

BSR/ASHRAE Addendum g
to ANSI/ASHRAE Standard 135-2004

Public Review Draft

ASHRAE® Standard

Proposed Addendum g to Standard 135-2004, *BACnet®—A Data Communication Protocol for Building Automation and Control Networks*

Second Public Review (March 2007)
(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 addendum, use the comment form and instructions provided with this draft. 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 web site) remains in effect. The current edition of any standard may be purchased from the ASHRAE Bookstore @ <http://www.ashrae.org> or by calling 404-636-8400 or 1-800-527-4723 (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 web site @ <http://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 ASHRAE of any product, service, process, procedure, or design, and ASHRAE expressly disclaims such.

© March 16, 2007. This draft is covered under ASHRAE copyright. Permission to reproduce or redistribute all or any part of this document must be obtained from the ASHRAE Manager of Standards, 1791 Tullie Circle, NE, Atlanta, GA 30329. Phone: 404-636-8400, Ext. 1125. Fax: 404-321-5478. E-mail: standards.section@ashrae.org.

AMERICAN SOCIETY OF HEATING,
REFRIGERATING AND AIR-CONDITIONING
ENGINEERS, INC.
1791 Tullie Circle, NE · Atlanta GA 30329-2305



[This foreword, the table of contents, and the "rationale" following the table of contents 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.

SSPC 135 wishes to recognize the efforts of the following people in developing this addendum: **tbd**.

135-2004g-1. Updating BACnet Network Security, p. 4.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-2004 and Addenda 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 as this time. All other material in this addendum is provided for context only and is not open for public review comment except as it relates to the proposed changes.

CONTENTS

24	NETWORK SECURITY	4
24.1	Overview	4
24.1.1	Shared Keys	4
24.1.2	Securing Messages.....	5
24.1.3	Network Security Policies.....	6
24.1.4	Device Level Security.....	6
24.1.5	Secure Tunnel Mode	6
24.1.6	User Authentication	7
24.1.7	Key Distribution	7
24.1.8	Deployment Options	7
24.1.9	Limitations and Attacks	8
24.2	Security Wrapper.....	9
24.2.1	Security Layer Protocol Control Information	9
24.2.2	Key Revision.....	10
24.2.3	Key Identifier.....	10
24.2.4	Source Device Instance.....	10
24.2.5	Destination Device Instance.....	10
24.2.6	Message Id	10
24.2.7	Timestamp	11
24.2.8	DNET/DLEN/DADR.....	11
24.2.9	SNET/SLEN/SADR.....	11
24.2.10	Authentication Mechanism	11
24.2.11	Authentication Data	11
24.2.12	Service Data.....	12
24.2.13	Padding	12
24.2.14	Signature	13
24.3	Security Layer Messages.....	13
24.3.1	Challenge-Request	13
24.3.2	Security-Payload.....	16
24.3.3	Security-Response	17
24.3.4	Request-Key-Update.....	19
24.3.5	Update-Key-Set	21
24.3.6	Update-Distribution-Key	23
24.3.7	Request-Master-Key	24
24.3.8	Set-Master-Key	25
24.4	Securing an APDU	25
24.5	Securing an NPDU	27
24.6	Securing a BVLL.....	27
24.7	Securing Messages	29
24.7.1	Message Id	29
24.7.2	Timestamp	29
24.7.3	Device Identification.....	30
24.7.4	Message Signature	30
24.7.5	Encrypted Messages	32
24.8	Network Security Network Trust Levels.....	33
24.8.1	Trusted Networks.....	33
24.8.2	Non-trusted Networks.....	33
24.9	Network Security Policies	33
24.9.1	Plain-Non-Trusted	33
24.9.2	Plain-Trusted.....	33
24.9.3	Signed-Trusted.....	34
24.9.4	Encrypted-Trusted	34
24.10	Network Security.....	34
24.11	End to End Security.....	35

24.11.1	Determining Exceptional Encryption Requirements.....	35
24.12	Wrapping and Unwrapping Secure Messages	35
24.13	Authenticating Messages.....	35
24.13.1	Validating the Source MAC Address.....	35
24.13.2	Validating the Destination Device Instance	36
24.13.3	Validating the Message Id	36
24.13.4	Validating the Timestamp.....	36
24.13.5	Validating the Signature	37
24.14	Wrapping, Unwrapping, and Routing Messages	37
24.14.1	Unwrapping Security Errors	38
24.14.2	Securing Response Messages.....	38
24.15	User Authentication.....	38
24.15.1	Proxied User Authentication.....	38
24.16	Time Synchronization Requirements	39
24.16.1	BACnet Time Synchronization Messages	39
24.16.2	Overcoming Non-synchronized Clocks	39
24.17	Secure PDU Sizes.....	40
24.18	BACnet Security In A NAT Environment.....	40
24.19	BACnet Firewalls	41
24.19.1	Security Proxy.....	41
24.19.2	Network Filtering.....	41
24.20	Security Keys	42
24.20.1	Key Identifiers	42
24.20.2	Key Sets	43
24.20.3	Key Distribution	43
24.21	Key Server.....	43
24.21.1	Key Generation	44
24.21.2	Distribution Method.....	44
24.21.3	Initial Key Distribution	45
24.21.4	Multiple Key Servers	45
6.2.4	Network Layer Message Type	46
6.4.10	Challenge-Request.....	47
6.4.11	Security-Payload.....	47
6.4.12	Security-Response.....	47
6.4.13	Request-Key-Update	47
6.4.14	Update-Key-Set.....	47
6.4.15	Update-Distribution-Key	47
6.4.16	Request-Master-Key.....	47
6.4.17	Set-Master-Key	47
6.4.18	What-Is-Network-Number.....	47
6.4.19	Network-Number-Is	48
12.11.17	Max_APDU_Length_Accepted	48
12.21	Network Security Object Type.....	48
12.21.1	Object_Identifier.....	49
12.21.2	Object_Name	49
12.21.3	Object_Type.....	49
12.21.4	Description.....	49
12.21.5	Base_Device_Security_Policy	49
12.21.6	Network_Security_Policies.....	49
12.21.7	Security_Time_Window	49
12.21.8	Distribution_Key_Revision	50
12.21.9	Key_Sets.....	50
12.21.10	Last_Key_Server	50
12.21.11	Profile_Name.....	50

135-2004g-1. Updating BACnet Network Security

Rationale

The existing BACnet Network Security architecture defined in Clause 24 of Standard 135-2004 is based on the 56-bit DES cryptographic standard and needs to be updated to meet the needs of today's security requirements.

Addendum 135-2004g-1

[Replace **Clause 24** in its entirety, pp. 440-447, with the following new **Clause 24**.]

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 (network, application, and BVLL)
- (e) Placing non-security-aware devices, if physically secure, behind a secure 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 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 device and object binding, for encryption tunnels, and by user interface devices that cannot authenticate, or are not trusted to authenticate, a user. All devices must 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 field 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 field 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.

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 may thus be limited to highly secure communication.

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 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.

24.1.2 Securing Messages

Security is applied at the network layer by creating a new NPDU message type. Plain BACnet messages are secured by placing them in 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.

NPDUs are similarly wrapped as are BVLL messages containing an NPDU. For BVLL messages without 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 must be loosely synchronized. If a timestamp on a message is outside the

security time window, then the message 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.

24.1.3 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 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 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.

24.1.4 Device Level Security

Secure Devices are not restricted to residing on trusted networks (plain-trusted, signed-trusted, or encrypted-trusted). Secure devices may 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 must 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 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) must reside on a plain network (plain-trusted or plain-nontrusted). Secure devices can also reside on this same network, but their Base Device Security Policy must be set to plain if they need to communicate with the incapable devices. Even if the Base Device Security Policy 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.

24.1.5 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.

24.1.6 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.

24.1.7 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 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 revisioned separately from other keys, as they may change less frequently.

A full description of the key distribution protocol is defined in 24.20.3.

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.

24.1.8 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 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.

24.1.9 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.

But secure devices should be designed in such a manner that there is no way, short of invasively changing the device's internal nonvolatile memory, to get a secured device to change its device instance number after installation (secure devices should not allow their instance numbers to be set by physical switches). 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 could be used rather than the address form, and services like "Subscribe COV" could 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 the incorrect 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.

24.2 Security Wrapper

All BACnet security messages use the same security wrapper consisting of a fixed 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
Destination Device Instance	3 octets
Message Id	3 octets
Timestamp	4 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 Layer 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.
0 indicates that the Payload contains an application layer message
The value of this bit shall be ignored when securing a BVLL.
- 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.

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.

24.2.2 Key Revision

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

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 encrypt the message.

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.

24.2.5 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 should take all reasonable steps to ensure that a value other than 4194303 is used in unicast messages.

24.2.6 Message Id

The Message Id is a 24 bit counter value that is present in all secure messages. It is used for matching security layer responses to security layer 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. A monotonically increasing counter will suffice for the Message Id, although implementations are allowed to use random values as long as uniqueness over the security time window is maintained.

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.

24.2.7 Timestamp

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

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.

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 must 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; 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. 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.

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.

If the Authentication Data field is present in a segmented message, it shall be present and contain the same value in all segments of the message.

This field is always at least two octets in length. The first two octets are an integer (most significant octet first) indicating the numeric User Id for the user (or group) that is authenticated for this message.

User Id values represent either unique human users, groups of 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.

A User Id of 0 is reserved to mean "the system itself" and may be used for device-to-device communication that is not initiated by human action. Other User Id 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 value 0 is the only one that is reserved specifically for this purpose and shall not be assigned to a human user.

If the Authentication Mechanism has a value of 0, then this 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 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 Standard 135.

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

24.2.12 Service Data

The Service Data field contains security specific data. Its content varies by message type.

The security NPCI message type values are:

- X'0A': Challenge-Request
- X'0B': Security-Payload
- X'0C': Security-Response
- X'0D': Request-Key-Update
- X'0E': Update-Key-Set
- X'0F': Update-Distribution-Key
- X'10': Request-Master-Key
- X'11': Set-Master-Key

24.2.13 Padding

The padding is present if and only if the message is encrypted. This field is sized to ensure that the length of the data being encrypted is a multiple of the encryption algorithm's block size. The padding field is added after the signature is calculated.

The first two octets of the padding field are a count (most significant octet first) that indicates the total number of octets of padding, including the count itself. The values of all remaining octets are unspecified.

Since the count includes itself, and cannot be zero, the padding field is always included if the message is encrypted.

The size of the padding field may be increased, by adding multiples of the block size to the minimum requirement, to allow devices to hide the true length of their encrypted messages.

24.2.14 Signature

The signature contains an HMAC of the message. See Clause 24.7.4 for details on generating the signature.

The signature (in whole or in part) is also used as the Initialization Vector for the encryption algorithm.

24.3 Security Layer Messages

24.3.1 Challenge-Request

The Service Data for a Challenge-Request message has the following form:

Table 24-2. Challenge-Request Service Data

Message Field	Size	Description
Message Challenge	1 octet	When set to 1, this field indicates that the Challenge-Request is being sent in response to a message. Otherwise, the Challenge-Request is being sent for some other reason and the following fields contain random data.
Original Message Id	3 octets	The Message Id from the message that caused the device to issue the challenge.
Original Timestamp	4 octets	The timestamp from the message that caused the device to issue the challenge.

Any device that receives a secure BACnet message may, at the device's discretion, challenge the message source. Specific cases where a device may want to challenge a message source are as follows: on receipt of an I-Am or I-Am-Router-To-Network where the source address does not match a previously cached value, on receipt of a secure message where the source MAC address does not match the source address in the secured NPDU and both devices are on the same BACnet/IP network, on receipt of a message where the source address is not present, and on receipt of a unicast message where the Destination Device Instance is not specified. When challenging a specific message, the Message Challenge field shall be set to 1.

A device may also arbitrarily Challenge another device simply by generating a Challenge-Request with a random Original Message Id, and any value for the Original Timestamp. When performing a challenge without reference to a specific message, the Message Challenge field shall be set to 0.

Upon receipt of a Challenge-Request that authenticates correctly according to Clause 24.13, with a Message Challenge field set to 1, the device shall attempt to verify that it originated the message identified by Original Message Id. If the device is unable to verify that it sent the specified message such as would occur if its Message Id cache were to overflow, then the device shall assume that it did send the request and send a positive Security-Response.

Upon receipt of a Challenge-Request that authenticates correctly according to Clause 24.13 with a Message Challenge field set to 0, a device shall respond with Security-Response of Success

Broadcasts of this Message Type shall be ignored.

Devices that do not know their own MAC address, such as BACnet/IP devices behind a NAT firewall, may use the Challenge-Request message to determine their own address by examining the DADR in the Security-Response message.

The possible error codes returned in response to a Challenge-Request are listed in Table 24-3 below. The order in which the error codes are checked is important, and is specified by the order in the table. When an ignorable error occurs, it is a local matter as to whether the device returns the error or does not respond at all. If multiple error conditions are present, the condition that occurs first in the table shall be the error that is returned. Note that if multiple errors are present and the one that occurs first in the table is ignorable, then the device has the choice to either return that error or not respond at all; the device does not have the option to return an error that occurs later in the table.

