server to using the following search parameters with UTF8 characters:

Subscribe name: Việt	Identifier	Identifier: identity Name: 中文 type: username
	Metadata	Xmlns: español=http://ns1.example.com Result-filter: español: ελληνικά[עברית=“العربية”]

5. Open an ARC channel and send an IF-MAP poll request to IF-MAP server.
6. Disconnect client

Expected Outcomes:

- Publish metadata in UTF8 MUST respond in "publishReceived"
- Subscribe with UTF8 search name MUST response in "subscribeReceived"
- Poll response MUST return all identifier, metadata and subscribe name in its original UTF8 values from step#3 and 4.

Anticipated Failures:

- If none UTF8 encoding has been found on the respond packet.

5.1.2.4 Server supported entire set of identifiers

CTNC-IFMAP2.0-SERVER-TC-4.

Purpose: To verify that the MAP server accepts the entire set of identifiers.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-7-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request(s) to IF-MAP server with the following entire set of identifiers:

Link	Identifier1	Identifier: access-request Name: id1
	Identifier2	Identifier: device Name: id2
Metadata		metadata: meta:access-request-device

Link	Identifier1	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
	Identifier2	Identifier: mac-address Value: 01:02:03:0a:0b:0c
Metadata		metadata: meta:ip-mac start-time: 2010-01-01T00:00:01 end-time: 2010-12-31T23:59:59 dhcp-server: 4.3.2.1

identifier	Identifier: identity Name: id1 type: username
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1

	Cardinality: singleValue Element name: name Element value: dummy1

4. Send an IF-MAP search request to IF-MAP server to retrieve the metadata that was published on step#3.
5. Disconnect client

Expected Outcomes:

- Publish to the identifiers in step3 MUST responds in “publishReceived”
- Search response MUST contain the identifier and values used during publish.

Anticipated Failures:

- If server is rejecting any of the 5 types of identifiers
- If server is responding any identifier types that are not supported.

5.1.2.5 Server supported identifiers size

CTNC-IFMAP2.0-SERVER-TC-5.

Purpose: To verify that the MAP server supports identifiers containing 1000 bytes of data

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-8-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server using the following identifiers containing 1000 bytes of data:

Identifier	Identifier: access-request Name: a string with 1000 ASCII characters
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

Identifier	Identifier: device Name: a string with 1000 ASCII characters
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

Identifier	Identifier: identity Type: username Name: a string with 1000 ASCII characters

Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

Identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4 Administrative-domain: a string with 993 ASCII characters
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

Identifier	Identifier: mac-address Value: 01:02:03:0a:0b:0c Administrative-domain: a string with 983 ASCII characters
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Send an IF-MAP search request to IF-MAP server to retrieve the identifiers that were published in step#3
5. Disconnect client

Expected Outcomes:

- Publish to the identifiers in step3 MUST responds in "publishReceived"
- Search response MUST contain the identifier and values used during publish.

Anticipated Failures:

- If server rejects identifier with size of 1000 bytes.
- If server does not return the exact identifier published.

5.1.2.6 "access-request" identifiers equivalence

CTNC-IFMAP2.0-SERVER-TC-6.

Purpose: To verify that the MAP server processes two access-request identifiers as equivalent if and only if ALL corresponding fields in the two access-request identifiers are equivalent, and MAP server processes the administrative domain field as CASE SENSITIVE

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-9-M] and [CTNC-IFMAP2.0-SERVER-REQ-10-M]

Preconditions:

- o Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.

2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server with following identifier and metadata:

identifier	Identifier: access-request Name: AbCd 1234 Administrative-domain: AbCd 1234
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Send an IF-MAP search request to IF-MAP server using the same identifier values as published.

identifier	Identifier: access-request Name: AbCd 1234 Administrative-domain: AbCd 1234

5. Send an IF-MAP search request to IF-MAP server using same identifier values but name in different case.

identifier	Identifier: access-request Name: ABCD 1234 Administrative-domain: AbCd 1234

6. Send an IF-MAP search request to IF-MAP server using same identifier values but administrative-domain in different case.

identifier	Identifier: access-request Name: AbCd 1234 Administrative-domain: ABCD 1234

7. Send an IF-MAP search request to IF-MAP server using same identifier values but missing administrative-domain.

identifier	Identifier: access-request Name: AbCd 1234

8. Disconnect client

Expected Outcomes:

- Publish to the identifiers in step3 MUST responds in “publishReceived”
- Search using exact string and case in step#4 MUST return the matching metadata that was published in step#3.
- Search using incorrect case or missing value in step #5~7 MUST NOT return the metadata that was published in step#3.

Anticipated Failures:

- If server returned the published metadata by searching the identifier in different casing, different value or different administrative-domain.

5.1.2.7 Empty string or unspecified administrative-domains are equivalent

CTNC-IFMAP2.0-SERVER-TC-7.

Purpose: To verify that the MAP Server processes as equivalent (i.e. belonging to the same administrative domain) administrative domains which are specified as an empty string or unspecified in access-request, identity, ip-address and mac-address identifiers.

This test case is for the following requirements:[CTNC-IFMAP2.0-SERVER-REQ-11-M] and [CTNC-IFMAP2.0-SERVER-REQ-15-M] and[CTNC-IFMAP2.0-SERVER-REQ-21-M] and [CTNC-IFMAP2.0-SERVER-REQ-24-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server using following identifier

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue

4. Send an IF-MAP search request to IF-MAP server unspecified administrative-domain.

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

5. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain.

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4 Administrative-domain: ‘’

6. Send IF-MAP publish update request to IF-MAP server using following identifier.

identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:6:7:8 Administrative-domain: ‘’
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue

7. Send an IF-MAP search request to IF-MAP server with unspecified administrative-domain.

identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:6:7:8

8. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain.

identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:6:7:8 Administrative-domain: ‘’

9. Send IF-MAP publish update request to IF-MAP server using following identifier

identifier	Identifier: mac-address Value: 01:02:03:04:05:0a
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue

10. Send an IF-MAP search request to IF-MAP server unspecified administrative-domain.

identifier	Identifier: mac-address Value: 01:02:03:04:05:0a

11. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain.

identifier	Identifier: mac-address Value: 01:02:03:04:05:0a Administrative-domain: ‘’

12. Send IF-MAP publish update request to IF-MAP server using following identifier.

identifier	Identifier: mac-address Value: 01:02:03:04:05:0b Administrative-domain: ‘’
Metadata	name:mac-address

13. Send an IF-MAP search request to IF-MAP server with unspecified administrative-domain

identifier	Identifier: mac-address Value: 01:02:03:04:05:0b

14. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain

identifier	Identifier: mac-address Value: 01:02:03:04:05:0b Administrative-domain: ‘’

15. Send IF-MAP publish update request to IF-MAP server using following identifier

identifier	Identifier: access-request Name: id1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue

16. Send an IF-MAP search request to IF-MAP server with unspecified administrative-domain.

identifier	Identifier: access-request Name: id1

17. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain.

identifier	Identifier: access-request Name: id1 Administrative-domain: ‘’

18. Send IF-MAP publish update request to IF-MAP server using following identifier.

identifier	Identifier: access-request Name: id2 Administrative-domain: ‘’
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

19. Send an IF-MAP search request to IF-MAP server with unspecified administrative-domain.

identifier	Identifier: access-request Name: id2

20. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain.

identifier	Identifier: access-request Name: id2 Administrative-domain: ‘’

21. Send IF-MAP publish update request to IF-MAP server using following identifier

identifier	Identifier: identity Type: username Name: id1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue

22. Send an IF-MAP search request to IF-MAP server unspecified administrative-domain.

identifier	Identifier: identity Type: username Name: id1

23. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain.

identifier	Identifier: identity Type: username Name: id1 Administrative-domain: ‘’

24. Send IF-MAP publish update request to IF-MAP server using following identifier.

identifier	Identifier: identity Type: username Name: id2 Administrative-domain: ‘’
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

25. Send an IF-MAP search request to IF-MAP server with unspecified administrative-domain.

identifier	Identifier: identity Type: username Name: id2

26. Send an IF-MAP search request to IF-MAP server with empty string administrative-domain.

identifier	Identifier: identity Type: username Name: id2 Administrative-domain: ‘’

27. Disconnect client.

Expected Outcomes:

- Publish to the identifiers in step#3 and step#9, step#15, step#21 MUST responds in "publishReceived"
- Searches using empty string or unspecified administrative-domain in step#4 and step#5 MUST find metadata published in step#3.
- Searches using empty string or unspecified administrative-domain in step#7 and step#8 MUST find metadata published in step#6.
- Searches using empty string or unspecified administrative-domain in step#10 and step#11 MUST find metadata published in step#9.
- Searches using empty string or unspecified administrative-domain in step#13 and step#14 MUST find metadata published in step#12.
- Searches using empty string or unspecified administrative-domain in step#16 and step#17 MUST find metadata published in step#15.
- Searches using empty string or unspecified administrative-domain in step#19 and step#20 MUST find metadata published in step#18.
- Searches using empty string or unspecified administrative-domain in step#22 and step#23 MUST find metadata published in step#21.
- Searches using empty string or unspecified administrative-domain in step#25 and step#26 MUST find metadata published in step#24.

Anticipated Failures:

- If an identifier with undefined administrative-domain is not being treated the same as a defined by empty administrative-domain. Vice versa.

5.1.2.8 "device" identifiers equivalence

CTNC-IFMAP2.0-SERVER-TC-8.

Purpose: To verify that the MAP server processes two device identifiers as equivalent if and only if the corresponding names in the two device identifiers are equivalent in both type (i.e. name or aik-name) and value.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-12-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server with metadata, using a “device” identifier containing the following value:

identifier	Identifier: device Name: string1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Send an IF-MAP search request to IF-MAP server to retrieve the “device” identifier using the same value.

identifier	Identifier: device Name: string1

5. Send an IF-MAP search request to IF-MAP server to retrieve the “device” identifier using the different value.

identifier	Identifier: device Name: string2

6. Send IF-MAP publish update request to IF-MAP server with metadata, using an “device” identifier contain the following value:

identifier	Identifier: device Aik-name: string1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

7. Send an IF-MAP search request to IF-MAP server to retrieve the “device” identifier using the same value.

identifier	Identifier: device Aik-name: string1

8. Send an IF-MAP search request to IF-MAP server to retrieve the “device” identifier using the different type.

identifier	Identifier: device Name: string1

9. Disconnect client

Expected Outcomes:

- Publish to the identifiers in step3 and 6 MUST responds in “publishReceived”
- Search with same string in the same type MUST find the matching identifier and metadata in step#4 and 7
- Search with different string or different type MUSTNOT find the matching identifier and metadata in step#5 and 8

Anticipated Failures:

- If server returned the published metadata by searching the identifier in different casing, different value or different administrative-domain

5.1.2.9 "identity" identifiers equivalence

CTNC-IFMAP2.0-SERVER-TC-9.

Purpose: To verify that the MAP server processes two identity identifiers as equivalent if and only if ALL corresponding fields in the two identity identifiers are equivalent, and MAP server processes the administrative domain field as CASE SENSITIVE. And a MAP Server MUST process two syntactically equal name fields as distinct if they are associated with distinct types.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-13-M] and [CTNC-IFMAP2.0-SERVER-REQ-14-M] and [CTNC-IFMAP2.0-SERVER-REQ-17-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server with metadata of any kind, using an "identity" identifier contain the following mixed cases values:

identifier	Identifier: identity Type: aik-name Name: AbCd 1234 Administrative-domain: AbCd 1234
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Send an IF-MAP search request to IF-MAP server to retrieve an "identity" identifier contain the same value and case as published.

identifier	Identifier: identity Type: aik-name Name: AbCd 1234 Administrative-domain: AbCd 1234

5. Send an IF-MAP search request to IF-MAP server to retrieve an "identity" identifier containing different type.:

identifier	Identifier: identity Type: username Name: AbCd 1234

	Administrative-domain: AbCd 1234

6. Send an IF-MAP search request to IF-MAP server to retrieve an "identity" identifier containing same value but different case.

identifier	Identifier: identity Type: aik-name Name: ABCD 1234 Administrative-domain: AbCd 1234

7. Send an IF-MAP search request to IF-MAP server to retrieve an "identity" identifier containing same value but different case.

identifier	Identifier: identity Type: aik-name Name: AbCd 1234 Administrative-domain: ABCD 1234

8. Send an IF-MAP search request to IF-MAP server to retrieve an "identity" identifier containing missing administrative-domain.

identifier	Identifier: identity Type: aik-name Name: AbCd 1234

9. Disconnect client

Expected Outcomes:

- Publish to the identifiers in step3 MUST responds in "publishReceived"
- Search with all type, name and administrative-domain in the same string and case MUST find the metadata published in step#3.
- Search with different type in step#5 MUST NOT find the metadata published in step#3.
- Search with name in difference casing in step#6 MUST NOT find the metadata published in step#3.
- Search with administrative-domain in difference casing in step#7 MUST NOT find the metadata published in step#3.
- Search with missing administrative-domain in step#8 MUST NOT find the metadata published in step#3.

Anticipated Failures:

- If server returned the published metadata by searching the identifier in different casing, different value or different administrative-domain

5.1.2.10 "identity" identifier name enumeration

CTNC-IFMAP2.0-SERVER-TC-10.

Purpose: To verify that the MAP Server processes the name field according to the syntax specified by the type enumeration.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-16-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type aik-name.

identifier	Identifier: identity Type: aik-name Name: abc
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type distinguished-name.

identifier	Identifier: identity Type: distinguished-name Name: abc
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

5. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type dns-name.

identifier	Identifier: identity Type: dns-name Name: abc.abc.com
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

6. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type email-address.

identifier	Identifier: identity Type: email-address Name: abc@abc.com
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

7. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type kerberos-principal.

identifier	Identifier: identity Type: kerberos-principal Name: primary/instance@REALM
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

8. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type username.

identifier	Identifier: identity Type: username Name: abc
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

9. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type sip-uri.

identifier	Identifier: identity Type: sip-uri Name: sip:user@host:1234
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

10. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type tel-uri.

identifier	Identifier: identity Type: tel-uri Name: tel:+1-23-456-7890
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

11. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type hip-hit.

identifier	Identifier: identity Type: hip-hit Name: 2001:10:7654:3210:fedc:ba98:7654:3210
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

12. Send IF-MAP publish update request to IF-MAP server using following identify identifier with type other.

identifier	Identifier: identity Type: other

	Other-type-definition: abc Name: abc
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

13. Send IF-MAP publish update request to IF-MAP server using following identify identifier with missing type.

identifier	Identifier: identity Name: abc
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

14. Disconnect client

Expected Outcomes:

- Publish with name and type MUST result in “publishReceived”
- Publish with unspecified type in step#13 MUST result in a proper error regarding the missing type. Error could come from SOAP level as an incorrect xml grammar or from IF-MAP server.

Anticipated Failures:

- If any of the identifier was rejected by the server.

5.1.2.11 HIP-HIT syntax

CTNC-IFMAP2.0-SERVER-TC-11.

Purpose: To verify that the MAP Server accepts HIT expressed as x:x:x:x:x:x:x:x, where the 'x's are the lowercase hexadecimal values of the eight 16-bit pieces of the HIT. No leading zeros are allowed except that the number 0 is represented by a single 0 character.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-18-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server with “identity” identifier with type=’hip-hit’ and a correct name of one hex in each column.

identifier	Identifier: identity Type: hip-hit Name: 1:2:3:4:c:d:e:f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Send IF-MAP publish update request to IF-MAP server with “identity” identifier with type=’hip-hit’ and a correct name of max allowed hexes in each column

identifier	Identifier: identity Type: hip-hit Name: 1234:5678:90ab:cdef:1234:5678:90ab:cdef
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

5. Send IF-MAP publish update request to IF-MAP server with “identity” identifier with type=’hip-hit’ and a correct name of all 0

identifier	Identifier: identity Type: hip-hit Name: 0:0:0:0:0:0:0:0
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

6. Send IF-MAP publish update request to IF-MAP server with “identity” identifier with type=’hip-hit’ and a name with an empty column

identifier	Identifier: identity Type: hip-hit Name: 1:2:3:4:5::7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

7. Send IF-MAP publish update request to IF-MAP server with “identity” identifier with type=’hip-hit’ and a name of extra hex.

identifier	Identifier: identity Type: hip-hit Name: 1:2:3:4:5:6:7:8:9
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

8. Send IF-MAP publish update request to IF-MAP server with “identity” identifier with type=’hip-hit’ and a name of containing upper case.

identifier	Identifier: identity Type: hip-hit Name: 1:2:F:4:5:6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

9. Send IF-MAP publish update request to IF-MAP server with “identity” identifier with type=’hip-hit’ and a name of contains leading zero

identifier	Identifier: identity Type: hip-hit

	Name: 1:2:3:04:5:6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

10. Disconnect client

Expected Outcomes:

- Publish in steps#3~5 MUST result in "publishReceived"
- Publish in steps#6~9 MUST result in proper errorResult from MAP server regarding the invalid syntax of hip-hit.

