23 EXTENDING BACnet TO ACCOMMODATE VENDOR PROPRIETARY INFORMATION

The objective of BACnet is to provide the mechanisms by which building automation equipment may exchange information. To aid in interoperability, BACnet defines a standardized set of data structures, called objects, which contain information that is common to most building systems. BACnet may also be used to exchange non-standardized information between devices that understand this information. There are four independent areas where BACnet may be extended to exchange non-standard information:

(a) A vendor may define proprietary extended values for enumerations used in BACnet.
(b) A vendor may invoke a proprietary service using the PrivateTransfer services.
(c) A vendor may add new proprietary properties to a standard object.
(d) A vendor may define new proprietary object types.

In each of these cases, the BACnet messages implicitly reference a vendor identification code that serves to unambiguously specify which vendor's proprietary enumerations, services, properties, or objects were intended. Vendor identification codes are administrated by ASHRAE and are assigned one per vendor. The special Vendor_Identifier of zero is permanently assigned to ASHRAE. The Vendor_Identifier for a given device may be determined by reading the Vendor_Identifier property of the Device object. A list of vendor identification codes may be obtained from the ASHRAE Manager of Standards.

23.1 Extending Enumeration Values

There may be instances when it is necessary for a vendor to extend BACnet by including additional possible values to an enumeration. This is accomplished by using enumeration values that are greater than the range reserved for BACnet for a given enumeration type. Table 23-1 defines those enumerations that may be extended and the range of enumerated values reserved for BACnet use. All other enumerations, which do not appear in Table 23-1, shall not be extended.

Table 23-1. Extensible Enumerations

Enumeration Name	Reserved Range	Maximum Value
error-class	0...63	65535
error-code	0...255	65535
BACnetAbortReason	0...63	255
BACnetAccessAuthenticationFactorDisable	0...63	65535
BACnetAccessCredentialDisable	0...63	65535
BACnetAccessCredentialDisableReason	0...63	65535
BACnetAccessEvent	0...511	65535
BACnetAccessUserType	0...63	65535
BACnetAccessZoneOccupancyState	0...63	65535
BACnetAuthorizationExemption	0...63	255
BACnetAuthorizationMode	0...63	65535
BACnetBinaryLightingPV	0...63	255
BACnetDeviceStatus	0...63	65535
BACnetDoorAlarmState	0...255	65535
BACnetDoorStatus	0...1023	65535
BACnetEngineeringUnits	0...255, 47808...49999	65535
BACnetEscalatorFault	0...1023	65535
BACnetEscalatorMode	0...1023	65535
BACnetEscalatorOperationDirection	0...1023	65535
BACnetEventState	0...63	65535
BACnetEventType	0...63	65535
BACnetLifeSafetyMode	0...255	65535
BACnetLifeSafetyOperation	0...63	65535
BACnetLifeSafetyState	0...255	65535
BACnetLiftCarDirection	0...1023	65535
BACnetLiftCarDriveStatus	0...1023	65535
BACnetLiftCarMode	0...1023	65535
BACnetLiftFault	0...1023	65535
BACnetLightingOperation	0...255	65535
BACnetLightingTransition	0...63	255
BACnetLoggingType	0...63	255

Table 23-1. Extensible Enumerations (*continued*)

Enumeration Name	Reserved Range	Maximum Value
BACnetMaintenance	0...255	65535
BACnetNetworkPortCommand	0...127	255
BACnetNetworkType	0...63	255
BACnetObjectType	0...127	1023
BACnetProgramError	0...63	65535
BACnetPropertyIdentifier	0...511	4194303
BACnetPropertyStates	0...63	254
BACnetRejectReason	0...63	255
BACnetRelationship	0...1023	65535
BACnetReliability	0...63	65535
BACnetRestartReason	0...63	255
BACnetSilencedState	0...63	65535
BACnetVTClass	0...63	65535

23.2 Using the PrivateTransfer Services to Invoke Non-Standardized Services

BACnet defines a set of application layer services that are specifically tailored to integrating building control systems. While this standard prescribes a set of application layer services that is intended to be comprehensive, vendors are free to create additional services. Standard services shall be used when possible.