Table 24-3. Challenge-Request Error Codes

Error Code	Ignorable	Description
functionNotConfigured	Yes	If the recipient is not configured for security on this port.
encryptionNotConfigured	Yes	If the Encrypted field is set to 1 and the receiving device is not configured to accept encrypted messages.
unknownEncryptionAlgorithm	Yes	If the Encryption Algorithm indicates an algorithm that the receiving device does not support or the value is outside the range of algorithms identifiers known to the device.
unknownKey	Yes	If the Key Identifier field indicates a security key that the receiving device does not know.
duplicateMessage	Yes	A message with the provided Message Id has already been received from the source device within the security time window.
unknownKeyRevision	Yes	If the Key Revision field indicates a revision that the receiving device does not know.
badSignature	Yes	If the signature is not correct.
badSourceAddress	No	If the source address information is missing or invalid
badDestinationAddress	Yes	If the destination address information is missing or invalid.
unknownSourceMessage	No	The specified message was not sent by the client. While devices are required to track all messages that have been sent, if a device is capable of detecting that it did not send the specified message, it shall use this error code to indicate that it was not the source of the challenged message. If the device cannot detect that it did not send the message, then it will assume that it did.
badTimestamp	No	The Timestamp in the security header of the message is not within the allowable timestamp window of the receiver.
encryptionRequired	No	If the Encrypted flag is set to 0,0 and the server's policy requires encryption.
unknownAuthenticationType	No	If the user authentication method in the message is unknown to the device.

24.3.2 Security-Payload

The Service Data for a Security-Payload message has the following form:

Table 24-4. Security-Payload Service Data

Message Field	Size	Description
Payload Length	2 octets	The number of octets of payload
Payload	Variable	The secure NSDU

The Security-Payload message is used to transfer non-security related BACnet messages between communicating parties. As with all secure BACnet messages, the message is signed and may be optionally encrypted.

If the recipient of this message cannot process the message for one of the reasons listed below, and if the message was unicast, a negative Security Response may be returned to the sender with a Response Code as shown in the following table.

The possible error codes returned in response to a Security-Payload message are listed in Table 24-5 below. The order in which the error codes are checked is important, and is specified by the order in the table. When an ignorable error occurs, it is a local matter as to whether the device returns the error or does not respond at all. If multiple error conditions are present, the condition that occurs first in the table shall be the error that is returned. Note that if multiple errors are present and the one that occurs first in the table is ignorable, then the device has the choice to either return that error or not respond at all; the device does not have the option to return an error that occurs later in the table.

Table 24-5. Security-Payload Error Codes

Error Code	Ignorable	Description
functionNotConfigured	Yes	If the recipient is not configured for security on this port.
encryptionNotConfigured	Yes	If the Encrypted field is set to 1 and the server is not configured to accept encrypted messages.
unknownEncryptionAlgorithm	Yes	If the Encryption Algorithm indicates an algorithm that the receiving device does not support or the value is outside the range of algorithms identifiers known to the device.
unknownKey	Yes	If the Key Identifier field indicates a security key that the receiving device does not know.
unknownKeyRevision	Yes	If the Key Revision field indicates a revision that the receiving device does not know.
badSignature	Yes	If the signature is not correct.
badSourceAddress	No	If the source address information is missing or invalid.
badDestinationAddress	Yes	If the destination address information is missing or invalid.
badTimestamp	No	The Timestamp in the security header of the message is not within the allowable timestamp window of the receiver.
encryptionRequired	No	If the Encrypted field is set to 0,0 and the server's policy requires encryption.
unknownAuthenticationType	No	If the user authentication method in the message is unknown to the device.

24.3.3 Security-Response

The Service Data for a Security-Response message has the following form:

Table 24-6. Security-Response Service Data

Message Field	Size	Description
Response Code	1 octet	The type of response (positive acknowledgement or error code).
Original Message Id	3 octets	The Message Id of the message that caused the response.
Original Timestamp	4 octets	The Timestamp of the message that caused the response.
Response Specific Parameters	Variable	The contents of this field are dependent on the value of the Response Code field.

This message is sent as a positive acknowledgement of another security message, or when a security error occurs and the reporting of that error is allowed by the security policy. A Security-Response message is never sent in response to a broadcast message. Certain errors may be suppressed if hiding from port scanners is desired.

The reaction of the recipient to this message is a local matter. Usually Security-Response messages that represent errors will be either ignored, logged, or delivered to a human operator.

The Security-Response messages are sent in response to a security message and indicate a success or failure to process the message. All security responses shall be sent with the same level of security (signed or encrypted) and with the same Key Identifier that the original message had except Security-Response messages that indicate an error of encryptionNotConfigured, unknownKey or unknownKeyRevision which shall not be encrypted.

All Security-Response messages shall be signed. If the generating device is not configured for security (e.g., has not received keys), or its keys are out of date (e.g., due to being off-line for an extended time), it will be unable to generate a Security-Response message.

Security-Response messages that indicate an error of unknownKey shall be secured with the General-Network-Access key.

Security-Response messages shall not be sent in response to Security-Response messages.

Broadcasts of this Message Type shall be ignored.

The following list defines the allowable security response codes.

- X'00': success
- X'01': badAuthenticationMethod
- X'02': badSignature
- X'03': badTimestamp
- X'04': correctKeyRevision
- X'05': encryptionNotConfigured
- X'06': encryptionRequired
- X'07': functionNotConfigured
- X'08': unknownAuthenticationType
- X'09': unknownKey
- X'0A': unknownKeyRevision
- X'0B': unknownSourceMessage
- X'0C': badSourceAddress
- X'0D': badDestinationAddress
- X'0E': unknownEncryptionAlgorithm
- X'0F': duplicateMessage

Descriptions of Response Specific Parameters follow. Response Codes for which no Response Specific Parameters are defined have no Response Specific Parameters and shall be transmitted without a Response Specific Parameters field.

24.3.3.1 badTimestamp

The Response Specific Parameter for a badTimestamp error is shown in the following table:

Table 24-7. badTimestamp Response Specific Parameters

Message Field	Size	Description
Expected Timestamp	4 octets	The current time of the device that generated the Security-Response message.

Devices that generate a badTimestamp error shall set the Timestamp field in the security header to the value provided in the original message to ensure that it will be accepted by the destination device. This is the only case where the Timestamp field in the security header does not represent the current time in the generating device.

24.3.4 Request-Key-Update

The Service Data for a Request-Key-Update message has the following form:

Table 24-8. Request-Key-Update Service Data

Message Field	Size	Description
Set 1 Key Revision	1 octet	The Key Revision of the device's first key set.
Set 1 Expiration Time	4 octets	The UTC time at which this key set expires after which the device shall no longer accept or generate message with keys from the set.
Set 2 Key Revision	1 octet	The Key Revision of the device's second key set.
Set 2 Activation Time	4 octets	The UTC time at which the device should switch to using the second key set.
Set 2 Expiration Time	4 octets	The UTC time at which this key set expires after which the device shall no longer accept or generate message with keys from the set.
Distribution Key Revision	1 octet	The revision for the Distribution key

This security message is used by secure devices that either do not have a valid key set, or want to ensure that both of the device's key sets are still the most current.

If a secure device does not have a valid Distribution key, it shall secure this message with its Device-Master key thus indicating to the Key Server that a Distribution key is required. In this case, all the revision number fields shall be 0, to indicate that no valid revision has been received, and the time fields are meaningless and are ignored by the Key Server.

Upon receipt of a valid Request-Key-Update message secured with the device's Device-Master key, a Key Server device shall respond to the device with an Update-Distribution-Key message and then send an Update-Key-Set message to the device.

Upon receipt of a valid Request-Key-Update message secured with the device's Distribution key, a Key Server shall respond to the device with a Security-Response message with a Response Code of correctKeyRevision if the key revision data provided are the same as the Key Server has recorded for the device and the Key Server does not have an outstanding set of keys to send to the device. Otherwise the Key Server shall respond to the device with an Update-Key-Set message.

Key Servers shall not restrict execution of this service based on the Authentication Mechanism.

A device that receives no response to a Request-Key-Update message, or receives an error of unknownKey should retry the request secured with the device's Device-Master key. This allows the device to obtain a new Distribution key and key set in those cases where the device and the Key Server have become out of sync.

Broadcasts of this Message Type shall be allowed.

The possible error codes returned in response to a Request-Key-Update are listed in Table 24-9 below. The order in which the error codes are checked is important, and is specified by the order in the table. Implementations are allowed to ignore errors up to, and including, badSignature so as to hide from devices that scan for secure devices. If multiple error conditions are present, the condition that occurs first in the table shall be the error that is returned. Note that if multiple errors are present and the one

that occurs first in the table is ignorable, then the device has the choice to either return that error, or not respond at all; the device does not have the option to return an error that occurs later in the table.

Table 24-9. Request-Key-Update Error Codes

Error Code	Ignorable	Description
functionNotConfigured	Yes	If the recipient is not configured for security on this port, or is not configured as a Key Server for the requesting device.
encryptionNotConfigured	Yes	If the Encrypted field is set to 1 and the server is not configured to accept encrypted messages.
unknownEncryptionAlgorithm	Yes	If the Encryption Algorithm indicates an algorithm that the receiving device does not support or the value is outside the range of algorithms identifiers known to the device.
unknownKey	Yes	If the Key Identifier field indicates a security key that the receiving device does not know.
duplicateMessage	Yes	A message with the provided Message Id has already been received from the source device within the security time window.
unknownKeyRevision	Yes	If the Key Revision field indicates a revision that the receiving device does not know.
badSignature	Yes	If the signature is not correct.
badSourceAddress	No	If the source address information is missing or invalid.
badDestinationAddress	Yes	If the destination address information is missing or invalid.
badTimestamp	No	The Timestamp in the security header of the message is not within the allowable timestamp window of the receiver.
encryptionRequired	No	If the Encrypted field is set to 0,0 and the server's policy requires encryption.

24.3.5 Update-Key-Set

The Service Data for a Update-Key-Set message has the following form:

Table 24-10. Update-Key-Set Service Data

Message Field	Size	Description
Clear Key Set	1 octet	Indicates whether the keys provided in this message should be added to the existing key sets or should replace the complete existing keys sets.
Set 1 Key Revision	1 octet	The Key Revision of the device's first key set.
Set 1 Expiration Time	4 octets	The UTC time at which this key set expires, after which the device shall no longer accept or generate message with keys from the set.
Set 1 Key Count	1 octet	The number of keys in the first key set.
Set 1 Keys	Variable	The first key set, consisting of a concatenated sequence of key entries.
Set 2 Key Revision	1 octet	The Key Revision of the device's second key set.
Set 2 Activation Time	4 octets	The UTC time at which the device should switch to using the second key set.
Set 2 Expiration Time	4 octets	The UTC time at which this key set expires, after which the device shall no longer accept or generate message with keys from the set.
Set 2 Key Count	1 octet	The number of keys in the second key set.
Set 2 Keys	Variable	The second key set, consisting of a concatenated sequence of key entries.

This security message is used by the Key Server to provide keys to secure devices. This message shall always be signed and encrypted with the destination device's Distribution key.

The message can be used to provide a new key set, to modify an existing key set, or to invalidate an existing key set.

If the Clear Key Set parameter has the value 1, the device shall discard its current key sets before updating them with the provided keys. If the Clear Key Set parameter has the value 0, then the provided keys are added to the existing key sets, replacing existing entries for the same keys, if present.

The Clear Key Set parameter allows the Key Server to use multiple Update-Key-Set messages when the total number of keys would overflow the maximum NPDU size transmittable to the secure device. In this case, the Key Server shall send a group of messages, where the first message shall set the Clear_Key_Set parameter to non-zero, and all subsequent messages in the group shall set the Clear_Key_Set parameter to zero. Because this group has a strict order requirement, the Key_Server shall ensure the successful delivery of the first message before sending the subsequent messages in the group.

When the Key Server generates a new key set for a device, the previous Set 2 key set is moved to Set 1 and the new set is made to be Set 2. The activation time for Set 2 is calculated with the intent that the set will be activated after the key sets in all devices have been updated but before Set 1 expires. The expiration time for Set 2 should be calculated based on the expected time at which the next key set will be generated. The Key Server then uses this message to update the key sets for each device.

Generally, when any key in a key set is revised, the Key Revision field shall be incremented by 1. However, a key distribution may be sent to a device using the same revision number if that device has been authorized to receive an additional key (e.g., a temporary Installation key or new Application-

Specific key) while other keys remain unchanged. Modification of existing key sets allows the Key Server to change expiration or activation times on key sets without changing the key values.

A key revision of 0 indicates that the key(s) have not been configured. Therefore, when the key revision number wraps after being incremented past 255, it wraps back to 1, not 0.

In some circumstances it may be desirable to update all existing key sets. In such a scenario, the Key Server can provide 2 completely new key sets to a device.

If local site policy allows for infinite key expiration, the Expiration Time of key set 1 shall be set to X'FFFFFFFF', and key set 2 shall have a key revision of 0 and shall be ignored. Infinite key expiration can be useful for small sites that do not wish to maintain an active Key Server at all times. In this case, the Key Server is a temporary device that assigns keys once and then is not needed again for normal operation. To allow for this deployment scenario, devices shall maintain key data in a non-volatile manner and shall not be dependent on a Key Server after a power up or reset.

