BSR/ASHRAE Addendum *cp* to
ANSI/ASHRAE Standard 135-2020

Public Review Draft

Proposed Addendum *cp* to Standard 135-2020, BACnet® - A Data Communication Protocol for Building Automation and Control Networks

Second Public Review (March 2024)
(Draft shows Proposed Changes to Current Standard)

This draft has been recommended for public review by the responsible project committee. To submit a comment on this proposed standard, go to the ASHRAE website at www.ashrae.org/standards-research--technology/public-review-drafts and access the online comment database. The draft is subject to modification until it is approved for publication by the Board of Directors and ANSI. Until this time, the current edition of the standard (as modified by any published addenda on the ASHRAE website) remains in effect. The current edition of any standard may be purchased from the ASHRAE Online Store at www.ashrae.org/bookstore or by calling 404-636-8400 or 1-800-727-4723 (for orders in the U.S. or Canada).

This standard is under continuous maintenance. To propose a change to the current standard, use the change submittal form available on the ASHRAE website, www.ashrae.org.

The appearance of any technical data or editorial material in this public review document does not constitute endorsement, warranty, or guaranty by ASHARE of any product, service, process, procedure, or design, and ASHRAE expressly disclaims such.

ASHRAE, 180 Technology Parkway NW, Peachtree Corners, GA 30092

[This foreword, the table of contents, the introduction, and the "rationales" on the following pages are not part of this standard. They are merely informative and do not contain requirements necessary for conformance to the standard.]

FOREWORD

The purpose of this addendum is to present a proposed change for public review. These modifications are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The proposed changes are summarized below.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-2020 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout. Only this new and deleted text is open to comment at this time. All other material in this document is provided for context only and is not open for public review comment except as it relates to the proposed changes.

The use of placeholders like XX, YY, ZZ, X1, X2, NN, x, n, ? etc., should not be interpreted as literal values of the final published version. These placeholders will be assigned actual numbers/letters only after final publication approval of the addendum.

135-2020*cp*-1 Addition of Authentication and Authorization

Rationale

BACnet/SC provided a mechanism to create secure networks where participation on the network is limited to those devices that possess a trusted TLS certificate. While this kept out rogue actors, it did not differentiate between the devices within the SC network, therefore they all have equal authority.

This framework provides a mechanism for adding strong identity (authentication) and permissions (authorization) to client devices so that target devices can allow or deny certain operations based on that identity and permission.

With this, devices are no longer equal, and fine-grained authorization policies can be specified as needed to limit certain operations to certain clients.

[Replace Clause 17, Virtual Terminal Services, p. 777]
[Note to reviewer: This new service category logically belongs with the other service clauses 13-16 and it had been separately decided that the outdated Virtual Terminal Services should be removed from the BACnet standard]

17 AUTHENTICATION AND AUTHORIZATION SERVICES

This clause describes mechanisms that are used for authentication and authorization of BACnet devices.

17.1 Overview

BACnet is fundamentally a protocol for the free interchange of information between devices. There are, however, some operations that need to be protected, either from an innocent but misconfigured device, or a deliberately malicious device.

BACnet/SC defines a way to create network segmentation that can be used as an alternative to VLAN or VPN technologies. These isolated network segments protect the devices from outside attacks by creating a "perimeter security". However, within the perimeter, all the devices have equal permissions as a collection of trusted peers. The concept of "zero trust" considers that there is no safe and trusted "inside" and that attacks from compromised peers is possible. The mechanism in this clause supports configurations where there are no trusted peers, and every protected operation is validated through selectable levels of authentication and authorization.

Historically, proprietary solutions were created to define which device is allowed to perform certain operations on other devices. For example, this could have been done with some form of "Allow List" in the end device or a "Firewall Router" that does packet inspection to allow or deny messages between networks.

Most operations in BACnet are "open operations", meaning they can be performed by any device without regard to a particular authorization policy. For example, this would include most "data sharing" that consists of reading or reporting public data, like sensor readings. However, some operations, such as writing control commands or changing configurations, are considered "protected operations" and should not be open to all devices to perform.

The process of defining which device can perform which protected operations in which other device(s) is known as authorization and is set by an authorization policy. The process by which a device asserts that it is the specific device that has been granted an authorization is known as authentication.

This clause defines mechanisms that can be used for a variety of situations that require a device to be authorized before it can perform a protected operation in another device. Performing a protected operation requires both authentication of the client device and the presence of policy that authorizes that client to perform the operation. See Clause 17.3.7 for a description of the variety of authentication options that are available for interactions among different capabilities of devices and networks.

For flexibility of deployment and interaction with newer and older client devices, the BACnet authorization mechanism supports two possible flows of authorization policy.

The first method is a "centralized policy" mechanism based on ACE-OAuth, RFC 9200. With this mechanism, the applicable authorization policy is delivered "on the fly" by the client device to the target device along with the request for a protected operation. The target device will trust this policy because it has been signed by a third device that it trusts. In OAuth terms, the target device is known as the "resource server", and the delivered policy is known as an "access token" and the trusted third party that signed the policy is the "authorization server". Aligning with modern OAuth practice, BACnet uses "sender-constrained" tokens that cannot be used if stolen, rather than "Bearer Tokens" that need to be protected.

The second method is a "distributed policy" flow that can be used where the client device does not, or cannot, send a token. This is analogous to a traditional Access Control List where the actions of the client are authorized by some locally configured rules in the target. In this case, the applicable authorization policy for the client must be pre-configured into the target before the protected operation is attempted by the client.

See Clause 17.4.2 for details of these two methods for deployment of policy information.

17.2 Trust of Intermediaries

Since BACnet is a routable protocol, messages often travel through intermediary devices such as hubs and routers. Therefore, in addition to the policy deployment options, the authorization policies provide flexibility to allow the site administrator to establish the required level of trust of the intermediary devices that is appropriate for a protected operation.

For example, one policy could state that only a direct connection is allowed, a different policy could state that the device must be on the same network, and another could state that any network is allowed as long as all the intermediaries are authorized to relay identity. See Clause 17.3.4.

The "direct connect" case might be appropriate for situations where the data to be transferred is confidential or critical and the devices do not trust any intermediaries. For example, in BACnet/SC, the intermediaries are "TLS terminating" devices, meaning that once a message has been delivered to the intermediary, it is no longer encrypted internally. It is therefore possible for a compromised intermediary to read the data in the message, modify the message, or even create a new message entirely. With direct connect, the receiver has immediate access to the sender's TLS certificate and thus sees its identity first-hand and data can be exchanged with end-to-end confidentiality.

The "same network" case might be appropriate for some operations because each BACnet/SC network has a common TLS certificate issuer, thus all devices have a common trust anchor for their identity. In this case, the devices have only one intermediary, the hub, and it is possible that all devices and the hub are maintained to a tighter level of monitoring and control than other networks.

Most situations will allow "any network" since that level of policy nonetheless still requires that the identity information is conveyed only by an authorized chain of intermediaries. As is true for any TLS terminating "middle box", e.g., a corporate proxy firewall, all BACnet hubs and routers must be maintained with proper patching and monitoring to ensure that they remain uncompromised.

For end devices, BACnet authentication and authorization provides assurance that a compromised device cannot obtain more privilege than it has normally. Since a device cannot spoof its own identity, it is not possible for it to pretend to be a more powerful device even if it obtains a powerful access token. In practical terms, this means that a compromised VAV box could send bogus information to its AHU, but it cannot start writing anywhere else as if it were the workstation.

17.3 Authentication

The "identity" of a BACnet device is its device instance number. Every device on the BACnet internetwork has a unique device instance number, and that number does not change as TLS certificates are rotated or if a defective device is replaced with a new device. In addition, there are many properties of the datatype BACnetDeviceObjectPropertyReference that use this single numeric identifier to locate the referenced device. Distributed authorization policies and access tokens use this persistent client identity so that they are not made invalid when the client certificate is changed or a client device is replaced. Therefore, for the mechanisms defined in this clause, the single interoperable and persistent identity of a device is its device instance number.

Authentication is the process of proving that a device's instance number can be trusted. Authentication of BACnet devices is provided by the BACnet datalink layer by some verifiable and unmodifiable means.

The identity of an originating device is determined by the first datalink that receives an original message, and then, if that datalink is not the final destination, the identity is relayed by authorized hubs and routers to the final destination. The mechanism for such relaying is defined in Clause 17.3.5.

17.3.1 Proof of Identity

In BACnet/SC, the device identity is provided by the operational certificate, and the trust in the certificate is the basis for trust in the identity. The device's instance number is provided by a Subject Alternative Name (SAN) entry in the TLS certificate. The SAN entry shall be a URI in the form defined by Clause Q.8. For example, "bacnet://1234".

17.3.2 Conveying Identity

The identity of the originator of a message is provided by a "Source" data attribute that accompanies the NPDU up the stack from the datalink layer. For BACnet/SC, the Source data attribute is conveyed between connection peers using the 'Source' Data Option. See Clause AB.2.3.X3.

The "Source" data attribute is defined to be an "Every Segment" data attribute, so it appears on every segment of a segmented message and its value must remain consistent. If any segment is received with a different or absent value, the entire message shall be dropped.

17.3.3 Relaying Identity

In addition to determining an originating device's identity, the datalink layer determines an intermediary device's authorization to relay the identity of other devices. This authorization consists of indications for both capability and permission, both of which must be true.

In BACnet/SC, the capability to relay identity according to the rules in Clause 17.3.3 is indicated by the 'identity relay' bit of the "Hello" Data Option, and the permission to relay is provided by query parameters on the SAN in the certificate. If the datalink is part of a router, the query parameter "router" is appended, and if the datalink hosts a hub function, the parameter "hub" is appended. For example, "bacnet://1234?router", "bacnet://1234?hub", "bacnet://1234?router&hub", "bacnet://1234?hub&router".

17.3.4 Datalink Rules

The "Source" data attribute is added to original messages as an attestation by the first datalink that it has directly seen the proof of identity of the originating device. Since the "Source" data attribute could be added maliciously and routed by older intermediaries, the acceptance or rejection of the attribute is determined by the downstream receivers.

The following rules are enforced at the data link layer upon receipt of a message containing an NPDU:

1. When a datalink receives an original message, if the identity is available for the originating connection peer, then the "Source" attribute shall be added before the message is sent up to the network layer.
2. When a datalink receives a forwarded or routed message, if the "Source" data attribute is present and the sending connection peer is not authorized to relay identity, then the attribute shall be removed.

The term “original message” means SNET and ‘Originating Virtual Address’ are both absent. And “forwarded or routed” means the opposite, i.e., SNET or ‘Originating Virtual Address’ are present. The phrase “identity is available” means that the originating peer has proof of identity as defined in Clause 17.3.1, and “authorized to relay” means that the sending peer is authorized to relay identity according to Clause 17.3.3

Note that if an intermediary is not authorized to relay a “Source” attribute, the attribute is removed from the message rather than dropping the message. This situation is not necessarily malicious and could simply be an older hub or router relaying a message from a newer device. In this case, the message gets through to the destination, but without strong authentication.

17.3.7 Authentication Scenarios

A BACnet internetwork often contains a mixture of device capabilities that were installed over time. Additionally, the devices likely have a variety of security requirements based on the criticality of their function. To support a wide range of appropriate interactions, BACnet enables tailoring authorization policies to meet the specific needs of the installation.

In a mixed system, there are generally four levels of security.

- A non-secure network is the lowest level and could allow a rogue device to be connected to the network and send messages to any other network.
- An externally-secured network is one where communication between the non-secure devices on that network are secured by some external means, such as a VLAN or physical security.
- A secure network is a network secured by cryptographic proof that a device has a right to join the network, e.g., BACnet/SC. All devices on this network, or on adjacent secure networks joined by secure routers, are known as "Secure Path" devices.
- An "Authenticated" device is a secure device that has been given a verifiable identity, e.g., a BACnet/SC certificate with an identity in it.

The difference between a Secure Path device and an Authenticated device is that a compromised Secure Path device can claim to be another device, whereas a compromised Authenticated device cannot. Therefore, even if an Authenticated device is compromised such that its behavior is no longer predictable or desirable, that device cannot claim to be a different device, and therefore its authorization is limited to what the uncompromised device was already authorized to do.

In addition to the mixture of device and network types, there is a mixture of requirements for protected operations that might need differing levels of authentication. Some policies protect operations that are critical and must be protected against malicious devices, while some policies are merely in place to protect against misconfigured client devices. See Clause 17.4 for details of device authorization.