Vendors may add proprietary services to BACnet using the PrivateTransfer services to invoke them. The service types and arguments are not restricted by BACnet, but they shall be conveyed using the Confirmed or Unconfirmed PrivateTransfer services. The protocol mechanisms used in the handling of these APDUs shall perform as specified in this standard.

The format of proprietary application layer services, invoked using PrivateTransfer, shall follow the encoding rules of this standard.

When using the PrivateTransfer service, it is important to note that segmentation is not permitted for Error APDUs. The implementor shall ensure that the parameters in the Error APDU do not expand to the point where segmentation is required.

23.3 Adding Proprietary Properties to a Standardized Object

BACnet defines a set of standard objects, each with a set of standard properties that can be accessed and manipulated with BACnet services. BACnet allows a vendor to add proprietary properties to extend the capabilities of a standard object. Proprietary properties receive the same support from BACnet services as standard properties and therefore can be accessed and manipulated in a manner identical to standard properties.

Objects may indicate conformance to an object profile by use of the Profile_Name property.

If a proprietary property is to be a commandable property, additional properties that fulfill the role of the standard Priority_Array and Relinquish_Default properties shall be provided for each commandable property. The priority arbitration mechanism described in Clause 19 shall apply.

Vendors may add proprietary properties to a standard object by modifying the object definition within a device. Proprietary properties are enumerated with Property_Identifier values of 512 and above. These property identifiers can be used in any BACnet service that uses a Property_Identifier as a parameter.

Proprietary property identifiers implicitly reference the Vendor_Identifier property of the Device object in the device where they reside. It is entirely possible, and expected, that different vendors will use the same enumeration values to represent completely different properties.

23.4 Adding Proprietary Object Types to BACnet

To accommodate building applications where the defined set of standardized objects is not adequate, BACnet allows a vendor to add proprietary object types. Standard object types shall be used when possible. To enhance extensibility, BACnet provides the same support for proprietary objects as for standard objects.

Objects may indicate conformance to an object profile by use of the Profile_Name property.

23.4.1 Proprietary Object_Type Enumerations

Vendors may add proprietary object types to BACnet by extending the BACnetObjectType enumeration. Proprietary object types are enumerated with Object_Type values of 128 and above. These Object_Type values may be used in any BACnet service that uses an Object_Type as a parameter.

23.4.2 Proprietary Property Datatypes

The properties of vendor proprietary objects may include both standard and proprietary datatypes. Proprietary datatypes may only be constructed from application datatypes defined in Clause 20.2.1.4.

23.4.3 Required Properties in Proprietary Object Types

Non-standard object types shall support the following properties:

Object_Identifier
Object_Name
Object_Type
Property_List

These properties shall be implemented to behave as they would in standard BACnet objects. This means that the Object_Identifier and Object_Name properties shall be unique within the BACnet device that maintains them. The Object_Name string shall be at least one character in length and shall consist of only printable characters. The Property_List property was added in Protocol Revision 14.

23.5 Restrictions on Extending BACnet

The following restrictions to extending BACnet apply:

(a) APDU types 8 through 15 are reserved for future ASHRAE use.

(b) Services may be added only via the Confirmed- and UnconfirmedPrivateTransfer services. That is, the enumerations BACnetConfirmedServiceChoice and BACnetUnconfirmedServiceChoice may not be extended.

24 NETWORK SECURITY

This clause defines a security architecture for BACnet. Network security in BACnet is optional. The intent of this architecture is to provide peer entity, data origin, and operator authentication, as well as data confidentiality and integrity. Other aspects of communications security, such as authorization policies, access control lists, and non-repudiation, are not defined by this standard. Systems that require these functions may add them to BACnet by using the proprietary extensibility features provided for by this architecture, or by some other proprietary means.

24.1 Overview

The BACnet network security architecture provides device authentication, data hiding, and user authentication. This has been accomplished within the constraints that BACnet security should allow for:

(a) Application to all BACnet media types (BACnet/IP, MS/TP, etc.)
(b) Application to all BACnet device types (devices, routers, BBMDs)
(c) Application to all message types (broadcast, unicast, confirmed, and unconfirmed)
(d) Application to all message layers (BVLL, network, and application)
(e) Placing non-security-aware devices, if physically secure, behind a security proxy firewall router
(f) Placing secure devices on non-security-aware networks.