Since key revision numbers will wrap around and be reused every 255 revisions, Key Servers that are using non-infinite expirations shall set the expiration times so that the keys are set to expire sooner than the Key Server is scheduled to have revised the keys 255 times.

Upon receiving a valid Update-Key-Set message signed and encrypted with the device's Distribution key, a device shall replace both of its key sets and then respond to the message with a Security-Response message containing a Response Code of Success.

Devices shall not restrict execution of this service based on the authentication mechanism; knowledge of the device's Distribution key shall always be sufficient authorization.

Each key entry in the message shall be of the form:

Table 24-11. Key entry description

Message Field	Size	Description
Key Identifier	2 octets	The Key Identifier for the key pairs
Key Size	1 octet	The size of the key, in octets.
Key	Variable	The key value, consisting of the signature key followed by the encryption key.

The correct key sizes by algorithm are:

Table 24-12. Key sizes by algorithm

Hash algorithm (key size bytes)	Encryption algorithm (key size bytes)	Key field size
MD5 (16)	AES (16)	32 octets
SHA-256 (16)	AES (16)	32 octets

Broadcasts of this Message Type shall be ignored.

The possible error codes returned in response to an Update-Key-Set are listed in Table 24-13 below. The order in which the error codes are checked is important, and is specified by the order in the table. Implementations are allowed to ignore errors up to, and including, badSignature so as to hide from devices that scan for secure devices. If multiple error conditions are present, the condition that occurs first in the table shall be the error that is returned. Note that if multiple errors are present and the one that occurs first in the table is ignorable, then the device has the choice to either return that error, or not respond at all; the device does not have the option to return an error that occurs later in the table.

Table 24-13. Update-Key-Set Error Codes

Error Code	Ignorable	Description
functionNotConfigured	Yes	If the recipient is not configured for security on this port.
unknownEncryptionAlgorithm	Yes	If the Encryption Algorithm indicates an algorithm that the receiving device does not support or the value is outside the range of algorithms identifiers known to the device.
unknownKey	Yes	If the Key Identifier field indicates a security key that the receiving device does not know.
duplicateMessage	Yes	A message with the provided Message Id has already been received from the source device within the security time window.
unknownKeyRevision	Yes	If the Key Revision field indicates a revision that the receiving device does not know.
badSignature	Yes	If the signature is not correct.
badSourceAddress	No	If the source address information is missing or invalid
badDestinationAddress	Yes	If the destination address information is missing or invalid.
badTimestamp	No	The Timestamp in the security header of the message is not within the allowable timestamp window of the receiver.
encryptionRequired	No	If the Encrypted field is not set to 0,1.

24.3.6 Update-Distribution-Key

The Service Data for a Update-Distribution-Key message has the following form:

Table 24-14. Update-Distribution-Key Service Data

Message Field	Size	Description
Key Revision	1 octet	The Key Revision of the device's Distribution key.
Key	Variable	The new Distribution key.

This security message is used by the Key Server to provide Distribution keys to secure devices. This message shall always be signed and encrypted with the destination device's Device-Master key. The Key Revision field of the security header shall be ignored by the destination device (no revision is associated with the Device Master Key).

This message is sent either to initially configure a Distribution key into a new device, to update a Distribution key in an existing device, or to provide a Distribution key at the request of a device.

Upon receiving a valid Update-Distribution-Key message signed and encrypted with the device's Device-Master key, a device shall replace its Distribution key and respond with a Security-Response message containing a Response Code of Success.

Devices shall not restrict execution of this service based on the authentication mechanism; knowledge of the device's Device-Master key shall always be sufficient authorization.

The Key field in the message shall be of the form specified in Table 24-11. The correct key sizes by algorithm are as given in Table 24-12.

Broadcasts of this Message Type shall be ignored.

The possible error codes returned in response to an Update-Distribution-Key are listed in Table 24-15 below. The order in which the error codes are checked is important, and is specified by the order in the table. Implementations are allowed to ignore errors up to, and including, badSignature so as to hide from devices that scan for secure devices. If multiple error conditions are present, the condition that occurs first in the table shall be the error that is returned. Note that if multiple errors are present and the one that occurs first in the table is ignorable, then the device has the choice to either return that error, or not respond at all; the device does not have the option to return an error that occurs later in the table.

Table 24-15. Update-Distribution-Key Error Codes

Error Code	Ignorable	Description
functionNotConfigured	Yes	If the recipient is not configured for security on this port.
unknownKey	Yes	If the Key Identifier field indicates a security key that the receiving device does not know.
duplicateMessage	Yes	A message with the provided Message Id has already been received from the source device within the security time window.
badSignature	Yes	If the signature is not correct.
badSourceAddress	No	If the source address information is missing or invalid
badDestinationAddress	Yes	If the destination address information is missing or invalid.
badTimestamp	No	The Timestamp in the security header of the message is not within the allowable timestamp window of the receiver.
encryptionRequired	No	If the Encrypted field is not set to 0,1.

24.3.7 Request-Master-Key

The Service Data for a Request-Master-Key message has the following form:

Table 24-16. Request-Master-Key Service Data

Message Field	Size	Description
<i>No Parameters</i>		

This security message is broadcast by a secure device to request a Device-Master key from a Key Server. It is expected that this message is only used when initially configuring a secure device to work with the Key Server and that the issuance of the request is the result of a physical interaction with the device. If a secure device times out waiting for a Set-Master-Key message, the length of the timeout shall be sufficient to allow for human interaction with the Key Server in order to allow the Key Server to respond to the request.

As the secure device cannot produce a signature, the message shall not include a valid signature (the signature shall be all zeros) nor be encrypted. The Key Identifier field shall be set to 0. This service might not work if either the Key Server or the destination device is connected to a network that

requires encryption. Secure routers shall, by default, not drop these messages regardless of the network security policies although secure routers are allowed to be configured to drop these packets.

This service is inherently insecure and as such its use must be protected through safety precautions taken both by the implementers of secure BACnet devices and site setup personnel. The expectation is that site policy will dictate whether this service is to be used on a physically secure network, such as would be accomplished by attaching a BACnet device directly to the Key Server via a port dedicated for this purpose, or whether this service is allowed to be used on insecure networks during site setup.

A Key Server that receives this message, and is in a mode that allows it to respond to Request-Master-Key messages, or is instructed to respond by a user, shall generate a Set-Master-Key message in response. The expectation is that a Key Server will only be in a mode that allows a response to this message if the Key Server has been specifically placed into such a mode. The method for placing a Key Server into a mode that will support execution of the Set-Master-Key service is a local matter. Key Servers are required to support such a mode.

Secure BACnet devices that are not configured as Key Servers shall silently drop broadcasts of this message.

The possible error codes returned in response to a Request-Master-Key are listed in Table 24-17 below.

Table 24-17. Request-Master-Key Error Codes

Error Code	Ignorable	Description
functionNotConfigured	Yes	If the recipient is not configured with a Device-Master key.

24.3.8 Set-Master-Key

The Service Data for a Set-Master-Key message has the following form:

Table 24-18. Set-Master-Key Service Data

Message Field	Size	Description
Key	variable	The Device-Master key.

This security message is sent by a Key Server in response to a Request-Master-Key message. This message shall not be encrypted by the source device. The message shall be signed with the Device-Master key thus allowing values for all of security header fields.

Secure BACnet devices shall ignore messages of this type unless they are specifically placed into a mode in which they will accept these messages. It is recommended that secure BACnet devices restore themselves to factory defaults, excluding network communication parameters, when this service is executed so as to protect against theft of confidential information.

See the Request-Master-Key description in 24.3.7 for more details on the use of this service.

The Key field in the message shall be of the form specified in Table 24-11. The correct key sizes by algorithm are as given in Table 24-12.

Device shall not broadcast this message type.

24.4 Securing an APDU

When an APDU is to be sent securely, the APDU portion of the message is placed into the Payload parameter of a Security-Payload message.

Security is applied at the network layer by creating a new NPDU message type. Therefore, when a BACnet APDU is encapsulated with security information, it is transported as a network layer message so 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 resulting NPDU will once again indicate that it contains an APDU.

An example of a secured APDU follows.

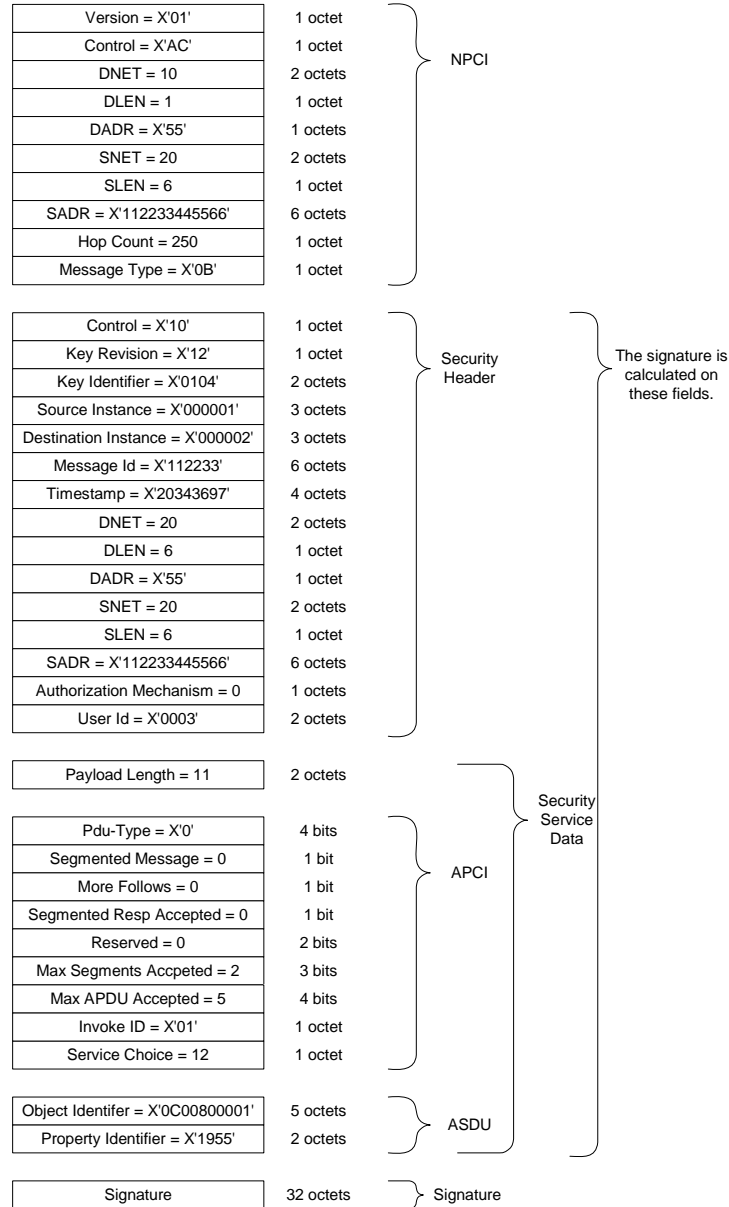


Figure 24-1. An example secured APDU (Read-Property).

24.5 Securing an NPDU

To secure an NPDU, the NSDU (NPDU payload starting with the Message-Type field) shall be placed into the Payload field of a Security-Payload message.

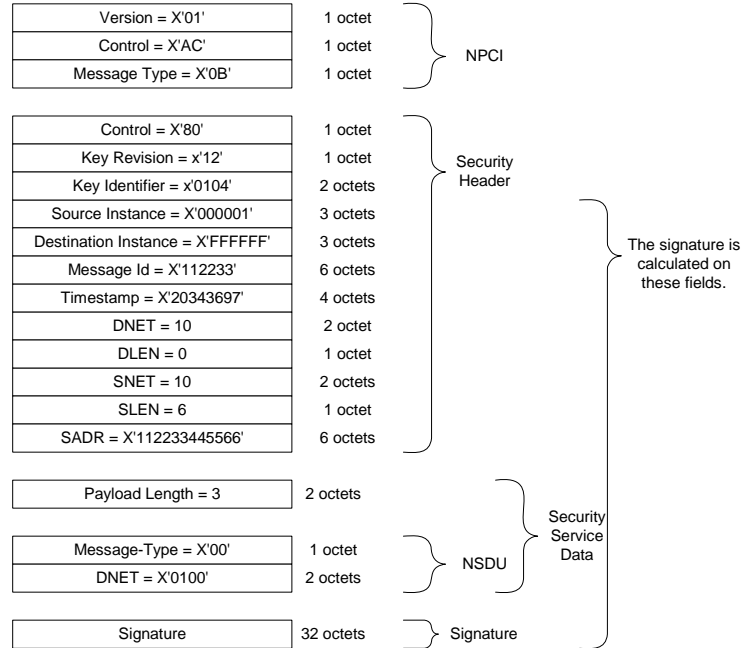


Figure 24-2. An example secured NPDU (Who-Is-Router-To-Network).