17.4 Authorization

Once a client device has been authenticated to a target device, the target device needs a second piece of information to allow the protected operation. This is provided by an authorization policy. The target device consults this policy to determine if the protected operation is allowed. If it is not allowed, a suitable error response will be returned for confirmed services, or the operation will be silently discarded for unconfirmed operations. The following clauses define the information in a policy and the options for deployment.

17.4.1 Policies

To perform a protected operation in a target device, the client device must be granted authorization to do so. Authorization of protected operations is provided by an authorization policy that considers the authentication of the sending device among other factors. An authorization policy specifies the following:

- The time range that the policy is valid.
- The client device(s) that the policy applies to.
- Constraints on the client’s network location and authentication method
- The network location of the client and how it was authenticated.
- The target device(s) that the policy applies to.
- The set of permissions that the client is granted to perform on the target device(s).

17.4.2 Policy Deployment

For flexibility of deployment and interaction with new and existing clients, the BACnet authorization mechanism supports two possible flows of information.

The first is a "centralized policy" mechanism based on ACE-OAuth, RFC 9200. The capitalized terms here are from the RFC.

With this mechanism:

- An Authorization Server issues Access Tokens to a Client that authorizes the actions that can be taken by that Client.
- The tokens are signed by the Authorization Server and bound to the Client, so they cannot be used by others if stolen.
- The Client either tries a protected operation and finds that it needs a token, or it knows to ask for one ahead of time.
- The Client requests an Access Token from the Authorization Server for a specific target device or group of devices.
- The request contains a specific "authorization scope", a set of permissions that are being requested by the Client.
- If the Authorization Server contains a policy allowing the request, it returns an Access Token for the requested scope.
- The Client presents the token to the Resource Server (target device) along with the protected operation.
- The Resource Server verifies the token's signature with a key that it has been configured to trust.
- If the protected operation is allowed by the token's scope, then the operation succeeds.

The second is a "distributed policy" mechanism where the policy information is preconfigured into the target devices. This can be used where a client device does not, or cannot, send a token. This is analogous to an Access Control List where the client is authenticated remotely but authorized locally.

With this mechanism:

- An administrator or an automated "Policy Distributor" function creates an entry in the target device that authorizes the actions that can be taken by a client device.
- The client device attempts a protected operation, and the target device looks for an active policy that matches the client's instance number, it's location on the network, and its method of authentication.
- If a policy is found, and the operation is allowed by the policy's scope, then the operation succeeds.
- If an operation is disallowed, an error is returned letting the client device know what scope is needed to authorize the operation.

In both cases, if there are failures of authorization, the failures can be made visible in the Authorization_Status property in the target. It is a local matter how human operators or automated tools are notified to correct a situation where authorization should have been granted or to take measures to correct a compromised device.

Each of these mechanisms has its advantages.

The centralized policy mechanism allows granting a policy to a client without the need to update the distributed authorization policies in the target devices. This is especially advantageous for granting temporary access, e.g., for a technician or installer. In this case, the client device carries its authorization with it, and the target devices trust the authorization policy because they can verify that the access tokens were issued by the authorization server that they trust.

The distributed policy has the advantage of working with older secure clients that may not be aware of tokens. For example, a first generation BACnet Secure Connect device can present a strong authentication with its TLS certificate, but it cannot present an access token. Therefore, the policy information that would be in the token can be pre-configured into the target device and the policy will be evaluated as if the policy had been delivered by the token.

The distributed policy can also be used by nonsecure devices that nonetheless want to protect certain operations against misconfigured devices or devices outside their local network. This is analogous to an Allow List based on Device ID or MAC address. This is also useful for a nonsecure network that is physically secure or secured by some other means like a VLAN, where all the devices on that network have a reasonable level of trust of each other but do not trust messages routed from other networks.

It is an important design feature that the local policy information is in the same format and has the same capabilities as the information which is delivered by an access token. Effectively, a token just "delivers a policy". Therefore, to the administrator

of authorization policy, the two mechanisms are as similar as possible, and policies can be defined in one way but deployed differently depending on the needs of the devices involved.

See Clause 17.4.4 for how policy constraints can be crafted to meet the needs of a site with a variety of device capabilities and security requirements.

17.4.3 Authorization Scopes

Authorization policies define the permissions a client device has for protected operations at the target device. The set of individual permissions constitutes the combined "authorization scope" for the policy. The individual permissions are referred to as "scope identifiers", or sometimes just "scopes".

This standard defines "standard scopes" for common operations. Use of standard scopes greatly improves interoperability; therefore, their use is strongly recommended wherever their semantics apply.

If one of the standard scopes does not apply to a particular protected operation, a device can define its own set of "extended scopes". A human-readable description of these extended scopes can be provided by the Authorization_Scopes property. User interfaces can use the information in this property to construct textual choices to be presented along with the standard scopes.

[Note to reviewer, this table is moved here from Clause W.3.5]

Table 17-1. Predefined Scopes

Scope Identifier	Meaning
view	View (view private data - public data does not need authorization)
adjust	Adjust setpoints
control	Runtime Control (write values for the purpose of controlling actions)
override	Command Override (placing objects out of service, commanding at priorities that would limit runtime control, etc.)
config	Configuration and Programming (configuration and programming actions, adding objects etc.)
bind	Configuration of external references for which the server device will use its own credentials to read or write.
install	Installation (more technical configuration actions, adding IO points, uploading different application programs)
auth	Configuration of authorization-related data.
infrastructure	Announcements of network topology and time information.

For brevity, standard scopes are encoded using bits in the BACnetAuthorizationScope construct. Extended Scopes are encoded as a list of character strings.

17.4.4 Authorization Constraints

Authorization policies can specify not only which clients can use the policy but also set minimum requirements for where the client is located with respect to the target and how the client authenticates itself to the target.

These constraints consist of settings that specify the minimum requirement for two separate aspects: how “close” is the client to the target, and how “strongly” is the client authenticated. This allows the authorization requirements to be tailored to the particular needs of the site, taking into consideration the capabilities of the devices, hubs and routers involved, as well as the minimum required level of security to protect certain operations.

Each of the two setting is “hierarchical”, meaning that some choices are clearly “better” than the others, in increasing order. This allows minimum requirements to be configured while also accepting “better” security, i.e., “closer” or “stronger”, when it is available.

The first setting is called "Origin" and it specifies where the client device must be located on the network with respect to the target device. This allows the configuration of the level of trust in the infrastructure, e.g., "trust no one", "trust only my hub", or "trust the hubs and routers". The choices are:

Direct Connect	The client is directly connected to the target, i.e., no hubs or routers.
Same Network	The client is on the same network as the target. For BACnet/SC, this means that the client and target are connected to the same hub. For routers, this applies to messages originating from clients on any of its directly connected networks, regardless of which network the router uses for its I-Am announcements.
Any Network	The client is anywhere, any number of hubs and routers can be traversed.

The second setting is called "Authentication" and specifies how the client device asserts it identity to the target device. This allows the configuration of the level of protection against a compromised device claiming to be another device, e.g., "don't trust what any device says", "trust what secure devices say", or "trust anyone". The choices are:

Certified	The client has a proven identity certified by a trusted third party. This identity cannot be changed even if the device is compromised. For BACnet/SC, this identity is provided by a signed certificate that is either directly observed or is relayed by authorized identity relays.
Secure Path	The client is a secure device, and communications between client and target is restricted to a path of one or more secure networks. For BACnet/SC, the device identity is determined with a Secure Path I-Am or a Secure Path ReadProperty of the device's instance number.
Any Method	The device identity can be determined with a nonsecure I-Am or a nonsecure ReadProperty of the device's instance number.

Both settings are hierarchical, ordered from "most restricted" to "least restricted". A setting of "Same Network" also accepts directly connected clients, and a setting of "Any Network" also accepts clients on the same network and those that are directly connected. A setting of "Secure Path" also accepts clients with certified identities, and a setting of "Any Method" accepts any method of determining identity.

17.4.5 Configuration

The distributed policies are defined by the Authorization_Policy property of the Device object. See Clause 12.11.X3. The centralized authorization policy, defining the information for the authorization server for a device, is defined by the Authorization_Server property. See Clause 12.11.X1.

Each device trusts one authorization server. However, for flexibility in deployment scenarios, there is no requirement that all devices trust the same authorization server. Therefore, a site might have more than one authorization server, each serving a different population of devices. Note that the targets of broadcasts and groups must share the same authorization server since a single access token must be trusted by all intended recipients.

Devices that initiate the AuthRequest service are only required to be able to request access tokens from their own authorization server. If a site uses different authorization servers for different populations of devices, there is a possibility that a client cannot naturally get an access token for a target in a different population. This can be overcome in several ways: 1) the target device could be configured with a distributed policy for the client device so that a token is not necessary, or 2) the client's authorization server could relay the request to the target's authorization server, or 3) an advanced client can read the target's Authorization_Server property to determine the authorization server to use.

17.4.6 Access Tokens

An access token is a mechanism for delivering an authorization policy to the target device. A client device can request an access token from the authorization server and then deliver that token along with the protected operation.

If a token is delivered with a protected operation, then that token is the only thing considered for authorization. Any distributed policies and any previously received tokens are ignored. If the delivered token is invalid, then there is no authorization for the

operation, i.e., there is no "fall back" to a distributed policy or previous tokens. Failures can be indicated in the Authorization_Status property of the Device object.

An access token is an access policy wrapped with information that allows the target device to trust the authenticity and validity of the policy. An access token contains the following information:

- The device instance of the authorization server that generated the token, known as the ‘issuer’.
- The date and time the token was issued.
- The "audience" of the token, i.e., the list of devices and groups that can be the target of the token.
- The identification of the client device that is authorized to use the token.
- In indication of what the client is authorized to do.
- A digital signature, signed by the authorization server, that ensures the authenticity of the preceding data.

An access token is requested from an authorization server using the AuthRequest service. This request explicitly contains the client identifier of the device that will use the token. Therefore, it is possible for one device to retrieve a token that will be used by another device. This is known as the "on behalf of" data flow.

An access token is bound to a particular client, i.e., it is "sender constrained", it is not a "bearer token". They can only be successfully used by a properly authenticated client device. Since access tokens are sender constrained, the only reason to keep them confidential is to hide the fact that one device has been given authorization for another device. Since this information is usually not secret in a typical BACnet installation, there is no reason to apply special protection to the acquisition or transmission of the tokens. They are effectively "public information". If it is desired that a particular policy contained by a token remain confidential, it is possible that the authorization server could be configured to only issue tokens directly to the client device and only via a direct connection. The client devices would then be required to keep this information private and only use the token in a direct connection. Note that it is not possible to hide the fact that a device is communicating with another device. However, a direct connection can hide the contents of the communication. The token policy contains a setting for ‘direct-connect’ to support the limitation of the client to target communications. However, any restriction on the process of issuing these tokens from authorization server to client is a local matter.

The details of the BACnetAccessToken construct are as follows:

Field	Type	R/O
issuer	Unsigned	R
issued	BACnetDateTime	R
not-before	BACnetDateTime	O
not-after	BACnetDateTime	O
audience	Sequence of Integer32	R
client	Unsigned32	R
constraint	BACnetAuthorizationConstraint	R
scope	BACnetAuthorizationScope	R
key-id	Unsigned	R
signature	OctetString	R

The 'issuer' field indicates the device instance of the authorization server that issued the token. The 'issued' field indicates the local date and time of the authorization server when the token was created by the authorization server.

The 'not-before' and 'not-after’ define the active period when the token can be used.

The 'audience' field indicates where the token can be used. This contains a list of devices and/or groups represented as signed integer values and is required to contain at least one value. Positive numbers are device instances and negative numbers are groups, e.g., group 5 is represented as -5. Group number 1 is reserved to mean "all devices".

The 'client' field controls which client device can use the token.

The 'constraint' field limits the client’s network location with respect to the target device and how it is authenticated. See Clause 17.4.4. The 'authentication' constraint for an access token shall default to "Certified" to cause a strong binding to the authorized client. However, a setting of "Secure Path" can be used to reduce this binding to make it apply to secure devices that do not

have certified identities. This has the advantage of being able to be used through first-generation BACnet/SC hubs and routers that are not capable of forwarding identity information. While unlikely, it is possible that a compromised secure device can claim a different identity and use a stolen token. Therefore, this use case is supported for situations where it is warranted but should generally be considered an interim situation during the transition to strongly authenticated devices with authorized identity forwarders.

The 'scope' field specifies which operations are authorized for the client at the target devices, See Clause 17.4.3.

The ‘key-id’ field indicates which of the two possible signing keys in the Authentication_Server property was used to sign the token. A value of 1 means ‘signing-key-1’ and a value of 2 means ‘signing-key-2’.

