Proposed Addendum *dd* to Standard 135-2024, BACnet® - A Data Communication Protocol for Building Automation and Control Networks

First Publication Public Review (November 2026)
(Draft shows Proposed Changes to Current Standard)

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

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

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

© 2026 ASHRAE. 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, 180 Technology Parkway NW, Peachtree Corners, GA 30092. Phone: 404-636-8400, Ext. 1125. Fax: 404-321-5478. E-mail: standards.section@ashrae.org.

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

(This foreword is not part of this standard. It is merely informative and does not contain requirements necessary for conformance to the standard. It has not been processed according to the ANSI requirements for a standard and may contain material that has not been subject to public review or a consensus process. Unresolved objectors on informative material are not offered the right to appeal at ASHRAE or ANSI.)

FOREWORD
Just as BACnet Authentication and Authorization in Addendum cp was an adaptation of the OAuth protocol, this addendum is an adaptation of the ACME protocol, modified for the needs of BACnet networking.

Note: This addendum makes proposed changes to the current standard. These changes are indicated in the text by underlining (for additions) and strikethrough (for deletions) except where the reviewer instructions specifically describe some other means of showing the changes. Only these changes to the current standard are open for review and comment at this time. Additional material is provided for context only and is not open for comment except as it relates to the proposed changes.

Addendum dd to Standard 135-2024

Problem:

There is no BACnet standard way for an installer or maintainer to obtain an Operating Certificate for a BACnet/SC device without human interaction.

Background:

Addendum cc defines a standard way to write an Operating Certificate into a BACnet/SC device.

Addendum cs defines CARI files - a way to package Certificate Signing Request files into a standard format that can be machine readable.

The BACCARI project is an example of a web-based server that can be used to process CARI files, including defining policies for *automatic signing* for authorized installers. However, this tool is still a human-facing operation involving uploading and downloading CARI files with a web browser.

What is missing is a **totally automated** machine-to-machine way to get a CARI file signed.

For public web servers, the Automatic Certificate Management Environment (ACME) protocol defined by RFC 8555 provides a mechanism to automatically sign CSRs by validating the request using HTTP or DNS to ensure that the host name being claimed in the certificate is valid.

BACnet uses a host name in the form "bacnet://<device-instance>". Unfortunately, this can't be validated using standard ACME "challenges" since it is not DNS or HTTP based. Therefore, we cannot use ACME as is, and all the currently published and proposed extensions to ACME (there are many) are nonetheless very global-IP-centric, e.g., they do not work for private IP addresses like 192.168.1.1.

Proposed Change:

Just as BACnet Authentication and Authorization in Addendum cp was an *adaptation* of the OAuth protocol, this proposal is an adaptation of the ACME protocol, modified for the needs of BACnet networking.

The proposal is called, of course, BACnet Automated Certificate Management Environment (BACME). It follows the same basic flow as ACME, but the validation is based on an authorized requestor and server policy, not an on-the-wire challenge. This follows the ACME "challenge-less" model used by CAs that support requests that are pre-vetted through the CA's management portal before an ACME request is made.

Using this protocol allows a tool operated by trusted installer to request a certificate for a device (either a new certificate or a rotation) without human interaction, i.e., the tool makes the request and automatically receives the answer.

The request can either be granted immediately, based on a policy at the BACME server, or it can be queued for human review if no such auto-signing policy exists. If the request is queued, the tool will automatically poll for the

result of the human review. As soon as the certificate becomes available it will be retrieved and configured into the end device without human action by the installer.

Document History:

Revision 1: Initial version, cast as a “discussion document”. Reviewed by IT-WG in Las Vegas, 2026-01.

Revision 2: Recast as “spec language”; removed “discovery”; made Retry-After mandatory; made it possible for an SC Device to be its own Client (TLS 1.3 suites, size limits); clarified opaqueness of API Key (*can* be JWT, but not required to be).