24.6 Securing a BVLL

BVLLs are divided into 2 groups: those that contain an NPDU, such as Original-Unicast-NPDU or Distribute-Broadcast-To-Network, and those that do not, such as Register-Foreign-Device or BVLL-Result.

To secure a BVLL message that does not contain an NPDU, the original BVLL shall be placed in the Service Data field of a Security Wrapper, and that Security Wrapper shall be used as the service data for a Secure-BVLL (BVLC Function X'0C'), message, as shown in Figure 24-3.

The ability to generate and consume Security Wrappers requires that the sender and receiver of secured BVLL messages are full BACnet devices with all the requirements thereof.

The Secure-BVLL message consists of the following fields:

Table 24-19. Secure-BVLL Message Fields

Field	Size	Description
BVLC Type	1-octet	BVLL for BACnet/IP (value = X'81')
BVLC Function	1-octet	Secure-BVLL (value = X'0C')
BVLC Length	2-octets	Length L, in octets, of the BVLL message
Security Wrapper	variable	As described in the Security Wrapper clause.

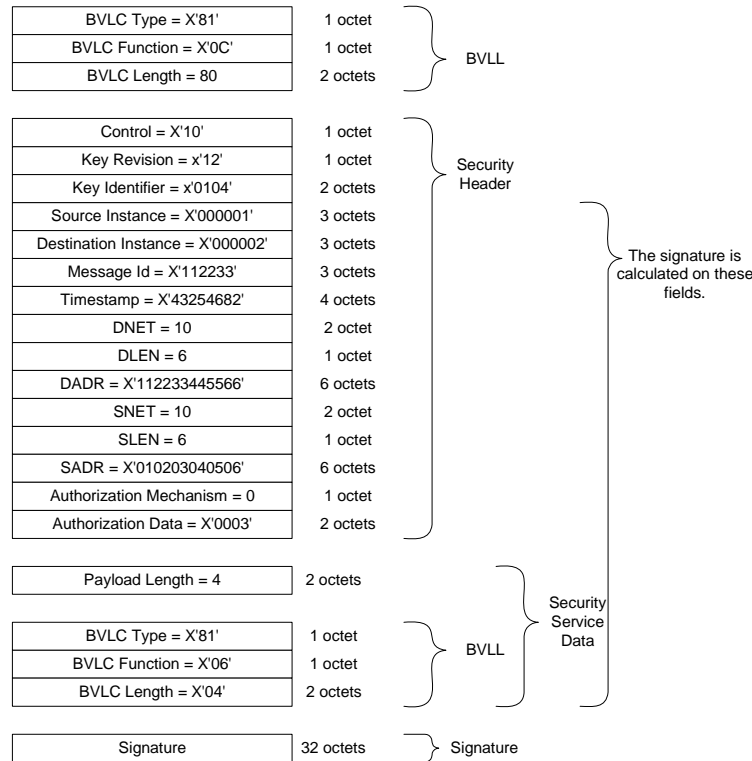


Figure 24-3. An example secured BVLL (Read-Foreign-Device-Table).

BVLL messages that contain an NPDU are transmitted without modification as specified in Annex J. However, there is no loss of security capability because those NPDUs may be secure NPDUs, containing their own Security Wrappers, based on the security policies of the network and sending device. If plain (non-secured) NPDUs are not desired in BVLL messages, then the network security policy of the BACnet/IP network should not be plain.

BBMDs do not perform wrapping/unwrapping functions on forwarded NPDUs the way routers do. The Network Security Policy that guides such operations in routers applies to an entire BACnet network, and BBMDs only serve to facilitate broadcasts between distant parts of a single network, which is governed by a single network security policy.

An example of a BVLL message containing a secure NPDU is shown in Figure 24-4.

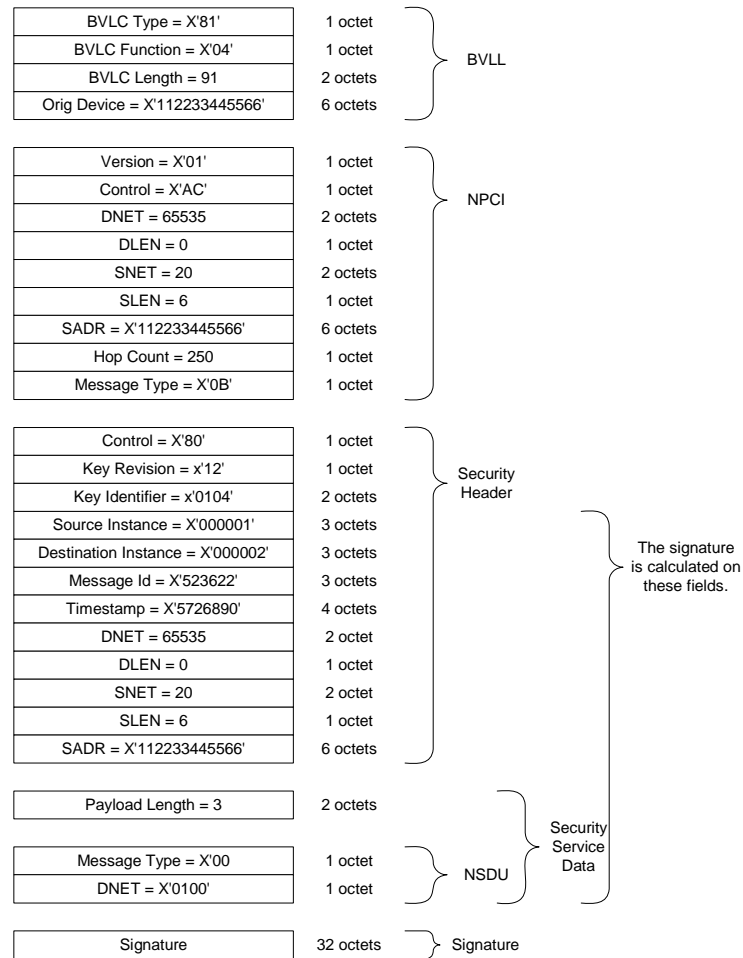


Figure 24-4. An example BVLL containing a Secured NPDU (Who-Is-Router-To-Network).

24.7 Securing Messages

The basic level of security that can be applied to a BACnet message consists of signing each message with an HMAC and of marking each message with the source and destination device instances, a Message Id and a timestamp.

24.7.1 Message Id

The security header contains a 24 bit Message Id which fulfills two 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.

To thwart replay, the Message Id is required to be unique across the security time window. This allows devices to track all received Message Id and Timestamp pairs over the security time window in order to detect replaying of messages.

24.7.2 Timestamp

The timestamp in the security header is used to detect the replay of messages.

The timestamp of a received message is compared to the device's local clock. If the timestamp is not within the security time window, then it is not accepted.

Timestamp along with Message Id provide variability in the message content so that messages that are repeated frequently do not generate the same signature.

24.7.3 Device Identification

All secure messages include the Source Device Instance, Destination Device Instance, SNET, SLEN, SADR, DNET, DLEN, and DADR fields in the security header. The inclusion of these values allows the security signature to be calculated on the source and destination device identities in order to stop redirection and identity switching attacks.

Requiring SNET in every secure message imposes the requirement that all secure BACnet devices know their network numbers.

When the device that applies the security wrapper does not know the device instance of the destination device, and is incapable of determining, or does not care to determine the device instance, the value 4194303 shall be used.

24.7.4 Message Signature

The signature included in all secure messages is computed using HMAC and a secure hash algorithm.

24.7.4.1 Secure Hash Algorithms

The first step in generating the signature is to generate a hash of the message using the secure hash algorithm specified by the security key. The hash shall be computed over the message contents starting with the initial octet of the security header and ending with the last octet of the Service Data field.

The signature is always calculated before encryption or after decryption. Before calculating the signature, the 'encrypted flag' of the control octet must be set to 0. Any padding required for encryption is not included in the input text of the signature.

A hash algorithm is considered "secure" because it is computationally infeasible either to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest. Any change to a message will, with a very high probability, result in a different message digest and thus verification failure.

The output of the secure hash algorithm will be used as the input to one of the keyed-hash message authentication code (HMAC) algorithms that follow.

24.7.4.1.1 MD5

The MD5 hash algorithm is defined by RFC1321. MD5 is included in the BACnet security framework as it is less computationally expensive than SHA-256. Sites whose security policies do not require SHA-256 will get better performance by using MD5. All secure BACnet devices shall support MD5 for use as a secure hash algorithm.

24.7.4.1.2 SHA-256

The SHA-256 hash algorithm is defined in NIST FIPS180-2. This algorithm is more computationally intensive than MD5, but is included in the BACnet security framework for use at sites that require it. All secure BACnet devices shall support SHA-256 for use as a secure hash algorithm.

24.7.4.2 HMAC

The HMAC algorithm is defined in section 5 of NIST FIPS198a. When performing this operation on a BACnet security message, the 'text' referred to below shall be the output of the secure hash algorithm

(256 bits for MD5 and SHA-256). To compute a signature of the data 'text' using the HMAC function, the following operation is performed:

$$\text{Signature}(\text{text})_t = \text{HMAC}(\mathbf{K}, \text{text})_t = \mathbf{H}((\mathbf{K}_0 \oplus \text{opad}) \parallel \mathbf{H}((\mathbf{K}_0 \oplus \text{ipad}) \parallel \text{text}))_t$$

where

- B** is the block size of the hash algorithm, which is 32,
- t** is the number of leftmost bytes of the output that will be used for the signature, which is 16,
- text** is the output of the secure hash algorithm applied to the message,
- H** is the hashing function (secure hash),
- K** is the signature key,
- K₀** is the padded key,
- ipad** is the inner pad, X'36' repeated 32 times,
- opad** is the outer pad, X'5C' repeated 32 times.
- \oplus is the XOR operation.

Table 24-20 illustrates the step-by-step process in the HMAC algorithm, which is depicted in Figure 24-5.

Table 24-20. The HMAC Algorithm

Step	Description
1	If the length of $K = B$: set $K_0 = K$. Go to step 4.
2	If the length of $K > B$: hash K to obtain an L byte string, then append $(B-L)$ zeros to create a B -byte string K_0 (i.e., $K_0 = H(K) \parallel 00\dots00$). Go to step 4.
3	If the length of $K < B$: append zeros to the end of K to create a B -byte string K_0 (e.g., if K is 20 bytes in length and $B = 64$, then K will be appended with 44 zero bytes 0x00).
4	Exclusive-Or K_0 with <i>ipad</i> to produce a B -byte string: $\mathbf{K_0} \oplus \text{ipad}$.
5	Append the stream of data 'text' to the string resulting from step 4: $(\mathbf{K_0} \oplus \text{ipad}) \parallel \text{text}$.
6	Apply H to the stream generated in step 5: $\mathbf{H}((\mathbf{K_0} \oplus \text{ipad}) \parallel \text{text})$.
7	Exclusive-Or K_0 with <i>opad</i> : $\mathbf{K_0} \oplus \text{opad}$.
8	Append the result from step 6 to step 7: $(\mathbf{K_0} \oplus \text{opad}) \parallel \mathbf{H}((\mathbf{K_0} \oplus \text{ipad}) \parallel \text{text})$.
9	Apply H to the result from step 8: $\mathbf{H}((\mathbf{K_0} \oplus \text{opad}) \parallel \mathbf{H}((\mathbf{K_0} \oplus \text{ipad}) \parallel \text{text}))$.
10	Select the leftmost t bytes of the result of step 9 as the HMAC.

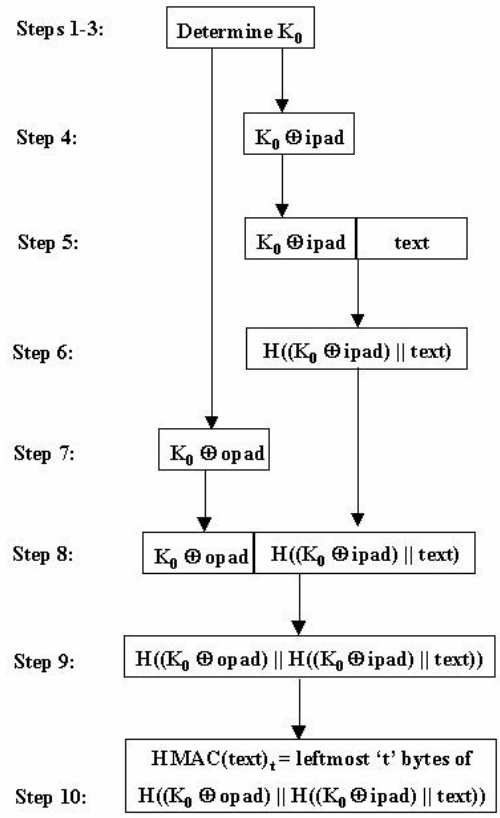


Figure 24-5. Diagram of the HMAC construction

24.7.5 Encrypted Messages

The BACnet security architecture allows for the encryption of a message's payload. The encryption of a BACnet message is done using AES. The definition of AES is contained in NIST FIPS 197.

When encrypting a BACnet message, the text to be encrypted starts with the Source Device Instance and includes all fields in the security wrapper up to and including the Padding field.