The ‘signature’ field is the final field of the ASN-1 production of a BACnetAccessToken. It is a digital signature of all the octets preceding the ‘signature’ production. The algorithm for the signature shall be "ES256" (ECDSA using P-256 and SHA-256). The client does not use this information. The public key(s) required to validate the signature is(are) configured into the Authorization_Server property of the Device object of the target device. The private key shall be maintained confidentially by the authorization server.

17.4.7 Token Validation

When a client attempts a protected operation that is not covered by a distributed policy, the client must present an access token in a "Token" data attribute along with the NPDU. A "Token" data attribute contains the information in the BACnetAccessToken construct. The fields of this datatype are discussed in Clause 17.4.6. For BACnet/SC, the "Token" data attribute is conveyed with a "Token" Data Option.

The following procedure is described as a function that returns "success" or an error code and possibly information to make a "Hint". The actual implementation is a local matter. Since validation of a digital signature is a computationally intense operation, it is a local matter where in this process the validation takes place. If a device is concerned with leakage of information, it can choose to validate the signature up front. If the device is more concerned with denial-of-service attacks, then the validation can be deferred until the end. In any case, the use of a token cache will greatly aid in the computational burden of repeated presentation of the same token.

Upon receipt of an access token, the target device shall, in any order:

a) Optionally check if the token has been revoked. If it has been revoked, return a REVOKED_TOKEN error code. The method for such optional detection is a local matter.
b) Check that the ‘audience’ field matches the target device or one of the groups listed in the Authorization_Groups property. If not, return an INVALID_AUDIENCE error code.
c) Check the ‘not-before’ and ‘not-after’ fields if present. If the token is not currently valid, issue an INVALID_TOKEN error code.
d) Check the 'authentication' field of the 'constraint'. If the value is “Certified”, then check that there is a "Source" data attribute. If the value is “Secure Path”, then check that client identity is known by a Secure Path mechanism. If the 'authentication' constraint is not met, return a "NOT_AUTHENTICATED" error code. Note that it is not possible to receive a token through a nonsecure path, so “Any Method” is not a valid choice for 'authentication'.
e) Check that the ‘client’ field matches the identity determined in the previous step. If not return an INVALID_CLIENT error code.
f) Check that the origin of the message matches the 'origin' field of the 'constraint' field. If not, return an INVALID_CLIENT_ORIGIN error code. Note that "Same Network” means that SNET is absent and "Any Network" means that SNET is not considered.
g) Check that the ‘scope’ will allow the protected operation. If not, return the appropriate standard error code for the required scope or the EXTENDED_SCOPE_REQUIRED error code and the required extended scope identifier to be used with a "Hint" data attribute with the response.

The signature is validated as follows:

1) Optionally consult a cache of previously received tokens to see if the signature has already been validated. If the result was a failure, then return an "INVALID_SIGNATURE" error code, else return success. Note that any cache of previous

tokens must contain either the entire token exactly as received, or a hash of the token using a secure hash algorithm. A simple checksum or the retention of a few key fields is not sufficiently secure to detect tampering.

2) Look at the 'key-id' field in the token to see which key in the Authorization_Server property should be used for validation. A 'key-id' value of 1 will select 'signing-key-1' and a value of 2 will select 'signing-key-2'.
3) Validate the 'signature' field against all octets coming before the 'signature' field in the BACnetAccessToken production using the selected public key.
4) Optionally store the token, or a secure hash of the token, for both success and failure, in a cache to avoid validation of the signature in the future.

17.4.8 Hints

When a protected operation fails, the client may need to know how to recover in an automated fashion. This usually means knowing what scope is needed. A "Hint" data attribute returned with the error response can provide that information. For BACnet/SC, the Hint data attribute is conveyed in the 'Hint' Data Option. See Clause AB.2.3.X5.

When the EXTENDED_SCOPE_REQUIRED error code is returned, the responding device shall include a "Hint" data attribute with the error response that indicates the required scope.

For error responses that can include multiple failures, e.g., ReadPropertyMultiple, the returned hint shall include the authorization scope that will allow at least the first failed operation to succeed. It is a local matter whether the responding device tries to make a combined authorization scope that will allow all protected operations to succeed.

17.5 Conformance Requirements

The BACnet authentication and authorization mechanism allows flexibility in implementation while maintaining compatibility for all customer and site deployment choices.

17.5.1 Client Conformance

Initiating the AuthRequest service is an optional functionality in client devices. Client devices can obtain access tokens by either actively using the AuthRequest service, or passively accepting access tokens configured into it by some local means.

Active token retrieval by the client can be advantageous in situations where the target device's requirements for authorization change. For example, if the target device is reconfigured to require a different scope for a given resource that the client is currently accessing, it will begin to reject the protected operation. If the client is able to automatically retrieve an access token from an online authorization server, then access can be automatically restored. This can also be used when a client begins to talk to a target for the first time and, through rejections, it "learns" what permissions are needed.

Passive configuration of access tokens can be advantageous in situations where the authorization server is not online, either because it is temporarily offline, or the site is designed such that the authorization server is a temporary function that is never expected to be online continuously. In this case, a "helper tool" retrieves the access token and configures them into the client device. It is a local matter how passive configuration is performed. Note that access tokens are not secrets and therefore can be configured "in the clear". See also Clause 12.11.X5.

These methods are not exclusive, and clients can be implemented to support both for flexible adaptation to deployment choices. The capabilities of client devices shall be declared on the PICS statement.

17.5.2 Client Helper Conformance

For client devices that do not retrieve their own access tokens, a helper tool can be used to retrieve the access tokens and configure them into the client by some means. To support site deployment choices for the authorization server function, a helper tool shall be capable of using the AuthRequest service to retrieve tokens. It is a local matter if the helper tool can also obtain tokens by some non-standard means. See Clause 17.5.4.

17.5.3 Target Conformance

To support the site deployment choices of centralized policy and/or distributed policy, secure target devices that support authorization are required to support both methods, and both Authorization_Policy and Authorization_Server properties shall be present. The capacity of distributed policy configuration shall be declared on the PICS statement.

Note that the Authorization_Policy property can also be used by non-secure target devices to protect against misconfigured clients or require same-network devices only. In this case, support of token-based authorization is not possible and the Authorization_Server property is not used.

17.5.4 Authorization Server Deployment Options

Since target devices are required to support distributed policies in the Authorization_Policy property, the use of access tokens and an authorization server is not required on every site.

It is a site deployment choice whether the authorization server is always online as a reachable BACnet device. If long term tokens are created and no subsequent changes are made to target scope requirements, the authorization server is not needed to be online. Note that BACnet access tokens are "self-contained" and therefore do not require an active connection between the target device and the authorization server for token validation.

Vendor-specific helper tools might retrieve tokens by some mechanism other than the standard AuthRequest service. See Clause 17.5.2. However, to support helper tools and clients from other vendors, all authorization servers are required to support the AuthRequest service as an online BACnet device when needed.

17.6 AuthRequest Service

When a client determines that it needs an access token for a particular device, it uses the AuthRequest service to request a token from an authorization server.

The determination of which authorization server to contact for a given target is a local matter, with the exception that a client is required to use its own authorization server by default. The ability for the client to determine a different authorization server, by reading the target device's Authorization_Server property or by some other means, is optional. Likewise, the ability of an authorization server to relay this request to another authorization server on behalf of the original requestor is also optional.

This service grants Access Tokens. This service does not require a secure datalink like BACnet/SC because tokens are non-confidential. The tokens are bound to a specific client so they cannot be used if stolen and they are signed so they cannot be forged.

The service explicitly specifies the client to bind the token to, so it naturally supports the "on behalf of" flow by "helpers" getting tokens to be used by other devices.

17.6.1 AuthRequest Service Structure

The structure of the AuthRequest service primitive is shown in Table 17-X1. The terminology and symbology used in this table are explained in Clause 5.6.

Table 17-X1. Structure of AuthRequest Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Access Token Request	U	U(=)		
Result(+)			S	S(=)
Access Token Response			U	U(=)
Result(-)			S	S(=)
Error Type			M	M(=)
Error Details			U	U(=)

17.6.2 Argument

This parameter shall convey the information for the AuthRequest confirmed service request. For future extensibility, this parameter is a construct consisting of multiple request types, each defined to be optional. All service invocations shall have at least one of these options present. Note that there is only one request option defined at this time, the "Access Token Request".

17.6.2.1 Access Token Request

This parameter specifies the individual parameters for the request for an access token bound to a specific client for a given authorization scope to be used for a particular audience.

Table 17-X2. Structure of 'Access Token Request' Parameters

Parameter Name	Req	Ind	Datatype
Client	M	M(=)	Unsigned
Audience	M	M(=)	Sequence of Integer32
Scope	U	U(=)	BACnetAuthorizationScope

The 'Client' parameter specifies the device instance of the client device that the token is to be bound to. Only a device authenticating with this instance can successfully use the token.

The 'Audience' parameter indicates where the token can be used. This contains a list of devices and/or groups represented as signed integer values and is required to contain at least one value. Positive numbers are device instances and negative numbers are groups, e.g., group 5 is represented as -5. Group number 1 is reserved to mean "all devices".

The 'Scope' parameter indicates the requested standard authorization scope for the given audience. The server can return a different set of authorizations than those that were requested. If the 'Scope' parameter is absent, the client is requesting the server to return the "default scope" configured for the combination of 'Client' and 'Audience'. If such a default does not exist, the server shall return an error.

17.6.3 Result(+)

The 'Result(+)' parameter shall indicate that the service request succeeded. A successful result includes the following parameters. For future extensibility, this parameter is a construct consisting of multiple response types, each defined to be optional. All Result(+) service responses shall have at least one of these options present, and it shall correspond to the option present in the service request. Note that there is only one response option defined at this time, the "Access Token".

17.6.3.1 Access Token Response

This parameter, of type BACnetAccessToken, specifies the requested access token. The scope of the token might be less (contain fewer permissions) than that of the request. See Clause 17.4.6 for the definition of the members of this construct.

17.6.4 Result(-)

The 'Result(-)' parameter shall indicate that the service request has failed. The reason for failure shall be specified by the 'Error Type' parameter and optionally the 'Error Description' parameter.

17.6.4.1 Error Type

This parameter shall be used to report a service execution errors that result in a failure to grant the request.

The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
The device is not configured to grant access tokens.	SERVICES	OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
The specified client has no policy.	SERVICES	UNKNOWN_CLIENT

The specified audience has no policy.	SERVICES	UNKNOWN_AUDIENCE
The server does not have any policies for the requested audience.	SERVICES	UNKNOWN_AUDIENCE
The server does not have any policies for the requested client.	SERVICES	UNKNOWN_CLIENT
The server does not recognize the requested scope.	SERVICES	UNKNOWN_SCOPE
A requested scope was not provided, and the server does not have a configured default scope matching the client and audience.	SERVICES	NO_DEFAULT_SCOPE
The server does not have a policy that allows the requested combination of parameters	SERVICES	NO_POLICY

17.6.4.1 Error Description

This optional parameter can be used to report a human readable description of the error condition.

17.6.5 Service Procedure

After verifying the validity of the request, the responding BACnet-user shall attempt to grant the request based on the parameters provided. Note that there is only one possibility defined at this time, the "Access Token Request".

17.6.5.1 Access Token Request Procedure

This request is used to get an access token that will grant an authorization scope to a client for a set of target devices.

Note that this is only a "request" for a token. The authorization server decides which parts of this request will be granted and which parts will need to be modified. Following OAuth practice, if a client requests "too much scope", the authorization server will return a reduced scope token rather than returning an error. This is true even if the return scope has no privileges at all. However, if the client asks for any other aspect (audience, client, etc.), that can't be granted, the authorization server will return an error.

The 'Client' field is not optional and must be filled out. Since this service does not require security, and the client can't know that its request will flow through a path that maintains the "Source" data attribute, there is no guaranteed way for the authorization server to know who the client is. Typically, this field will just contain the instance of the client itself, but it could be different if a "helper" is requesting the token "on behalf of" another client that will use the token.

The 'Audience' list will typically contain the one device that the client is wishing to talk to. If the client knows it is talking to a group, it will include that in the audience. If the client needs to talk to all devices, it can request "Group 1", thus asking for the requested scope in "all devices". Again, it is up to the authorization server to decide if that is granted. It is typically best if the client asks for the minimum audience required and lets the authorization server decide to broaden its applicability if appropriate, based on its knowledge of what the client needs to do. However, the authorization server shall also be guided by the doctrine of least privilege, so broadening will likely be rare.