Revision 3: Aligned with ACME "order creation" by changing the initial "202 Accepted" to a "201 Created" + Location, and removed the Location from subsequent polls. Simplified and moved the "Resource retention" requirements to the "Asynchronous response" clause. Other clarifying and terminology changes based on feedback from Las Vegas meeting and mailing list.

Revision 4: 1) Clarified that the intended use case and target audience is for installation tools. Removed implications that the device does this itself as part of an in-band device-oriented onboarding protocol like BRSKI or EST. Removed TLS version, cipher suite, and size constraints that were designed for Devices. 2) Following the ACME spec, the initial poll delay is a local matter, as is the subsequent backoff delay if a Retry-After is not present. 3) expanded the text for the vendor-specific tool so that it *might* be the one creating the keys and CSR for the device in addition to assembling the CARI file.

Revision 5: 1) Clarified that the CARI file format, and this protocol, is for a single issuer. 2) Mentioned the common term “certificate pinning” for Server Authentication. 3) Added clause for “Policies” to describe possible limitations to the API Key’s authorization scope.

Revision 6: editorial cleanup.

Revision 7: 1) Added Authorization header to examples for the GETs. 2) “Automated”/”Automatic” consistency. 3) removed reference to “Retention Policy”

Changes to Standard 135-2024:

[Add new Annex XX, p. 1505]

ANNEX XX - BACnet AUTOMATED CERTIFICATE MANAGEMENT ENVIRONMENT (NORMATIVE)

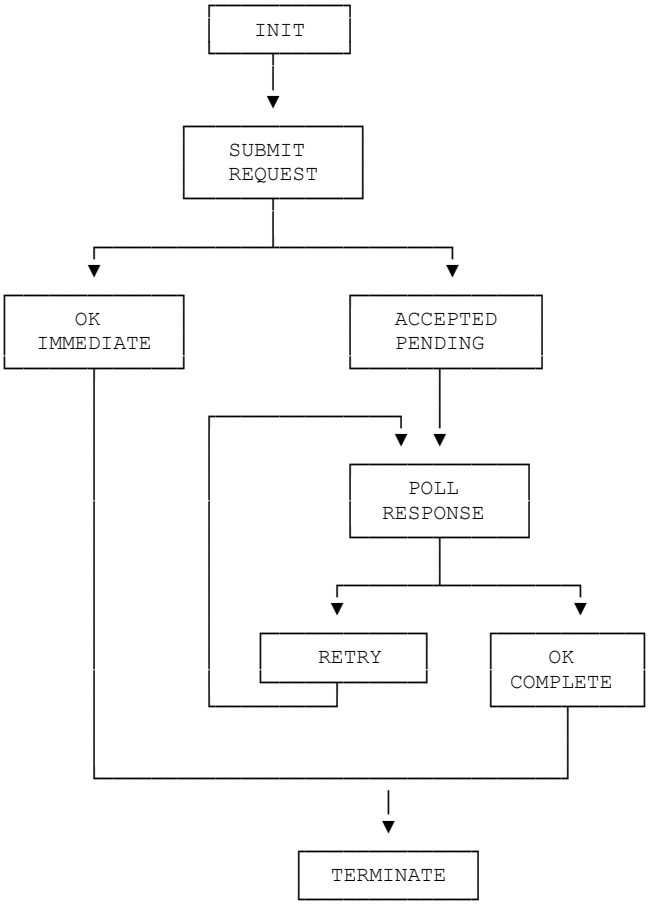
(This annex is part of this standard and is required for its use.)

XX.1 Overview

This annex defines a protocol that can be used for automatically obtaining TLS certificates for BACnet/SC devices. The BACnet Automated Certificate Management Environment (BACME) is patterned after the ACME protocol described in RFC 8555 that is used for automatically obtaining TLS certificates for public webservers.

The purpose of the protocol is to turn a Certificate Signing Request (CSR) into an operational certificate for use in a BACnet/SC device. Since the protocol uses CARI files, as defined in Clause AA.2, it can be used to request certificates for more than one device at a time.