To achieve these network security goals, the BACnet standard is extended with a set of network layer security messages. Other security standards, such as IPsec and Kerberos, were designed to operate only on TCP/IP networks and as such do not meet the above requirements. However, the BACnet security architecture was developed by applying the best security practices of those standards that fit the requirements listed above.

24.1.1 Security Layer

The security functionality is added into the BACnet stack as a set of network layer messages. As such, there is no actual security layer, although the discussion of security processing is easiest to understand if it is conceptually separated into a distinct layer. For this reason the security processing and the related messages are referred to as the security layer although in fact they are part of the network layer.

24.1.2 Shared Keys

The BACnet security model relies on the use of shared secrets called keys. Device and user authentication is achieved through the use of message signatures and shared signature keys. Data hiding is achieved through encryption of the secure payload and shared encryption keys.

In BACnet security, keys are always distributed as key pairs, where one half is the signature key and the other half is the encryption key. There are 6 types of key pairs: General-Network-Access, User-Authenticated, Application-Specific, Installation, Distribution, and Device-Master.

The General-Network-Access key is used for broadcast network layer messages, for encryption tunnels, and by user interface devices that cannot authenticate, or are not trusted to authenticate, a user. All devices shall be given the General-Network-Access key pair to interoperate on a BACnet network. BACnet server devices that receive requests signed with the General-Network-Access key should assume that the User Id and User Role fields included in the message may not have been properly authenticated by the source device and may want to restrict access accordingly.

The User-Authenticated key is distributed to client devices that are trusted to authenticate a user's identity by some means, or to devices that do not contain a user interface (where the user identity to use in BACnet messages is configured into the device and is not based on human interaction). This key is also distributed to BACnet server devices that restrict operations based on the identity of an authenticated user. Servers that receive requests that are signed with the User-Authenticated key can assume that the User Id and User Role fields included in the message has been properly authenticated by the client device, or was configured into a trusted device with no user interface. While the client device may restrict a user's actions based on its authorization policies prior to sending the message, the server device is also free to restrict access based on the received User Id and User Role.

An Application-Specific key may be used to provide security boundaries between application areas, such as access control and HVAC. Application-Specific keys are distributed only to those devices sharing a particular application and can thus be limited to highly secure communication. Devices using Application-Specific keys for highly secure communications should be designed to be able to restrict which services can be executed with lesser keys. For example,

such devices might be configured to disallow time synchronization or network configuration via the General-Network-Access key.

Installation keys are distributed temporarily to small sets of devices, usually the configuration tool of a technician and a set of BACnet devices that require configuration. These keys are provided to allow temporary access to a specific set of controllers through a configuration tool that would not normally have access to the BACnet network. There may be multiple Installation keys in use by different devices simultaneously, so that different configuration tools could use different Installation keys, if desired.

The Distribution keys are used to distribute the General-Network-Access, User-Authenticated, and Application-Specific keys, which may change over time as needed to meet local security policies. They are also used to distribute the temporary Installation keys.

The Device-Master keys are used only for the distribution of the Distribution keys and remain the most secure of all key types because they are unique for every device and their use on the wire is very limited.

24.1.3 Securing Messages

Security is applied at the network layer by creating a new NPDU message type. Plain BACnet messages are secured by placing the NSDU portion of the message into the Payload of a Security-Payload message. Therefore, when a BACnet APDU is encapsulated with security information, it is transported as a network layer message and the control bit in the NPCI is changed to indicate that the message now contains a network layer message rather than an APDU. The security header will indicate that the encapsulated message is an APDU so that this information is not lost. Upon unwrapping this message, this control bit will change back so that the plain NPDU will once again indicate that it contains an APDU.

For NPDUs and BVLLs containing NPDUs, the portion of the message starting with the network layer Message_Type field is placed into the Payload of a Security-Payload message. For BVLL messages that do not contain an NPDU, the original BVLL is embedded in a Secure BVLL message.