24.7.5.1 Cipher Block Chaining (CBC)

In order to encrypt a complete BACnet message, cipher block chaining (CBC) mode is used. The standard formula for CBC is:

$$C_i = E(K, P_i \oplus C_{i-1}) \quad \text{for } i = 1, \dots, k$$

where

- C_0 is the initialization vector,
- C_i is the i^{th} block of output,
- K is the encryption key,
- P_i is the i^{th} block of the portion of the message to encrypt.
- \oplus is the XOR operation.
- E is the encryption function.
- k is the number of blocks that make up the input message.

The initialization vector consists of the first B bytes of the message signature where B is the block size. For AES the block size is 16.

24.8 Network Security Network Trust Levels

There are 2 trust levels that describe the capabilities and security requirements of a device.

24.8.1 Trusted Networks

A trusted network consists of devices that are trusted either inherently, or because all communications are secured by the protocol.

When devices are inherently trusted, access to the devices and networks must be controlled. Depending on the installation policy, this may mean that there are no PCs and there are no direct network connections outside of locked space, or it may be that all devices are in locked cabinets and all network cabling is in metal conduit. The exact requirements will be determined on an installation-by-installation basis. Non-secured messages exchanged on these networks are trusted.

24.8.2 Non-trusted Networks

A non-trusted network is one where access to the network is not controlled, and as such no non-secured messages exchanged on the network can be trusted. Non-secured messages received from non-trusted networks must not be trusted. In order to identify such messages when they are routed onto trusted networks, the Non-Trusted-Source flag is set to 1. This allows devices to identify and optionally ignore or drop messages from non-trusted networks.

Secure BACnet routers shall be configurable to route non-secured messages from non-trusted networks onto trusted networks. The mechanism for enabling and disabling this feature is a local matter.

24.9 Network Security Policies

There are 4 standard network security policy levels. The following definitions are presented in increasing order of security policy level.

The policy level for a network specifies the minimum level of security required for messages on the network. It also specifies the default level of security for messages that are forwarded onto a network. A secure router will change the security on a message when it is forwarded onto a network with a different security policy if not restricted by the do-not-unwrap and do-not-decrypt flags.

24.9.1 Plain-Non-Trusted

Plain-non-trusted networks have no security requirements, and are not physically secure. Devices on these networks are allowed to generate and consume messages that are not signed. Secure devices on plain-non-trusted networks should assume that all plain messages are suspect and only execute requests contained in plain messages that are considered harmless.

The router that connects a plain-non-trusted network to a network with a different security policy must be a secure router. If the router's security policy allows it to route non-secured messages from the plain-non-trusted network onto a secure network, the router shall add the security wrapper and set the Non-Trusted-Source flag and the Do-Not-Unwrap flag.

24.9.2 Plain-Trusted

Plain-trusted networks have no security requirements but are physically secure. Devices on these networks are allowed to generate and consume messages that are not signed or encrypted. Devices on plain-trusted networks may assume that all plain messages are from sources that are allowed to communicate on the network.

Devices on plain-trusted networks should treat plain messages in the same manner as secure messages that use the General-Network-Access key.

24.9.3 Signed-Trusted

Signed-Trusted networks require that all messages be at least signed. All devices connected to signed-trusted networks must therefore be secure devices. All plain messages received on a signed-trusted network shall be silently dropped.

24.9.4 Encrypted-Trusted

Encrypted-trusted networks require that all messages be encrypted. All devices connected to encrypted-trusted networks must therefore be secure devices and support encryption. All non-encrypted messages received on an encrypted-trusted network shall be silently dropped.

24.10 Network Security

The BACnet security architecture allows for both secure networks and end-to-end security. Secure networks are created by setting the security policy for the network and configuring all devices on the secure network with the security policy of the network. This is accomplished by setting the `Base_Device_Security_Policy` property of the Network Security Settings object in all devices on the network to a level at or above the security policy for the network. This policy dictates the device's minimum level of security for sending or receiving messages.

In addition, each security enabled router shall contain a network policy table that defines the security policy for each directly connected network. Each entry in the network security policy table is a data pair of the form:

<network number, security policy>

Where the Base Device Security Policy differs from an entry in the network security policy table, the higher of the two values shall dictate the minimum level of security for the communication to or from that network.

In contrast, end-to-end security is determined on a device-by-device and request-by-request basis. A security enabled device may contain a more detailed security policy to guide end-to-end secure communications. This policy may require security based on any number of factors, such as the service being requested, the object or property being operated on, the source of the request, etc.

The limitations on device security policy are:

No secure devices shall require that requests be plain. If a device receives a well-formed valid secure request, the device is not allowed to return an error indicating that the message should have been sent plain.

No secure device connected to a network protected by a security proxy (see Clause 24.19.1) shall have a device policy that requires broadcast messages be secured.

If a network contains any incapable devices, then the local policy for the network must be 'plain-trusted' or 'plain-non-trusted'. If a network contains any devices that do not support encryption, then the local policy for the network must not be 'encrypted-trusted'.

The security policy of a network specifies the minimum security that messages must have if they are to be accepted from the network. A network that is physically secure might allow plain messages to be sent between local devices for efficiency reasons, but a router from that network might be configured to require signed or encrypted messages for communication beyond the local network.

24.11 End-to-End Security

In order to ensure that devices can determine the security expectations of peers, a response to a request shall use the same level of security as the request. For requests that result in a broadcast response, such as the unconfirmed application service Who-Is, the responding device is allowed to duplicate and send the response at other security levels as long as the other security levels do not violate the device's base local security policy or the network security policy over which the message is sent. The choice to send duplicates at other security levels is a local matter. Note that a device that is incapable of consuming a broadcast request due to its security level shall not respond in any way to the broadcast request.

24.11.1 Determining Exceptional Encryption Requirements

A device is allowed to require a higher security level on a per-service, per-object, or per-property basis.

Where the base device security policies and network security policies of communicating devices do not indicate the need for encryption, a secure device that is not configured to know ahead of time the sensitivity of a particular piece of data that is to be written to another secure device should attempt to read that data first before writing it without encryption. This is to allow the receiving device an opportunity to indicate that the data requires encryption by returning the error class SECURITY and error code ENCRYPTION_REQUIRED. Upon receiving this error, the client device now knows that the data should not be transmitted in the clear. If the sending device is configured to know ahead of time which data needs encryption, then pre-reading is not necessary.

24.12 Wrapping and Unwrapping Secure Messages

Messages can be secured either at the source device or en route in a router. Where the security is added or upgraded depends on the security policies and capabilities of the source device, destination device, and intervening networks.

When a router encrypts an already-signed secure message, the router shall pad and encrypt the message, and update the message header to indicate that the message is encrypted. The router shall not set the do-not-decrypt flag (except in the case of a security proxy device as defined in Clause 24.19.1).

When operating in tunneling mode, a router shall pad and encrypt the already-signed message using the General-Network-Access encryption key, and indicate the tunnelled encryption mode by updating the 'encrypted flag' (control bit 6) to 1 while leaving the 'do-not-decrypt flag' (control bit 3) as 0. If the received message is plain, the router shall sign with the General-Network-Access signature key prior to encrypting. The router shall not set the do-not-decrypt flag. If a received message is already encrypted, the tunneling router shall not decrypt or apply additional encryption. In this case, the control bits will not be changed and the encryption key in use is indicated by the Key Identifier field.

24.13 Authenticating Messages

Whenever security is removed from a message, either at the destination device or enroute through a router where the destination network policy differs from the source network, the message should be authenticated. The authentication process consists of validating:

- (a) source MAC address.
- (b) uniqueness of the Message Id.
- (c) timestamp.
- (d) message signature.
- (e) destination device instance (only checked by the destination device)

24.13.1 Validating the Source MAC Address

All devices that receive and process or forward a secure message have the option to validate the MAC address that the message was received from.

When validating the source MAC address of a message, there are 3 cases to consider:

- (a) The message originated from a device that resides on the network from which the message was received.
- (b) The message originated from a device that resides on a remote network.
- (c) The message does not contain source MAC address information in the security header.

When a message is received from a local device, the MAC address should be compared to the signed SADR in the security header. If the values match, then the source MAC address validation succeeded. If the values do not match, then the check failed and the receiving device has the option to challenge the source of the message. If the challenge succeeds, then the source MAC address validation will be considered to have succeeded.

When a message is received from a remote device, the MAC address should be compared against any locally cached value for a router to the source network. If the MAC address matches the cached value, then the source MAC address validation succeeded. If the values do not match or the device does not have a locally cached router address for the network, then the check failed and the receiving device has the option to challenge the router of the message. If the challenge succeeds, then the source MAC address validation will be considered to have succeeded.

To challenge a router to a network, a device may send a secure unicast Who-Is-Router-To-Network to the MAC address from the message. If a valid secure I-Am-Router-To-Network is received in response, then the challenge will have succeeded.

An alternative method to challenging the router is to challenge the originator of the message, ensuring that the MAC address sent to is the address the original message was received from. If the challenge succeeds, then the source device is reachable via the address the message was received from.

If the SLEN field of the security header is 0, then the source address information is not signed. The only option for validating the source address is to challenge the source device. If the challenge succeeds, then the source MAC address validation will be considered to have succeeded.

24.13.2 Validating the Destination Device Instance

Secure messages include a Destination Device Instance field that indicates the device that the message is intended for. A secure device shall only accept a message that has either a Destination Device Instance field of 4194303 or its own device instance.

24.13.3 Validating the Message Id

All secure devices that receive and process a secure message shall validate the Message Id of the message. Routers are not required to validate the Message Id of a message unless the router modifies or removes the security wrapper of the message, or the router is the final destination for the message.

All secure devices shall cache data records about recently received messages. The cached data records shall consist of the Message Id, the source device instance, and the timestamp. When a message is received, the cache shall be checked and if the Message Id is already present in the cache and was received within the current security time window, then the Message Id validation shall fail. If the Message Id validation, destination device instance validation, timestamp validation, and signature validation all pass, then a new record shall be added to the cache and the Message Id validation shall succeed.

The size of the cache is a local matter. Determining when to remove entries from the cache is a local matter.

24.13.4 Validating the Timestamp

All secure devices that receive and process a secure message shall validate the Timestamp of the message. Routers shall not validate the Timestamp of a message unless the router changes or removes the security wrapper of the message, or the router is final destination for the message.

The timestamp shall be checked against the clock of the receiving device. If the Timestamp is within the configured time window, then the validation shall succeed, otherwise the validation shall fail.

All secure devices shall have a default security time window of 3 minutes with a granularity of 1 second. The time window shall be configurable and shall support a range of at least 1 through 10 minutes.

24.13.5 Validating the Signature

All secure devices that receive and process a secure message shall validate the Signature of the message. Routers are not required to validate the Signature of a message unless the router changes or removes the security wrapper of the message, or the router is the final destination for the message.

The Signature is validated by calculating the signature as described in Clause 24.7.4. The validation succeeds if the calculated signature is the same as the Signature from the message.

24.14 Wrapping, Unwrapping, and Routing Messages

The application and removal of secure wrappers to/from BACnet messages occur at different points in a BACnet network. Each secure device along the communication path will evaluate the level of security required for the message and whether or not the message shall be forwarded along its route, or dropped. The rules shall be evaluated in the order presented, and only 1 rule shall be applied to a message.

When unwrapping messages, a router shall validate the signature and timestamp, and verify uniqueness of the Message Id across the timestamp window. If the signature or timestamp is invalid, or the Message Id is not unique then the message shall not be routed. In such a case, the router shall either remain quiet, or respond with an appropriate security error code as described in Clause 24.3.

Note: The rules governing wrapping and routing of messages are:

1. A router not configured for security processing (i.e., contains no configured security policy table, or does not support security NPDUs) shall route all unicast messages according to this standard's Clause 6 rules.

The remaining rules only apply to routers configured for security processing:

2. A message shall be dropped if any of the following are true:
 - a. The message is plain and was received on a network with a local policy of 'signed' or 'encrypted'.
 - b. The message is not encrypted and was received on a network with a local policy of 'encrypted'.
 - c. The message is signed or encrypted, the Non-Trusted-Source flag is set, and the message is to be placed onto a plain-trusted network.
3. A plain message shall be signed if the security policy of the outgoing port is 'signed'. The do-not-unwrap flag shall be set to 0. If the message was received on a non-trusted network, then the Non-Trusted-Source flag shall be set to 1.
4. A plain message shall be encrypted if the security policy of the outgoing port is 'encrypted'. The do-not-decrypt flag shall be set to 0. If the message was received on a non-trusted network, then the Non-Trusted-Source flag shall be set to 1.
5. A signed message shall be encrypted if the security policy of the outgoing port is 'encrypted'. The do-not-decrypt flag shall be set to 0.
6. A signed message shall be unwrapped if the security policy of the outgoing port is 'plain' and the do-not-unwrap flag is set to 0.
7. An encrypted message shall be decrypted and unwrapped if the security policy of the outgoing port is 'plain' and both the do-not-decrypt and do-not-unwrap flags are set to 0.