The overall successful flow of a BACME client is conceptually the same as an ACME client, as shown here:



XX.2 Participants

The following terms and roles are used in this annex. The roles are specified for a single certificate issuer. Some advanced software might support more than one issuer, but the CARI file format, and this protocol, are designed for retrieving certificates from a single issuer.

Role	Responsibilities
Device	The BACnet/SC recipient of the certificates created by this protocol.
Client	Initiates certificate requests, authenticates server, validates responses, implements retry logic.
Server	Authenticates clients, validates CSRs, issues or queues certificates, enforces policy.
User	A human operator that has been granted rights to access the Server using a Client.
Credentials	Proof of the User's identity that is needed by the Server to validate requests from the Client.
Policy	Holds the policies defined by the site owner for auto-approval of requests. For example, these policies could be based on: user, time, device instance range, device type (router, hub).

The Client is a conceptual role. It can be implemented in a variety of ways:

(a) it could be a vendor-specific tool that creates the CARI file, communicates with the Server, and provides the resultant certificates to the Device by some vendor-specific means. In this case, the creation of the key pair and the CSR is also a local matter, either done by the Device or the tool,
(b) it could be a standards-based tool that uses the methods of Clause 19 to use the Network Port object to create keys, retrieve the CSR, and update the File objects with the resultant certificates, or
(c) other possibilities, as long as they conform to the BACME protocol.

The Server is also a conceptual role that may be implemented as a stand-alone tool or as part of a larger site HMI. It could be acting either as the CA itself, issuing certificates directly based on its own policy, or as a pass-through to the site CA. In the pass-through case, the Server's role is to provide a standards-based BACME interface to possibly different backends.

The User is a human operator that has been granted a limited authority by the site CA to install or refresh BACnet/SC certificates in Devices using the Client.

The Server provides the User with Credentials to allow the Client to perform operations on the User's behalf. How these Credentials are given to the User and how the User provides these to the Client depends on the method in which the Server and Client are implemented and are out of scope for this annex. These credentials are called an "API Key" in the following clauses.

XX.3 Operation

The BACME protocol uses HTTP/1.1 or later and follows Representational State Transfer (REST) architectural principles. All communications use TLS (https). The protocol defines two operational flows: synchronous and asynchronous, based on whether the Server can immediately return a result.

XX.3.1 Prerequisites

Prior to initiating protocol operations, the following shall be established:

(a) User obtains a valid API Key through a secure out-of-band mechanism and configures it into the Client,
(b) Client is configured with the URL for BACME requests, e.g., https://example.com/bacme/request, and
(c) Client has obtained and trusts the Server's TLS certificate for that URL.

Non-normative examples of the above could include:

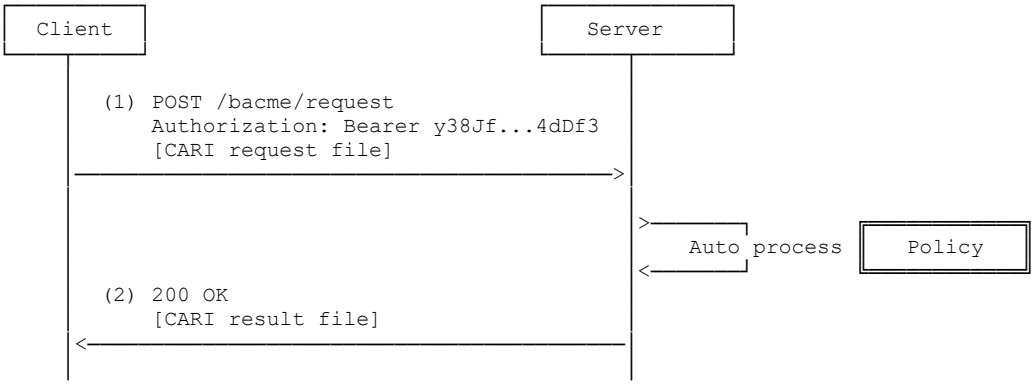
1. User securely logs into the BACME Server as a web server and requests an API Key from a web page.
2. The server generates and presents an API Key for the User to copy and keep safely. See Clause XX.4.3.
3. The web page provides the URL of the BACME submission endpoint (possibly more than one if the Server supports more than one site CA).
4. The web page provides a downloadable server certificate for the BACME server. See Clause XX.4.1.