The basic level of security that can be applied to a BACnet message consists of signing each message using HMAC (keyed-hash message authentication algorithm) and MD5 or SHA-256 (commonly used hash algorithms), and of marking each message with the source and destination Device instances, a Message Id and a timestamp. Including source and destination addresses and source and destination device instances assures that messages cannot be spoofed or redirected. However, this requires that all secure devices, even routers and BBMDs, contain an application layer and Device object.

Message Id fulfills several purposes in securing BACnet messages. It is used to detect the replay of messages, to associate security responses with security requests, and along with the Timestamp field, to provide variability in otherwise identical messages.

Timestamp is used mainly for prevention of message replay but also serves as a source of variability in the message content so that messages that are repeated frequently do not generate the same signature. The clocks of secure devices are required to be loosely synchronized. If a timestamp on a message is outside the security time window, then an error is returned and clock issues need to be addressed. Within the security time window, Message Ids are checked to confirm that a message has not been replayed.

A higher level of security is provided by encrypting BACnet messages so that the content of the message cannot be determined without the possession of an appropriate key. Even the length of the message can be obscured by using a varying amount of hidden padding.

24.1.4 Network Security Policies

There are two network trust levels - trusted and non-trusted. Networks can be designated as trusted due to being physically secure, or due to the use of protocol security (signatures and/or encryption). Non-trusted networks are those which are both physically non-secure and not configured to require protocol security.

BACnet messages that do not have any security information in them are referred to as "plain" messages. Therefore, there are four corresponding network security policies: plain-trusted (requires physical security; no protocol security applied), signed-trusted (physical security not required; secured with signatures), encrypted-trusted (physical security not required; secured with encryption), and plain-non-trusted (not physically secure; no signature or encryption applied). A common example of a plain-trusted network is an MSTP network where all devices are locked up and no direct network connections are available outside of the locked space. Devices that do not support the BACnet security messages must reside only on plain-trusted networks for their communications to be trusted by secure devices. A common example of a

plain-non-trusted network would be the corporate LAN. However, the LAN may be re-designated as signed-trusted or encrypted-trusted by requiring all BACnet devices on the LAN to implement BACnet security and sign/encrypt all messages.

24.1.5 Device Level Security

Secure Devices are not restricted to residing on trusted networks (plain-trusted, signed-trusted, or encrypted-trusted). Secure devices can be located on non-trusted networks and rely on end-to-end (device level) security for secure communications. While trusted networks are created by setting the security policy for a network, and all devices on a trusted network are required to be configured with the security policy of the network, end-to-end security is determined on a device by device and request by request basis.

Secure BACnet devices are configured with a base device security policy that dictates the device's minimum level of security for sending or receiving messages. This policy may be higher than, but not lower than, the network access security policy.

Incapable Devices (devices that are not capable of processing BACnet security messages or those that have been configured to not be able to process BACnet security messages) shall reside on a plain network (plain-trusted or plain-non-trusted). Secure devices can also reside on this same network, but their Base_Device_Security_Policy property is required to be set to PLAIN if they need to communicate with the incapable devices. Even if the Base_Device_Security_Policy property is set to PLAIN for interoperability with incapable devices, the secure devices are free to use secured messages, for communicating with other secure devices, for any traffic that needs to be secured.

24.1.6 Secure Tunnel Mode

The standard allows for a tunnelling mode whereby plain and signed packets arriving at one end of the tunnel (e.g., router A on subnet A) can be tunnelled to another device (e.g., router B on subnet B across a non-physically-secure network segment). The tunnelling router applies encryption (and signature if needed) using the General-Network-Access key and forwards the packet along to the other end of tunnel. Control bits in the security header indicate that the packet has been tunnelled. If a packet is already encrypted, the tunnelling router passes the message as is.

To avoid inverted networks, it is recommended that only BACnet/IP be used for secure tunnels when connecting non-secure BACnet/IP or BACnet/Ethernet networks. BACnet/IP is preferred for secure tunnels since it is the only medium through which full size BACnet packets (1476 octets of APDU) can be transferred when security is enabled. In such installations, all BACnet products can take advantage of the secure tunnel, not just those that are security aware or only communicate with smaller PDU sizes.