8. An encrypted message shall be decrypted, but not unwrapped, if the security policy of the outgoing port is not 'encrypted' and the do-not-decrypt flag is set to 0.
9. All other messages shall be forwarded as is.

24.14.1 Unwrapping Security Errors

When a router receives a security error that requires routing and the router has determined that the security wrapper should be removed before routing the message on, the router shall generate and forward a Reject-Message-To-Network message to the destination device. The reject reason shall be code 5 indicating that a security error stopped the message from being processed by the ultimate destination device.

24.14.2 Securing Response Messages

When a device responds to a secure request, whether it is a network layer request, an application layer request, or a BVLL request, the response shall be secured using the same key and to the same level (signed or encrypted) as the request, when possible. This requires that security information associated with a request be passed along with the request as it moves between layers of the protocol so that it can be used to create a response that matches the request. The only exception to this requirement is when the responder is unable to provide the same level of security, such as occurs when reporting certain security errors back to the requestor.

24.15 User Authentication

The BACnet standard provides a simple user authentication mechanism. It is expected that future versions will provide additional authentication mechanisms that employ standard IT user authentication technologies.

24.15.1 Proxied User Authentication

The Proxied Authentication mechanism is identified by an Authentication Mechanism field value of 0. When using this mechanism, the User Id field shall contain a 2 octet User Id. The User Id in the message is assumed to have been authenticated by the source device when the key is anything other than the General-Network-Access key. See also 24.2.1.11.

The User Id identifies the user or user group requesting the operation. Secure devices shall not expect to use a fixed set of User Ids. User Ids will be assigned during site installation and setup and as such shall be configurable in both client and server devices. Devices that are capable of providing User Id values in secure messages shall be capable of being configured to provide any User Id value for each of the users or user groups that it supports. Devices that do not ignore User Id values in received messages shall be capable of being configured to accept and process any User Id value. The method used to configure User Ids and authorization rules is a local matter.

A User Id of 0 shall indicate that a request is not initiated by a human user, that no user authentication was performed, and no user authentication was required.

Proxied user authentication information is not useful in responses or transmission control. Proxied user authentication will be useful in any PDU type that initiates a request such as Confirmed-Request, Unconfirmed-Request, Initialize-Routing-Table, Establish-Connection-To-Network, Write-Broadcast-Distribution-Table, Read-Broadcast-Distribution-Table, Register-Foreign-Device, Read-Foreign-Device-Table, and Delete-Foreign-Device-Table-Entry PDUS. It shall be ignored in all other PDU types. 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.

A server device that receives a request with a User Id that is secured with the User-Authenticated key is allowed to apply local user authorization to determine whether the user is authorized to perform the requested service. If the user does not have sufficient rights for the operation, the server device shall return an error class of SECURITY and an error code of ACCESS_DENIED. Note that the error is

returned by the application layer in a Complex-ACK-PDU or an Error-PDU; the error is not returned in a Security-Response message.

If a server is not configured to apply authorization rules, then the user authentication information may be ignored. If a server is configured to apply authorization rules and the message is secured with the General-Network-Access key, it is a local matter as to whether or not the request is granted. Exceptions to this rule are the Who-Is and Who-Has services that shall be executed.

The authorization scheme details and the method of configuring and maintaining the scheme, are a local matter.

24.16 Time Synchronization Requirements

Because message timestamps are used to prevent replay attacks, secure devices are required to maintain the current time. Also, because message timestamps are in UTC, secure devices are required to support the UTC_Offset property in their Device object.

24.16.1 BACnet Time Synchronization Messages

Secure BACnet devices should not accept non-secured TimeSynchronization or UTCTimeSynchronization requests, unless the local network security policy is plain-trusted. Doing so will leave the secure device vulnerable to replay attacks.

24.16.2 Overcoming Non-synchronized Clocks

The network security architecture relies on devices having loosely synchronized clocks. If there are secure devices within the system whose clocks are not within the time window of each other, then those devices will be unable to communicate with each other.

24.16.2.1 Clock Offsets

One method of overcoming this problem is for devices to maintain clock offsets for peer devices with which they communicate. When a message sent to a peer device fails with a badTimestamp error, the source of the secure message will be informed of the actual clock value of the other device. From this clock value an offset representing the difference in the clocks can be saved and added to the timestamp in any message that is subsequently sent to that device.

24.16.2.2 Devices Without Real Time Clock Chips

Devices that do not keep time over a reset or while turned off will need to acquire the time before they can communicate safely. Each of the methods outlined requires that the device have the ability to generate a random value that will be used in Message Id and/or Timestamp fields of the security header. Use of standard programming language random functions will not meet this requirement. The device must use its interaction with the outside world to generate a random value (inter-message delay, total network traffic, values found in DRAM if it is not cleared on startup, etc.). If a predictable Message Id counter value is used on startup, then an attacker can record a sequence of messages when a device resets and then replay them at a later date when the device resets again.

24.16.2.3 Monitoring Broadcast Traffic

The device can determine the time by monitoring the network for broadcasts. If a broadcast is seen that is secure and validates, then the time can be taken from the message and used in a Challenge-Request containing a random Message Id and aimed at the source of the message. If the challenge succeeds, then the time can be accepted.

24.16.2.4 Monitoring For Any Traffic

The device can determine the time by monitoring the network for any traffic. Once a message is received, the source MAC address can be removed from the message and a Challenge-Request formed and sent to the source of the message. If the challenge succeeds, then the time can be accepted.

24.16.2.5 Wait For TimeSynchronization

The first two solutions may result in incorrect times being used by the device. If the device chooses a message from a device that has an invalid time, then the device will end up with the same invalid time.

A third option is for the device to be configured to challenge a time master on restart, or be programmed to remember the source address of the last successfully processed TimeSynchronization or UTCTimeSynchronization request and challenge this address on start up. The device should update its time only if the Challenge succeeds.

Alternatively, the device could wait for a TimeSynchronization or UTCTimeSynchronization request. When one is received, the device should challenge the message source and only accept the time if the Challenge succeeds

24.17 Secure PDU Sizes

The addition of the BACnet security layer into the BACnet PDU reduces the amount of space for BACnet NSDUs and APDUs.

For secure devices, the value of the Max_APDU_Length_Accepted property of the Device object shall be calculated assuming the largest size of valid Authentication Data the device is configured to accept and the largest encryption block size of any BACnet security encryption algorithm the device is configured to use.

In order to reduce the chance of inverted networks being used when security tunnels are in place, the maximum APDU for secure BACnet/IP segments is kept at 1476. To allow for this, the maximum length BACnet/IP NPDU is increased to 1562 for secure BACnet/IP messages. This allows full size non-secured APDUs to be secured and passed through BACnet/IP networks.

The following table of maximum NSDU and APDU lengths is calculated assuming an Authentication Mechanism of 1 with 2 octets of Authentication Data and an encryption block size of 16 octets.

Table 24-21. Maximum APDU Lengths for Secure BACnet Data Link Layers

Data Link Technology	Maximum APDU Length
ISO 8802-3 ("Ethernet"), as defined in Clause 7	1420 octets
ARCNET, as defined in Clause 8	412 octets
MS/TP, as defined in Clause 9	412 octets
Point-To-Point, as defined in Clause 10	412 octets
LonTalk, as defined in Clause 11	140 octets
BACnet/IP, as defined in Annex J	1476 octets

When reporting the Max_APDU_Length_Accepted in the APCI of a ConfirmedRequest-PDU, a secure device might be forced to indicate a value smaller than indicated by its Device object. While this will result in successful communications, the network bandwidth usage of large segmented messages will be sub-optimal. Therefore devices responding to ConfirmedRequest-PDUs may ignore the value indicated in the APCI and use the value indicated in the client’s Device object instead, if it is known.

24.18 BACnet Security In A NAT Environment

A secure foreign device that resides behind a NAT cannot include a correct SADR in secure messages because the device does not know it. While BBMDs are given static IP addresses, a foreign device's address is not usually static. To overcome this issue, secure foreign devices that communicate through a NAT to reach the BACnet internetwork may first send a Challenge-Request message to the BBMD

that they intend to register with as a foreign device. The resulting Security-Response will let them know their SADR as seen by other BACnet devices.

24.19 BACnet Firewalls

The BACnet security extensions allow for the implementation of special BACnet routers that act as firewalls within the BACnet network. There are numerous techniques that can be employed. A few examples are described in the following clauses.

24.19.1 Security Proxy

In order to provide security for devices without requiring the devices themselves to be secure, a new type of device called a Security Proxy is defined.

A BACnet security proxy device is a secure router that strips and adds security on behalf of the devices "behind" it that are incapable of, or not configured for, secure communications. The proxy will remove security from messages destined for the protected devices, even if the do-not-unwrap or do-not-decrypt flags are set. The proxy will also determine when security should be applied to messages that are originated by protected devices and will apply it accordingly. The methods used by a security proxy device to determine when to apply security, what level of security, and what user information to use are a local matter.

In its simplest form, a security proxy device protects a complete BACnet network or collection of BACnet networks but it is not restricted to operate in such a manner. A security proxy device could be implemented such that it protects a subset of the devices on the protected networks.

The BACnet security proxy functionality is optional and is not required to be supported by secure BACnet routers.

24.19.2 Network Filtering

Security in some network configurations will benefit from enhanced routers that filter traffic based on the security settings within the secure messages. Where a BAS network is connected to the Internet or to a building's PC LAN, a BACnet firewall that filters out certain message types can be used to reduce the threats to the BAS network.

24.19.2.1 Filtering Messages with Invalid Signatures

A secure router that authenticates all messages coming from a physically insecure network (such as the Internet) will help to stop Denial Of Service attacks consisting of well formed BACnet secure messages that have incorrect signatures. Such a router would be configured to authenticate each message received from the Internet in the same manner as the final destination device would authenticate the message. This allows the router to drop invalid messages from the physically insecure network before they get to the destination control devices, or the control networks. Doing so reduces the network load when such a Denial Of Service attack is underway.

While the secure router may be overcome by the network load, all of the devices behind the router will be spared the task of authenticating the messages generated by the attacker. Communications through the router may be stopped by the attack, but the BACnet devices behind the device are able to continue to communicate with each other and to perform their control actions.

24.19.2.2 Filtering Messages Based On Authentication Information

An advanced BACnet router that examines BACnet messages in detail would be capable of applying authorization restrictions on traffic destined for other devices. Such a device would be capable of restricting requests destined for devices behind it based on the received authentication information and the request's contents.

Such a device would analyze the contents of messages being transmitted to devices behind it and intercept those that do not have sufficient authorization based on the user authentication information contained in the security header. A BACnet firewall of this form that performs access decisions based solely upon the Service Choice could be implemented as a simple filter firewall. In contrast, a BACnet firewall that analyzes the contents of messages so as to restrict requests based on the object or property being accessed would almost certainly have to be a BACnet-to-BACnet gateway so as to allow it to pass on some parts of requests, and not others.

24.20 Security Keys

In BACnet security there are six types of keys: General-Network-Access, User-Authenticated, Installation, Application-Specific, Device-Master, and Distribution. Each key actually consists of a pair of key values, one used for signatures and one used for encryption.

The General-Network-Access key is used for device and object binding, for encryption tunnels, and by user interface devices that cannot authenticate or are not trusted to authenticate a user. All secure BACnet devices shall support use of the General-Network-Access key.

The User-Authenticated key is used by devices that are allowed to authenticate the user's identity that is included in BACnet messages. It is also given to devices that do not contain a user interface. All secure BACnet devices shall support use of the User-Authenticated key.

Installation keys are distributed to pairs 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 should not normally have access to the BACnet network. All secure BACnet devices shall support use of the Installation key.

In order to provide security boundaries between application areas, such as access control and HVAC, the BACnet security framework provides Application-Specific keys. Support for the Application-Specific keys is optional. The semantics of the Application-Specific keys are site-specific, and as such all devices that support them shall be capable of communicating with at least one of the Application-Specific keys.

The Device-Master key is a unique key per device that is either given to the device before the device is installed, or provided to the device by the Key Server during installation. In theory, the Device-Master key is never changed for a device other than during initial site setup. In practice, a device's master key will have to be changed if a Key Server's key database is lost and cannot be recovered. The Device-Master key is only used to provide Distribution keys to devices.

Distribution keys are unique per device and are used only to distribute key sets.

24.20.1 Key Identifiers

Keys are identified by their 2 octet key identifier. The identifier indicates which key is to be used and the signature and encryption algorithms that the key is used with.

The first octet of the Key Identifier field indicates the encryption and signature algorithms to be used. The values are:

Table 24-22. Key Identifier Algorithm Enumeration

Value	Algorithm (Encryption/ Signature)
0	AES / MD5
1	AES / SHA-256
2..255	Reserved

The second octet of the Key Identifier indicates the key being used. The values are:

Table 24-23. Key Identifier Key Number Enumeration

Value	Key number
0	(not used)
1	Device-Master
2	Distribution
3	Installation
4	General-Network-Access
5	User-Authenticated
6..127	Application-Specific Keys
128..255	Reserved

While the key identifier format would allow the specification of multiple keys of each type (such as the General-Network-Access key) differing only in the algorithms used, the intent is that only one key of each type would be used in practice.