The 'Scope' field is optional. The absence of the 'scope' fields indicates that the token requestor does not know the specific scope that is being requested and is therefore requesting that the authorization server return the "default scope" that matches the client and audience. This can be useful for implementing simple policies for simple client devices that do not need to, or cannot, manage multiple tokens per target.

The server can return a scope in the token that is different from the requested scope. A token with a different scope might succeed if the authorization server intentionally issued the different scope based on its knowledge of the scopes in use in the target device and the expected range of operations of the client.

Since the client can look at the token's 'audience' as part of its strategy for picking a token from its cache for a given audience, the authorization server is not allowed to turn a requested device into a group if the client did not ask for a group to begin with.

In other words, clients are either "group-aware" or not. Returning a group instead of a device could confuse a non-group-aware client.

When a request is denied, either by the return of a Result(-) or a reduction of scope, the specific behavior of the authorization server is a local matter. However, since such denial or reduction is either the result of an attempt of an unauthorized action (i.e., an attack) or, more likely, the result of a configuration change in a legitimate device that now needs a new policy to perform its legitimate function, some kind of notification to a human operator or authorization management system is required. The manner of such notification is a local matter with the exception that, since clients will likely retry the request periodically, it is required that authorization servers provide features to rate limit repetition of identical notifications or provide a feature to silence specific notifications for a period of time to allow field devices to be reconfigured. The manner of notification shall be indicated on the PICS. See Annex A.

It is a local matter whether an authorization server has the option to be configured to hide error conditions to prevent leakage of information about its policies. In this case, the error situation table in Clause 17.6.4.1 is not required to be followed and the server can return any error code it chooses or not respond at all.

135-2020*cp*-2 BACnet/SC Changes to Support Authentication and Authorization

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 defines new data options for authentication and authorization information that can accompany encapsulated NPDUs.

Additionally, a new destination option is defined to indicate the capabilities of the SC connection peer.

This section also makes it clear that Subject Alternative Name is used for device identity but is not considered for datalink connectivity.

[Change Table AB-3, p. 1386]
[Note to reviewer: These clause numbers start at "X2" to reduce the chance of errors since the assigned values will very likely start at 2]

Table AB-3 BVLC Header Options

Header Option Type	Numeric Header Option Type	Description
Secure Path	1	See Clause AB.2.3.1
Hello	*x2*	*See Clause AB.2.3.X2*
Client Identity	*x3*	*See Clause AB.2.3.X3*
Hint	*x4*	*See Clause AB.2.3.X4*
Token	*x5*	*See Clause AB.2.3.X5*
Proprietary Header Option	31	See Clause ~~AB.2.3.2~~ *AB.2.3.[X5+1]*

[Rename Clause AB.2.3.1 to AB.2.3.1.1 and make new parent Clause AB.2.3.1]

AB.2.3.1 Standard Header Options

~~AB.2.3.1~~ *AB.2.3.1.1* Secure Path Header Option

The 'Secure Path' header option specifies, by its presence, whether the service being requested represents a message which has only been transferred by BACnet/SC data links and secure connect BACnet routers.
...

[Add new Clauses AB.2.3.1.2-6, p. 1386]

AB.2.3.1.2 Hello Header Option

The 'Hello' header option specifies the BACnet/SC protocol version and the capabilities of a connecting peer.

The 'Hello' header option consists of the following fields.

Header Marker	1-octet	'More Options' = 0 or 1, 'Must Understand' = 0, 'Header Data Flag' = 1, 'Header Option Type' = x2
Header Length	2-octets	Length of 'Header Data' field, in octets. Shall be 1
Header Data	1-octet	Required to include:

Capabilities	1-octet	8 bit flags. Bit 0 is 'authentication relay'. Bits 1-7 are reserved and shall each be 0.

This header option, if present, shall be a destination option in the 'Destination Options' parameter of the BVLC messages 'Connect Request' and 'Connect Accept'.

This header option shall be included with the 'Connect Request' and 'Connect Accept' messages and shall only be included with those messages. An SC connection shall record the capabilities of the connecting peer and this information will remain for the lifetime of the connection. If the capabilities of a peer change, that peer shall drop the connection so that a new Hello Option can be sent upon reconnection.

AB.2.3.1.3 Client Identity Header Option

The 'Client Identity' header option specifies the identity information for the sender of a message.

When passed to or from the network layer, a 'Client Identity' option is conveyed as data attribute with the encapsulated NPDU. It is designated as a "every segment" attribute. See Clause 17.3.2 for its definition and usage.

The 'Client Identity' header option consists of the following fields.

Header Marker	1-octet	'More Options' = 0 or 1, 'Every Segment' = 1, 'Header Data Flag' = 1, 'Header Option Type' = x3
Header Length	2-octets	Length of 'Header Data' field, in octets.
Header Data	7-octets	Required to include:
Device	3-octets	Device instance number, with most significant octet first

This header option, if present, shall be a data option in the 'Data Options' parameter of BVLC messages conveying an NPDU.

AB.2.3.1.4 Hint Header Option

The 'Hint' header option provides information about what authorization is required for a failed operation. See Clause 17.4.8 for meaning and usage.

When passed to or from the network layer, a 'Hint' option is conveyed as data attribute with the encapsulated NPDU. It is designated as a "first segment" attribute.

A 'Hint' header option consists of the following fields.

Header Marker	1-octet	'More Options' = 0 or 1, 'Every Segment' = 0, 'Header Data Flag' = 1, 'Header Option Type' = x4
Header Length	2-octets	Length of 'Header Data' field, in octets.
Header Data	3-N octets	Required to include:
Scope	Variable	The required scope represented as the ASN.1 production of a BACnetAuthorizationScope

This header option, if present, shall be a data option in the 'Data Options' parameter of BVLC messages conveying an NPDU.

AB.2.3.1.5 Token Header Option

The 'Token' header option conveys an access token to authorize a protected operation. See Clause 17.4.8 for meaning and usage.

When passed to or from the network layer, a 'Token' option is conveyed as data attribute with an encapsulated NPDU. It is designated as a "first segment" attribute.

A 'Token' header option consists of the following fields.

Header Marker	1-octet	'More Options' = 0 or 1, 'Every Segment' = 0, ' Header Data Flag' = 1, 'Header Option Type' = x5
Header Length	2-octets	Length of 'Header Data' field, in octets.
Header Data	N octets	Required to include:
Token	Variable	The ASN.1 production of a BACnetAccessToken

This header option, if present, shall be a data option in the 'Data Options' parameter of BVLC messages conveying an NPDU.

[Change Clause AB.7.4, p. 1398]

AB.7.4 Connection Security

The use of secure WebSocket connections as of RFC 6455 and TLS V1.3 as of RFC 8446 for BACnet/SC connections provides for confidentiality, integrity, and authenticity of BVLC messages transmitted across the connection.

The establishment of a secure WebSocket connection shall be performed as defined in RFC 6455. For establishing a secure WebSocket connection, mutual TLS authentication shall be performed. "Mutual authentication" in this context means that both the initiating peer and the accepting peer shall:

(a) Validate that the peer's operational certificate is well formed.
(b) Validate that the peer's operational certificate is active as of the current date and not expired.
(c) Validate that the peer's operational certificate is not revoked, if such information is available.
(d) Validate that the peer's operational certificate is directly signed by one of the locally configured CA certificates.

To ensure interoperability, no additional checks beyond the above shall be performed by default, and none are required to be supported. Any additional checks, e.g., Common Name, Distinguished Name, or Subject Alternate Names matches, shall only be performed if specifically enabled, as directed by the installation. The support and update of revocation information is a local matter.

Note that Clause 17 uses an optional Subject Alternative Name in the certificate for authentication of the device identity for authorization policies. This is not used as a connection criteria at the datalink layer. If the above criteria are satisfied, the BACnet/SC connection shall succeed regardless of the presence or content of any Subject Alternative Name fields in the certificate.

135-2020*cp*-3 Device Object Properties to support Authentication and Authorization

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 defines new properties for the Device object for configuring and monitoring the behavior of authorization clients and servers.

[Change Table 12-13, p. 222]

…	…	…
Authorization_Server	*BACnetAuthorizationServer*	*O*[s]
Authorization_Status	*BACnetAuthorizationStatus*	*O*[s]
Authorization_Policy	*BACnetARRAY of BACnetAuthorizationPolicy*	*O*
Authorization_Groups	*BACnetARRAY of Unsigned*	*O*
Authorization_Cache	*BACnetLIST of BACnetAccessToken*	*O*[t]
Authorization_Scopes	*BACnetARRAY of BACnetAuthorizationScopeDescription*	*O*
…	…	…

[s] *required for secure devices that perform server-side authorization*
[t] *required for secure devices that contain client tokens that are configured by a nonstandard mechanism*

[Add new clauses 12.11.X1-X6, p. 231]

12.11.X1 Authorization_Server

This property, of type BACnetAuthorizationServer, contains information about the authorization server that controls the authorization policies for this device.

The ‘auth-server’ field indicates the device instance of the authorization server for this device. Authorization clients can use this to know where to request an access token for this device using the AuthRequest service.

The ‘signing-key-1’ and ‘signing-key-2’ fields contain the public keys that can be used to validate the signature in access tokens. Only one key is required for operation and either field can be used for a single key. The ‘key-id’ field in the token selects which key is to be used. This allows for key rotation, if desired. For example, if the authorization server wishes to rotate keys, one of these fields can be updated with the future key, then the authorization begins using that key, and then the previous key can be removed.

The public keys shall be formatted as a Subject Public Key Info structure as defined by X.509 in Section 4.1.2.7 of RFC 5280, in binary DER format.

12.11.X2 Authorization_Status

This read-only property, of type BACnetAuthorizationStatus provides information about this device’s current security posture and recent authentication and authorization activity.

The ‘posture’ field indicates the current security posture of the device. The enumerated values have the following meaning:

Enumeration Value	Meaning
open	no authorization is in effect
proprietary	proprietary authorization is in effect (temporary situation)
configured	everything looks good and is properly secured
misconfigured-partial	something is wrong, but things are mostly working
misconfigured-total	nothing works, may need reset to factory defaults

The ’error’ field is used when a known error condition can be indicated for a ‘misconfigured’ posture. If the error is known to be caused by a particular property of an object, then that shall be indicated in the ‘error-source’ field. If there are human readable details of the error condition available, they shall be indicated in the ‘error-details’ field. If no error condition is in effect, or the reason is unknown, these fields shall be absent.

There are four variably sized lists of events that can be us used for diagnostics. The intent of the four lists is to be helpful to humans. The selection of which events to include, if any, is a local matter. When present, the events shall be in timestamp order, oldest record first.

Authentication events shall consist of the following fields:

Field Name	Data Type	R/O	Description
timestamp	BACnetDateTime	R	When the event occurred
peer	BACnetAuthenticationPeer	R	The peer’s address, identity, and capabilities
client	BACnetAuthenticationClient	R	From the "Client Identity" data attribute
decision	BACnetAuthenticationDecision	R	allow or deny reason
decision-details	CharacterString	O	human readable details for the decision

Authorization events shall consist of the following fields:

Field Name	Data Type	R/O	Description
timestamp	BACnetDateTime	R	When the event occurred
address	BACnetAddress	R	SNET/SADR of originator
client	BACnetAuthenticationClient	O	From the "Client Identity" data attribute, if provided
token	BACnetAccessToken	O	the access token, if provided
decision	BACnetAuthorizationDecision	R	allow or deny reason
decision-details	CharacterString	O	human readable details for the decision

The effective size of this property shall be dynamically adjusted based on the capabilities of the device reading this property. For example, if the reading device does not support segmentation, then the event lists shall be limited to that which can be conveyed in the response without segmentation. The ‘token’ field, while useful for diagnostics, can be omitted from BACnetAuthorizationEvent to reduce size.

12.11.X3 Authorization_Policy

This property, of type BACnetARRAY of BACnetAuthorizationPolicy, is a collection of distributed policies containing the same information that would be delivered by access tokens across the wire. If no token is provided by a client device, the target device can consult this distributed policy list to see what authorization has been configured for the client device.

The distributed policy capability supports older client devices by effectively pre-sending authorization information to the target device, to be used when the client doesn't/can't send policy information on its own with an access token. This supports non-secure clients, secure clients that pass through non-secure networks, and original BACnet/SC clients that are "Secure Path" capable but don't use access tokens. This is useful for target devices that want to protect themselves with standard mechanisms but still want to be able to interact with earlier clients.

Each entry in the array contains the following information:

Field	Type	R/O
not-before	BACnetDateTime	O
not-after	BACnetDateTime	O
clients	Sequence of Unsigned32	R
constraint	BACnetAuthorizationConstraint	R
scope	BACnetAuthorizationScope	R

The 'not-before' and 'not-after' define the active period when the policy is active.