24.1.7 User Authentication

The BACnet security architecture allows for multiple methods for user authentication. Currently only a single method of user authentication is defined: Proxied User Authentication.

Proxied User Authentication relies on site policy and trust of selected software to perform user authentication. To allow for some clients to be trusted to perform user authentication, and some clients that do not perform, or are not trusted to perform, user authentication, different security keys are provided. Clients with user interfaces that are trusted to perform user authentication are given the User-Authenticated key, or an Application-Specific key. Other clients that need access to the network but are not trusted to securely authenticate users are given the General-Network-Access key.

24.1.8 Key Distribution

BACnet security keys are distributed to all devices by a BACnet Key Server. The General-Network-Access, User-Authenticated, Application-Specific, and Installation keys are bundled into a set and distributed together with a single key revision number, each device receiving a specific set of keys appropriate for that device. While different devices may receive different key sets (differing in Application-Specific or Installation keys, for example), the key sets shall share the same revision number across all devices after a key distribution is complete.

Each BACnet device shall either have a unique factory-fixed Device-Master key, or support initiation of Request-Master-Key service and execution of the Set-Master-Key service. The Key Server will use a device's Device-Master key to securely provide the device with a device specific Distribution key. The Key Server will then use the Distribution key to send the device its set of security keys. Distribution keys are therefore revised separately from other keys, as they may change less frequently.

A full description of the key distribution protocol is defined in Clause 24.21.3.

All secure devices shall support the key distribution messages defined in this standard. In addition, they may also support proprietary mechanisms for setting keys. For example, an installation tool may configure an initial key set as part of its programming and commissioning operations.

24.1.9 Deployment Options

Security deployment always involves careful consideration for balancing costs, complexity, and time of configuration and maintenance against the likelihood of various attack scenarios and the sensitivity of the data or actions being protected. This standard provides for a continuum of protection from very simple and coarse grained to very powerful and fine grained.

Using the architecture defined here, very simple deployments can be made. Some deployments may not require a live Key Server. In these cases, the function of the Key Server is performed by the installation tool(s) and all devices are given infinite duration keys so that no Key Server is needed after installation. In addition, all key values can be set to be the same value if only a moderate level of security is needed to protect moderately critical resources.

Also using the architecture defined here, highly specific and highly secure deployment requirements can be met by segregating collections of devices using Application-Specific keys and tightly controlling the distribution of those keys to a limited number of devices. In addition, a live Key Server can be used to distribute expiring keys periodically according to site policy. User information is provided so that fine grained authorization policies (e.g., access control lists) can be based on the source device and/or the source user or process. The authentication mechanism can be extended to support complex proprietary methods, if required.

24.1.10 Limitations and Attacks

Highly secure communications between peer devices requires not only the knowledge of the proper key(s), but also the knowledge of a peer device's device instance number as well. This is because there are attack scenarios where it may be possible for the source and destination address information (SNET, SADR, DNET, DADR) to be altered. The relative ease or difficulty of these attacks is affected by the site's physical access policies and the skill and equipment of the attackers.

Altering the addressing information may be accomplished by gaining physical access to a secured device and changing its MAC address (e.g., by changing its address switches), by causing its IP address to change (e.g., by spoofed DHCP messages or a physically inserted NAT device), or by placing it on another network, either by physically moving the device or by remotely rewiring the networks.

Secure devices should not allow their instance numbers to be changed by physical switches after installation; device instance numbers should only be changeable via secured communications with a configuration tool. Therefore, the device instance number is the most trustworthy form of identifying the source or destination of a message, and highly secured communications should always include the destination device instance number (the source instance is always known and always included).

Devices receiving messages where the device instance of the destination is unknown should act accordingly based on their internal policies for the operation being requested. The device instance of the source is always known and may be used by the destination device's internal policies for determining how to handle these messages. In many cases, the knowledge by the destination of the authorized source instances may be sufficient to relieve the source of having to know the destination's instance.