XX.3.2 Request Submission

To begin the protocol flow, the Client submits the certificate requests by POSTing a CARI file to the request endpoint. The POST shall include the API Key using the Authorization header. See Clause XX.4.2. The Accept header of the request and the Content-Type header of the response shall have the value 'application/zip'.

XX.3.3 Synchronous response

If the Server can immediately process the request, then it returns the result using an HTTP 200 response.



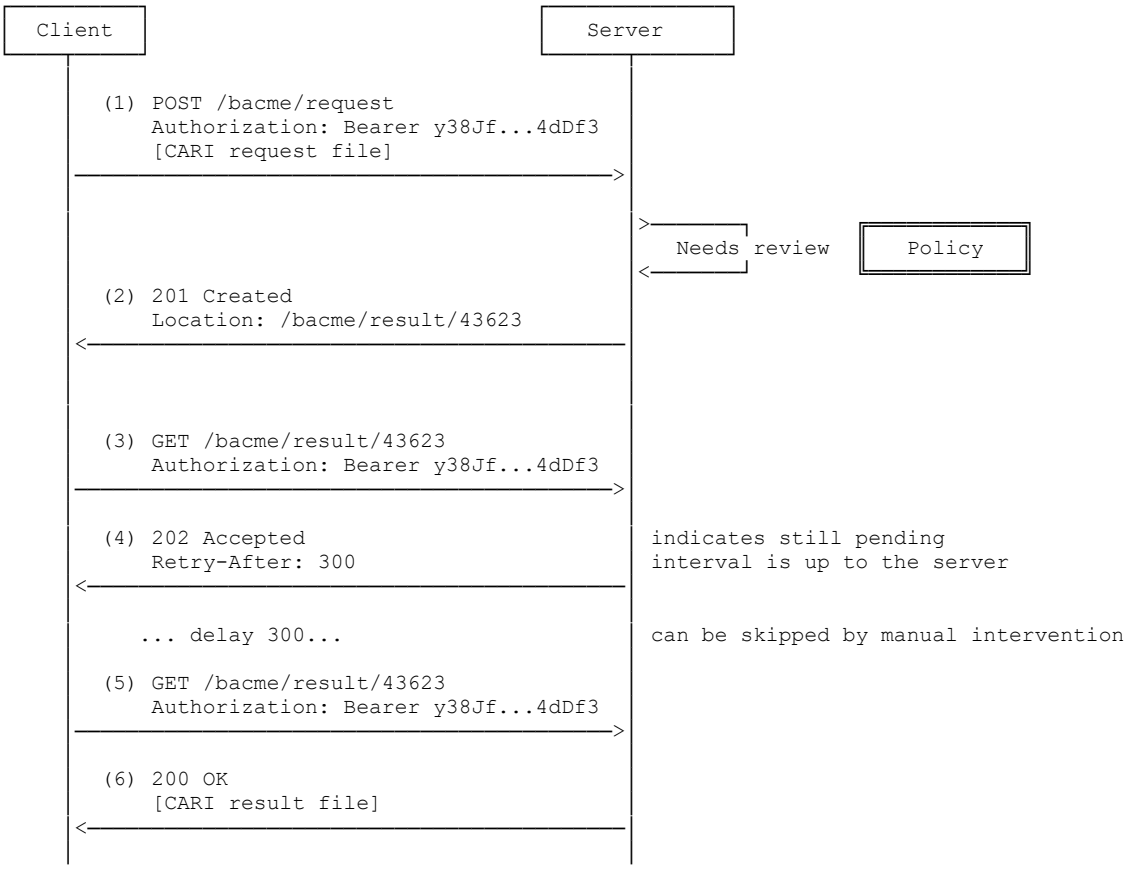
XX.3.4 Asynchronous response

If the Server cannot immediately process the request, then it returns an acknowledgment of the request using an HTTP 201 Created response and a URL in the Location header indicating where to poll for the result. The delay for this initial poll is a local matter.