The 'clients' field controls which client devices this policy applies to. An empty list means that this policy applies to any client that meets the requirements of the 'constraint' field.

The 'constraint' field limits the client's network location and authentication method. See Clause 17.

The 'scope' field specifies which operations are authorized for the client at the target devices, See Clause 17.4.3.

This property can be used in non-secure devices, with the restriction that the 'constraint' field is limited to have an 'authentication' of "Any Method", and an 'origin' of either "Same Network" or "Any Network". An attempt to write any other values shall result in a Result(-) to be returned with an 'Error Class' of PROPERTY and an 'Error Code' of VALUE_OUT_OF_RANGE..

12.11.X4 Authorization_Scopes

This read-only property, of type BACnetARRAY of BACnetAuthorizationScopeDescription, is a way of describing the non-standard scopes that are in use in this device. It shall be present in devices that require the use of non-standard scopes. It shall not be present in devices that only use standard scopes. If present, it shall not contain descriptions for standard scopes.

Its presence allows an authorization server to read this information and populate its user interface with meaningful scope descriptions for the user to pick from.

The 'name' field is a short string restricted to the format that can be used as an OAuth "scope-token". See RFC 6749 Section A.4. The intent of the 'name' field is for interoperation with OAuth; therefore, the names should be meaningful to the extent possible.

The 'description' field provides the human-readable description of the scope identifier. The description should be sufficiently detailed to provide guidance to the user as to its meaning. However, considering that it is expected to be used in user interfaces, possibly in tables or drop-down choices, it shall not contain newline characters and is not intended to be a lengthy description.

12.11.X5 Authorization_Cache

This read-only property, of type BACnetLIST of BACnetAccessToken, is a list of access tokens that are in use by the client functions within this device. The tokens in this list have either been acquired by this device with the AuthRequest service or have been configured into this device by a nonstandard mechanism.

12.11.X5 Authorization_Groups

This property, of type BACnetARRAY of Unsigned, indicates the list of group numbers that this device is a member of. The 'audience' field of an access token can specify one of these group numbers as an alias for this device.

135-2020*cp*-4 Data Structures to support Authentication and Authorization

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 defines new data structures used for configuring, monitoring, and conveying authentication and authorization information.

[Add the following new Application Type to Clause 21.5, p 871]

21.5 Application Types

...
Integer32 *::=* *INTEGER (-2147483648 to 2147483647)*
...

[Add the following productions to Clause 21.6, p. 882, in alphabetical order]

```
BACnetAuthorizationPolicy ::= SEQUENCE {
        not-before          [0] BACnetDateTime OPTIONAL,          -- local time
        not-after           [1] BACnetDateTime OPTIONAL,          -- local time
        clients             [2] SEQUENCE OF Unsigned32,           -- "who" this policy applies to
        constraint          [3] BACnetAuthorizationConstraint,    -- where the client is and how it is identified
        scope               [4] BACnetAuthorizationScope          -- the granted permissions
        }

BACnetAuthorizationConstraint ::= SEQUENCE {
        origin ::= ENUMERATED {
                direct-connect      (0),
                same-network        (1),
                any-network         (2)
                }
        authentication ::= ENUMERATED {
                certified           (0),
                secure-path         (1),
                any-method          (2)
                }
        }



BACnetAuthorizationScope ::= SEQUENCE {
        standard            BIT STRING {  -- see standard scopes in Clause 17.4.3
                view                (0),      -- for viewing confidential information, not needed for public information
                adjust              (1),
                control             (2),
                override            (3),
                config              (4),
                bind                (5),
                install             (6),
                auth                (7),
                infrastructure      (8),
                reserved-9          (9),
                reserved-10         (10),
                reserved-11         (11),
```



```
                reserved-12     (12),
                reserved-13     (13),
                reserved-14     (14),
                reserved-15     (15),
                reserved-16     (16),
                reserved-17     (17),
                reserved-18     (18),
                reserved-19     (19),
                reserved-20     (20),
                reserved-21     (21),
                reserved-22     (22),
                reserved-23     (23)
                }
        extended        [0] SEQUENCE OF CharacterString OPTIONAL
        }

BACnetAccessToken ::= SEQUENCE {
        issuer          [0] Unsigned,                       -- device instance of authorization server issuing the token
        issued          [1] BACnetDateTime,                 -- when the token was issued
        audience        [2] SEQUENCE OF Integer32,          -- negative numbers are groups; group 1 is "all",
        not-before      [3] BACnetDateTime OPTIONAL,        -- local time
        not-after       [4] BACnetDateTime OPTIONAL,        -- local time
        client          [5] Unsigned32,                     -- "who" this policy is bound to
        constraint      [6] BACnetAuthorizationConstraint,  -- where the client is and how it is identified
        scope           [7] BACnetAuthorizationScope        -- the granted permissions
        key-id          [8] Unsigned                        -- 1=signing-key-1, 2=signing-key-2
        signature       [9] OCTET STRING                    -- digital signature of all the preceding ASN encoded octets
        }

BACnetAuthorizationServer ::= SEQUENCE {
        auth-server     [0] Unsigned,               -- instance of authorization server for this device, 4194303=unconfigured,
        signing-key-1   [1] OCTET STRING OPTIONAL,  -- SubjectPublicKeyInfo in binary DER, absent=unconfigured
        signing-key-2   [2] OCTET STRING OPTIONAL,  -- SubjectPublicKeyInfo in binary DER, absent=unconfigured
        }

BACnetAuthorizationScopeDescription ::= {
        name            CharacterString,            -- usable as OAuth/JWT scope token
        description     CharacterString
        }

BACnetAuthorizationStatus ::= SEQUENCE {
        posture                 [0] BACnetAuthorizationPosture,
        error                   [1] Error OPTIONAL,
        error-source            [2] BACnetObjectPropertyReference OPTIONAL,
        error-details           [3] CharacterString OPTIONAL,
        authentication-success  [4] SEQUENCE OF BACnetAuthenticationEvent OPTIONAL,
        authentication-failure  [5] SEQUENCE OF BACnetAuthenticationEvent OPTIONAL,
        authorization-success   [6] SEQUENCE OF BACnetAuthorizationEvent OPTIONAL,
        authorization-failure   [7] SEQUENCE OF BACnetAuthorizationEvent OPTIONAL
        }

BACnetAuthorizationPosture ::= ENUMERATED {
        open                    (0),
        proprietary             (1),
        configured              (2),
        misconfigured-partial   (3),
```



```
        misconfigured-total         (4)
        }

BACnetAuthenticationEvent ::= SEQUENCE {
        timestamp                   [0] BACnetDateTime,
        peer                        [1] BACnetAuthenticationPeer,
        client                      [2] BACnetAuthenticationClient,
        decision                    [3] BACnetAuthenticationDecision,
        decision-details            [4] CharacterString OPTIONAL
        }

BACnetAuthenticationPeer ::= SEQUENCE {
        host            BACnetHostNPort,
        device          Unsigned32,  -- 4194303 if unknown
        auth-aware      Boolean,
        router          Boolean,
        hub             Boolean
        }

BACnetAuthenticationClient ::= SEQUENCE {
        authenticated   Boolean,
        device          Unsigned32,
        }

BACnetAuthenticationDecision ::= ENUMERATED {
        allow-match     (0),
        deny-mismatch   (1),
        deny-non-router (2)
        }

BACnetAuthorizationEvent ::= SEQUENCE {
        timestamp       [0] BACnetDateTime,
        address         [1] BACnetAddress,  -- SNET and SMAC
        client          [2] BACnetAuthenticationClient OPTIONAL,
        token           [3] BACnetAccessToken OPTIONAL,
        decision        [4] BACnetAuthorizationDecision
        decision-details [5] CharacterString OPTIONAL
        }

BACnetAuthorizationDecision ::= ENUMERATED {
        allow-by-token          (0),
        allow-by-local-policy   (1),
        deny-no-token-or-policy (2),
        deny-not-before         (3),
        deny-not-after          (4),
        deny-target-device      (5),
        deny-target-group       (6),
        deny-client-device      (7),
        deny-client-method      (8),
        deny-scope              (9),
        deny-issuer             (10),
        deny-stale              (11),
        deny-revoked            (12),
        deny-signature          (13),
        deny-other              (14)
        }
```


[Change Clause 21.6, p. 882]

21.6 Base Types

...

BACnetServicesSupported ::= BIT STRING {

...

-- Remote Device Management Services

device-communication-control	(17),
confirmed-private-transfer	(18),
confirmed-text-message	(19),
reinitialize-device	(20),
-- who-Am-I	(47),
-- you-Are	(48),

-- Security Services

-- auth-request	(x),

...

-- Services added after 2016

confirmed-audit-notification	(44), -- Alarm and Event Service
audit-log-query	(45), -- Object Access Service
unconfirmed-audit-notification	(46), -- Alarm and Event Service
who-Am-I	(47), -- Remote Device Management Service
~~you-Are~~	~~(48) -- Remote Device Management Service~~
you-Are	*(48), -- Remote Device Management Service*

[note to reviewer: the above change just adds a comma to the above line]

-- *Services added after 2020*

auth-request	*(x)*

...

}

...

[Change Clause 21.2, p. 858]

BACnetConfirmedServiceChoice ::= ENUMERATED {

...

-- Remote Device Management Services

device-communication-control	(17),
confirmed-private-transfer	(18),
confirmed-text-message	(19),
reinitialize-device	(20),

-- *Security Services*

auth-request	*(x),*

...

-- Services added after 2016

-- confirmed-audit-notification	(32) see Alarm and Event Services
-- audit-log-query	(33) see Object Access Services

-- *Services added after 2020*


```
        -- auth-request                 (x)  see Security Services
...

BACnet-Confirmed-Service-Request ::= CHOICE {
...
-- Remote Device Management Services
        device-communication-control    [17]  DeviceCommunicationControl-Request,
        confirmed-private-transfer      [18]  ConfirmedPrivateTransfer-Request,
        confirmed-text-message          [19]  ConfirmedTextMessage-Request,
        reinitialize-device             [20]  ReinitializeDevice-Request,

-- Security Services
        auth-request                    [x] AuthRequest-Request,
...
-- Services added after 2016
        -- confirmed-audit-notifications [32]  see Alarm and Event Services
        -- audit-log-query              [33]  see Object Access Services
        }

-- Services added after 2020
        -- auth-request                 [x] see Security Services
 ...

BACnet-Confirmed-Service-ACK  ::= CHOICE {
...

-- Remote Device Management Services
        confirmed-private-transfer      [18]  ConfirmedPrivateTransfer-ACK,

-- Security Services
        auth-request                    [x]    AuthRequest-ACK,
...
```

[Add new Clause 21.2.5, p. 866]

21.2.X Authentication and Authorization Services

```
AuthRequest-Request ::= CHOICE {            -- CHOICE of the "sub service" allows future extensibility
        token-request       [0] SEQUENCE {
                client              [0] Unsigned,                       -- client device instance to bind the token to
                audience            [1] SEQUENCE OF Integer32,          -- target device(s) and/or group(s).
                scope               [2] BACnetAuthorizationScope,       -- requested scope
                }
        }

AuthRequest-ACK ::= CHOICE
        token-response     [0] BACnetAccessToken
        }
```

[Change Clause 21.4, p 872]

```
...
BACnet-Error ::= CHOICE {
...
-- Remote Device Management Services
```



```
        device-communication-control    [17]  Error,
        confirmed-private-transfer      [18]  ConfirmedPrivateTransfer-Error,
        confirmed-text-message          [19]  Error,
        reinitialize-device             [20]  Error,

-- Security Services
        auth-request                    [x]   AuthRequest-Error,
...
-- Services added after 2016
        -- confirmed-audit-notification [32]  see Alarm and Event Services
        -- audit-log-query              [33]  see Object Access Services
        }

-- Services added after 2020
        -- auth-request                 [x]   see Security Services
...
        }
```

[Add new codes to the Error production, interspersing new items in alphabetical order]

```
Error ::= SEQUENCE {
        ...
        error-code          ENUMERATED { -- see below for numerical order
                              ...
                              adjust-scope-required                (n),
                              ...
                              auth-scope-required                  (n+1),
                              ...
                              bind-scope-required                  (n+2),
                              ...
                              config-scope-required                (n+3),
                              ...
                              control-scope-required               (n+4),
                              ...
                              extended-scope-required              (n+5),
                              ...
                              incorrect-client                     (n+6),
                              ...
                              install-scope-required               (n+7),
                              ...
                              insufficient-scope                   (n+8),
                              ...
                              no-default-scope                     (n+9),
                              ...
                              no-policy                            (n+10),
                              ...
                              override-scope-required              (n+11),
                              ...
                              revoked-token                        (n+12),
                              ...
                              unknown-audience                     (n+13),
                              ...
                              unknown-client                       (n+14),
                              ...
                              unknown-scope                        (n+15),
                              ...
```



```
                        view-scope-required             (n+16),
                        ...
-- numerical order reference
                        -- see other                    (0),
                        ...
                        -- see adjust-scope-required    (n)
                        -- see auth-scope-required      (n+1)
                        -- see bind-scope-required      (n+2)
                        -- see config-scope-required    (n+3)
                        -- see control-scope-required   (n+4)
                        -- see extended-scope-required  (n+5)
                        -- see incorrect-client         (n+6)
                        -- see install-scope-required   (n+7)
                        -- see insufficient-scope       (n+8)
                        -- see no-default-scope         (n+9)
                        -- see no-policy                (n+10)
                        -- see override-scope-required  (n+11)
                        -- see unknown-audience         (n+12)
                        -- see unknown-client           (n+13)
                        -- see unknown-scope            (n+14)
                        -- see view-scope-required      (n+15)
                    }
        ...
        }

...
WritePropertyMultiple-Error ::= SEQUENCE {
        error-type                      [0]  Error,
        first-failed-write-attempt      [1]  BACnetObjectPropertyReference
        }

AuthRequest-Error ::= SEQUENCE {
        error           Error,
        error-details   CharacterString OPTIONAL
        }
...
```

135-2020*cp*-5 Error Codes to support Authentication and Authorization

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 defines new error situations that need new error code definitions.