There are also ways to avoid the condition of a source device not knowing the instance of a destination. For example, the device instance form of a recipient address should be used rather than the address form, and services like "Subscribe COV" should record the requesting device instance along with its address.

Secure devices should restrict the setting of their device instance number to communications that are secured with an Installation key, which may be temporary and unique to the device. Site policies should restrict user access to software that is authorized to change instance numbers in secure devices. But since this software is likely the same software that can completely reprogram the devices, this policy may already be in place. Site policies should also restrict physical access to highly secured devices so that their internal memory cannot be physically tampered with. Here again, this is likely to be an existing policy for such devices.

Many of the above attacks involve physical access to either secured devices themselves or to the wiring between devices. Given this opportunity, Denial of Service attacks are trivial and obvious and this standard does not address their

prevention. However, to limit over-the-wire Denial of Service attacks, this standard allows some error conditions to be ignorable. For example, devices that want to hide from scanners are allowed to ignore messages that are using an unknown key or appear to be replayed.

In general, error responses are helpful for diagnosing or recovering from some forms of legitimate network problems, however, some devices may want to limit repeated error responses to repeated receipt of erroneous messages, which may actually be an attempt at a Denial of Service attack. Legitimate devices should be designed to recover from errors like outdated key sets or incorrect timestamps in a reasonable manner or should limit their rate of sending unsuccessful messages to avoid creating an inadvertent Denial of Service attack by repeatedly sending erroneous messages to other secure devices.

24.1.11 Minimum Device Requirements

In order to implement BACnet network security in a device, the device shall:

(a) have an application layer;
(b) support execution of WriteProperty;
(c) be able to track time;
(d) have non-volatile re-writable storage in which to retain some run-time and configuration data;
(e) not be an MS/TP slave.

In addition, it is recommended that secure devices have a real-time clock that is persistent across resets and extended power down periods.

24.2 Security Wrapper

All BACnet security messages use the same security wrapper consisting of a header, an optional body, and a required signature. The format of the wrapper is:

Table 24-1. Security Wrapper Format

Field Name	Size
Control	1 octet
Key Revision	1 octet
Key Identifier	2 octets
Source Device Instance	3 octets
Message Id	4 octets
Timestamp	4 octets
Destination Device Instance	3 octets
DNET	2 octets
DLEN	1 octet
DADR	Variable
SNET	2 octets
SLEN	1 octet
SADR	Variable
Authentication Mechanism	1 octet
Authentication Data	Variable
Service Data	Variable
Padding	Variable
Signature	16 octets

All multi-octet fields shall be conveyed with the most significant octet first. The DADR and SADR fields shall be encoded as described in Clause 6.

24.2.1 Security Header Protocol Control Information

Each security message NPDU shall start with a control octet that includes indications of the presence or absence of particular security header fields.

Bit 7: 1 indicates that the Payload contains a network layer message or a Secure-BVLL.
0 indicates that the Payload contains an application layer message

Bit 6: 1 indicates that the message is encrypted.
0 indicates that the message is not encrypted.

This bit is referred to as the 'encrypted flag' and shall always be 0 when calculating the signature for the message.

Bit 5: Reserved. Shall be 0.

Bit 4: 1 indicates that the Authentication Mechanism and Authentication Data fields are present.
0 indicates that the Authentication Mechanism and Authentication Data fields are absent.
The Authentication Mechanism and Authentication Data fields are optionally present on request messages but shall be absent from response messages (e.g., Complex Ack, Simple Ack, Security Response)

Bit 3: 1 indicates that the Security Wrapper should not be removed, except by the destination device. This bit shall be 1 if Bit 2 (the 'do-not-decrypt flag') is set to 1.
0 indicates that the Security Wrapper should be removed before placing the message on a plain network segment.
This bit is referred to as the 'do-not-unwrap flag'.

Bit 2: 1 indicates that encryption should not be removed, except by the destination device. If this bit is set to 1, then Bit 3 (the 'do-not-unwrap flag') shall be set to 1. This bit shall not be 1 when Bit 6 ('encrypted flag') is set to 0.
0 indicates that encryption should be removed before placing the message on a network segment that does not require encryption.
This bit is referred to as the 'do-not-decrypt flag'.