Anticipated Failures:

- If server rejects a properly formatted hip-hit.
- If server accepted a invalid hip-hit

5.1.2.12 "ip-addresses" identifiers in canonical form

CTNC-IFMAP2.0-SERVER-TC-12.

Purpose: To verify that the MAP Server rejects IP addresses which are not in canonical form.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-19-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IF-MAP publish update request to IF-MAP server to publish IPv4 address with proper IPv4 canonical syntax.

identifier	Identifier: ip-address Type: ipv4 Value: 0.0.0.0
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue

identifier	Identifier: ip-address

	Type: ipv4 Value: 255.255.255.255
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

4. Send the following IF-MAP publish update requests to IF-MAP server to publish IPv4 address with Incorrect IPv4 canonical syntax: (send each request individually)

Missing an octet:

identifier	Identifier: ip-address Type: ipv4 Value: 192.0.0.
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Extra octet:

identifier	Identifier: ip-address Type: ipv4 Value: 192.0.0.1.1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Negative octet:

identifier	Identifier: ip-address Type: ipv4 Value: 192.0.-1.1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Out of bound octet:

identifier	Identifier: ip-address Type: ipv4 Value: 192.256.0.1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2

	cardinality: multiValue

Non-integer octet:

identifier	Identifier: ip-address Type: ipv4 Value: 192.a.0.1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Wildcard octet:

identifier	Identifier: ip-address Type: ipv4 Value: 192.0.*.1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Leading zero:

identifier	Identifier: ip-address Type: ipv4 Value: 192.0.01.1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

5. Send an IF-MAP publish update request to IF-MAP server to Publish IPv6 IpAddres identifier with proper IPv6 canonical syntax.

identifier	Identifier: ip-address Type: ipv6 Value: ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

identifier	Identifier: ip-address

	Type: ipv6 Value: 0:0:0:0:0:0:0:0
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

6. Send following IF-MAP publish update requests to IF-MAP server to Publish IPv6 IpAddres identifier with incorrect IPv6 canonical syntax. (send each request individually)

Extra hex:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:6:7:8:9
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Missing hex:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:6:7
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Uppercase hex:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:A:5:6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Leading zero:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:03:4:5:6:7:8

Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Negative hex:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:-6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Out of bound hex:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:g:5:6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Out of bound hex:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:aabbcc:5:6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

IPv6 short format:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3::7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Wildcard hex:

Identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:*:5:6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

7. Disconnect client.

Expected Outcomes:

- Publish Ipv4 with proper canonical syntax in step#3 MUST respond in “publishReceived“.
- Publish Ipv6 with proper canonical syntax in step#5 MUST respond in “publishReceived“.
- Publish Ipv4 with each incorrect canonical syntax in step#4 MUST respond in error.
- Publish Ipv6 with each incorrect canonical syntax in step#6 MUST respond in error.

Anticipated Failures:

- If server rejects a properly formatted canonical IP address.
- If server accepts an invalid IP address.

5.1.2.13 Administrative-domain is CASE SENSITIVE.

CTNC-IFMAP2.0-SERVER-TC-13.

Purpose: To verify that the MAP Server is processing the administrative domain field as CASE SENSITIVE in ip-address and mac-address identifiers.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-20-M] and [CTNC-IFMAP2.0-SERVER-REQ-23-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server with metadata of any kind, using an “ip-address” identifier containing the following mixed case values:

identifier	Identifier: ip-address Type: ipv4 Value: 1.1.1.1 Administrative-domain: AbCd 1234
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

4. Send an IF-MAP search request to IF-MAP server using the same identifier values as published.

identifier	Identifier: ip-address Type: ipv4 Value:1.1.1.1 Administrative-domain: AbCd 1234

5. Send an IF-MAP search request to IF-MAP server using same identifier values but name in different case.

identifier	Identifier: ip-address Type: ipv4 Value: 1.1.1.1 Administrative-domain: ABCD 1234

6. Send an IF-MAP search request to IF-MAP server to retrieve an “ip-address” identifier containing missing administrative-domain.

identifier	Identifier: ip-address Type: ipv4 Value: 1.1.1.1

7. Send IF-MAP publish update request to IF-MAP server with metadata, using a “mac address” identifier containing the following mixed case values:

identifier	Identifier: mac-address Value: 0a:0b:0c:01:02:03 Administrative-domain: AbCd 1234
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

8. Send an IF-MAP search request to IF-MAP server to retrieve a “mac-address” identifier containing the same value and case as published.

identifier	Identifier: mac-address Value: 0a:0b:0c:01:02:03 Administrative-domain: AbCd 1234

9. Send an IF-MAP search request to IF-MAP server to retrieve an “mac-address” identifier containing same value but different case

identifier	Identifier: mac-address Value: 0a:0b:0c:01:02:03 Administrative-domain: ABCD 1234

10. Send an IF-MAP search request to IF-MAP server to retrieve an “ip-address” identifier containing missing administrative-domain

identifier	Identifier: mac-address Value: 0a:0b:0c:01:02:03

11. Disconnect client.

Expected Outcomes:

- Publish to the identifiers in step#3 and step#7 MUST responds in “publishReceived”
- Search with all type, name and administrative-domain in the same string and case MUST find the metadata published in step#4 and step#8.
- Search with administrative-domain in difference casing in step#5 MUST NOT find the metadata published in step#3.
- Search with missing administrative-domain in step#6 MUST NOT find the metadata published in step#3.
- Search with administrative-domain in difference casing in step#9 MUST NOT find the metadata published in step#7.
- Search with missing administrative-domain in step#10 MUST NOT find the metadata published in step#7.

Anticipated Failures:

- If an identifier with undefined administrative-domain is not being treated the same as a defined by empty administrative-domain. Vice versa.

5.1.2.14 mac-address syntax

CTNC-IFMAP2.0-SERVER-TC-14.

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

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-22-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send a IFMAP publish update request to IFMAP server with proper Mac address syntax

identifier	Identifier: mac-address Value: 00:00:00:00:00:00
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

identifier	Identifier: mac-address Value: ff:ff:ff:ff:ff:ff
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

4. Send following IFMAP publish update requests to IFMAP server with incorrect Mac address syntax. (Send each request individually)

Extra hex:

Identifier	Identifier: mac-address Value: 01:02:03:04:05:06:07
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Missing hex:

Identifier	Identifier: mac-address Value: 01:02:03:04:05

Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Missing hex:

Identifier	Identifier: mac-address Value: 01:02:03:0A:05:06
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Missing leading zero:

Identifier	Identifier: mac-address Value: 01:02:3:04:05:06
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Negative hex:

Identifier	Identifier: mac-address Value: 01:02:03:-04:05:06
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Out of bound hex:

Identifier	Identifier: mac-address Value: 01:02:03:0g:05:06
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Out of bound hex:

Identifier	Identifier: mac-address Value: 01:02:03:aaa:05:06
Metadata	Xmlns: myns1=http://ns1.example.com

	metadata: myns1: mymeta2 cardinality: multiValue

Incorrect separators:

Identifier	Identifier: mac-address Value: 01-02 03,04.05_06
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

Wildcard hex:

Identifier	Identifier: mac-address Value: 01:02:03:*:05:06
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue

5. Disconnect client.

Expected Outcomes:

- Publish Mac address with proper syntax in step#3 MUST result in “publishReceived”.
- Publish Mac address with each incorrect syntax in step#4 MUST result in error.

Anticipated Failures:

- If server rejects a properly formatted MAC address.
- If server accepts a invalid MAC address.

5.1.2.15 Operations attributes in searchResponse, and default time zone.

CTNC-IFMAP2.0-SERVER-TC-15.

Purpose: To verify that MAP Servers include operational attributes in search responses and interpret a ifmap-timestamp without a timezone component as UTC time.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-25-M] and [CTNC-IFMAP2.0-SERVER-REQ-33-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server. Capture session-id.
3. Send publish request with the following data:

identifier	Identifier: device Name: id1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Send search request with the following data:

identifier	Identifier: device Name: id1

5. Disconnect client.

Expected Outcomes:

- Searched metadata from step#4 MUST contain attribute “ifmap-publisher-id” the same as client’s publisher-id.
- Searched metadata from step#4 MUST contain attribute “ifmap-timestamp”. The value of “ifmap-timestamp” MUST be the time of publication at step#3. This value MUST be in ISO8601 format with the value in UTC.

Anticipated Failures:

- If ifmap-publisher-id and ifmap-timestamp is not returned with each metadata.
- If the timestamp is not in ISO8601 format.

5.1.2.16 Ignore unrecognized attributes with ifmap- prefix

CTNC-IFMAP2.0-SERVER-TC-16.

Purpose: To verify that the MAP Server ignores any unrecognized attributes beginning with the ifmap- prefix.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-27-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server. Capture session-id.
3. Send publish request with the following data:

identifier	Identifier: device Name: id1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue ifmap-blah: dummystring

4. Send search request with the following data:

identifier	Identifier: device Name: id1

5. Disconnect client.

Expected Outcomes:

- •Searched metadata from step#4 MUST NOT contain attribute “ifmap-blah.

Anticipated Failures:

- •If returned metadata contains unknown attribute with prefix of ifmap-

5.1.2.17 Uniqueness of ifmap-publisher-id

CTNC-IFMAP2.0-SERVER-TC-17.

Purpose: To verify that the MAP Server does not assign the same ifmap-publisher-id to multiple different MAP Clients, the MAP Server consistently uses the same ifmap-publisher-id for a particular MAP Client, and the ifmap-publisher-id is not a function of time or of any connection specific information

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-28-M] and [CTNC-IFMAP2.0-SERVER-REQ-29-M] and [CTNC-IFMAP2.0-SERVER-REQ-32-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.

2. Client requests new session from server. Capture publisher-id.
3. Disconnect client.
4. Client2 request new session from server. Capture publisher-id.
5. Disconnect client2.
6. Client requests another new session from server. Capture publisher-id.
7. Disconnect client.
8. Client2 request another new session from server. Capture publisher-id.
9. Disconnect client2.
10. Change interface IP from 192.168.1.10 to 192.168.1.20, and have client request new session from server. Capture publisher-id.
11. Disconnect client.

Expected Outcomes:

- Publisher-id for client and client2 MUST NOT be the same.
- Publisher-id for client in both step#2, step#6 and step#10 MUST be the same.
- Publisher-id for client2 in both step#4 and step#8 MUST be the same.

Anticipated Failures:

- If the same publisher-id is being used by two different clients.

5.1.2.18 Credential changes shouldn’t affect ifmap-publisher-id

CTNC-IFMAP2.0-SERVER-TC-18.

Purpose: To verify that the MAP Server ifmap-publisher-id is not a function of specific credentials, and the MAP Client is able to change credentials (e.g. new cert after expiration) and continue to be assigned the same ifmap-publisher-id.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-30-M] and [CTNC-IFMAP2.0-SERVER-REQ-31-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server. Capture publisher-id.
3. Disconnect client.
4. From MAP server, change client credential (certificate, or username/password)

5. Update client to use the updated matching credential, then request new session from server, capture publisher-id.
6. Disconnect client.

Expected Outcomes:

- Client's publisher-id MUST be the same before and after change credential.

Anticipated Failures:

- If publisher-id is changed after credential change.

5.1.2.19 Invalid Cardinality

CTNC-IFMAP2.0-SERVER-TC-19.

Purpose: To verify that every metadata item includes a valid ifmap-cardinality, and when the MAP Server detects missing ifmap-cardinality or a change of ifmap-cardinality, it returns an Invalid Metadata error.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-34-M][CTNC-IFMAP2.0-SERVER-REQ-35-M] [CTNC-IFMAP2.0-SERVER-REQ-36-M] and [CTNC-IFMAP2.0-SERVER-REQ-37-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1 Begin tracking SOAP transactions contents from client side.

2 Client requests new session from server.

3 Send an IF-MAP publish update request to IF-MAP server with multiValue standard metadata on, but specify cardinality as singleValue.

Identifier	Identifier: access-request Name: id1
Metadata	metadata: capability name: dummy cardinality: singleValue

4 Send an IF-MAP publish update request to IF-MAP server with singleValue standard metadata on given link, but specify cardinality as multiValue.

Link	Identifier1	Identifier: access-request Name: id1
	Identifier2	Identifier: device Name: id2

Metadata	metadata: meta:access-request-device cardinality: multiValue

5 Send an IF-MAP publish update request to IF-MAP server with a vendor-specific metadata on given identifier as a multiValue metadata.

Identifier	Identifier: ip-address Type: ipv4 Name: 2.2.2.2
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue Element name: name Element value: dummy1

6 Send an IF-MAP publish update request to IF-MAP server with the same vendor-specific metadata on given link, but specify cardinality as singleValue.

Identifier	Identifier: ip-address Type: ipv4 Name: 2.2.2.2
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: singleValue Element name: name Element value: dummy2

7 Send an IF-MAP publish update with null cardinality

Identifier	Identifier: ip-address Type: ipv4 Name: 2.2.2.2
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: (null) Element name: name Element value: dummy2

8 Search for metadata on ip-address 2.2.2.2

identifier	Identifier: ip-address Type: ipv4

	Value: 2.2.2.2

9 Disconnect client

Expected Outcomes:

- Publish mismatching metadata cardinality in step#3, 4, 6, and 7 MUST respond in "InvalidMetadata"
- Update in step#5 succeeds.
- Search in step 8 returns only mymeta2 name=dummy1.

Anticipated Failures:

- If server accepts incorrect cardinality

5.1.2.20 Metadata with lifetime="session"

CTNC-IFMAP2.0-SERVER-TC-20.

Purpose: To verify that if an element was published with lifetime="session" and the client session ends, either due to inactivity (see Section 4.1.1) or at the client's request, the MAP server deletes the metadata, and the deletion is completed before the publishing client is allowed to create another session

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-40-M] and [CTNC-IFMAP2.0-SERVER-REQ-42-M]

Preconditions:

- Devices configured to "Common Setup".
- On devices which support a configurable session timeout, session timeout (period of client inactivity) set to the minimum duration supported by the server.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP publish update request to IFMAP server to publish metadata with lifetime=session.

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1

	Cardinality: singleValue Element name: name Element value: dummy1
Lifetime	Session

4. End the client session by closing the TCP connection or staying inactive for more than the minimum session timeout supported by the server.
5. Client requests new session from server and searches for the following metadata:

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

6. Send an IFMAP publish update request to IFMAP server to publish metadata with lifetime=session.

identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:6:7:8
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1
Lifetime	Session

7. End the client session by sending an endSession request.
8. Client requests new session from server and searches for the following metadata:

identifier	Identifier: ip-address Type: ipv6 Value: 1:2:3:4:5:6:7:8

9. Send an IFMAP publish update request to IFMAP server to publish metadata with lifetime=session.

identifier	Identifier: mac-address Value: 01:02:03:04:05:06
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

	Element name: name Element value: dummy1
Lifetime	Session

10. End the existing client session by requesting another session instance using the same client credential.
11. Using new client session, search for the following metadata:

identifier	Identifier: mac-address Value: 01:02:03:04:05:06

12. Disconnect the client.

Expected Outcomes:

- Publish metadata in step#3, step#9 and step#9 MUST respond in “publishReceived“.
- Search MUST NOT find metadata in step#5, step#8 and step#11.

Anticipated Failures:

- If metadata with lifftime=session does not get deleted when the publish client has ended the current session.

5.1.2.21 Notify with lifetime

CTNC-IFMAP2.0-SERVER-TC-21.

Purpose: To verify that the lifetime attribute is meaningful only in update requests, and the MAP Server ignores it if it appears in a notify request.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-41-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP subscribe request with following search, and then poll twice (ignore initial poll result).

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

4. Send an IFMAP publish notify request to IFMAP server to publish metadata element with lifetime=session.

Lifetime	Session
identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata	Metadata: meta:event Name: attack Discovered-time: 2009-05-30T13:10:11 Discovered-id: 1273 Magnitude: 45 Confidence: 100 Significance: important Type: worm infection

5. Check poll result, and also send an IFMAP search request to search for the metadata elements published.

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

6. Poll again using existing subscription.
7. Send an IFMAP publish notify request to IFMAP server to publish metadata elements with Lifetime=”forever”.