[Change Clause 18.5, p. 793]

18.5 Error Class - SECURITY

...

BAD_SIGNATURE - The signature in a secure message is incorrect. ~~(Reserved for future use)~~

...

SOURCE_SECURITY_REQUIRED - The operation requested requires that the source secure or encrypt the request. ~~(Reserved for future use)~~

...

UNKNOWN_AUTHENTICATION_TYPE - The authentication method in a secure message is unknown to the receiving device. ~~(Reserved for future use)~~

[Add new error codes to Clause 18.5, Error Class - SECURITY, p. 793, in alphabetical order]

INCORRECT_AUDIENCE - The audience specified in an access token does not match the recipient.

INCORRECT_CLIENT - The client specified in an access token does not match the sender.

INSUFFICIENT_SCOPE - The protected resource requires an authorization scope that was not provided.

REVOKED_TOKEN - The presented access token has been revoked.

VIEW_SCOPE_REQUIRED - The protected resource requires the ‘view’ scope.

ADJUST_SCOPE_REQUIRED - The protected resource requires the ‘adjust’ scope.

CONTROL_SCOPE_REQUIRED - The protected resource requires the ‘control’ scope.

OVERRIDE_SCOPE_REQUIRED - The protected resource requires the ‘override’ scope.

CONFIG_SCOPE_REQUIRED - The protected resource requires the ‘config’ scope.

BIND_SCOPE_REQUIRED - The protected resource requires the ‘bind’ scope.

INSTALL_SCOPE_REQUIRED - The protected resource requires the ‘install’ scope.

AUTH_SCOPE_REQUIRED - The protected resource requires the ‘auth’ scope.

EXTENDED_SCOPE_REQUIRED - The protected resource requires a non-standard scope.

[Add new error codes to Clause 18.6, Error Class - SERVICES, p. 795, in alphabetical order]

UNKNOWN_AUDIENCE - There is no policy defined for the given audience.

UNKNOWN_CLIENT - There is no policy defined for the given client.

UNKNOWN_SCOPE - The requested scope is not recognized for the target device.

NO_DEFAULT_SCOPE - There is no default scope for the requested client and audience.

NO_POLICY - There is no policy defined for the requested operation.

135-2020*cp*-6 PICS statements to support Authentication and Authorization capabilities

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 defines new capabilities for devices that need to be indicated on a PICS statement.

[Add new section to Annex A, p.]

ANNEX A - PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT (NORMATIVE)

...

Authentication Options:

If this product is an authorization server, describe the methods by which administrators are notified of denied authorization requests and the options for filtering, pacing, and silencing such notifications:

_____

If this product is an authorization client,

- *Can it use the AuthRequest service to obtain its own tokens dynamically? ____*
- *Can it be statically configured with access tokens? ____ If so, how many? _____*

If this product is an authorization target, how many distributed polices can be configured? ___

135-2020*cp*-7 New definitions for Authentication and Authorization

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 uses new abbreviations.

[Add new abbreviations to Clause 3.3, p. 17]

3.3 Abbreviations and Acronyms Used in this Standard

...
ACL *Access Control List*
...
VPN *Virtual Private Network*
...
VLAN *Virtual Local Area Network*
...

135-2020*cp*-8 New BIBBs and Profiles for Authentication and Authorization

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 describes new services and behaviors that need to be described with BIBBs and Profiles.

[Add new Clause K.X, p. 1117]

K.X Authentication and Authorization BIBBs

K.X.1 BIBB - Dynamic Authorization Client - A (AA-DAC-A)

The A device is an authorization client that is capable of getting its own access tokens from an authorization server. When a protected operation fails with an authorization error code like CONFIG_SCOPE_REQUIRED, a device claiming conformance to AA-DAC-A will attempt to retrieve an access token for the required scope from the authorization server configured into its Authorization_Server property. See Clause 17.

BACnet Service	Initiate	Execute
AuthRequest	x	

Devices claiming conformance to this BIBB are interoperable with devices claiming conformance to AA-AS-B and AA-AT-B.

K.X.2 BIBB - Static Authorization Client - A (AA-SAC-A)

The A device is an authorization client that accepts configuration of access tokens by a helper tool. The manner of such configuration is a local matter. The capacity of such configuration shall be claimed on the PICS. The device shall support a minimum capacity to hold two access tokens. The device shall support the Authorization_Cache property with a minimum capacity of two tokens. However, it is recommended that the Authorization_Cache have the capacity to indicate the same number of tokens that can be configured. See Clause 17.

Devices claiming conformance to this BIBB are interoperable with devices claiming conformance to AA-AT-B.

K.X.3 BIBB - Authorization Target - B (AA-AT-B)

The B device is a target device that has protected operations that require authorization. The B device has an Authorization_Server property that supports both of the signing keys and a group membership list with at least two members. The Authorization_Policy property shall be present and support at least four distributed policies. Support for all optional policy fields is required. See Clause 17.

The B device shall process access tokens presented to it by an authorization client.

The B device shall generate appropriate authorization error codes like CONFIG_SCOPE_REQUIRED. If the device supports non-standard scopes, then the device shall also generate "Hint" data attribute for the error responses and the Authorization_Scopes property shall be present.

Devices claiming conformance to this BIBB are interoperable with devices claiming conformance to AA-DAC-A and AA-SAC-A.

K.X.4 BIBB - Non-secure Authorization Target - B (AA-NAT-B)

The B device is a non-secure device that nonetheless supports authorization policies to create an "allow list" for protected operations to defend against misconfigured devices. The Authorization_Policy property shall be present and support at least four distributed policies. Support for all optional policy fields is required. See Clause 17.

K.X.5 BIBB - Authorization Server - B (AA-AS-B)

The A device is an authorization server that is capable of issuing access tokens to authorization clients or their helpers. See Clause 17.

BACnet Service	Initiate	Execute
AuthRequest		x

Devices claiming conformance to this BIBB are interoperable with devices claiming conformance to AA-DAC-A.

[Add new Clause L.X, p. 1143]

L.X Authentication and Authorization Profiles

The following table indicates which BIBBs shall be supported by the device types of this family, for each interoperability area.

Data Sharing

B-AS
DS-RP-B

Alarm & Event Management

B-AS

Scheduling

B-AS

Trending

B-AS

Device & Network Management

B-AS
DM-DDB-B
DM-DOB-B
DM-DCC-B

Authentication & Authorization

B-AS
AA-AS-B

L.X.1 BACnet Authorization Server (B-AS)

The B-AS is an authorization server that issues access tokens to authorization clients or their helpers. The server has a persistent database of access policies configured to meet the installation's needs. The server has a User Interface suitable for configuring the authorization policies for the tokens that it issues. Support for all optional policy fields is required. See Clause 17.

135-2020*cp*-9 Examples for Authentication and Authorization

Rationale

The BACnet Authentication and Authorization framework defined in Clause 17 describes new services and behaviors. Examples are created to provide clarity to the various interactions' patterns and scenarios.

[Add new Annex XX, p. 14xx]

ANNEX XX - EXAMPLES OF AUTHENTICATION AND AUTHORIZATION (INFORMATIVE)
(This annex is not part of this standard but is included for informative purposes only.)

These are examples of the various flows of data used in authentication and authorization. In many cases, only the interesting items are shown for clarity. For example, "Secure Path" option is assumed. The absence of any required items should not be considered significant.

The following abbreviations are used in the examples:

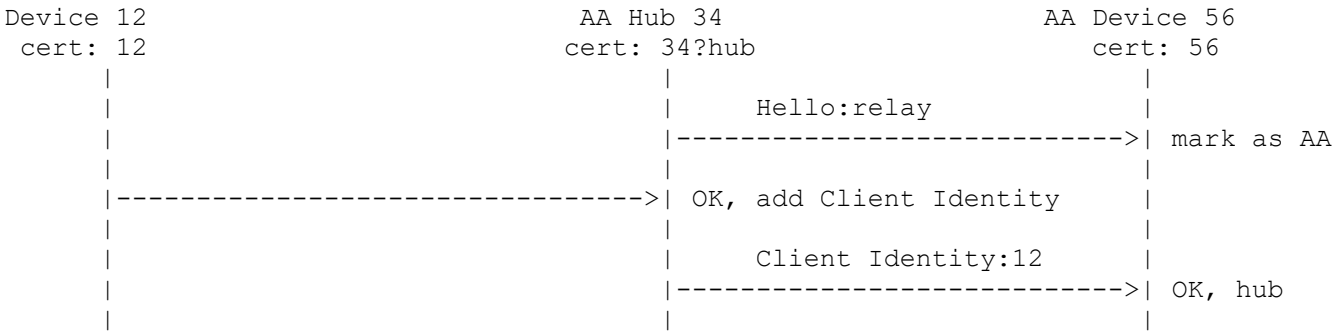
AA	an auth-aware device, following the rules in Clause 17.
non-AA	a non-auth-aware device, predating the rules in Clause 17
cert	this shows just the "..." part of the "bacnet://..." Subject Alternative Name URI
n/a	not applicable to the example; may or may not be present and its value is not important

XX.1 Authentication Examples

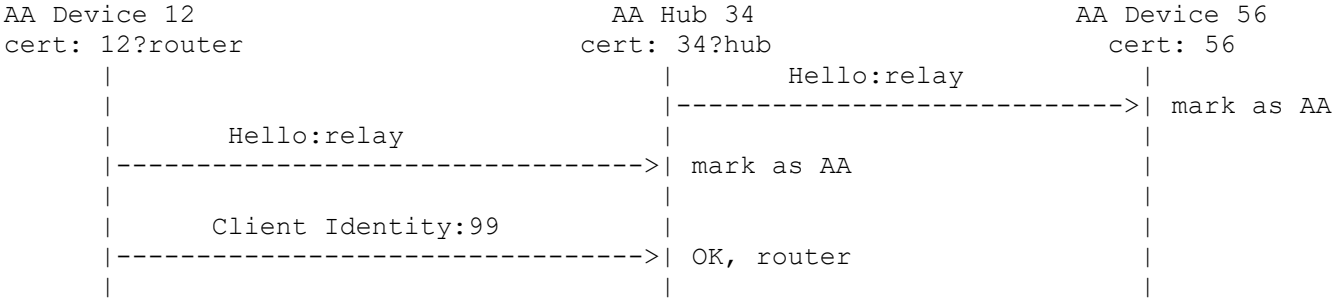
The following examples show the flow of information for authenticating a client to a target, either directly or relayed through authorized intermediaries.