Bit 1: 1 indicates that the message was received from a plain-non-trusted network and that the security information was placed on the message by the router from the plain-non-trusted network to a trusted-signed or trusted-encrypted network. Routers should not route plain messages from plain-non-trusted network to a plain-trusted network.
0 indicates that the message originated on a trusted network, or that the originator applied the security header.
This bit is referred to as the 'non-trusted-source flag'.

Bit 0: 1 indicates that the message was secured by an intervening router.
0 indicates that the message was secured by the originator.
This bit is referred to as the 'secured-by-router flag'.

24.2.2 Key Revision

This field shall contain the key revision for the key identified by the Key Identifier field.

This field shall be 0 when the Key Identifier indicates the Device-Master key.

24.2.3 Key Identifier

The Key Identifier field specifies the key that is used to sign the message. If the do-not-decrypt flag has a value of 1, then it also specifies the key used to decrypt the message. If the do-not-decrypt flag has a value of 0, the General-Network-Access key is used to decrypt the message as it is the only key that is guaranteed to be known by intermediate routers (see Clause 24.21.1).

24.2.4 Source Device Instance

The Source Device Instance is the Device object instance of the device that applied security to the message.

The field shall be restricted to the range 0 through 4194302. This requires that all secure BACnet devices, even those that are only routers or BBMDs, contain an application layer and a Device object.

Note that this field cannot be used to identify the source device of a message when the secured-by-router flag is set as it will indicate the router's Device object instance.

24.2.5 Message Id

The Message Id is a 32 bit monotonically increasing counter value that is present in all secure messages. It is used for matching security responses to security requests, for preventing replay attacks, and along with the Timestamp provides variability between messages that might otherwise be identical.

In the normal course of operation, a device shall not generate more than one message with the same Message Id within the security time window.

If a device does not remember its Message Id across resets, then the device may have problems communicating for the first security time window period. Such a condition should be expected if the device resets within the first security time window period of a previous reset, and it always resets its Message Id counter to the same value on reset. Waiting 2 * Security_Time_Window seconds before communicating will overcome this problem.

24.2.6 Timestamp

The Timestamp field, an unsigned 32 bit integer, indicates the time of the message in UTC as seconds since 12:00 AM January 1, 1970 (standard Unix timestamp).

24.2.7 Destination Device Instance

The Destination Device Instance is the Device object instance of the destination device for the message. A value of 4194303 shall be used in all broadcast messages and when the device instance of the destination device is unknown to the device applying the security. Secure devices shall attempt to determine the Device object instance of the destination device, and only if attempts to determine the value fail, shall a secure device resort to the use of 4194303 in unicast messages.

24.2.8 DNET/DLEN/DADR

These fields contain the values of the fields with the same name from the NPCI portion of the message. They are always present and are included in the security header to allow the signing of the values.

When the message is to be placed onto the destination network, or is received from the destination network, the NPCI will not contain the DNET/DLEN/DADR fields. Regardless of whether the NPCI contains the DNET/DLEN/DADR fields, the security header shall contain these fields and they shall contain the correct destination address information.

For the Update-Key-Set, Update-Distribution-Key, and Set-Master-Key, the DNET shall be set to 0 in the security header if the SNET was 0 in the corresponding Request-Key-Update or Request-Master-Key message.

24.2.9 SNET/SLEN/SADR

These fields correspond to the fields with the same name from the NPCI portion of the message. They are always present and are included in the security header to allow the signing of the values. As such, the values are required to be known and filled in by the device. This is in contrast to non-secure BACnet messages where these fields in the NPCI are only present when added by a router when routing remote messages.

When a security header is placed in a message by a router on behalf of another device, these fields shall contain the address information of the originating device and not the address information of the router.

There are exceptions where the values are not known and cannot be filled in by the sending device. In the What-Is-Network message, the SNET shall be set to 0. In the Request-Key-Update, and Request-Master-Key, the SNET shall be set to 0 only when the device does not know its network number. However, the SLEN and SADR shall be set to valid values, if they are known. In the case where a device temporarily does not know its own SADR, such as a BACnet/IP device behind a NAT firewall, the SLEN shall be set to 0 and the SADR shall be empty. These devices shall learn their SADR by reading the destination address of any properly authenticated message sent to it.