Lifetime	Forever
identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

8. Check poll result, and also send an IFMAP publish search request to IFMAP server:

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

9. Disconnect client.

Expected Outcomes:

- Publish metadata in step#4 and step#7 with lifetime MUST respond in "publishReceived".
- Poll results from step#5 and step#8 MUST contain the notifications published at step#4 and step#7.
- Search result from step#5 and step#8 MUST NOT contain the notifications published at step#4 and step#7.

Anticipated Failures:

- Server throws error if lifetime is specified on notify

5.1.2.22 Filter in delete

CTNC-IFMAP2.0-SERVER-TC-22.

Purpose: To verify that if a delete request has no filter, the MAP server deletes all metadata associated with the identifier or link, and if a delete request has a filter that is the empty string, the MAP server deletes nothing.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-42-M] and [CTNC-IFMAP2.0-SERVER-REQ-56-M] and [CTNC-IFMAP2.0-SERVER-REQ-57-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP publish update request to IFMAP server.

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab

Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Link	Identifier1	Identifier: mac-address Value: 01:02:03:04:05:ab
	Identifier2	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata		metadata: meta:ip-mac Cardinality: multiValue Element name: name Element value: dummy1

4. Send an IFMAP publish delete request to IFMAP server with empty string.

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab
Result-filter	“” (empty string)

Link	Identifier1	Identifier: mac-address Value: 01:02:03:04:05:ab
	Identifier2	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Result-filter		“” (empty string)

5. Send an IFMAP search request to IFMAP server.

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab
Max-depth	1

6. Send an IFMAP publish delete request to IFMAP server with no filter.

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab

Link	Identifier1	Identifier: mac-address Value: 01:02:03:04:05:ab
	Identifier2	Identifier: device Name: id2

7. Send an IFMAP search request to IFMAP server.

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab
Max-depth	1

8. Disconnect client.

Expected Outcomes:

- Publish update metadata MUST respond in “publishRecieved”.
- Publish delete metadata with empty string filter in step#4 MUST respond in “publishReceived”.
- Search for metadata in step#5 MUST still find all the identifiers, link and metadata that was published in step#3.
- Publish delete metadata with no filter in step#6 MUST respond in “publishReceived”.
- Search for metadata in step#7 MUST NOT find the identifiers, link and metadata that was published in step#3.

Anticipated Failures:

- If server is deleting any of the metadata when an empty filter is specify in delete request.
- If server is not deleting all of the metadata when no filter is specified in delete request.

5.1.2.23 purgePublisher requests appear atomic.

CTNC-IFMAP2.0-SERVER-TC-23.

Purpose: To verify Server MUST ensure that purgePublisher requests delete the metadata and appear atomic.

This test case is for the following requirements:[CTNC-IFMAP2.0-SERVER-REQ-42-M] and [CTNC-IFMAP2.0-SERVER-REQ-95-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send a publish update request as preparation data:

identifier	Identifier: device Name: id1
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy

identifier	Identifier: device Name: id2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy

4. Send subscribe request with the following data. Poll twice and ignore initial result.

Search1	Identifier: device Name: id1
Search2	Identifier: device Name: id2

5. Send a Purge request on client's own publisher_id.
6. Disconnect client.

Expected Outcomes:

- Subscribe and poll request from step#4MUST result in subscribeReceived.
- Purge from step#5 MUST result inPurgeReceived.
- A single poll result MUST be received after step#5, and pollResult MUST return the following results:

  -deleteResult:

identifier	Identifier: device Name: id1
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy

  -deleteResult:

identifier	Identifier: device Name: id2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy

Anticipated Failures:

- If purge does not trigger poll result.
- If multiple poll results are received from a single purge.

5.1.2.24 Supported metadata size

CTNC-IFMAP2.0-SERVER-TC-24.

Purpose: To verify that the MAP server supports metadata containing 100000 bytes of data.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-43-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send IF-MAP publish update request to IF-MAP server using standard metadata containing 100000 bytes:

identifier	Identifier: access-request Name: ar1
Metadata	metadata: capability name: a string with 100000 ASCII characters

4. Send IF-MAP publish update request to IF-MAP server using vendor-specific metadata containing 100000 bytes:

Link	Identifier1	Identifier: access-request Name: id1
	Identifier2	Identifier: device Name: id2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue Element name: name Element value: a string with 100000 ASCII characters

5. Send an IF-MAP search request to IF-MAP server to retrieve the identifiers that were published in step#3 and Step#4
6. Disconnect client.

Expected Outcomes:

- Publish to the identifiers in step$3 and step#4MUST responds in "publishReceived"
- Search response MUST contain the metadata published in step#3 and step#4.

Anticipated Failures:

- If server is rejecting the large metadata during publish.
- If server is not returning the complete metadata during search.

5.1.2.25 Filter in search

CTNC-IFMAP2.0-SERVER-TC-25.

Purpose: To verify that the MAP Server considers a filter consisting of the empty string as a request to match nothing, and if no result-filter is present in a search or subscription, all metadata that matches the search is returned.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-44-M] and [CTNC-IFMAP2.0-SERVER-REQ-67-M]

Preconditions:

- o Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP publish update request to IFMAP server:

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Link	Identifier1	Identifier: mac-address Value: 01:02:03:04:05:ab
	Identifier2	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata		metadata: meta:ip-MAC Cardinality: multiValue Element name: name

	Element value: dummy1

4. Send an IFMAP search request to IFMAP server with empty filter.

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab
Result-filter	“” (empty string)
Max-depth	1

5. Send an IFMAP subscribe request to IFMAP server with no filter, and then poll once for initial pollResult.

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab
Max-depth	1

6. Disconnect client.

Expected Outcomes:

- Publish metadata MUST respond in “publishReceived”.
- Search with empty filter string in step#4 MUST return all identifiers and link published at step#3, but not the metadata.
- Poll with no filter in step#5 MUST return all identifiers, link and its metadata published at step#3.

Anticipated Failures:

- If server returns any metadata when an empty result-filter is specified in search or subscribe request.
- If server does not return all metadata when no result-filter is specified in search or subscribe request.

5.1.2.26 Namespaces declaration for filter

CTNC-IFMAP2.0-SERVER-TC-26.

Purpose: To verify that the namespace prefixes in a filter are declared in the XML document containing the filter, and the declaration of the namespace prefix applies to the scope of the filter.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-45-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Publish following vendor-specific metadata with prefix declaration.

identifier	Identifier: device Name: id1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

4. Search following vendor-specific metadata with prefix declaration.

identifier	Identifier: device Name: id1
Result-filter	myns1: mymeta1 (with prefix declaration in SOAP envelope xmlns: myns1=http://ns1.example.com)

5. Subscribe following vendor-specific metadata with prefix declaration, and then poll twice (ignore the initial pollResult).

identifier	Identifier: device Name: id1
Result-filter	myns2: mymeta1 (with prefix declaration in SOAP envelop xmlns: myns2=http://ns1.example.com)

6. Delete following vendor-specific metadata without prefix declaration

identifier	Identifier: device Name: id1
Result-filter	myns3: mymeta1 (without prefix declaration)

7. Disconnect client

Expected Outcomes:

- Name space declaration is applied to the scope of the filter in step#4 and 5, search result MUST be return matching with step#3 data.
- Without supply namespace declaration in step#6, no matching data MUST be found or deleted. Or error may be given for missing namespace declaration, depending on the server implementation.

Anticipated Failures:

- If search result is not filtered based on the declared namespace.

5.1.2.27 Failure in a publish request containing multiple operations.

CTNC-IFMAP2.0-SERVER-TC-27.

Purpose: To verify that a successful metadata publish results in a publishReceived message; also, that an unsuccessful metadata publish fails without effect, and the response contains an errorResult element with an error Code attribute indicating the cause of the failure.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-48-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send the following IFMAP publish update request to IFMAP server as a preparation data.

identifier	Identifier: identity Type: username Name: id1
Metadata	Metadata: meta:capability

4. Send an IFMAP subscription request to IFMAP server to subscribe for the following metadata, and then poll twice (ignore the initial pollResult).

identifier	Identifier: identity Type: username Name: id1

5. Send the following IF-MAP publish requests to IFMAP server in a single request packet.

1 - update	identifier	Identifier: identity Type: username Name: id1
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1
2 - delete	identifier	Identifier: identity Type: username Name: id1
	filter	Metadata: meta:capability
3 - notify	identifier	Identifier: identity Type: username Name: id1
	metadata	Metadata: meta:event Name: attack Discovered-time: 2009-05-30T13:10:11 Discovered-id: 1273 Magnitude: 45 Confidence: 100 Significance: important Type: worm infection
4 – update (invalid identifier/meta data)	identifier	Identifier: dummy Type: dummy Name: dummy
	metadata	Metadata: meta:dummy

6. Send an IF-MAP search request to IFMAP server:

identifier	Identifier: identity Type: username Name: id1

7. Disconnect client

Expected Outcomes:

- Publish to the identifiers in step#3 MUST responds in "publishReceived"

- Publish to the identifiers in step#5 MUST responds in error due to invalid identifier.
- Poll result from step#4 MUST NOT receive notify from step#5.
- Search result from step#6 MUST NOT find the metadata published in step#5.
- Search result from step#6 MUST find the metadata published in step#3

Anticipated Failures:

- If only partial operations were performed in a multiple operations request.

5.1.2.28 Multiple operations in a single publish request.

CTNC-IFMAP2.0-SERVER-TC-28.

Purpose: To verify that a MAP Server processes valid publish messages which contain any combination of the three publish subtypes (update, delete, notify). And if the publish request contains multiple update, notify and/or delete, the server must ensure they appear atomic.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-49-M] and [CTNC-IFMAP2.0-SERVER-REQ-94-M] and [CTNC-IFMAP2.0-SERVER-REQ-126-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send the following IFMAP publish update request to IFMAP server as a preparation data.

Identifier	Identifier: identity Type: username Name: id1
Metadata	Metadata: meta:capability

4. Send an IFMAP subscription request to IFMAP server to subscribe for the following metadata, and then poll twice (ignore the initial pollResult).

Identifier	Identifier: identity Type: username Name: id1

5. Send the following IF-MAP publish requests to IFMAP server in a single request packet.

1 - update	identifier	Identifier: identity Type: username Name: id1
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1
2 - delete	identifier	Identifier: identity Type: username Name: id1
	filter	Metadata: meta:capability
3 - notify	identifier	Identifier: identity Type: username Name: id1
	metadata	Metadata: meta:event Name: attack Discovered-time: 2009-05-30T13:10:11 Discovered-id: 1273 Magnitude: 45 Confidence: 100 Significance: important Type: worm infection

6. Send an IF-MAP search request to IFMAP server:

Identifier	Identifier: identity Type: username Name: id1

7. Disconnect client

Expected Outcomes:

- Publish to the identifiers in step#3 MUST responds in “publishReceived”
- Publish to the identifiers in step#5 MUST responds in “publishReceived”
- Poll result in step#4 MUST receive all update, delete, and notify published in step#5, and in the same order.

- Search result from step#6 MUST find the metadata published in step#5 update1.
- Search result from step#6 MUST NOT find the metadata published in step#3

Anticipated Failures:

- If requested operations are not all performed, or not in same sequence as requested.

5.1.2.29 singleValue Cardinality

CTNC-IFMAP2.0-SERVER-TC-29.

Purpose: To verify that an update request with ifmap-cardinality="single Value" replaces the previous value for that metadata type if it exists on the MAP Server.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-38-M] and [CTNC-IFMAP2.0-SERVER-REQ-50-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IF-MAP publish update request to IF-MAP server with a singleValue vendor-specific metadata.

identifier	Identifier: ip-address Type: ipv4 Value: 2.2.2.2
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 cardinality: singleValue Element name: name Element value: before

4. Send an IF-MAP publish update request to IF-MAP server with the same identifier and metadata, but different metadata value

identifier	Identifier: ip-address Type: ipv4 Value: 2.2.2.2

Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 cardinality: singleValue Element name: name Element value: after

5. Send an IF-MAP search request to IF-MAP server to search for the singleValue vendor-specific metadata.

identifier	Identifier: ip-address Type: ipv4 Value: 2.2.2.2
Result-filter	myns1: mymeta1

6. Send an IF-MAP publish update request to IF-MAP server with a singleValue standard metadata.

Link	Identifier1	Identifier: access-request Name: id1
	Identifier2	Identifier: device Name: id2
Metadata		Metadata: wlan-information Wlan-security-type=”open”

7. Send an IF-MAP publish update request to IF-MAP server with the same identifier and metadata, but different metadata value

Link	Identifier1	Identifier: access-request Name: id1
	Identifier2	Identifier: device Name: id2
Metadata		Metadata: wlan-information Wlan-security-type=”wep”

8. Send an IF-MAP search request to IF-MAP server to search for the previous singleValue standard metadata.

identifier	Identifier: access-request Name: id1
Result-filter	meta:wlan-information
Max-depth	1

9. Disconnect client

Expected Outcomes:

- Search for metadata in step#5 MUST return matching metadata published in step#4, and only 1 result MUST be returned.
- Search for metadata in step#8 MUST return matching metadata published in step#7, and only 1 result MUST be returned.

Anticipated Failures:

- If publish singleValue metadata does not replace existing same metadata.

5.1.2.30 multiValue Cardinality

CTNC-IFMAP2.0-SERVER-TC-30.

Purpose: To verify that an update request with ifmap-cardinality="multiValue" appends the new value to a list of values for that metadata type on the MAP Server

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-39-M] and [CTNC-IFMAP2.0-SERVER-REQ-51-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IF-MAP publish update request to IF-MAP server with a multiValue standard metadata.

identifier	Identifier: access-request Name: id1
Metadata	Metadata: capability Name: value1

4. Send an IF-MAP publish update request to IF-MAP server with the same identifier and metadata, but different metadata value

identifier	Identifier: access-request Name: id1
Metadata	Metadata: capability Name: value2

5. Send an IF-MAP search request to IF-MAP server to search for the multiValue standard metadata.

identifier	Identifier: access-request Name: id1
Result-filter	meta:capability

6. Send an IF-MAP publish update request to IF-MAP server with a multiValue vendor-specific metadata.

Link	Identifier1	Identifier: access-request Name: id1
	Identifier2	Identifier: device Name: id2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue Element name: name Element value: value1

7. Send an IF-MAP publish update request to IF-MAP server with the same link and metadata, but different metadata value.

Link	Identifier1	Identifier: access-request Name: id1
	Identifier2	Identifier: device Name: id2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta2 cardinality: multiValue Element name: name Element value: value2

8. Send an IF-MAP search request to IF-MAP server to search for the previous multiValue vendor-specific metadata.

identifier	Identifier: access-request Name: id1
Result-filter	myns1: mymeta2
Max-depth	1

9. Disconnect client.

Expected Outcomes:

- Search for metadata in step#5 MUST return matching metadata published in both step#3 and step#4.
- Search for metadata in step#8 MUST return matching metadata published in both step#6 and step#7.

Anticipated Failures:

- If publish multiValue metadata replaces existing same metadata

5.1.2.31 notifyResult in pollResult

CTNC-IFMAP2.0-SERVER-TC-31.

Purpose: To verify that metadata published with ‘notify’ is queued for delivery to all clients with subscriptions whose searches match the published data, delivered to those clients inside notifyResult elements within pollResult elements, and is not delivered to clients in response to a search request.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-52-M] and [CTNC-IFMAP2.0-SERVER-REQ-53-M] and [CTNC-IFMAP2.0-SERVER-REQ-54-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Subscribe for ‘event’ metadata on an ip-address identifier, and then poll twice (ignore the initial pollResult).

Identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11

4. Send Publish notify message with the following data:

Identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
Metadata	Metadata: meta:event

	Name: attack1 Discovered-time: 2010-01-01T01:01:01 Discovered-id: 1234 Magnitude: 1 Confidence: 100 Significance: critical Type: worm infection

5. Check poll result from step#3. Then continue to poll on the existing subscription list.
6. Send Publish notify message with the following data:

Identifier	Identifier: ip-address Type : ipv6 value: 1:2:3:4:5:6:7:8
Metadata	Metadata: meta:event Name: attack2 Discovered-time: 2010-12-31T12:59:59 Discovered-id: 4321 Magnitude: 99 Confidence: 1 Significance: informational Type: worm policy violation

7. Check poll result from step#5. Send search request with the following data:

Identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
result-filter	meta:event

8. Disconnect client.

Expected Outcomes:

- Poll result in step#3 MUST return the metadata published in step#4 in a notifyResult element
- Step#7 MUST NOT receive any poll result
- Step#7 MUST NOT find the metadata published in step#3

Anticipated Failures:

- Notify is not sent to the matching subscribers.