24.20.2 Key Sets

A key set consists of all of the keys that a BACnet device is configured to have, excluding the Device-Master key and the Distribution key, a time period during which the keys are valid, a second time period during which the Installation key is valid and a key revision number. The key revision number applies to all of the keys in a key set.

A key shall not be used outside of the timeframe defined for the key set. To allow for variations in clocks between devices, a key's acceptability shall be determined by using the timestamp placed in the message by the sending device rather than the local time in the receiving device.

Each BACnet device has 2 key sets with possibly overlapping valid time periods. This allows a device to have a current and a new key set for transitioning between key sets.

24.20.3 Key Distribution

In general, the keys used by the BACnet security framework are not unique to each device, or device pair. The General-Network-Access key is shared by all of the BACnet devices and the User-Authenticated key is shared by most devices. In order to increase the security of the keys, they should be changed periodically.

Device-Master keys must be configured in a secure device and shared with the Key Server before the Key Server can provide keys to a device. Initial key distribution is discussed in more detail in Clause 24.21.3.

Keys are distributed as a set, excluding the Distribution key, which is distributed on its own. The key revision number applies to all of the keys in the key set. Each time a new set of keys is generated, the key revision number is incremented.

To protect the keys, the distribution messages shall be encrypted. The encryption algorithm used for distribution of keys shall be the same encryption algorithm as used with the most secure key in the distribution.

24.21 Key Server

To automatically update security keys, each BACnet installation will require a Key Server that is responsible for generating and distributing keys to all of the BACnet devices.

The Key Server is configured with a list of the devices to update, the keys that each should be given, and the period at which the keys should be distributed. In addition, the Key Server shall allow the operator to initiate a key update at any time.

24.21.1 Key Generation

The Key Server will be responsible for generating all keys, except for factory-fixed Device-Master keys. The algorithm used to generate the keys shall be a local matter.

24.21.2 Distribution Method

When the Key Server has generated a new set of keys, the Key Server will increment the Key Revision and distribute the keys to each device. Each device receives the General-Network-Access key plus any other keys it has been authorized to receive. The set of keys are sent encrypted to the device using the Distribution key for that device. Distribution keys are delivered separately using the Device-Master key and need not be distributed as frequently as the key sets.

The Key Server packages both the device's current key set and the newly generated key set into an Update-Key-Set message, signs and encrypts the message with the device's Distribution key, and sends it to the device. Upon receipt of a valid Update-Key-Set message, the device shall update both of its key sets, and acknowledge the message with a Security-Response message containing response code Success. The device shall start using the new key set as soon as its valid time period will allow.

If the sending of the set of keys to the device is successful, the Key Server shall save the key set for the device. If the Key Server is unable to update the device, it shall re-try the update periodically until successful.

The Key Server shall update each device. If a device cannot be contacted, or the update fails, the Key Server shall continue with the next device. The order in which the Key Server updates devices is a local matter.

If both of a device's key sets become invalid due to the current time being outside each key set's valid time period, then the device shall cease generating requests, other than Request-Key-Update messages signed with the device's Distribution key, or Device-Master key. The device shall periodically request a new key set from the Last_Key_Server. If the Key Server is on a remote network and the device does not know the MAC address of the router to use, then the device shall use a local MAC broadcast to transmit the Request-Key-Update message. If the device does not have a value for Last_Key_Server, or does not receive an update from the Last_key_server, the device shall globally broadcast the Request-Key-Update message. The device shall not request a key set more than once every 5 minutes. If, upon power up, a device detects that its key sets have expired, the device shall wait at least 1 minute after power up before requesting a key set so as to allow the Key Server to distribute key sets after power outages.

When the Key Server receives a Request-Key-Update from a device where the Key Revision of the message's Distribution key does not match the recorded Distribution key for the device, the Key Server shall respond with an error code of unknownKeyRevision and then update the device's Distribution key set with an Update-Distribution-Key message.

For devices that the Key Server is unable to provide a key set to, the Key Server shall periodically attempt to update the device using the Update-Key-Set message. The period at which the Key Server retries the key update is a local matter.

The Key Server shall periodically update each device's Distribution key. To update a device's Distribution Key, the Key Server sends the device an Update-Distribution-Key message that is signed and encrypted with the device's Device-Master key.

If a device loses its keys, it can request a new set by sending the Key Server a Request-Key-Update message signed with the device's Device-Master key. To such a request, the Key Server shall respond with an Update-Distribution-Key message.

24.21.3 Initial Key Distribution

A Key Server shall provide a mechanism for accepting manually input Device-Master keys. For example, the Key Server might have a user interface through which the Device-Master key values can be entered, or the Key Server may be provided with a software tool that will provide keys to the Key Server through a secure channel. This mechanism allows for devices that have a fixed, factory configured, Device-Master key. All such devices shall provide a method of reporting the Device-Master key. For example, a device could have the Device-Master key printed on a tear-off label.

As an alternative to having predefined Device-Master keys, secure devices may optionally support initiation of the Request-Master-Key and execution of the Set-Master-Key messages. It is left up to site policy as to whether or not the use of these inherently insecure services is allowed during site setup.

24.21.4 Multiple Key Servers

When there are multiple Key Servers in a BACnet installation, the method by which the Key Servers generate new key sets, distribute keys between themselves and determine which Key Server distributes keys to which devices is a local matter. In addition, the method for selecting which Key Server responds to any particular Request-Master-Key service, and the subsequent sharing of generated Device-Master keys between the Key Servers is a local matter.

[Add the following new definitions to **Clause 3**, pp. 1-5, inserting them in appropriate alphabetical positions and renumbering subsequent clauses]

...

3.1.X1 authentication: the act of verifying identity

...

3.1.X2 authorization (network security): the control of access to network resources based on known identity and access rules

...

3.1.X3 encrypted message: a message that is wrapped in a security header, signed, and encrypted.

...

3.1.X4 incapable device: a device that is inherently incapable, or has been configured to appear to be incapable, of producing or consuming secure BACnet messages. All incapable devices are plain devices.

...

3.1.X5 - inverted network: a BACnet internetwork where two or more networks are connected by a network with an NPDU size smaller than the networks it joins.

...

3.1.X6 physically insecure: not physically secure.

...

3.1.X7 plain device: a device that does not normally produce or consume secure BACnet messages. All incapable devices are plain devices. However, a plain device that is not an incapable device is capable of producing or consuming secure BACnet messages when communicating with another device that requires it.

...

3.1.X8 plain network: a network that does not require signed or encrypted traffic.

...

3.1.X9 plain message: a message that is not secured by a BACnet security wrapper.

...

3.1.X10 physically secure: a device or network that is protected from physical access by unauthorized individuals.

...

3.1.X11 signed message: a message that is wrapped in a security header, signed, and not encrypted.

...

3.1.X12 secure network: a network on which all traffic is required to be signed or encrypted.

...

3.1.X13 trusted: a term that refers to devices or networks from which messages are believed to be authentic, based either on the use of secure messages or on the physical security of that device or network.

...

[Change clause 6.2.4, p. 52]

6.2.4 Network Layer Message Type

If Bit 7 of the control octet described in 6.2.2 is 1, a message type octet shall be present as shown in Figure 6-1. The following message types are indicated:

X'00': Who-Is-Router-To-Network
X'01': I-Am-Router-To-Network
X'02': I-Could-Be-Router-To-Network
X'03': Reject-Message-To-Network
X'04': Router-Busy-To-Network
X'05': Router-Available-To-Network
X'06': Initialize-Routing-Table
X'07': Initialize-Routing-Table-Ack
X'08': Establish-Connection-To-Network
X'09': Disconnect-Connection-To-Network
X'0A': *Challenge-Request*
X'0B': *Security-Payload*
X'0C': *Security-Response*
X'0D': *Request-Key-Update*
X'0E': *Update-Key-Set*
X'0F': *Update-Distribution-Key*
X'10': *Request-Master-Key*
X'11': *Set-Master-Key*
X'12': *What-Is-Network-Number*
X'13': *Network-Number-Is*
X'0A14' to X'7F': Reserved for use by ASHRAE
X'80' to X'FF': Available for vendor proprietary messages

Figures 6-5 through 6-10 provide examples of NPDUs containing network layer messages.

[Change clause 6.4.4, p. 54]

6.4.4 Reject-Message-To-Network

This message is indicated by a Message Type of X'03' followed by an octet indicating the reason for the rejection and a 2- octet network number (see Figure 6-7). It is directed to the node that originated the message being rejected, as indicated by the source address information in that message. The rejection reason octet shall contain an unsigned integer with one of the following values:

0: Other error.

- 1: The router is not directly connected to DNET and cannot find a router to DNET on any directly connected network using Who-Is-Router-To-Network messages.
- 2: The router is busy and unable to accept messages for the specified DNET at the present time.
- 3: It is an unknown network layer message type.
- 4: The message is too long to be routed to this DNET.
- 5: *The source message was rejected due to a BACnet security error and that error cannot be forwarded to the source device. See Clause 24.14.1 for more details on the generation of Reject-Message-To-Network messages indicating this reason.*

[Insert new clauses **6.4.11** through **6.4.20**, p. 56]

6.4.11 Challenge-Request

This message is indicated by a Message Type of X'0A'. It is described in Clause 24.

6.4.12 Security-Payload

This message is indicated by a Message Type of X'0B'. It is described in Clause 24.

6.4.13 Security-Response

This message is indicated by a Message Type of X'0C'. It is described in Clause 24.

6.4.14 Request-Key-Update

This message is indicated by a Message Type of X'0D'. It is described in Clause 24.

6.4.15 Update-Key-Set

This message is indicated by a Message Type of X'0E'. It is described in Clause 24.

6.4.16 Update-Distribution-Key

This message is indicated by a Message Type of X'0F'. It is described in Clause 24.

6.4.17 Request-Master-Key

This message is indicated by a Message Type of X'10'. It is described in Clause 24.

6.4.18 Set-Master-Key

This message is indicated by a Message Type of X'11'. It is described in Clause 24.

6.4.19 What-Is-Network-Number

This message is indicated by a Message Type of X'12'. It is used to request the local network number from other devices on the local network. This message may be transmitted with a broadcast or a unicast address. This message shall never be routed.

If the message is generated on a secure network that requires the inclusion of the network address fields SNET and DNET, then the SNET and DNET fields shall be set to 0 and Hop Count shall be set to 1.

Upon receipt of a What-Is-Network-Number message, a device that knows the local network number shall transmit a local broadcast Network-Number-Is message back to the source device. Any SNET, DNET, DLEN, DADR and Hop Count fields in the What-is-Network-Number message shall be ignored. If the What-Is-Network-Number message was broadcast, then a non-routing device may

optionally wait for up to 10 seconds before sending the Network-Number-Is message. If during that time, a different device broadcasts the Network-Number-Is message, the non-routing device may choose to not send a Network-Number-Is message.

Upon receipt of a What-Is-Network-Number message, a device that does not know the local network number shall discard the message.

A device shall cache its local network number and not repeatedly issue this service.

6.4.20 Network-Number-Is

This message is indicated by a Message Type of X'13' followed by a 2-octet network number (most significant octet first), followed by a 1-octet flag, where a value of 1 indicates that the network number was configured, and a value of 0 indicates that the network number was learned by receipt of a previous Network-Number-Is message. This message is used to indicate the local network number to other devices on the local network. It shall be transmitted with a local broadcast address and shall never be routed.

For a device that has not been configured to know its network number, when it receives this message indicating a configured network number, it shall set its network number from this message. If it receives this message indicating a learned network number, then it shall set its network number only if it has not previously received a message indicating a configured network number. If a device resets, it shall reduce the quality of its last received network number to “learned”.

For a device that has been configured to know its network number, when it receives this message indicating a configured network number that is in conflict with its configuration, it should report the conflict to a local or remote management entity.

Upon startup, routers that have been configured to know their network numbers shall broadcast out each port a Network-Number-Is message containing the network number for the port, and indicate that this number is configured, not learned.

[Change 12.11.17, p 180]

12.11.17 Max_APDU_Length_Accepted

This property, of type Unsigned, is the maximum number of octets that may be contained in a single, indivisible application layer protocol data unit. The value of this property shall be greater than or equal to 50. The value of this property is also constrained by the underlying data link technology. See Clauses 6 through 11.

If the value of this property is not encodable in the 'Max APDU Length Accepted' parameter of a ConfirmedRequest-PDU, then the value encoded shall be the highest encodable value less than the value of this property. In such cases, a responding device may ignore the encoded value in favor of the value of this property, if it is known.

[Insert new **Clause 12.X**, after p. 251]

12.X Network Security Object Type

The Network Security object type defines a standardized object whose properties represent the externally visible network security settings and status of a BACnet Device. Secure BACnet devices shall contain exactly one Network Security object, and it shall have an instance of 1. A detailed description of BACnet security and secure BACnet devices can be found in Clause 24.

Network Security object type and its properties are summarized in Table 12-24 and described in detail in this subclause.