When the message is to be placed onto the source network, or is received from the source network, the NPCI will not contain the SNET/SLEN/SADR fields. Regardless, the security header shall contain these fields and they shall contain the correct source address information.

24.2.10 Authentication Mechanism

If present, the Authentication Mechanism field is a 1 octet value that indicates the user authentication mechanism being used. This field shall be present when the User-Authenticated or an Application-Specific key is used and the PDU type is one that initiates a request (see below). It shall be absent when the Device-Master, Distribution, or Installation key is used. And it shall be optional when the General-Network-Access key is used.

User authentication information is not useful in responses or transmission control. User authentication is useful in any PDU type that initiates a request:

APDU PDU Types: Confirmed-Request, Unconfirmed-Request,

NPDU PDU Types: Initialize-Routing-Table, Establish-Connection-To-Network, Disconnect-Connection-To-Network,

B/IP BVLL Types: Write-Broadcast-Distribution-Table, Read-Broadcast-Distribution-Table, Register-Foreign-Device, Read-Foreign-Device-Table, Delete-Foreign-Device-Table-Entry,

B/IPv6 BVLL Types: Address-Resolution, Forwarded-Address-Resolution, Virtual-Address-Resolution, Register-Foreign-Device, Delete-Foreign-Device-Table-Entry.

It shall not be included in all other PDU types, but if it is present a receiving device shall ignore it. If user authentication information is provided in a segmented APDU, the authentication information shall be the same in all segments. User authentication information in response PDUs and transmission control PDUs shall not be present.

The proxied user authentication mechanism is indicated by a value of 0 and is the only standardized mechanism at this time.

Values in the range 200 through 255 are reserved for vendor specific mechanisms.

24.2.11 Authentication Data

The Authentication Data field is a variable length identifier that provides authentication information in a format specific to the mechanism defined by the Authentication Mechanism field.

This may be used by the server's authorization mechanism to verify that the user is allowed to perform the requested action.

A client device that authenticates users may be given the User-Authenticated key or an Application-Specific key. It shall indicate the authenticated user's identity when initiating communication.

This field is always at least three octets in length. The first two octets are an unsigned integer (most significant octet first) indicating the numeric User Id for the user that is authenticated for this message. The third octet is the user's role or group. The user role values are site specific values dictated by site policy and are used to group access rights. Example roles are: HVAC operator, technician, etc.

User Id values represent either unique human users, or processes within a BACnet system. Assignment of the values is based on local site policy, but they should be unique across all BACnet devices, such that User Id 1234, for example, means the same regardless of its source or destination.

User Roles 0 and 1 are reserved to mean "the system itself". User Role 0 is used for programmed device-to-device communication that is not initiated by human action. A User Role of 1 is used for device-to-device communication that is initiated by an "unknown human", such as the changing of a setpoint based on button presses on a thermostat.

Other User Role values may also be used for device-to-device communication to indicate a particular subsystem that is performing the action, but those values are not restricted by this standard and are taken from the same set of numbers as are used for human users and groups. The values 0 and 1 are the only ones that are reserved specifically for this purpose and shall not be assigned to human user roles.

User Id 0 is reserved to indicate that the source user is unknown. It is commonly used in conjunction with User Role 0 or 1.

If the Authentication Mechanism has a value of 0, then the Authentication Data field contains no further information since the authentication has been performed by the source.

If the Authentication Mechanism has a value of 1 through 199, then the next 2 octets of this field shall be an unsigned integer (most significant octet first) indicating the length, in octets, of the entire field. The meaning of the remaining octets is not currently defined by this version of this standard.

If the Authentication Mechanism has a value of 200 through 255, then the next 2 octets of this field shall be an unsigned integer (most significant octet first) indicating the length, in octets, of the entire field. Following that, the next 2 octets shall be an unsigned integer (most significant octet first) indicating a BACnet Vendor Identifier. The meaning of the remaining octets is defined by that vendor.