•Notify is received as a searchResult instead of a pollResult.

5.1.2.32 publishDelete

CTNC-IFMAP2.0-SERVER-TC-32.

Purpose: To verify that the MAP server deletes metadata specified by a valid delete message.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-55-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Ste0070s:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send publish update request with the following data:

Identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Link	Identifier1	Identifier: ip-address Type : ipv4 value: 192.168.1.11
	Identifier2	Identifier: mac-address value: 00:11:22:33:44:55
Metadata		Metadata: meta:ip-MAC

4. Send search request with the following data:

Identifier	Identifier: ip-address Type : ipv4

	value: 192.168.1.11
Max-depth	1

5. Send publish delete request with the following data:

Identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
filter	myns1:mymeta1 (include namespace definition in SOAP envelope: Xmlns: myns1=http://ns1.example.com)

Link	Identifier1	Identifier: ip-address Type : ipv4 value: 192.168.1.11
	Identifier2	Identifier: mac-address value: 00:11:22:33:44:55
filter		meta:ip-MAC

6. Send search request with the following data:

Identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11`
Max-depth	1

7. Disconnect client.

Expected Outcomes:

- In Step #3, publish update metadata MUST respond with “publishReceived”.
- In Step #4, search MUST respond with all published metadata from step#3.
- In Step #5, publish delete metadata MUST respond with “publishReceived”
- In Step #6, search MUST respond with not find the metadata published from step#3.

Anticipated Failures:

- Publish delete returns errorResult.
- Publish delete does not delete the matching metadata.

5.1.2.33 Multiple operations in a single publish request should be atomic

CTNC-IFMAP2.0-SERVER-TC-33.

Purpose: To verify that when a publish request contains multiple update and/or delete elements which operate on the same identifiers or links, the result of the publish request is consistent with the update and delete elements having been applied to the IFMAP database in the order in which they are specified by the client.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-58-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IF-MAP publish update request to IF-MAP server to publish the following metadata as a preparation step.

Link	Identifier1	Identifier: ip-address Type : ipv4 value: 192.168.1.11
	Identifier2	Identifier: mac-address value: 00:11:22:33:44:55
Metadata		Metadata: meta:ip-mac Start-time: 2009-10-31T10:30:30 end-time: 2009-10-31T20:30:30 dhcp-server: 1.2.3.4

4. Send an IF-MAP search request to IF-MAP server the data published.

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
Max-depth	1

5. Send an IFMAP publish update and delete request to IFMAP server in a single request.

1 - update	identifier		Identifier: ip-address Type : ipv4 value: 192.168.1.11
	metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1
2 - update	identifier		Identifier: ip-address Type : ipv4 value: 192.168.1.11
	metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy2
3 - delete	link	identifier1	Identifier: ip-address Type : ipv4 value: 192.168.1.11
		identifier2	Identifier: mac-address value: 00:11:22:33:44:55
	filter		meta:ip-mac
4 - update	link	identifier1	Identifier: ip-address Type : ipv4 value: 192.168.1.11
		identifier2	Identifier: mac-address value: 00:11:22:33:44:55
	metadata		Metadata: meta:ip-mac Start-time: 2010-10-31T10:30:30 end-time: 2010-10-31T20:30:30 dhcp-server: 4.3.2.1

6. Send an IF-MAP search request to IF-MAP server the data published:

identifier	Identifier: ip-address Type : ipv4

	value: 192.168.1.11
Max-depth	1

7. Disconnect client.

Expected Outcomes:

- Publish in step#3 MUST respond in “publishReceived”.
- Search in step#4 MUST return the metadata published in step#3.
- Publish in step#5 MUST respond in “publishReceived”.
- Search in step#6 MUST return only the following metadata:

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy2

Link	Identifier1	Identifier: ip-address Type : ipv4 value: 192.168.1.11
	Identifier2	Identifier: mac-address value: 00:11:22:33:44:55
Metadata		Metadata: meta:ip-mac Start-time: 2010-10-31T10:30:30 end-time: 2010-10-31T20:30:30 dhcp-server: 4.3.2.1

Anticipated Failures:

- If multiple operations in a single request is not processed in atomic order.

5.1.2.34 Search algorithm.

CTNC-IFMAP2.0-SERVER-TC-34.

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

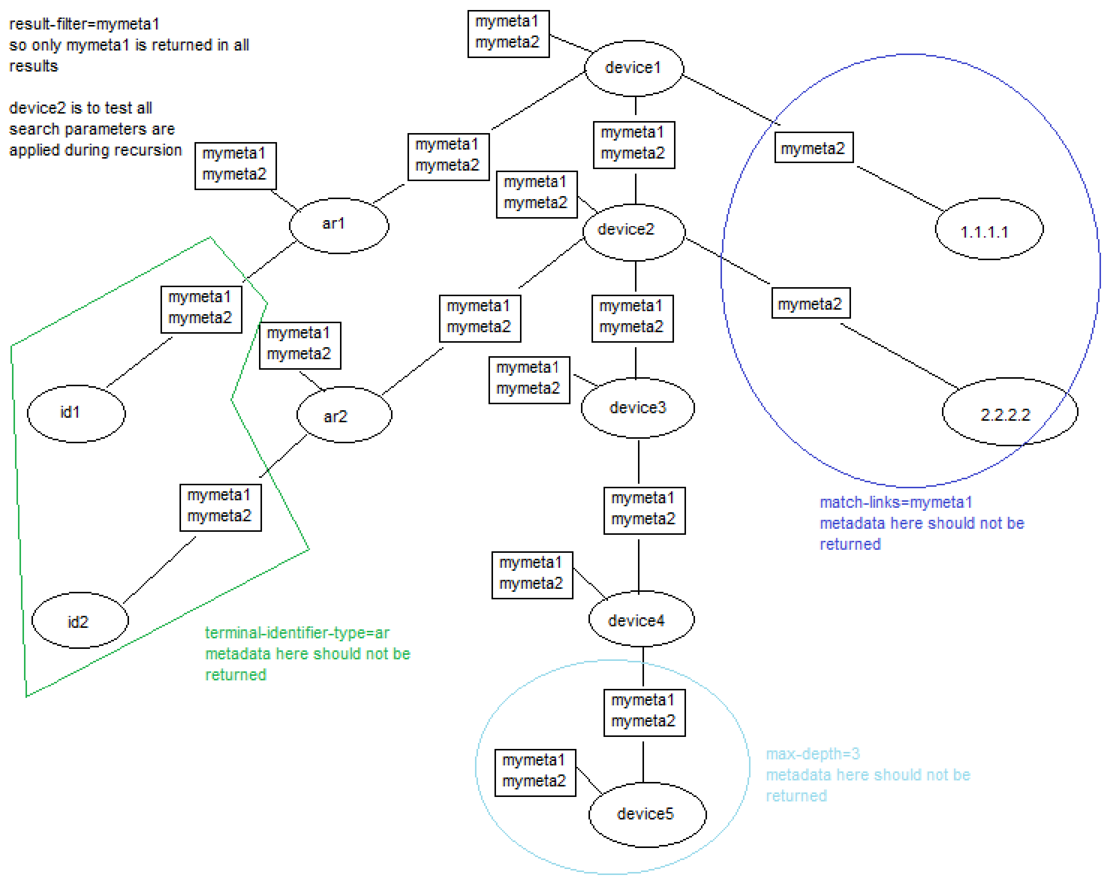
This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-59-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server. Capture session-id.
3. Send publish request with the following data (depth0 and 1 both have 3 links, first link to test terminal-identifier-type, second link to test match-links, and third link to test max-depth. The identifier at the third link will also have additional 3 links similar to depth0 to test for recursion algorithm. Two metadata are attached to each identifier and link, one of the metadata will be tested for result-filter)


result-filter=mymeta1
so only mymeta1 is returned in all results
device2 is to test all search parameters are applied during recursion
device1
device2
device3
device4
device5
ar1
ar2
id1
id2
1.1.1.1
2.2.2.2
mymeta1
mymeta2
match-links=mymeta1
metadata here should not be returned
terminal-identifier-type=ar
metadata here should not be returned
max-depth=3
metadata here should not be returned

identifier	Identifier: device Name: device1

Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 0
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 0

Link	Identifier1	Identifier: device Name: device1
	Identifier2	Identifier: access-request Name: ar1
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 0-1
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 0-1

identifier	Identifier: access-request Name: ar1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 1

Link	Identifier1	Identifier: access-request Name: ar1
	Identifier2	Identifier: identity Name: id1 Type: username
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1-2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 1-2

Link	Identifier1	Identifier: device Name: device1
	Identifier2	Identifier: ip-address Type: ipv4 value: 1.1.1.1
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 0-1

Link	Identifier1	Identifier: device Name: device1
	Identifier2	Identifier: device Name: device2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 0-1
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 0-1

identifier	Identifier: device Name: device2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 1

Link	Identifier1	Identifier: device Name: device2
	Identifier2	Identifier: access-request Name: ar2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1-2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth

	Element value: 1-2

identifier	Identifier: access-request Name: ar2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 2
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 2

Link	Identifier1	Identifier: access-request Name: ar2
	Identifier2	Identifier: identity Name: id2 Type: username
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 2-3
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 2-3

Link	Identifier1	Identifier: device Name: device2
	Identifier2	Identifier: ip-address Type: ipv4 value: 2.2.2.2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 2

Link	Identifier1	Identifier: device Name: device2
	Identifier2	Identifier: device Name: device3
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1

	cardinality: singleValue Element name: depth Element value: 1-2
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 1-2

identifier	Identifier: device Name: device3
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 2
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 2

Link	Identifier1	Identifier: device Name: device3
	Identifier2	Identifier: device Name: device4
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 2-3
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 2-3

identifier	Identifier: device Name: device4
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 3
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 3

Link	Identifier1	Identifier: device Name: device4
	Identifier2	Identifier: device Name: device5
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 3-4
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 3-4

identifier	Identifier: device Name: device5
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 4
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 cardinality: multiValue Element name: depth Element value: 4

4. Send search request with the following data:

identifier	Identifier: device Name: device1
Max-depth	3
Result-filter	metadata: myns1:mymeta1 (include namespace definition in SOAP envelop: Xmlns: myns1=http://ns1.example.com)
Match-link	metadata: myns1:mymeta2 (include namespace definition in SOAP envelop: Xmlns: myns1=http://ns1.example.com)
Terminal-identifier-type	access-request
Max-size	100000

5. Send another search request with the following data:

identifier	Identifier: device Name: device1
Max-depth	3
Result-filter	metadata: myns1:mymeta1 (include namespace definition in SOAP envelop:

	Xmlns: myns1=http://ns1.example.com)
Match-link	metadata: myns1:mymeta2 (include namespace definition in SOAP envelop: Xmlns: myns1=http://ns1.example.com)
Terminal-identifier-type	access-request
Max-size	50

6. Disconnect client.

Expected Outcomes:

- Publish metadata from step#3 MUST result in "publishRecieved".
- Searched metadata from step#4 MUST result the following identifier/metadata:

identifier	Identifier: device Name: device1
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 0

Link	Identifier1	Identifier: device Name: device1
	Identifier2	Identifier: access-request Name: ar1
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 0-1

identifier	Identifier: access-request Name: ar1
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1

Link	Identifier1	Identifier: device Name: device1
	Identifier2	Identifier: device Name: device2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 0-1

identifier	Identifier: device

	Name: device2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1

Link	Identifier1	Identifier: device Name: device2
	Identifier2	Identifier: access-request Name: ar2
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1-2

identifier	Identifier: access-request Name: ar2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 2

Link	Identifier1	Identifier: device Name: device2
	Identifier2	Identifier: device Name: device3
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 1-2

identifier	Identifier: device Name: device3
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 2

Link	Identifier1	Identifier: device Name: device3
	Identifier2	Identifier: device Name: device4
Metadata		Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 2-3

identifier	Identifier: device Name: device4
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 cardinality: singleValue Element name: depth Element value: 3

- Searched metadata from step#5 MUST return "searchResultTooBig", and no identifier or metadata MUST be returned.

Anticipated Failures:

- If searchResult does not match expected outcome.

5.1.2.35 Default max-depth value in search

CTNC-IFMAP2.0-SERVER-TC-35.

Purpose: To verify that if a MAP Client does not specify max-depth, the MAP Server MUST process the search with a max-depth of zero.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-60-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IF-MAP publish update request to IF-MAP server

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue

	Element name: name Element value: dummy1

Link	Identifier1	Identifier: ip-address Type : ipv4 value: 192.168.1.11
	Identifier2	Identifier: mac-address value: 00:11:22:33:44:55
Metadata		Metadata: meta:ip-mac Start-time: 2010-10-31T10:30:30 end-time: 2010-10-31T20:30:30 dhcp-server: 1.2.3.4

4. Send an IF-MAP search request to IF-MAP server the data published will be retrieved

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11

5. Disconnect client.

Expected Outcomes:

- Publish in step#3 MUST responds in “publishReceived”
- Search in step#4 MUST only retune the following identifier/metadata.

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Anticipated Failures:

- If search returns any metadata on link level while max-depth is not specified.

5.1.2.36 Default and supported max-size in search

CTNC-IFMAP2.0-SERVER-TC-36.

Purpose: To verify that a MAP Servers MUST support size constraints up to and including 100KB. If a MAP Client does not specify max-size, the MAP Server MUST process the search with a max-size of 100KB

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-61-M] and [CTNC-IFMAP2.0-SERVER-REQ-62-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP publish update request to IFMAP server with link metadata to publish data equal 100kilobyte (<= 100kilobyte)

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11
metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: a string of 97kilobytes

4. Send an IF-MAP search request to IF-MAP server to retrieve the data published

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11

5. Send an IFMAP publish update request to IFMAP server with link metadata to publish data equal 101kilobyte (> 100kilobyte).

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11

metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: a string of 101kilobyte

6. Send an IF-MAP search request to IF-MAP server the data published will not be retrieved

identifier	Identifier: ip-address Type : ipv4 value: 192.168.1.11

7. Disconnect client.

Expected Outcomes:

- Publish in step#3 MUST respond in “publishReceived“.
- Search in step#4 MUST return metadata published in step#3.
- Publish in step#5 MUST respond in “publishReceived“
- Search in step#6 MUST respond in “SearchResultTooBig”.

Anticipated Failures:

- If SearchResultTooBig is returned for search less than 100k.
- If no error were given when resultResult is larger than 100k and max-size is not defined.

5.1.2.37 Max-size support in search

CTNC-IFMAP2.0-SERVER-TC-37.

Purpose: To verify that MAP Client specifies a max-size that exceeds what the MAP Server can support, the MAP Server enforce its own maximum size constraints. MAP server will return an error if result exceeds max-size.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-63-M] and [CTNC-IFMAP2.0-SERVER-REQ-64-M]

Preconditions:

- Devices configured to “Common Setup”.
- MAP server supported max search result size from pre-test questionnaire.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send the following IF-MAP publish update requests to IFMAP server, send multiple publishes enough to exceed the server supported search size that test executor entered when initiate the test.

identifier	Identifier: ip-address Type: ipv4 Name: 1.2.3.4
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: a string with 100000 characters

4. Send an IF-MAP search request to IF-MAP server to retrieve the identifiers that were published in step#3

Identifier	Identifier: ip-address Type: ipv4 Name: 1.2.3.4
Max size	4294967295

5. Disconnect client.

Expected Outcomes:

- Publish request to the IFMAP server in step3 MUST responds in “publishRecieved”
- Search response in step#4 MUST either result in ALL metadata returned (no partial results), or error due to “SearchResultTooBig”.

Anticipated Failures:

- If MAP server only returns partial searchResult, or no result.

5.1.2.38 Successful searches

CTNC-IFMAP2.0-SERVER-TC-38.