Table 12-Y. Properties of the Network Security Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Base_Device_Security_Policy	BACnetSecurityLevel	W
Network_Security_Policies	List of BACnetNetworkSecurityPolicy	O
Security_Time_Window	Unsigned	W
Distribution_KeyRevision	Unsigned	R
Key_Sets	BACnetArray[2] of BACnetSecurityKeySet	R
Last_Key_Server	BACnetAddressBinding	R
Profile_Name	CharacterString	O

12.X.1 Object_Identifier

This property, of type BACnetObjectIdentifier, is a numeric code that is used to identify the object. It shall be unique within the BACnet Device that maintains it.

12.X.2 Object_Name

This property, of type CharacterString, shall represent a name for the object that is unique within the BACnet Device that maintains it. The minimum length of the string shall be one character. The set of characters used in the Object_Name shall be restricted to printable characters.

12.X.3 Object_Type

This property, of type BACnetObjectType, indicates membership in a particular object type class. The value of this property shall be NETWORK_SECURITY.

12.X.4 Description

This property, of type CharacterString, is a string of printable characters whose content is not restricted.

12.X.5 Base_Device_Security_Policy

This writable property, of type BACnetSecurityLevel, specifies the minimum level of security that the device requires allowing client devices to know the level of security to use when communicating with the device.

While devices may require higher security levels for some operations, this property shall be readable using the security level defined by this property.

12.X.6 Network_Security_Policies

This optional property, of type List of BACnetNetworkSecurityPolicy, specifies the security policy for each network directly connected to the device.

This property shall be readable via the base security level for the device.

12.X.7 Security_Time_Window

This writable property, of type Unsigned, specifies the security time window for the device in seconds. The recommended default value for this property is 180 (3 minutes). The property shall be restricted to the range 1 through 600.

12.X.8 Distribution_Key_Revision

This read only property, of type Unsigned, identifies the device's Distribution key revision. This property shall be 0 if the device does not have a Distribution key.

12.X.9 Key_Sets

This read only property, of type BACnetArray of BACnetSecurityKeySet, describes the contents of the device's 2 key sets. The actual key values are not included in the contents of this property.

12.X.10 Last_Key_Server

This read-only property, of type list of BACnetAddressBinding, identifies the device identifier and address of the last Key Server that successfully updated a security key in the device.

12.X.11 Profile_Name

This optional property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name must begin with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here. This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.

[Add the following new productions to **Clause 21**, p. 369, in proper alphabetical position]

...

```
BACnetNetworkSecurityPolicy ::= SEQUENCE {  
    network-number [0] Unsigned,  
    security-level [1] BACnetSecurityLevel  
}
```

...

```
BACnetSecurityKeySet ::= SEQUENCE {  
    key-revision [0] Unsigned, -- 0 if key set is not configured  
    activation-time [1] BACnetDateTime, -- UTC time, all wild if unknown  
    expiration-time [2] BACnetDateTime, -- UTC time, all wild if infinite  
    key-ids [3] SEQUENCE OF Unsigned8  
}
```

```
BACnetSecurityLevel ::= ENUMERATED {  
    incapable (0), -- indicates that the device is configured to not use security  
    plain (1),  
    signed (2),  
    encrypted (3)  
}
```

[Change **Table 12-13**, p. 178]

Table 12-13. Properties of the Device Object Type

Property Identifier	Property Datatype	Conformance Code
...
List_Of_Session_Keys	List of BACnetSessionKey	Θ
...

[Delete clause **12.11.29**, p. 181, and renumber subsequent clauses]

~~12.11.29 List_Of_Session_Keys~~

~~This property is a List of BACnetSessionKey each of which is one of the cryptographic keys used to communicate with other security conscious BACnet Devices. This property shall not be readable or writable by any device except a device designated the "Key Server." A session key shall consist of a 56-bit encryption key and a BACnet Address of the peer with which secure communications is requested.~~

[Change Clause **18**, pp. 355-356]

...
18.5 Error Class – SECURITY

This Error Class pertains to problems related to the execution of security services. Without exception, these errors signal the inability of the responding BACnet-user to carry out the desired service in its entirety and are thus "fatal."

~~**18.5.1 AUTHENTICATION_FAILED** – The message being authenticated was not generated by the service provider.~~

~~**18.5.2 CHARACTER_SET_NOT_SUPPORTED** – A character string value was encountered that is not a supported character set.~~

~~**18.5.3/18.5.1 INCOMPATIBLE_SECURITY_LEVELS** - The two clients do not have the same security authorization. An unsecured request was received for an operation that requires a secured request.~~

~~**18.5.4 INVALID_OPERATOR_NAME** – The 'Operator Name' is not associated with any known operators.~~

~~**18.5.5 KEY_GENERATION_ERROR** – The key server was unable to generate a Session Key (SK).~~

~~**18.5.6 PASSWORD_FAILURE** – The 'Operator Name' and 'Operator Password' did not associate correctly.~~

~~**18.5.7/18.5.2 SECURITY_NOT_SUPPORTED** - The remote client does not support any BACnet security mechanisms.~~

~~**18.5.8 TIMEOUT** – The APDU with the expected Invoke ID was not received before the waiting time expired.~~

~~**18.5.9/18.5.3 OTHER** - This error code is returned for a reason other than any of those previously enumerated for this Error Class.~~

...

[Change clause 21, pp. 390-431]

```
...
***** Confirmed Service Productions *****

BACnetConfirmedServiceChoice ::= ENUMERATED {
...
-- Virtual Terminal Services
    vtOpen          (21),
    vtClose         (22),
    vtData          (23),

-- Security Services
-- authenticate (24),
-- requestKey (25),
-- this enumeration was removed (24),
-- this enumeration was removed (25),

-- Services added after 1995
-- readRange          (26) see Object Access Services
-- lifeSafetyOperation (27) see Alarm and Event Services
-- subscribeCOVProperty (28) see Alarm and Event Services
-- getEventInformation (29) see Alarm and Event Services
}
-- Other services to be added as they are defined. All enumeration values in this production are reserved
for definition by
-- ASHRAE. Enumeration values 24 and 25 are not used. Proprietary extensions are made by using the
ConfirmedPrivateTransfer or UnconfirmedPrivateTransfer
-- services. See Clause 23.

...

BACnet-Confirmed-Service-Request ::= CHOICE {
...
-- Virtual Terminal Services
    vtOpen          [21] VT-Open-Request,
    vtClose         [22] VT-Close-Request,
    vtData          [23] VT-Data-Request,

-- Security Services
-- authenticate [24] Authenticate Request,
-- requestKey [25] RequestKey Request
-- this enumeration was removed [24],
-- this enumeration was removed [25],

-- Services added after 1995
    -- readRange          [26] see Object Access Services
    -- lifeSafetyOperation [27] see Alarm and Event Services
    -- subscribeCOVProperty [28] see Alarm and Event Services
    -- getEventInformation [29] see Alarm and Event Services
}
-- Context-specific tags 0..29 are NOT used in the encoding. The tag number is transferred as the service-
choice parameter
-- in the BACnet-Confirmed-Request-PDU.
--
```

-- Other services will be added as they are defined. *Tag numbers 24 and 25 are not used.* All choice values in this production are reserved for definition by
 -- ASHRAE. Proprietary extensions are made by using the ConfirmedPrivateTransfer service. See Clause 23.

...

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

...

-- Virtual Terminal Services
 vtOpen [21] VT-Open-ACK,
 vtData [23] VT-Data-ACK,

~~— Security Services~~

~~— authenticate — [24] Authenticate ACK
 -- this enumeration was removed [24],~~

-- Context-specific tags 3..29 are NOT used in the encoding. The tag number is transferred as the service-ACK-choice
 -- parameter in the BACnet-ComplexACK-PDU.

--

-- Other services to be added as they are defined. *Tag number 24 is not used.* All choice values in this production are reserved for definition by
 -- ASHRAE. Proprietary extensions are made by using the ConfirmedPrivateTransfer service.
 -- See Clause 23.

}

...

~~***** Confirmed Security Services *****~~

~~**Authenticate-Request ::= SEQUENCE {**~~

~~— pseudoRandomNumber — [0] Unsigned32,
 — expectedInvokeID — [1] Unsigned8 OPTIONAL,
 — operatorName — [2] CharacterString OPTIONAL,
 — operatorPassword — [3] CharacterString (SIZE (1..20)) OPTIONAL,
 — startEncipheredSession — [4] BOOLEAN OPTIONAL
 — }~~

~~**Authenticate-ACK ::= SEQUENCE {**~~

~~— modifiedRandomNumber — Unsigned32
 — }~~

~~**RequestKey-Request ::= SEQUENCE {**~~

~~— requestingDeviceIdentifier — BACnetObjectIdentifier,
 — requestingDeviceAddress — BACnetAddress,
 — remoteDeviceIdentifier — BACnetObjectIdentifier,
 — remoteDeviceAddress — BACnetAddress
 — }~~

...

BACnet-Error ::= CHOICE {

 other [127] Error,

...

-- Virtual Terminal Services

```

    vtOpen          [21] Error,
    vtClose        [22] VTClose-Error,
    vtData         [23] Error,

-- Security Services
-- authenticate [24] Error,
-- requestKey   [25] Error
-- this enumeration was removed [24],
-- this enumeration was removed [25],

-- Services added after 1995
-- readRange      [26] see Object Access Services
-- lifeSafetyOperation [27] see Alarm and Event Services
-- subscribeCOVProperty [28] see Alarm and Event Services
-- getEventInformation [29] see Alarm and Event Services
}
-- Context-specific tags 0..29 and 127 are NOT used in the encoding. The tag number is transferred as the
error-choice
-- parameter in the BACnet-Error-PDU.
--
-- Other services to be added as they are defined. Tag numbers 24 and 25 are not used. All choice values
in this production are reserved for definition by
-- ASHRAE. See Clause 23.

...

Error ::= SEQUENCE {
...
    error-code    ENUMERATED {
        other                (0),
-- authentication failed (1),
-- this enumeration was removed (1),
        character-set-not-supported (41),
...
        invalid-file-start-position (11),
-- invalid operator name (12),
-- this enumeration was removed (12),
        invalid-parameter-data-type (13),
        invalid-time-stamp (14),
-- key generation error (15),
-- this enumeration was removed (15),
        missing-required-parameter (16),
...

BACnetServicesSupported ::= BIT STRING {
...
-- Virtual Terminal Services
    vtOpen          (21),
    vtClose        (22),
    vtData         (23),

-- Security Services
-- authenticate (24),
-- requestKey   (25),
-- this enumeration was removed (24),
-- this enumeration was removed (25),

```

```

-- Unconfirmed Services
  i-Am (26),
  i-Have (27),
  unconfirmedCOVNotification (28),
  unconfirmedEventNotification (29),
  unconfirmedPrivateTransfer (30),
  unconfirmedTextMessage (31),
  timeSynchronization (32),
  -- utcTimeSynchronization (36),
  who-Has (33),
  who-Is (34),

-- Services added after 1995
  readRange (35), -- Object Access Service
  utcTimeSynchronization (36), -- Remote Device Management Service
  lifeSafetyOperation (37), -- Alarm and Event Service
  subscribeCOVProperty (38), -- Alarm and Event Service
  getEventInformation (39) -- Alarm and Event Service
}

```

```

BACnetSessionKey ::= SEQUENCE {
  sessionKey OCTET STRING (SIZE(8)), -- 56 bits for key, 8 bits for checksum
  peerAddress BACnetAddress
}
...

```

[Add to **Clause 25**, pp. 448-449]

[Note: the normal convention for the use of italics in this public review draft does not apply in this section because the lines to be added will contain italics in the printed standard. Therefore, the following lines are to be added exactly as shown below.]

FIPS 180-2 (2002) *Federal Information Processing Standards – Secure Hash Standard*

FIPS 197 (2002) *Federal Information Processing Standards – Advanced Encryption Standard*

[Change **Annex C**, pp. 457-458]

```

DEVICE ::= SEQUENCE {
  ...
  number-of-APDU-retries [73] Unsigned,
  list of session keys [55] SEQUENCE OF BACnetSessionKey OPTIONAL,
  time-synchronization-recipients [116] SEQUENCE OF BACnetRecipient OPTIONAL,
  -- required for time
  ...
}

```

[Change clause **D.11**, pp. 471-472]

D.11 Examples of a Device Object

Example 1: A "sophisticated" BACnet device.

```
...  
Property:   Number_Of_APDU_Retries = 3  
Property:   List_Of_Session_Keys = ((X'3799246237984589', 1, X'03'),  
                                     (X'4446214686489744', 1, X'05'))  
Property:   Time_Synchronization_Recipients =(Device, Instance 18)  
...
```

[Delete **Clause E.6**, "Security Services," pp. 502-503]

[Delete **Clause F.6**, "Encoding for Example E.6 - Security Services," pp. 536-537.]

[Insert new clause **J.2.13**, p. 568]

J.2.13 Secure-BVLL: Purpose

This message is used to secure BVLL messages that do not contain NPDUs. Its use is described in Clause 24.

J.2.13.1 Secure-BVLL: Format

The Secure-BVLL message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'0A'	Secure-BVLL
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
Security Wrapper:		Variable length	

The BVLL to be secured is placed into the Service Data field of the Security Wrapper. For more details on securing BACnet message, see Clause 24.