XX.1.1 AA to AA interactions

AA hub receives original message, adds Client Identity
AA destination receives Client Identity from authorized identity relay --> success

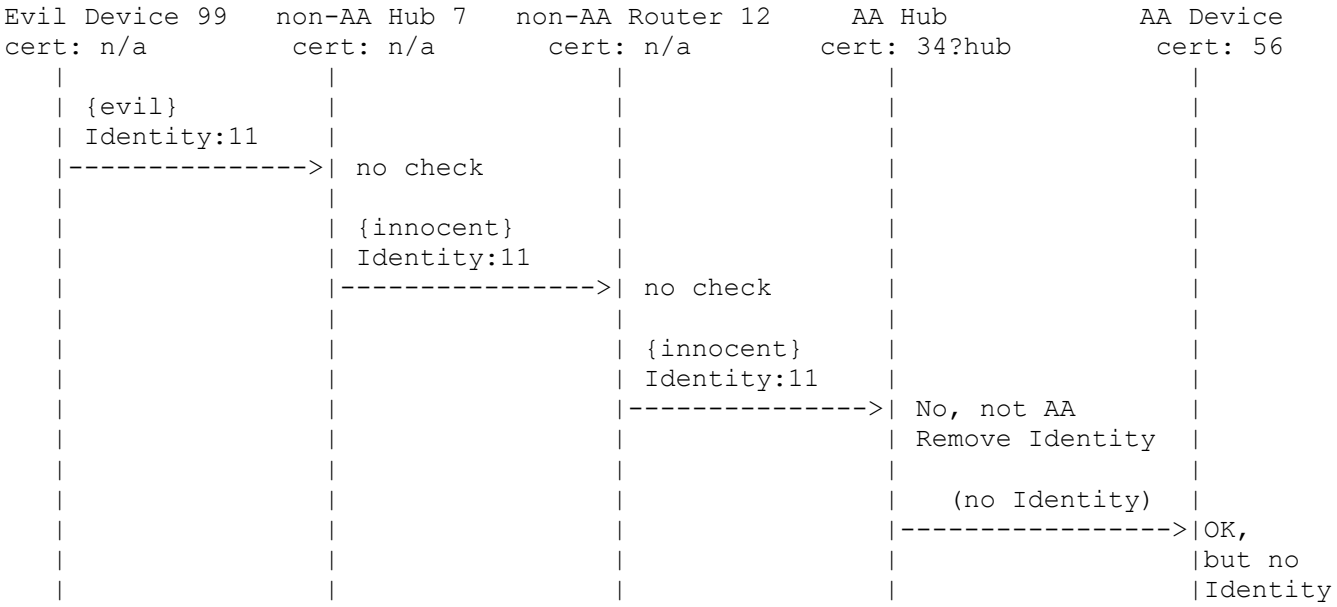


AA hub receives Client Identity from router --> success

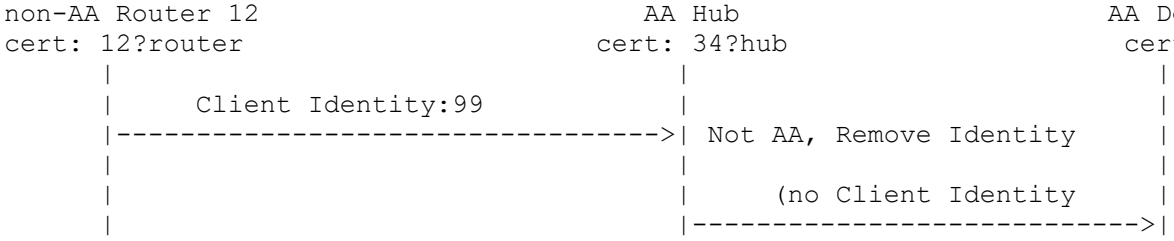


XX.1.2 non-AA to AA interactions

An evil device trying to sneak a Client Identity through a non-AA path will fail, eventually. It is temporarily a lie in transit, but it will be dropped at the first AA peer because there is no innocent way for this to happen. Since AA peers will remove Client Identity received from a non-AA peer, the only way it can be sent by a non-AA peer is a hack upstream.

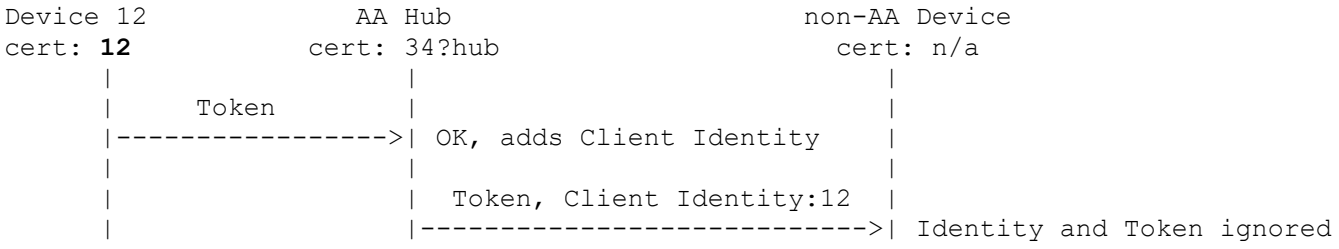


A non-AA peer with a new cert is still a non-AA peer. It can't relay a Client Identity. Just because a non-AA peer has been given a new cert does not make it an authorized identity relay.



XX.1.2 AA to non-AA interactions

A non-AA end device might receive a Client Identity option and/or a Token. This is harmless since the non-AA device does not understand those options and their presence does not grant any extra permissions.



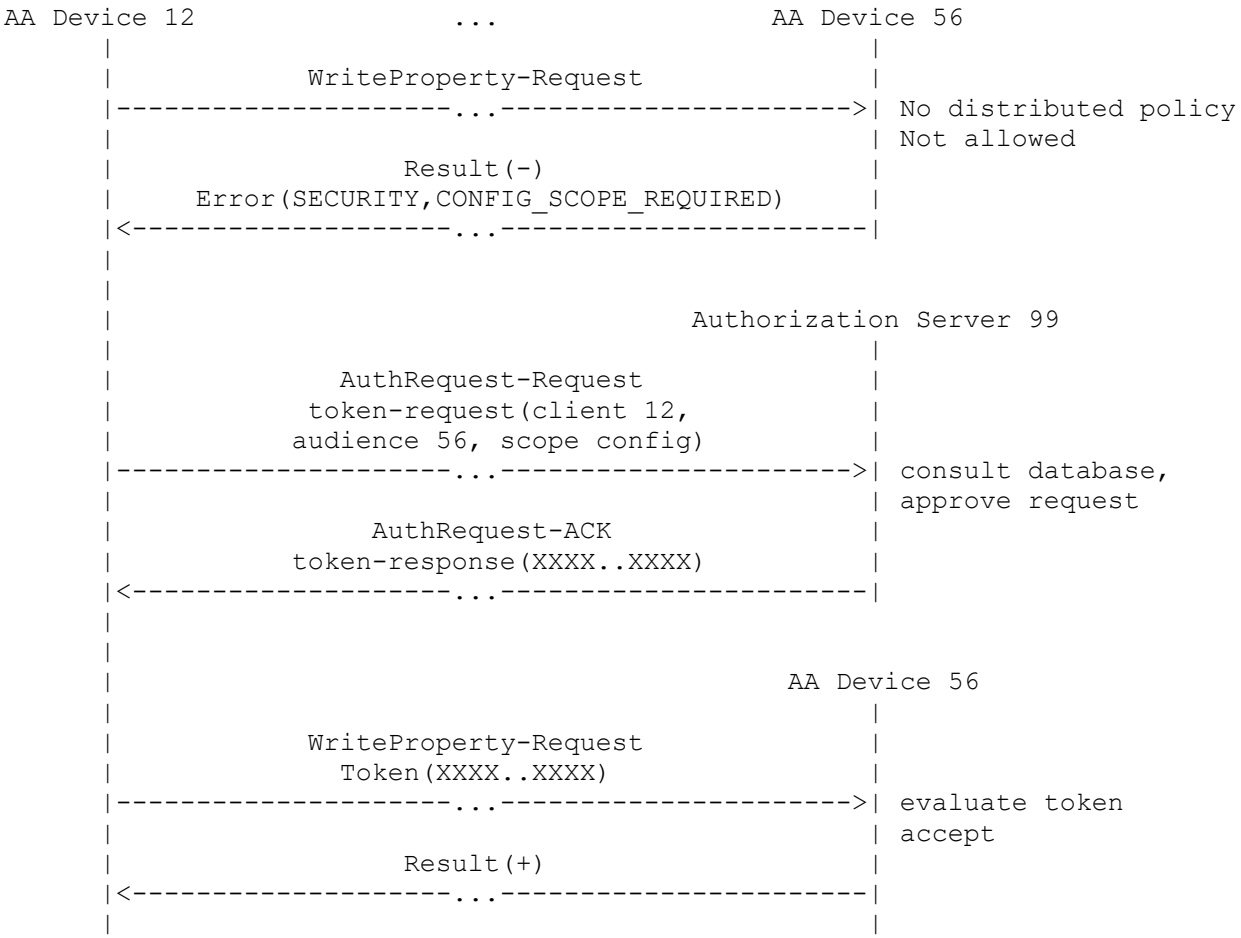
XX.2 Authorization Examples

The following examples show the flow of information for authorizing the operations of an authenticated client by the target device. In these diagrams, an ellipsis in the message arrows indicates that it is not material to the example whether the path is direct or relayed through hubs and routers.

XX.2.1 Typical Token request flow

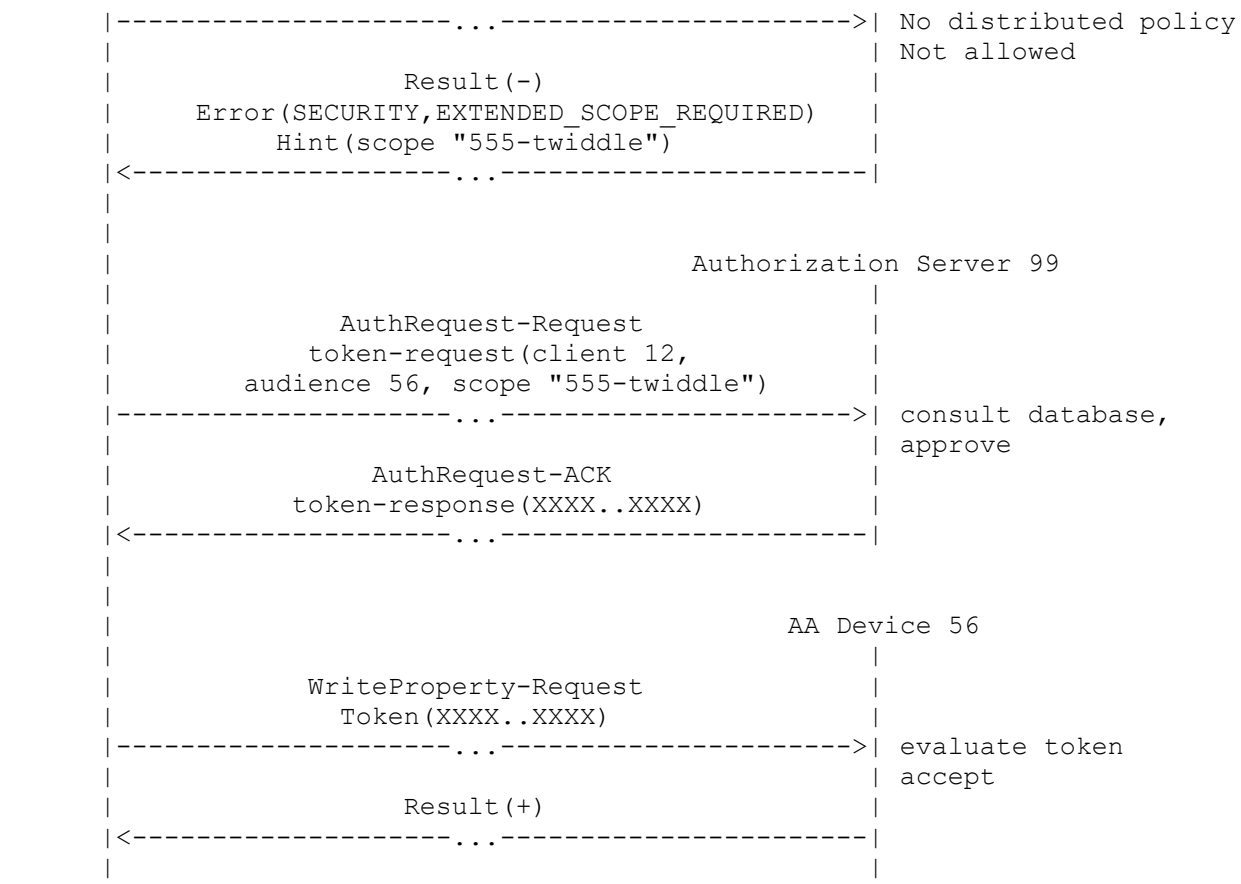
The client device does not have a distributed policy configured at the target. So, when a protected operation is attempted, it is denied. The client uses the error code to know what kind of access token to request for the protected operation. The client requests a token with that scope and is granted a token. The client then retries the protected operation, this time presenting the token, and the operation succeeds.