Purpose: To verify that a successful search result in a response message containing a searchResult element comprised of identifiers and links along with their associated metadata.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-65-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP publish update request to IFMAP server to publish the following.

identifier	Identifier: access-request Name: ar1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Link	Identifier1	Identifier: access-request Name: ar1
	Identifier2	Identifier: mac-address Value: 01:02:03:04:05:ab
Metadata		metadata: meta:access-request-mac cardinality: singleValue

4. Send an IFMAP search request to IFMAP server to search for the published identifiers metadata elements.

identifier	Identifier: access-request Name: ar1
Max-depth	1

5. Disconnect client.

Expected Outcomes:

- Publish to the identifiers in step#3 MUST responds in “publishReceived”
- Search response MUST contain the following 3 results.

identifier	Identifier: access-request Name: ar1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Link	Identifier1	Identifier: access-request Name: ar1
	Identifier2	Identifier: mac-address Value: 01:02:03:04:05:ab
Metadata		metadata: meta:access-request-mac cardinality: singleValue

identifier	Identifier: mac-address Value: 01:02:03:04:05:ab

Anticipated Failures:

- If search returns partial result, or missing identifiers, link or metadata.

5.1.2.39 searchResult with no metadata due to result-filter

CTNC-IFMAP2.0-SERVER-TC-39.

Purpose: To verify result-filter filters out all metadata associated with an identifier or link, the identifier or link MUST be included in the searchResult even though no metadata is associated with the identifier or link in the search result.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-66-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP publish update request to IFMAP server to publish the following.

identifier	Identifier: access-request Name: ar1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Link	Identifier1	Identifier: access-request Name: ar1
	Identifier2	Identifier: mac-address Value: 01:02:03:04:05:ab
Metadata		metadata: meta:access-request-mac cardinality: singleValue

4. Send an IFMAP search request to IFMAP server to search for the published identifiers metadata elements.

identifier	Identifier: access-request Name: ar1
Max-depth	1
Result-filter	meta:ip-mac

5. Disconnect client.

Expected Outcomes:

- Publish to the identifiers in step#3 MUST responds in "publishReceived"
- Search response MUST contain the following 3 results, all results does not contain metadata.

identifier	Identifier: access-request Name: ar1

Link	Identifier1	Identifier: access-request

		Name: ar1
	Identifier2	Identifier: mac-address Name: 01:02:03:04:05:ab

identifier	Identifier: mac-address Name: 01:02:03:04:05:ab

Anticipated Failures:

- If result-filter filtered out metadata, and empty identifier is not returned.

5.1.2.40 Search will always return identifier

CTNC-IFMAP2.0-SERVER-TC-40.

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

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-68-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP search request to IFMAP server to search for the metadata elements with the identifier.

identifier	Identifier: mac-address Type: mac Value: 01:02:03:04:05:06

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

identifier	Identifier: access-request Name: ar1

identifier	Identifier: identity Type: username Name: id1

identifier	Identifier: device Name: device1

4. Disconnect client.

Expected Outcomes:

- Search request to IFMAP server with different identifier MUST return the identifier, but no metadata. There MUST be no error message.

Anticipated Failures:

- If empty identifier is not returned.

5.1.2.41 One subscription list per connected MAP Client.

CTNC-IFMAP2.0-SERVER-TC-41.

Purpose: To verify MAP Server MUST maintain only one subscription list per connected MAP Client. When a MAP Client initially connects to a MAP Server, the MAP Server MUST delete any previous subscriptions corresponding to the MAP Client.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-69-M] and [CTNC-IFMAP2.0-SERVER-REQ-71-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Client sends an IFMAP Subscribe requests to IFMAP server, and then poll once (ignore initial pollResult)

name	Client_search1
Identifier	Identifier: access-request Name: ar1

4. Client send subscribe update on existing search, and then poll twice (ignore first pollResult)

name	Client_search1
Identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

5. Client send an IFMAP Publish update requests to IFMAP server:

identifier	Identifier: access-request Name: ar1
Metadata	meta: capability name: dummy

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata	meta: unexpected-behavior discovered-time: 2009-05-30T13:10:11 discovered-id: 1374 magnitude: 10 significance: critical

6. Client ends existing session.
7. Client requests new session from server, and send poll request
8. Disconnect client.

Expected Outcomes:

- Subscription request in step#3 and step#4 MUST respond in “subscribeReceived”.
- Poll result from step#4 MUST only contain the following metadata:

name	Client_search1
Identifier	Identifier: ip-address Type: ipv4

	Value: 1.2.3.4
metadata	meta: unexpected-behavior discovered-time: 2009-05-30T13:10:11 discovered-id: 1374 magnitude: 10 significance: critical

- Client polling at step#7MUSTnot receive any results, since the session with a subscription was ended..

Anticipated Failures:

- If the same search name returns different identifiers.

5.1.2.42 subscribeResult

CTNC-IFMAP2.0-SERVER-TC-42.

Purpose: To verify the MAP Server MUST respond to a valid subscribe message with a subscribeReceived message. If the subscribe Request is not valid, the MAP Server MUST respond with an appropriate errorResult

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-70-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP Subscribe request to IFMAP server to subscribe the metadata elements with valid identifiers.

identifier	Identifier: access-request name: ar1
Metadata	metadata: meta:capability

4. Send an IFMAP Subscribe request to IFMAP server to subscribe the metadata elements with invalid identifiers.

identifier	Identifier: mac-address Value: dummy

5. Disconnect the client.

Expected Outcomes:

- Subscribe in step#3 MUST respond in “subscribeReceived”.
- Subscribe using invalid identifier in step#4MUST respond in errorResult.

Anticipated Failures:

- If subscribe does not return subscribeReceived or errorResult.

5.1.2.43 Invalid poll response

CTNC-IFMAP2.0-SERVER-TC-43.

Purpose: To verify that if a poll is not valid, the MAP Server MUST respond with an appropriate errorResult

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-73-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Add a subscription for ip-address identifier.

Identifier	Identifier: ip-address Type: ipv4 Name: 10.0.0.3

4. Send an IF-MAP poll request to IF-MAP server with missing session ID.

Expected Outcomes:

- Add Subscription list in step 3 MUST respond in “subscribeReceived”.
- Poll in step 4 MUST respond with proper error “InvalidSessionID”.

Anticipated Failures:

- If no error is given for invalid polls.

5.1.2.44 Invalid poll removes subscription

CTNC-IFMAP2.0-SERVER-TC-44.

Purpose: To verify that if a server responds to a poll with an errorResult(most likely SearchResultsTooBig or PollResultsTooBig), all of the client’s subscriptions are automatically invalidated and MUST be removed by the server

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-74-M] and [CTNC-IFMAP2.0-SERVER-REQ-75-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send a subscription request to IF-MAP server with limited max-size, then poll twice (ignore the initial pollResult).

Identifier	Identifier: ip-address Type: ipv4 Name: 10.0.0.5
Max-size	100

4. Send an IF-MAP publish update request to IF-MAP server to publish Ipv4 Ip address with size greater than 100byte.

identifier	Identifier: ip-address Type: ipv4 Name: 10.0.0.5

	Xmlns: myns1=http://ns1.example.com
	metadata: myns1:mymeta1
	Cardinality: singleValue
	Element name: name
Metadata	Element value: a string with 101 byte

5. Send an IF-MAP poll request again to IF-MAP server.
6. Disconnect client.

Expected Outcomes:

- Add Subscription list in step 3 MUST respond in "subscribeReceived".
- Publish update in step 5 MUST respond in "publishReceived".
- Poll result from step#3 MUST respond with "PollResultTooBig".
- Poll in step#5 MUSTblock due to no subscriptions being available since theywere removed by the previous error.

Anticipated Failures:

- If poll still receives result or failure response after it has received failure and hasn't re-subscribed.

5.1.2.45 Initial poll returns searchResult

CTNC-IFMAP2.0-SERVER-TC-45.

Purpose: To verify that the first time a pollResult contains search results for a new subscription, the search results MUST consist of the complete set of identifiers, links, and metadata for the subscription. Then MAP Server checks to see if any search results are available for subscriptions the client has made. If no results are available, the MAP Server MUST NOT immediately send a response. At some future time when metadata changes occur that match client subscriptions, the MAP Server MUST send a pollResult response containing search results. The server won't return searchResult, updateResult or deleteResult elements for a subscription that has no changes associated with it

This test case is for the following [CTNC-IFMAP2.0-SERVER-REQ-76-M] and [CTNC-IFMAP2.0-SERVER-REQ-77-M] and [CTNC-IFMAP2.0-SERVER-REQ-78-M] and [CTNC-IFMAP2.0-SERVER-REQ-84-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.

3. Send IF-MAP publish update request(s) to IF-MAP server with the following entire set of identifiers:

Link1	Identifier1	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
	Identifier2	Identifier: device Name: dev1
Metadata		metadata: meta: discovered-by

identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
Metadata	Xmlns: myns1=http://ns1.example.com metadata:myns1: mymeta2 Cardinality: multiValue Element name: name Element value: dummy

4. Send IF-MAP subscribe contain above link and identifier request to IF-MAP server

identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
Max-depth	1

5. Send a first time poll request to IF-MAP server
6. Send second time poll request to IF-MAP server
7. Send an publish update to modify metadata for identifier id1

identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
Metadata	Xmlns: myns1=http://ns1.example.com metadata:myns1: mymeta1 Cardinality: singleValue Element name: name Element value: dummy2

8. Disconnect client.

Expected Outcomes:

- Publish to the identifiers in step3 MUST response in “publishReceived”
- Send IF-MAP subscribe request in step#4 MUST response in “subscribeReceived”
- Poll response from step#5 MUST contain the complete search result contain the metadata published in step#3.
- Poll response from step#6 MUST contain poll result which only the metadata published in step#7.

Anticipated Failures:

- If initial poll does not return matching search results.

5.1.2.46 Second poll returns only the changes, and atomicity.

CTNC-IFMAP2.0-SERVER-TC-46.

Purpose: To verify that MAP server subsequent pollResult responses contain all of the updates which have occurred since last poll, pollResults MUST contain updates in updateResult elements as metadata is added and deletes in deleteResult elements as metadata is removed. And updateResult and deleteResult elements should not cancel each other out.

This test case is for the following requirements:[CTNC-IFMAP2.0-SERVER-REQ-79-M] [CTNC-IFMAP2.0-SERVER-REQ-80-M] [CTNC-IFMAP2.0-SERVER-REQ-81-M] **Error! Reference source not found.** and [CTNC-IFMAP2.0-SERVER-REQ-97-M]

Preconditions:

- o Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send following IF-MAP subscribe request to IF-MAP server and send the initial poll request:

identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4

4. Send following IF-MAP update containing the following data

update	identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

5. Wait 1 second, then send IF-MAP delete containing the following data

delete	identifier	Identifier: ip-address Type: ipv4

		Value: 11.2.3.4
	filter	myns1:mymeta1

6. Send the second poll request to the server
7. Wait 1 second, then send following IF-MAP update containing the following data

update	identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

8. Wait 1 second, then send IF-MAP delete containing the following data

delete	identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
	filter	myns1:mymeta1

9. Send following IF-MAP update containing the following data

update	identifier	Identifier: ip-address Type: ipv4 Value: 11.2.3.4
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

10. Send a third poll request to the server.
11. Disconnect client

Expected Outcomes:

- Subscribe request in step3 MUST respond in “subscribeReceived”
- First poll in step#3 MUST return the identifier without any metadata.

- Publish requests in steps#4 and 5 MUST respond in “publishReceived”.
- Second poll in step#6 MUST return both the update and delete published in step#4 and 5 in the same order, based on timestamp, they were published.
- Third poll request in Step#10 MUST return pollResult for the update AND pollResult for the delete AND pollResult for the update.

Anticipated Failures:

- If second poll returns no search results, or the second poll does not return both update and delete results, the test fails.
- If the third poll does not return the two pollResults, one each for the delete and update.

5.1.2.47 Elements in pollResult

CTNC-IFMAP2.0-SERVER-TC-47.

Purpose: To verify that the server MUST return a separate searchResult, updateResult, deleteResult or errorResult element for each subscription that has changes. Each searchResult, updateResult, deleteResult and errorResult element in a pollResult response MUST include a name attribute which identifies the subscription corresponding to the result. When a MAP client publishes using notify, the MAP server MUST add the published metadata to the poll results for each subscriber whose subscription matches the published metadata

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-82-M]and [[CTNC-IFMAP2.0-SERVER-REQ-83-M]] and [[CTNC-IFMAP2.0-SERVER-REQ-85-M]]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send the following IFMAP publish update request to IFMAP server as a preparation data.

identifier	Identifier: identity Type: username Name: id1
Metadata	Metadata: meta:capability

identifier	Identifier: access-request Name: ar1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

Link1	Identifier1	Identifier: ip-address Type: ipv4 Value: 1.1.1.1
	Identifier2	Identifier: mac-address Name: 01:01:01:01:01:01
Metadata		metadata: meta:ip-mac

4. Send an IFMAP subscription request to IFMAP server to subscribe for the following metadata, and then poll once to get initial result. Then poll again.

Subscription name: sub1	Identifier	Identifier: identity Type: username Name: id1
	Max-size	2000
Subscription name: sub2	Identifier	Identifier: access-request Name: ar1
	Max-size	2000
Subscription name: sub3	Identifier1	Identifier: ip-address Type: ipv4
	Max-depth	1
	Max-size	2000

5. Send the following IF-MAP publish requests to IFMAP server in a single request packet.

1 - update	identifier	Identifier: identity Type: username Name: id1
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1
2 - delete	identifier	Identifier: identity Type: username Name: id1
	filter	Metadata: meta:capability
3 – notify	identifier	Identifier: identity Type: username Name: id1

	metadata	Metadata: meta:event Name: attack Discovered-time: 2009-05-30T13:10:11 Discovered-id: 1273 Magnitude: 45 Confidence: 100 Significance: important Type: worm infection
4 - update	Identifier1	Identifier: ip-address Type: ipv4 Value: 1.1.1.1
	Identifier2	Identifier: mac-address Name: 01:01:01:01:01:01
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1
5 - delete	identifier	Identifier: identity Type: username Name: id1
	filter	myns1:mymeta1 (include namespace definition in SOAP envelop: Xmlns: myns1=http://ns1.example.com)
6 - notify	identifier	Identifier: access-request Name: ar1
	metadata	Metadata: meta:event Name: attack Discovered-time: 2009-05-30T13:10:11 Discovered-id: 1273 Magnitude: 45 Confidence: 100 Significance: important Type: worm infection

6. Continue to poll with existing subscription list. And send another publish with following metadata:

1 - update	identifier	Identifier: identity

		Type: username Name: id1
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: a string with 2500 characters
2 - update	identifier	Identifier: access-request Name: ar1
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: a string with 2500 characters
3 - update	Identifier1	Identifier: ip-address Type: ipv4 Value: 1.1.1.1
	Identifier2	Identifier: mac-address Name: 01:01:01:01:01:01
	metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: a string with 2500 characters

7. Disconnect client

Expected Outcomes:

- Publish request at step#3 MUST result in “publishRecieve”
- Initial poll result MUST return pollResult containg multiple searchResults, each searchResult MUST contain the subscription name used in step#4. The searchResults MUST match the metadata published instep#3.
- After step#5, second poll result MUST contain the result from the 3 subscription names, total of 1 updateResults, 1 deleteResults and 2 notifyResults

- delete(sub1):identity(username:id1)/capability
- notity(sub1):identity(username:id1)/event
- notify(sub2):access-request(ar1)/event

- update(sub3):ipv4(1.1.1.1)-mac(01:01:01:01:01:01)/mymeta1

•After step#6, poll result MUST return 3 errorResults with the 3 subscription names.

Anticipated Failures:

- If each poll result element does not contains a name that matches the matching subscription.

5.1.2.48 Notifies are queued until poll or session end.

CTNC-IFMAP2.0-SERVER-TC-48.

Purpose: To verify a client that does not have an active poll request, the MAP server MUST retain metadata published using notify until the client issues a poll request or the client's session ends.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-86-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP subscription request to IFMAP server to subscribe for the following metadata, and then poll once to get initial result.

identifier	Identifier: identity Type: username Name: id1

4. Send the following IF-MAP publish requests to IFMAP server in a single request packet.

1 - notify	identifier	Identifier: identity Type: username Name: id1
	metadata	Metadata: meta:event Name: attack Discovered-time: 2009-05-30T13:10:11 Discovered-id: 1273 Magnitude: 45

		Confidence: 100 Significance: important Type: worm infection
2 - notify	identifier	Identifier: identity Type: username Name: id1
	metadata	Metadata: meta:event Name: attack Discovered-time: 2010-01-01T01:01:01 Discovered-id: 4321 Magnitude: 10 Confidence: 1 Significance: informative Type: worm infection

5. Send a poll request with the existing subscription list.
6. Disconnect client.

Expected Outcomes:

- Subscription request at step#3 MUST result in "subscribeRecieve".
- Poll request from step#5 MUST result in 2 notifyResults with the metadata published in step#4.

Anticipated Failures:

- If notify is lost while client is still connected.

5.1.2.49 notifyResult is only for notifies.

CTNC-IFMAP2.0-SERVER-TC-49.

Purpose: To verify notifyResult is only used to return metadata that was published using notify, even if other metadata matches the result-filter in the subscription.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-87-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP subscription request to IFMAP server to subscribe for the following metadata, and then poll twice (ignore initial poll result).

identifier	Identifier: identity Type: username Name: id1
Result-filter	meta:capability

4. Send the following IF-MAP publish requests to IFMAP server in a single request packet.

1 - notify	identifier	Identifier: identity Type: username Name: id1
	metadata	Metadata: meta:capability Name: from_notify
2 - update	identifier	Identifier: identity Type: username Name: id1
	metadata	Metadata: meta:capability Name: from_publish

5. Disconnect client.

Expected Outcomes:

- Subscription request at step#3 MUST result in "subscribeRecieve".
- Publish request at step#4 MUST result in "subscribeRecieve".
- Poll request from step#3 MUST result in 1 notifyResult and 1 updateResult with the metadata published in step#4.

Anticipated Failures:

- If notify is delivered by searchResult or any results other than notifyResult.

5.1.2.50 Minimum buffer for poll results

CTNC-IFMAP2.0-SERVER-TC-50.

Purpose: To verify a MAP Server MUST buffer at least 5,000,000 bytes of poll results for each MAP Client.If the size of a MAP Client's poll results exceeds the MAP Server's limit, the MAP Server MUST indicate this to the MAP client by responding to a poll request with an errorResult containing an errorCode of PollResultsTooBig.

This test case is for the following requirements: [[CTNC-IFMAP2.0-SERVER-REQ-88-M]] and [[CTNC-IFMAP2.0-SERVER-REQ-89-M]]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an IFMAP subscription request to IFMAP server to subscribe for the following metadata, and then poll once to get initial result

identifier	Identifier: identity Type: username Name: id1
Max-size	6,000,000

4. Send the following IF-MAP publish update requests to IFMAP server, send 40 copies (to reach total size close to but not exceed 5,000,000 bytes)

identifier	Identifier: identity Type: username Name: id1
metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: a string with 100000 characters

5. Send a poll request with existing subscription list.
6. Disconnect client.
7. Client requests a new session from server with "max-poll-result-size" of "2147483647". Check the responded allocated size.

8. A) If the responded allocated "max-poll-result-size" is under 10,000,000, then send the following IF-MAP publish requests to IFMAP server, send multiple requests which combine with metadata in step#4, are enough to exceed the "max-poll-result-size" received from step#7.

identifier	Identifier: identity Type: username Name: id1
metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: a string with 100000 characters

B) If the responded allocated "max-poll-result-size" from step#7 is over 10,000,000, then send the following IF-MAP publish requests to IFMAP server, send multiple requests which combine with metadata in step#4, are enough to reach the 10,000,000 bytes.

identifier	Identifier: identity Type: username Name: id1
metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: a string with 100000 characters

9. Send an IFMAP subscription request to IFMAP server to subscribe for the following metadata, and then poll once to get initial result.

identifier	Identifier: identity Type: username Name: id1

10. Disconnect client.

Expected Outcomes:

- Subscription request at step#4 MUST result in "subscribeRecieve".
- Poll request from step#5 MUST result in all the metadata that was published in step#4.
- If 8B is used, then step#9 MUSTresult in all the metadata that was published in step#8B.

- If 8A is used, then poll request from step#9 MUST result in errorResult due to pollResultTooBig

Anticipated Failures:

- If MAP server throw error for poll results smaller than 5,000 kilobytes.

5.1.2.51 Purge response

CTNC-IFMAP2.0-SERVER-TC-51.

Purpose: To verify a MAP Server MUST respond to a purge Publisher request with either a purge Publisher Received message or an error Result message.A MAP Server MAY forbid a MAP Client to use the purgePublisher request to remove data published by a different MAP Client, in which case the MAP Server MUST respond with an AccessDenied error.

This test case is for the following requirements: and [CTNC-IFMAP2.0-SERVER-REQ-91-M] and [CTNC-IFMAP2.0-SERVER-REQ-91-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send an update publish request

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1: mymeta1 cardinality: multiValue

4. Send a purgeRequest from Client with an unknown publisher_id.
5. Search for metadata on ip-address 1.2.3.4

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

6. Send a purgeRequest from Client with missing publisher_id.

7. Search for metadata on ip-address 1.2.3.4

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

8. Send a purgeRequest from Client with client's own publisher_id.
9. Search for metadata on ip-address 1.2.3.4

identifier	Identifier: ip-address Type: ipv4 Value: 1.2.3.4

10. Disconnect client.

Expected Outcomes:

- Purge request in Step 4 and 6 MUST receive "AccessDenied" response
- Searches in Steps 5 and 7 MUST return mymeta1 data.
- Purge request in Step 8 MUST receive "purgePublisherReceived"
- Search in Step 9 MUST NOT return any metadata (data purged).

Anticipated Failures:

- If purges with unknown or missing publisher IDs do not return an error and/or if they delete any data.
- If valid purge publisher request does not return purgePublisherReceived and purge data.

5.1.2.52 Metadata delete by lifetime="session" should be atomic

CTNC-IFMAP2.0-SERVER-TC-52.

Purpose: To verify when server deletes all metadata that came from a particular client and had lifetime="session", the Server MUST ensure that this bulk deletion appears atomic.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-96-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client2 requests new session from server.
3. Client2 sends publish update request with the following data in a single request packet.

identifier	Identifier: device Name: id1
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy
Lifetime	session

identifier	Identifier: device Name: id2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy
Lifetime	session

4. Client requests new session from server.
5. Client send subscribe request with the following data. Poll twice (ignore initial poll result):

Search1	Identifier: device Name: id1
Search2	Identifier: device Name: id2

6. Client2 send endSession request.
7. Disconnect client.

Expected Outcomes:

- Publish request at step#3 MUST respond in “publishReceived”.
- Subscribe from step#5 MUST respond in “subscribeReceived”.
- A single poll result MUST be received by Client after step#6, and pollResult MUST return the following results in any order:

  1. deleteResult:

identifier	Identifier: device Name: id1
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy

  2. deleteResult:

identifier	Identifier: device Name: id2
Metadata1	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy
Metadata2	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta2 Cardinality: multiValue Element name: name Element value: dummy

Anticipated Failures:

- If metadata delete caused by Client2 endSession and data published with lifetime=session does not trigger poll result to Client with relevant subscription.
- If poll results received by Client does not include all metadata published in Step 3, the test fails.

5.1.2.53 SSRC MUST NOT be used for poll requests.

CTNC-IFMAP2.0-SERVER-TC-53.

Purpose: To verify that the SSRC MUST NOT be used for poll requests.

This test case is for the following requirements: [[CTNC-IFMAP2.0-SERVER-REQ-100-M]]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Client send subscribe request with the following data:

identifier	Identifier: device Name: id1

4. Client send poll request using SSRC.
5. Disconnect client

Expected Outcomes:

- Subscribe request from step#3 MUST result in "subscribeReceived".
- Poll using SSRC at step#4 MUST result in error

Anticipated Failures:

- If poll request from SSRC does not return error.

5.1.2.54 Allowed messages on ARC.

CTNC-IFMAP2.0-SERVER-TC-54.

Purpose: To verify the following IFMAP messages MAY be communicated over the ARC: poll and response. Other messages MUST NOT be communicated over the ARC.

This test case is for the following requirements:[CTNC-IFMAP2.0-SERVER-REQ-101-M]

Preconditions:

- Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Client send publish update request with the following data using an ARC:

identifier	Identifier: device Name: id1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

4. Client send search request with the following data using an ARC:

identifier	Identifier: device Name: id1

5. Client send purge request of client's own publisher_id using an ARC
6. Client send subscribe request with the following data using an ARC:

identifier	Identifier: device Name: id1

7. Client send renewSession request using ARC.
8. Client send endSession request using ARC.
9. Disconnect client

Expected Outcomes:

- Publish request from step#3 MUST result in error.
- Search request from step#4 MUST result in error.
- Purge request from step#5 MUST result in error.
- Subscribe request from step#6 MUST result in error.
- renewSession request from step#7MUST result in error.
- endSession request from step#8 MUST result in error.

Anticipated Failures:

- If requests other than poll does not trigger error in ARC.

5.1.2.55 HTTP compression

CTNC-IFMAP2.0-SERVER-TC-55.

Purpose: To verify that, HTTP compression for responses, if used, MUST be negotiated according to HTTP 1.1. MAP servers MUST accept requests that have been compressed using gzip and identity transformations.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-104-M]] and [[CTNC-IFMAP2.0-SERVER-REQ-105-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client send a newSession request to server, with following http packet:

```
GET /index.html HTTP/1.1
Accept-Encoding: gzip
Encoded-Content: gzip

[GZIP COMPRESSED REQUEST]
```

3. Disconnect client

Expected Outcomes:

- newSession request with compressed response MUST return in one of the following http packet:

```

HTTP/1.1 200 OK
Content-Type: text/html
Content-Encoding: gzip

[GZIP COMPRESSED RESPONSE]
```

.

```
HTTP/1.1 200 OK
Content-Type: text/html

[NON COMPRESSED RESPONSE]
```

Anticipated Failures:

- If MAP server returns error for compressed MAP request.
- If response from MAP server is encoded using some format other than gzip format.

5.1.2.56 newSession request with max-poll-result-size.

CTNC-IFMAP2.0-SERVER-TC-56.

Purpose: To verify if the MAP Client included a max-poll-result-size attribute in its newSession request, the MAP Server MUST include a max-poll-result-size in its response indicating the actual amount of buffer space available for poll results for the client. A MAP Server MUST support buffer sizes of at least 5,000,000 bytes.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-106-M] and [CTNC-IFMAP2.0-SERVER-REQ-108-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server with a max-poll-result-size attribute of size 214748364 bytes. (max allowed for a signed integer)
3. Disconnect client

Expected Outcomes:

- Server MUSTreturn a new session containing max-poll-result-size of at least 5,000,000 bytes.

Anticipated Failures:

- If client includes max-poll-result-size in newSession request, and MAP server does not specify the max-poll-result-size in the newSessionResult.

5.1.2.57 newSessionResult content

CTNC-IFMAP2.0-SERVER-TC-57.

Purpose: To verify if the MAP Server can create a session for the client, the MAP Server’s response MUST contain a “newSessionResult” element that has a “session-id” attribute that specifies the MAP Client’s session-id along with a “ifmap-publisher-id” attribute that the MAP Client can use to recognize metadata that it published by examining operational attributes.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-107-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client send a newSession request to server:
3. Disconnect client

Expected Outcomes:

- newSession request MUST return the a newSessionResult containing “session-id” and “ifmap-publisher-id”.

Anticipated Failures:

- IF newSessionResult is not returned to authenticated client’s newSession request.

5.1.2.58 Multiple newSession requests

CTNC-IFMAP2.0-SERVER-TC-58.

Purpose: To verify if a MAP Client sends more than one SOAP request containing a “newSession” element in the SOAP body, the MAP Server MUST respond by ending the previous session and starting a new session. The server’s response MUST contain a “newSessionResult” element, and any state associated with the old session MUST be discarded. If an ARC is associated with the old session, the server MUST send an endSessionResult on the ARC.

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-109-M] and [CTNC-IFMAP2.0-SERVER-REQ-110-M] and [CTNC-IFMAP2.0-SERVER-REQ-111-M]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send following subscription request to IF-MAP server, then poll twice (ignore the initial pollResult).

Identifier	Identifier: ip-address Type: ipv4 Name: 10.0.0.6

4. Using the same connection, request a new session from the server.
5. Using the session created at step#2, send a dummy search request:

Identifier	Identifier: ip-address Type: ipv4 Name: 10.0.0.6

6. Disconnect client.

Expected Outcomes:

- Subscribe request at step#3 MUST respond in “SubscribeReceived”.
- Request new session at step#4MUST respond with “newSessionResult”.
- Poll result from step#3 MUST respond in “EndSessionResult”
- Search request using old session at step#5MUST respond with “EndSessionResult”

Anticipated Failures:

- If the client creates a new session on the same connection, and the existing ARC is not closed.

5.1.2.59 All requests must have session-id.

CTNC-IFMAP2.0-SERVER-TC-59.

Purpose: To verify each subsequent request (including a “poll” request on an ARC) MUST contain a “session-id” attribute in the top level element of the SOAP body, specifying the session-id assigned to the connection by the MAP Server.

This test case is for the following requirements: [[CTNC-IFMAP2.0-SERVER-REQ-112-M]]

Preconditions:

- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client send a newSession request to server. Capture session-id.
3. Send following publish request without specifying session-id.

identifier	Identifier: mac-address Name: 01:02:03:0d:0e:0f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

4. Send following search request without specifying session-id.

identifier	Identifier: mac-address

	Name: 01:02:03:0d:0e:0f

5. Send following subscribe request without specifying session-id.

identifier	Identifier: mac-address Name: 01:02:03:0d:0e:0f

6. Open an ARC to poll without specifying session-id.
7. Send a purge request without specifying session-id.
8. Send a renewSession request without specifying session-id.
9. Send a endSession request without specifying session-id.
10. Disconnect client

Expected Outcomes:

- Publish in step#3 without session-id MUST respond in error.
- Search in step#4 without session-id MUST respond in error.
- subscribe in step#5 without session-id MUST respond in error.
- poll in step#6 without session-id MUST respond in error.
- purge in step#7 without session-id MUST respond in error.
- renewSession in step#8 without session-id MUST respond in error.
- endSession in step#9 without session-id MUST respond in error

Anticipated Failures:

- If MAP server does not respond in error for any request not specifying a valid sessionID.

5.1.2.60 Incorrect session-id

CTNC-IFMAP2.0-SERVER-TC-60.

Purpose: To verify if the MAP Client specifies an invalid session-id, the MAP Server MUST indicate an InvalidSessionID errorResult in its response. A MAP Server MUST NOT accept a request from a MAP Client unless the session-id is valid for that client. A MAP Server MUST NOT permit one MAP Client to use a session that is created by another MAP Client.If the session-id is valid, the server MUST respond with a renewSessionResult element. Otherwise, the server MUST respond with an errorResult element, specifying an InvalidSessionID errorCode.

This test case is for the following requirements: [[CTNC-IFMAP2.0-SERVER-REQ-113-M]] and [[CTNC-IFMAP2.0-SERVER-REQ-114-M] and [[CTNC-IFMAP2.0-SERVER-REQ-115-M]] and [CTNC-IFMAP2.0-SERVER-REQ-119-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client send a newSession request to server. Capture session-id.
3. Send following publish request with an invalid session-id as “dummy”.

identifier	Identifier: mac-address Name: 01:02:03:0d:0e:0f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

4. Client2 send a newSession request to server. Capture session-id.
5. Client send following publish request using client2’s session-id.

identifier	Identifier: mac-address Name: 01:02:03:0d:0e:0f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

6. Disconnect clients

Expected Outcomes:

- Publish in step#3 with a non-existing session-id MUST respond in InvalidSessionID
- Publish in step#5 with a client2’s session-id MUST respond in InvalidSessionID

Anticipated Failures:

- If MAP server does not respond in error for any request with an invalid sessionID.

5.1.2.61 endSession request

CTNC-IFMAP2.0-SERVER-TC-61.

Purpose: To verify that a MAP Client MAY send an endSession request. If the session is valid, the MAP Server MUST respond with an endSessionResult response.

This test case is for the following requirements:[CTNC-IFMAP2.0-SERVER-REQ-116-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Send a valid IF-MAP endSession request to IF-MAP server.
4. Send a dummy search request to IF-MAP server

Identifier	Identifier: ip-address Type: ipv4 Name: 10.0.0.5

5. Disconnect client.

Expected Outcomes:

- Client MUST beresponded with new session in step2.
- The IF-MAP server MUSTrespond with endSessionResult to client in step#3.
- Search from step#4 MUST respond in “InvalidSession”

Anticipated Failures:

- If endSessionResult is not responded to endSession request.
- If session ID is still usable after a successful endSession.

5.1.2.62 Duplicate ARC

CTNC-IFMAP2.0-SERVER-TC-62.