To simplify these examples, a hub is not shown. They apply to either Hub Connect (where Client Identity would be present) or Direct Connect (where Client Identity would be absent).



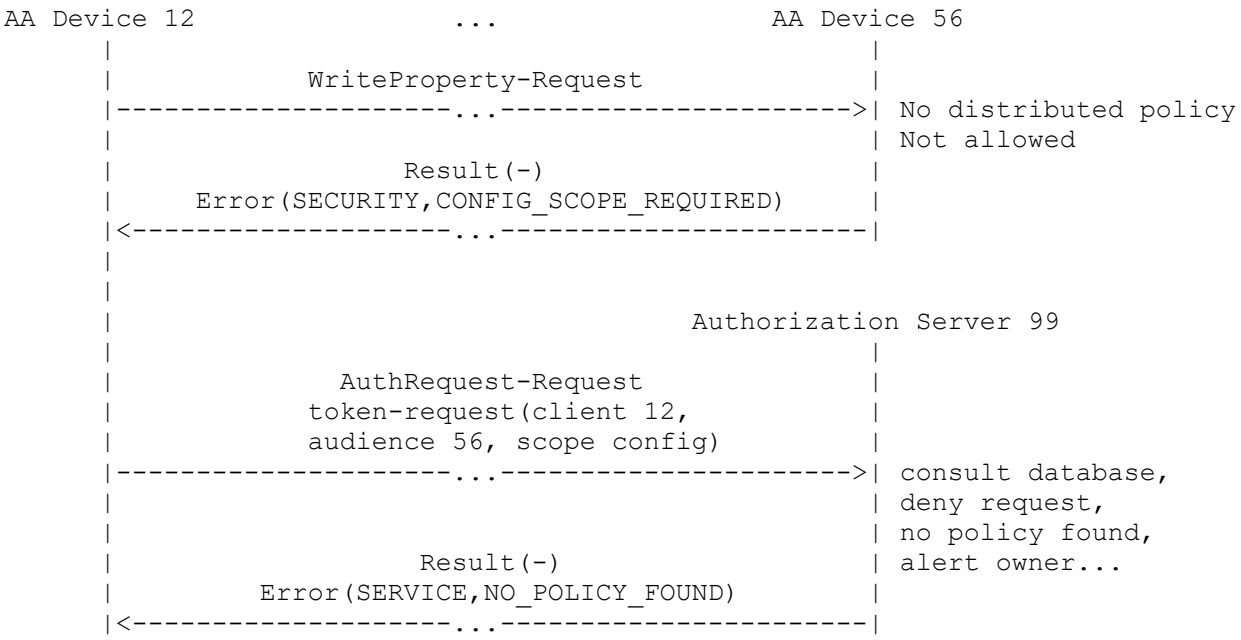
XX.2.1 Extended Scope

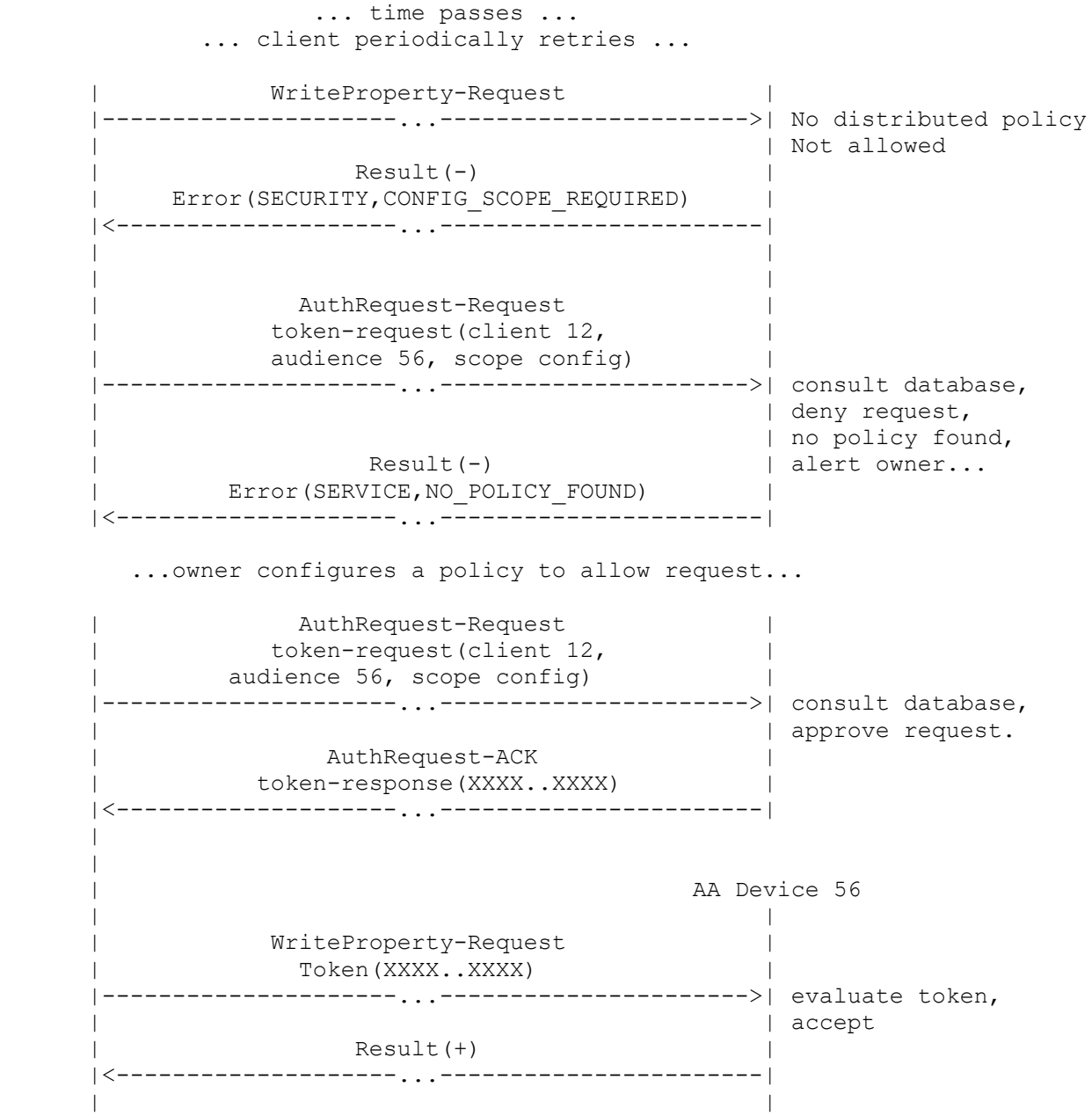
The target device requires an extended (non-standard) scope identifier for a protected operation. It returns that requirement in a Hint data attribute.



XX.2.1 No Policy Exists, Resolved with Authorization Server

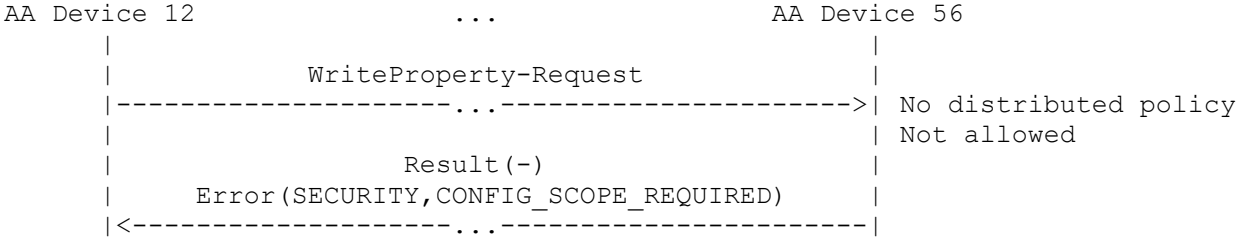
The Authorization server does not have a policy for a requested token, so there is a delay while the owner configures a policy to allow the token to be issued. Client also retries original operation in case a distributed policy has been created.

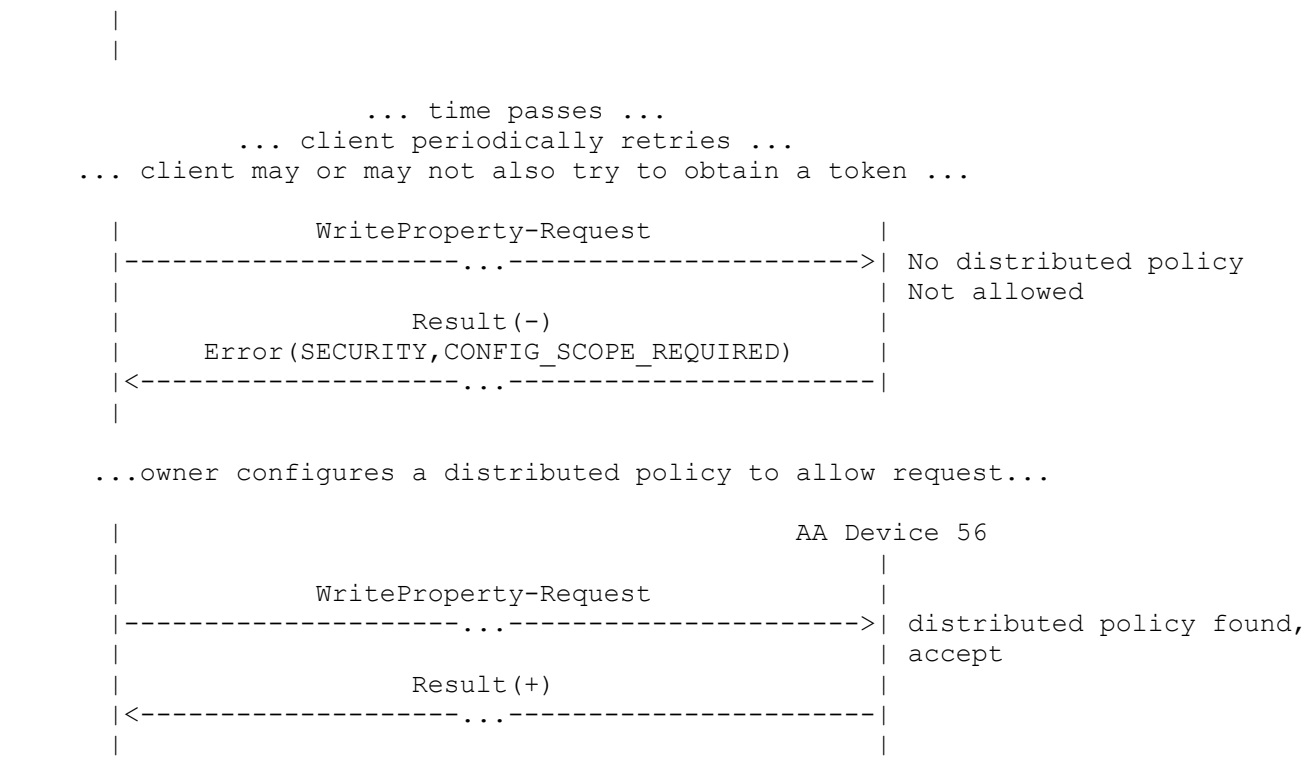




XX.2.1 No Policy Exists, Resolved with Distributed Policy

The target device does not have a policy for a requested operation, so there is a delay while the owner configures a distributed policy in the Authorization_Policy property to allow the operation to succeed.





[Add a new entry to **History of Revisions**, p. 1429]

(This History of Revisions is not part of this standard. It is merely informative and does not contain requirements necessary for conformance to the standard.)

HISTORY OF REVISIONS

. . .	. . .	**. . .**
1	X	**Addendum *xx* to ANSI/ASHRAE Standard 135-2020** Approved by ASHRAE on MONTH DAY, 20XX; and by the American National Standards Institute on MONTH DAY, 20XX. 1. **Addition of Authentication and Authorization** **2. BACnet/SC Changes to Support Authentication and Authorization** **3. Device Object Properties to support Authentication and Authorization** **4. Data Structures to support Authentication and Authorization** **5. Error Codes to support Authentication and Authorization** **6. PICS statements to support Authentication and Authorization capabilities.** **7. New definitions for Authentication and Authorization** **8. New BIBBs and Profiles for Authentication and Authorization** **9. Examples for Authentication and Authorization**