Purpose: When a session ends for any reason, and there is an outstanding poll request on the ARC, the MAP Server MUST send an endSessionResult to the MAP Client on the ARC. If a MAP Server receives a message containing a SOAP body containing a poll element that specifies a session which already has an ARC with an outstanding poll request, the MAP Server MUST:
- end the session
- respond to the poll request on the older ARC with an endSessionResult
- respond to the poll request on the newer ARC with an errorResult response with an errorCode of InvalidSessionID

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-117-M] and [CTNC-IFMAP2.0-SERVER-REQ-118-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client requests new session from server.
3. Client sends a subscription request to IF-MAP server, thensends two poll requests, ignoring the initial pollResult).

Identifier	Identifier: ip-address Type: IPv4 Value: 10.0.0.6

4. Send another IF-MAP poll request while the previous poll is still in progress.

Session ID	Specify the Session ID of the client

5. Client send a dummy IF-MAP search request to IF-MAP server from the existing SSRC.

Identifier	Identifier: ip-address Type: IPv4 Value: 10.0.0.6

6. Disconnect client.

Expected Outcomes:

- Add Subscription list in step 3 MUST respond in “subscribeReceived”.
- Poll result from step#3 MUST return “endSessionResult” once step#4 is requested.
- Poll result from step#4 MUST return “InvalidSessionID”.
- Search MUST return “endSessionResult” or “InvalidSessionID” in step#5

Anticipated Failures:

- If the same client sends two poll requests using same session ID, and all the SSRC and ARC is not terminated.

5.1.2.63 Error at TCP or SSL layer of an ARC

CTNC-IFMAP2.0-SERVER-TC-63.

Purpose: To verify that a MAP Server or MAP Client detects an error at the TCP or SSL layer of an ARC, the MAP Server or Client MUST end the session.

This test case is for the following requirements: [[CTNC-IFMAP2.0-SERVER-REQ-120-M]]

Preconditions:

- o Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client sends a newSession request to server over https.
3. Client sends following subscribe request and then polls once to get initial poll result

identifier	Identifier: mac-address Value: 01:02:03:0d:0e:0f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

4. Client sends an unencrypted or otherwise invalid SSL data instead of a valid SSL encrypted poll.
5. If the SSRC TCP connection is still open, client sends the following search request:

identifier	Identifier: mac-address Value: 01:02:03:0d:0e:0f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

6. Disconnect client

Expected Outcomes:

- Subscribe request in step#3 MUST respond in "subscribeReceived".
- In step#4 server MUST respond by closing the TCP ARC connection. The server may also close the TCP SSRC connection.
- Poll with corrupted SSL or TCP layer in step#4 MUST return in either TCP connection closed, or EndSessionResult.

- Search request in step#5 MUST either return EndSessionResult or return an errorResult containing invalidSessionID.

.

Anticipated Failures:

- If session ID is still valid after a corrupted TCP or SSL layer of ARC.

5.1.2.64 Identify client's role, and allows read-only client.

CTNC-IFMAP2.0-SERVER-TC-64.

Purpose: To verify that upon successful authentication, the trusted client entities MUST be verified for authorization to serve the MAP Client role. MAP Servers MUST allow the administrator to configure read-only access for MAP Clients.

This test case is for the following requirements: [[CTNC-IFMAP2.0-SERVER-REQ-124-M]] and [[CTNC-IFMAP2.0-SERVER-REQ-125-M]]

Preconditions:

- o Devices configured to "Common Setup".

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client-readonly sends a newSession request to server.
3. Client-readonly sends the following publish request:

identifier	Identifier: mac-address Value: 01:02:03:0d:0e:0f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

4. Client-readonly sends a purge request to purge its own publisher-id.
5. Cliend-readonly sends the following search request:

identifier	Identifier: mac-address Value: 01:02:03:0d:0e:0f
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

6. Disconnect clients

Expected Outcomes:

- Search request in step#5 MUST result with identifier returned.
- Publish in step#3 MUST result with AccessDenied.
- Purge in step#4 MUST result with AccessDenied

Anticipated Failures:

- If the read-only client is allowed to publish.

5.1.2.65 Multiple SSRC handling.

CTNC-IFMAP2.0-SERVER-TC-65.

Purpose: To verify that when a new SSRC connection is established for a client that is already connected, the new connection succeeds and the old connection is closed or invalidated

This test case is for the following requirements: [CTNC-IFMAP2.0-SERVER-REQ-99-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from client side.
2. Client sends a newSession request to server.
3. Client sends the following publish request:

identifier	Identifier: ip-address Value: 1.1.1.1
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

4. Client establishes a new TCP connection and searches for identifier ipv4(1.1.1.1) using the session ID from step 2
5. Using the connection from step 2 search for identifier ipv4(1.1.1.1)
6. Client establishes a new TCP connection and sends a newSessionRequest
7. Using the connection from step 4 search for identifier ipv4(1.1.1.1)
8. Using the connection from step 6 search for identifier ipv4(1.1.1.1)

Expected Outcomes:

- Search request in step#4MUSTfind custom metadata.
- newSession request in step #6 must return a new session id

- Search in step #8 MUST find custom metadata

Anticipated Failures:

- Search request in step#5 MUST result in an endSessionResponse,AccessDenied error response, or a dropped TCP connection.
- Search request in step#7 MUST result in either an endSessionResponse or AccessDenied error.

5.2 IF-MAP Compliance Test Cases for MAP Client

In the IF-MAP Compliance Test Cases for MAP Clients, the Device Under Test (DUT) is the IF-MAP Client. To verify that the MAP client correctly implements the IF-MAP 2.0 specification, the user should trigger all possible IF-MAP requests from MAP Client, while the test program will monitorand validate these requests to ensure anticipated requests are received, as well as respond with various packets.

When the test is initiated, the following questionnaire needs to be answered by the test executor to allow the test program to validate the correct client implementation.

1. What IF-MAP operation(s) are used by the DUT?
2. What IF-MAP identifier(s) are used by the DUT?

5.2.1 Always Available

Following are “Always Available” (AA) test cases, this behaviour should be monitored through out the test, and any mismatching behaviour will immediately be flagged as a failure.

5.2.1.1 Supports both standard and vendor-specific metadata

[CTNC-IFMAP2.0-CLIENT-AA-1]

Purpose: To verify that MAP client must support both standard and vendor-specific metadata, MUST NOT send metadata that does not comply with the relevant schema, and does not include publisher-id or timestamp in published metadata.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-4-M] and [CTNC-IFMAP2.0-CLIENT-REQ-6-M] and [CTNC-IFMAP2.0-CLIENT-REQ-48-M]

Preconditions:

- o None.

Expected Outcomes:

- •If a client sends publish update or notify request, metadata MUST be standard or vendor-specific metadata, containing appropriate namespace, syntax and attributes
- •Standard metadata MUST have following namespace

  xmlns:meta=”http://www.trustedcomputinggroup.org/2010/IFMAP-METADATA/2

Anticipated Failures:

- •If client sends publish update or notify with metadata that does not match standard or vendor-specific schema.
- •If client includes publisher-id and/or timestamp in published metadata.

5.2.1.2 Client supports UTF8

[CTNC-IFMAP2.0-CLIENT-AA-2]

Purpose: To verify that MAP client must supportUTF8.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-7-M]

Preconditions:

- o None.

Expected Outcomes:

- If a client sends publish update or notify request, all values MUST be in ASCII or UTF8. Including SOAP envelope, namespaces, prefix, identifier values, metadata name, metadata values.
- If a client sends publish delete, search or subscribe, all values MUST be in ASCII or UTF8, including SOAP envelope, namespaces, prefix, filter values, subscription name.

Anticipated Failures:

- If a non ASCII or UTF8 encoding is found in any of the request packet.

5.2.1.3 Access-request without administrative-domain

[CTNC-IFMAP2.0-CLIENT-AA-3]

Purpose: To verify that when utilizing the AccessRequestType specified in IFMAP, a MAP Client MUST NOT specify an administrative domain.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-9-M]

Preconditions:

- o None.

Expected Outcomes:

- When a client sends publish, search or subscribe requests, if the identifier type is ‘access-request’, then it MUST NOT specify ‘administrative-domain’ in the identifier.

Anticipated Failures:

- If client sends a request containing ‘access-request’ type identifier, and contains ‘administrative-domain’.

5.2.1.4 device without aik-name

[CTNC-IFMAP2.0-CLIENT-AA-4]

Purpose: To verify that MAP Clients MUST NOT create a device identifier with an aik-name.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-11-M]

Preconditions:

- None.

Expected Outcomes:

- When a client sends publish, search or subscribe requests, if the identifier type is ‘device’, then it MUST NOT specify ‘aik-name’ in the identifier.

Anticipated Failures:

- If a client request contains identifier type ‘identity’ and also includes ‘aik-name’.

5.2.1.5 Identity identifier syntax

[CTNC-IFMAP2.0-CLIENT-AA-5]

Purpose: To verify that a MAP Client MUST specify a name field in an identity identifier consisting of a non-empty string. A MAP Client MUST specify a type in an identity identifier in order to provide a hint to MAP Clients how to parse the name field. Also, any HIT identifiers MUST be expressed as x:x:x:x:x:x:x:x, where the 'x's are the lowercase hexadecimal values of the eight 16-bit pieces of the HIT. No leading zeros are allowed except that the number 0 is represented by a single 0 character. And if a MAP Client specifies identity type as “other”, the client MUST specify a non-empty string for the other-type-definition field with the format of “Vendor-ID:Name” or “Name”.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-13-M], [CTNC-IFMAP2.0-CLIENT-REQ-14-M], [CTNC-IFMAP2.0-CLIENT-REQ-15-M], [CTNC-IFMAP2.0-CLIENT-REQ-16-M]

Preconditions:

- None.

Expected Outcomes:

- When a client sends publish, search or subscribe requests, if the identifier type is ‘identity, then it MUST specify a non-empty string for the ‘name’ field in the identifier.
- When a client sends publish, search or subscribe requests, if the identifier type is ‘identity, then it MUST specify one of the following valid ‘type’ in the identifier:
  - aik-name
  - distringuished-name
  - dns-name
  - email-address
  - Kerberos-principal
  - username
  - sip-uri
  - tel-uri
  - hip-hit
  - other
- If type is specified as ‘hip-hit’, then the name field MUST be expressed as x:x:x:x:x:x:x:x, where the 'x's are the lowercase hexadecimal values of the eight 16-bit pieces of the HIT.

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

- If type is specified as 'other', then the client MUST specify a non-empty string for the other-type-definition field. The other-type-definition field's value must take one of two forms:
    - Vendor-ID:Name
    - Name

Anticipated Failures:

- When client's request contains identifier type as 'identity', and if any of the expected behavior is not met.

5.2.1.6 Ip-address identifier syntax

[CTNC-IFMAP2.0-CLIENT-AA-6]

Purpose: To verify that aIP addresses MUST be canonicalized by MAP Clients.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-17-M]

Preconditions:

- None.

Expected Outcomes:

- When a client sends publish, search or subscribe requests, if the identifier type is 'ip-address, and if 'type' is 'IPv4' or if 'type' is not defined, then the value of the identifier MUST follow the following syntax:

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

  IPv4 address => octet "." octet "." octet "." octet

  octet => 0..255

- When a client sends publish, search or subscribe requests, if the identifier type is 'ip-address, and if 'type' is 'IPv6', then the value of the identifier MUST follow the following syntax:

  The canonical form of an IPv6 address is x:x:x:x:x:x:x:x, where the 'x's are the lowercase hexadecimal values of the eight 16-bit pieces of the address.

  Examples:

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

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

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

Anticipated Failures:

- When client's request contains identifier type as 'ip-address', and if any of the expected behavior is not met.

5.2.1.7 mac-address identifier syntax

[CTNC-IFMAP2.0-CLIENT-AA-7]

Purpose: To verify that a MAP Clients MUST specify the MAC address as six groups of two lowercase hexadecimal digits, separated by colons (:) in transmission order.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-18-M]

Preconditions:

- None.

Expected Outcomes:

1. When a client sends publish, search or subscribe requests, if the identifier type is 'mac-address', then the value of the identifier MUST be a MAC address as six groups of two lowercase hexadecimal digits, separated by colons (:) in transmission order, e.g. 01:23:45:67:89:ab. Leading zero is required.

Anticipated Failures:

- When client's request contains identifier type as 'mac-address', and if any of the expected behavior is not met.

5.2.1.8 Filter syntax

[CTNC-IFMAP2.0-CLIENT-AA-8]

Purpose: To verify that all FilterType values MUST be constructed with the IF MAP filter syntax.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-24-M]

Preconditions:

- None.

Expected Outcomes:

- When a client sends publish delete, search or subscribe requests, if filter, result-filter or match-links is supplied, then the value MUST follow the filter syntax below:

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

Examples:

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

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

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

Anticipated Failures:

- All publish delete, search and subscribe request packet with filter, result-filter or match-links does not use the above grammar.

5.2.1.9 Filter prefix declaration

[CTNC-IFMAP2.0-CLIENT-AA-9]

Purpose: To verify that the namespace prefixes in a filter MUST be declared in the XML document containing the filter, and the declaration of the namespace prefix MUST apply to the scope of the filter.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-25-M]

Preconditions:

- o None.

Expected Outcomes:

- When a client sends publish delete, search or subscribe requests, if filter, result-filter or match-links is supplied, and QName prefixes is specified, then the namespace prefix(es) MUST be declared in the SOAP envelope.

Anticipated Failures:

- If a client’s search, subscribe, delete request contains filter with QNAME, but namespace declaration is missing.

5.2.1.10 Session-id in all requests

[CTNC-IFMAP2.0-CLIENT-AA-10]

Purpose: To verify that all requests except for newSession MUST include a session-id attribute, which is specified in the IF-MAP schema using the sessionAttributes attributeGroup. renewSession request MUST also specify a valid session-id.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-26-M] and [CTNC-IFMAP2.0-CLIENT-REQ-42-M]

Preconditions:

- o None.

Expected Outcomes:

- For renewSession, endSession, publish, search, subscribe, notify, poll and purge requests, session-id MUST be supplied.

Anticipated Failures:

- if renewSession, endSession, publish, search, subscribe, notify, poll and purge request is missing session-id.

5.2.1.11 Poll on ARC

[CTNC-IFMAP2.0-CLIENT-AA-11]

Purpose: To verify that a MAP Client that issues subscribe requests MUST create an ARC and issue poll requests, SSRC MUST NOT be used for poll requests.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-35-M] and [CTNC-IFMAP2.0-CLIENT-REQ-31-M]

Preconditions:

- None.

Expected Outcomes:

- If a client sends a validate subscription request, after server has responded with subscription received, the client MUST open an ARC and request for poll.

Anticipated Failures:

- If a client only subscribe, but doesn't poll.
- If poll is requested on a SSRC.

5.2.1.12 One SSRC and one ARC

[CTNC-IFMAP2.0-CLIENT-AA-12]

Purpose: To verify that an IFMAP session consists of one synchronous send-receive channel (SSRC) and no more than one optional asynchronous receive channel (ARC); a MAP Client MUST open exactly one SSRC and no more than one ARC. The following IFMAP messages MAY be communicated over the ARC: poll and response. Other messages MUST NOT be communicated over the ARC .A MAP Client MUST have no more than one ARC active at a time, meaning that a MAP Client MUST NOT have more than one outstanding “poll” request for a particular session.A MAP Client MUST NOT send any IF-MAP request other than “poll” on an ARC. A MAP Client must have a valid session with a MAP Server before opening an ARC.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-34-M] and [CTNC-IFMAP2.0-CLIENT-REQ-36-M] and [CTNC-IFMAP2.0-CLIENT-REQ-38-M] and

[CTNC-IFMAP2.0-CLIENT-REQ-37-M] and [CTNC-IFMAP2.0-CLIENT-REQ-40-M] [CTNC-IFMAP2.0-CLIENT-REQ-40-M]

Preconditions:

- None.

Expected Outcomes:

- At one time, the same session-id MUST only be used on one SSRC (renew, publish, search, subscribe, purge). And this session-id MUST only be used on one ARC (poll).

Anticipated Failures:

- If client uses the same session-id in two SSRC or two ARC sessions.
- If client uses ARC for NewSession, RenewSession, publish, search, subscribe, notify, purge request, or uses SSRC for poll request.

5.2.1.13 Ifmap- attributes in publish are correct

[CTNC-IFMAP2.0-CLIENT-AA-13]

Purpose: To verify that a MAP client MUST define the ifmap-cardinality attribute of any metadata as singleValue or multivalued, and it must not contain any additional ifmap- prefixed attributes or operational attributes in a publish request.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-19-M] [CTNC-IFMAP2.0-CLIENT-REQ-20-M] [CTNC-IFMAP2.0-SERVER-REQ-22-M] and CTNC-IFMAP2.0-CLIENT-REQ-47-M

Preconditions:

- This test case only applies if the IF-MAP client can do publish operation.
- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a publish update request.
3. Disconnect client

Expected Outcomes:

- The publish request from client MUST contain metadata attribute ‘ifmap-cardinality’ with value of either “singleValue” or “multiValue”.
- The publish request from the client MUST NOT contain metadata attributes ‘ifmap-publisher-id’ or ‘ifmap-timestamp’

Anticipated Failures:

- N/A

5.2.1.14 Validity of subscription request names

[CTNC-IFMAP2.0-CLIENT-AA-14]

Purpose: To verify that the value of the name attribute for a subscribed search MUST be a string between 1 and 20 characters in length, such as a unique integer value for each search the client subscribes to. The client MUST specify the name attribute in every update element of a subscriptionRequest.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-27-M] and [CTNC-IFMAP2.0-CLIENT-REQ-28-M]

Preconditions:

- This test case only applies if the IF-MAP client can do subscribe operations.
- Devices configured to “Common Setup”.

Test Steps:

4. Begin tracking SOAP transactions contents from server side.
5. Trigger client to send a subscribe update request.
6. Disconnect client

Expected Outcomes:

- Each search in the subscription request MUST contain a name.
- All the search names in the subscription request MUST be unique (consistently using the same name for a particular search over time and not reusing that name for another search) and have length between 1~20 characters.

Anticipated Failures:

- N/A

5.2.2 Client Test Cases

A number of client test cases might require additional interaction by a tester, such as running an external script to publish events. Some test cases will require no specific action by the server other than accepting and correctly handling requests. Test cases that require specific custom behavior from the test server will have a server requirements section.

5.2.2.1 Security between MAP Server and Client

CTNC-IFMAP2.0-CLIENT-TC-1

Purpose: To verify that communication between MAP Clients and Servers MUST be authenticated and integrity-protected against unauthorized modifications enroute, provide confidentiality against unauthorized disclosure, and MUST NOT be susceptible to replay attacks.The IF MAP binding for SOAP described in this document requires that the IF MAP protocol MUST be carried over TLS. The MAP Client MUST verify the MAP Server’s certificate and determine whether the MAP Server is trusted by this MAP Client before completing the TLS handshake.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-1-M] and [CTNC-IFMAP2.0-CLIENT-REQ-2-M] and [CTNC-IFMAP2.0-CLIENT-REQ-3-M] and [CTNC-IFMAP2.0-CLIENT-REQ-45-M] and [CTNC-IFMAP2.0-CLIENT-REQ-46-M]

Preconditions:

- Devices configured to “Common Setup”.

Server Requirements: server must be configured to only accept TLSv1 encrypted connections.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to request new session to server using SOAP over HTTPS with configured authentication method. Server will provide valid SSL server certificate.
3. Trigger client to send a request, capture packet, ensure packet is encrypted properly.
4. Disconnect client
5. Trigger client to request new session from server using SOAP over HTTPS with configured authentication method. Server will provide a SSL certificate that mismatches with the CA configured on the client.
6. Disconnect client

Expected Outcomes:

- Step#2 client MUST validate the correct newSession response from server.
- Step#3 captured packets between client and server MUST be encrypted.
- Step#5 clients MUST reject the connection from server due to incorrect certificate.

Anticipated Failures:

- If MAP client is not using HTTPS to communicate with MAP server.
- If MAP client does not validate MAP server certificate.

5.2.2.2 Client ignores unknown metadata

CTNC-IFMAP2.0-CLIENT-TC-2

Purpose: To verify that a MAP Client MUST ignore vendor-specific metadata that it does not understand. MAP Clients MUST ignore unrecognized vendor-specific metadata returned by searches and subscriptions.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-5-M]and [CTNC-IFMAP2.0-CLIENT-REQ-32-M]

Preconditions:

- This test case only applies if the IF-MAP client can do search or subscribe operation.
- Devices configured to “Common Setup”.

Server Requirements: server must respond to all subscribe/poll requests with valid vendor specific metadata.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a search or subscribe/poll request.

3. Server responds by including a dummy vendor-specific metadata:

identifier	<same identifier as used for search/subscribe>
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1 Cardinality: singleValue Element name: name Element value: dummy1

4. If possible, trigger the client to send another request to verify the client is still working.
5. Disconnect client

Expected Outcomes:

- Client MUST be able to accept the response containing dummy vendor-specific metadata and client does not crash.

Anticipated Failures:

- N/A

5.2.2.3 Client supports UTF8

CTNC-IFMAP2.0-CLIENT-TC-3

Purpose: To verify that MAP client mustsupportUTF8.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-7-M]

Preconditions:

- This test case only applies if the IF-MAP client can do search or subscribe operation.
- Devices configured to “Common Setup”.

Server Requirements: The server must respond to every search/poll request with a search response containing metadata with a valid UTF-8 prefix.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a search or subscribe/poll request.
3. Server responds by including metadata containing UTF8 prefix, name and value.
4. If possible, trigger the client to send another request to verify the client is still working.
5. Disconnect client

Expected Outcomes:

- Client MUST be able to accept the response containing UTF8 encoding and client does not crash.

Anticipated Failures:

- N/A

5.2.2.4 Client supports entire set of identifiers

CTNC-IFMAP2.0-CLIENT-TC-4

Purpose: To verify that all MAP Client implementations MUST support the entire set of identifiers.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-8-M]

Preconditions:

- o This test case only applies if the IF-MAP client can do search or subscribe operation or depth greater than 0.
- o Devices configured to "Common Setup".

Server Requirements: Server must respond to every search/poll request with a response containing a set of all possible identifiers linked to the requested identifier.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a search or subscribe/poll request with depth > 0.
3. Server responds by including linked identifiers of the following types:

   - access-request

   - device

   - identity

   - ip-address

   - mac-address
4. If possible, trigger the client to send another request to verify the client is still working.
5. Disconnect client

Expected Outcomes:

- Client MUST be able to accept the response containing all types of identifiers and client does not crash.

Anticipated Failures:

- N/A

5.2.2.5 Uniqueness of access-request name

CTNC-IFMAP2.0-CLIENT-TC-5

Purpose: To verify that MAP Clients MUST choose the value of the name attribute in such a way that the odds of having two logically different access-request identifiers or two device identifiers with the same name are negligible.

Server Requirements: provide logging of all publish requests.

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-10-M] and [CTNC-IFMAP2.0-CLIENT-REQ-12-M]

- This test case only applies if the IF-MAP client can do publish operation on access-request or device identifier or link.
- Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a publish request on access-request and/or device identifier/link. Save all the names for the identifiers/links and submit with the test report for human evaluation.
3. Disconnect client

Expected Outcomes:

- Human evaluator determines that name attribute values that have been chosen are not likely to be randomly duplicated.

Anticipated Failures:

- Human evaluators determine that name attribute values are likely to conflict.

5.2.2.6 Ignore unknown ifmap- attributes

CTNC-IFMAP2.0-CLIENT-TC-6

Purpose: To verify that metadata elements MUST NOT include attributes that begin with the ifmap- prefix other than attributes specified in this document, and any unrecognized metadata attributes beginning with the *ifmap-* prefix MUST be ignored by MAP Clients.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-20-M] and [CTNC-IFMAP2.0-CLIENT-REQ-21-M]

Preconditions:

- This test case only applies if the IF-MAP client can do search or subscribe operation.
- Devices configured to “Common Setup”.

Server Requirements: every search / poll request must result in a metadata response containing an ifmap-dummy attribute.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a search or subscribe/poll request.
3. In server response, include a metadata attribute of ‘ifmap-dummy = dummy’
4. If possible, trigger the client to send another request to verify the client is still working.
5. Disconnect client

Expected Outcomes:

- Client MUST be able to accept the response containing dummy metadata and client does not crash.

Anticipated Failures:

- N/A

5.2.2.7 Supported metadata size

CTNC-IFMAP2.0-CLIENT-TC-7

Purpose: To verify that MAP Clients MUST support metadata at least 100000 bytes in length

This test case is for the following requirements:[CTNC-IFMAP2.0-CLIENT-REQ-23-M]

Preconditions:

- This test case only applies if the IF-MAP client can do search or subscribe operation.
- Devices configured to “Common Setup”.

Server Requirements: server must respond to every publish/poll request with vendor specific metadata containing 100000 bytes.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a search or subscribe/poll request.
3. In server response, include a vendor specific metadata that is 100,000 bytes in length.

identifier	<same identifier as used for search/subscribe>
Metadata	Xmlns: myns1=http://ns1.example.com metadata: myns1:mymeta1

	Cardinality: singleValue Element name: name Element value: a string of 90,000 ASCII characters.

4. If possible, trigger the client to send another request to verify the client is still working.
5. Disconnect client

Expected Outcomes:

- Client MUST be able to accept the response containing dummy metadata and client does not crash.

Anticipated Failures:

- N/A

5.2.2.8 notifyResult with missing metadata

CTNC-IFMAP2.0-CLIENT-TC-8

Purpose: To verify that a MAP Client receiving a notifyResult MUST NOT assume that metadata missing from a notifyResult has been deleted.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-29-M]

Preconditions:

- This test case only applies if the IF-MAP client can do subscribe operation.
- Devices configured to “Common Setup”.

Special Note: since the server cannot know what metadata might be relevant to the client, the tester must use a utility provided in the test suite to publish location metadata and event metadata.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Client script publishes location metadata associated with an endpoint ip-address.
3. Trigger client to do a subscribefor the endpoint ip-address, and poll.
4. A client notifies on an event associated with the endpoint, e.g. botnet or worm infection.
5. Client polls again.
6. Send an IFMAP search request to IFMAP server with location filter.
7. If possible, trigger the client to send another request to verify the client is still working.

8. Disconnect client

Expected Outcomes:

1. Client MUST be able to accept the notifyResult response containing missingmetadata and does not crash

Anticipated Failures:

1. Client crashes or is otherwise inoperable after tests.

5.2.2.9 Communicating URI

CTNC-IFMAP2.0-CLIENT-TC-9

Purpose: To verify that a MAP Client MUST be capable of communicating with a MAP Server using any valid URI as specified in [10]and [7]. In particular, a MAP Client MUST be capable of communicating with a MAP Server using a tcp port specified in an https URI.

This test case is for the following requirements[CTNC-IFMAP2.0-CLIENT-REQ-33-M]

Preconditions:

- Devices configured to “Common Setup”.

Server Requirements:

- it must be possible to configure the server to listen on a non-default port
- the server must be configurable to accept different path configurations

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Configure MAP server to listen on https default port 443.
3. Trigger client to request a new session to MAP server without specify port.
4. Disconnect client
5. Trigger client to request a new session to MAP server and specify port 443.
6. Disconnect client
7. Configure MAP server to listen on https non-default port 1234.
8. Trigger client to request a new session to MAP server on port 1234.
9. Disconnect client
10. Configure MAP server to listen on a URI with an absolute path.

    https://<host>/abs_path
11. Trigger client to request a new session to MAP server with the absolute path.
12. Disconnect client
13. Configure MAP server to listen on a URI without an absolute path.

    https://<host>/
14. Configure MAP server to listen on an IDN(internationalized domain name) URI without an absolute path.

https://<host>/

15. Trigger client to request a new session to MAP server without the absolute path.
16. Disconnect client

Expected Outcomes:

- Client MUST be able to send valid request packets to MAP server using the URI in steps#3, 5, 8, 11, 15

Anticipated Failures:

- N/A

5.2.2.10 Poll on ARC

CTNC-IFMAP2.0-CLIENT-TC-10

Purpose: To verify that the following IFMAP messages MAY be communicated over the ARC: poll and response. Other messages MUST NOT be communicated over the ARC. A “poll” request MUST contain a session-id attribute referring to a valid session

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-36-M], [CTNC-IFMAP2.0-CLIENT-REQ-39-M]

Preconditions:

- This test case only applies if the IF-MAP client can do subscribe operation.
- Devices configured to “Common Setup”.

Server Requirements: the test server must implement SSRC/ARC connection handling as per the specification.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send a subscribe request on the SSRC
3. Server responds with a subscribeReceived message on the SSRC
4. Trigger client to send a poll request on the ARC
5. Server responds with the initial poll result in the ARC that client used for poll request.
6. If possible, trigger the client to send another request to verify the client is still working.
7. Disconnect client

Expected Outcomes:

- The poll request MUST be on an ARC, and MUST contain the same session-id that was used in subscribe SSRC.
- Client MUST be able to accept the poll result on the ARC and client does not crash.

Anticipated Failures:

- N/A

5.2.2.11 IFMAP protocol on SOAP v1.2

CTNC-IFMAP2.0-CLIENT-TC-11

Purpose: To verify that all implementations of the IFMAP protocol MUST support Simple Object Access Protocol (SOAP) v. 1.2 as defined in [5].

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-41-M]

Preconditions:

- o Devices configured to “Common Setup”.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Execute SOAP 1.2 compliant test suite.
3. Disconnect client

Expected Outcomes:

- All test cases MUST pass.

Anticipated Failures:

- N/A

5.2.2.12 Error at TCP or SSL layer of an ARC

CTNC-IFMAP2.0-CLIENT-TC-12

Purpose: To verify that if a MAP Server or MAP Client detects an error at the TCP or SSL layer of an ARC, the MAP Server or Client MUST end the session.

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-43-M]

Preconditions:

- o This test case only applies if the IF-MAP client can do subscribe operation.
- o Devices configured to “Common Setup”.

Server Requirements: the server must respond to any subscribe / poll requests by injecting random data into the ssl stream.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Trigger client to send subscribe/poll request.
3. Server responds poll/subscribe request with a corrupted SSL or TCP layer message.
4. If possible, trigger the client to send another request to verify the client is still working.
5. Disconnect client

Expected Outcomes:

- Step#3 client MUST NOT crash when receives packet with corrupted SSL or TCP layer.
- Step#3 serverMUST see client send endSession request on the SSRC, or simply request a newSession on a new SSRC.
- If newSession request sent, then Step 4 MUSTrespond successfully, with success criteria dependent upon request type.

Anticipated Failures:

- N/A

5.2.2.13 Certificate-based and Basic authentication

CTNC-IFMAP2.0-CLIENT-TC-13

Purpose: To verify that all MAP Servers and MAP Clients MUST implement both mutual certificate-based authentication and Basic Authentication. And per the HTTP specification [7], when basic authentication is in use, a MAP server MAY respond to any request that lacks credentials with HTTP code 401. A client MAY avoid this code by submitting basic auth credentials with every request. If it does not do so, a client MUST respond to this code by resubmitting the same request with credentials (unless the client is shutting down).

This test case is for the following requirements: [CTNC-IFMAP2.0-CLIENT-REQ-46-M] and [CTNC-IFMAP2.0-CLIENT-REQ-47-M]

Preconditions:

- Devices configured to “Common Setup”.

Server Requirements: the server must implement certificate and basic authentication correctly.

Test Steps:

1. Begin tracking SOAP transactions contents from server side.
2. Configure client and server with certificate based authentication.
3. Trigger client to send newSession using certificate based authentication.
4. Configure client and server with basic authentication.

5. Trigger client to send newSession using basic authentication.
6. Trigger client to send additional requests. Server will respond request with code 401.
7. If possible, trigger the client to send another request to verify the client is still working.
8. Disconnect client

Expected Outcomes:

- Step#3 serverMUST receive the client’s configured certificate.
- Step#6 client MUST resend the request with basic authentication credential when the server responds with an HTTP 401 error. Client MUST NOT crash or idle when receiving an HTTP 401 error.

Anticipated Failures:

- N/A

References

This section lists specifications and other documents that are referred to in the document. Since this document is informative (not normative), all of these references are informative with respect to this document.

Informative References

[1] Trusted Computing Group, TNC Architecture for Interoperability with IF-MAP, Specification Version 1.1, Feb 2010.

[2] Trusted Computing Group, TNC IF-MAP Binding for SOAP, Specification Version 2.0 R38, March, 2011.

[3] Trusted Computing Group, TNC Compliance and Interoperability Principles, Specification Version 1.01, Revision 0.06, 20 November, 2006 (Draft)

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

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

[6] Trusted Computing Group, *TNC IF-MAP Metadata for Network Security*, Revision 1.0, August 2010.

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

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

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

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

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

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

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

[14] TCG Compliance and Interoperability Guidelines, Revision 45, Apr 2008.