The Client shall poll the URL until receiving a 200 OK or an error response. See Clause XX.3.6 for error status codes. The GET shall include the API Key using the Authorization header. See Clause XX.4.2. The Accept header shall have the value 'application/zip'.

If the Client receives a 202 Accepted response, then the process is still pending, and the Client shall retry again after a delay. When polling for results, the Client shall honor the latest Retry-After interval unless instructed to retry immediately by a human operator, perhaps upon an out-of-band indication that the response file is now available. If a Retry-After is not provided, then it is a local matter how the Client implements its backoff delay.

It is a local matter whether the Server retains the results after they are successfully retrieved. It is also a local matter whether the Server times-out a pending request so that the polled URL returns an error code. If a Client receives a terminal error code, e.g., 410 Gone or 404 Not Found, when polling, it shall stop polling and resubmit another request if needed.



XX.3.5 Response File

Both synchronous and asynchronous flows shall result in the Server returning a CARI file, even if policy or human review has determined that none of the requests can be satisfied. Therefore, the receipt of a response CARI file by the Client does not indicate that all of the CSRs have been signed, and it is possible that none of them have been signed. The absence of an operational certificate is an indication of a failure or denial.

If some certificate requests within the CARI file are not processed due to policy constraints or other errors, the server can include a response-notes.txt file within the CARI file explaining the reasons. The presence and format of this file is a local matter, but could include details such as:

```
Device-2134, Port-A is rejected: Key type is not supported.
Device-5 is rejected: Device instance not within a pre-approved range.
```

XX.3.6 Error Responses

Error responses shall use appropriate HTTP status codes. The response body can optionally contain a human-readable explanation of the error condition. The contents of the body is a local matter.

XX.3.6.1 Bad Request (400)

This status code indicates the CARI file could not be parsed or contains invalid data. For example, a bad zip file format, or the zip file is missing required elements.

XX.3.6.2 Payload Too Large (413)

This standard does not set an upper bound on the size of the CARI file that Servers shall accept for processing, although it is expected that Servers are not constrained devices. It is a local matter when a Server might use this error code to indicate a failure to process. The Client shall not attempt to resubmit the same CARI file and shall indicate this error to the User.

XX.3.6.3 Unauthorized (401)

This status code indicates a missing or invalid Authorization header. The response shall include a WWW-Authenticate header with the value "Bearer".

XX.3.6.4 Forbidden (403)

This status code indicates the API key is valid but lacks necessary permissions for certificate operations.

XX.3.6.5 Gone (410)

This status code indicates a previously available response file has been deleted and is no longer accessible.

XX.3.6.6 Not Found (404)

This status code indicates that the requested URL is unknown to the Server. This could be because the path was never valid, or a previously valid response path has been forgotten.

XX.3.6.7 Server Error Responses (5xx)

Standard HTTP 5xx status codes might be returned for server-side errors. Errors other than 503 (Service Unavailable) shall be terminal. It is a local matter whether Clients retry after receiving a 503, and if they do, they shall honor the Retry-After header value if present.

XX.3.6.8 Redirect Responses (3xx)

Clients are not required to follow 3xx redirects and can treat them as errors. Servers shall ensure that the URL provided in the Location header is directly accessible without redirection.

XX.4 Security Requirements

All communication between Client and Server shall use TLS.

XX.4.1 Server Authentication

To prevent disclosure of API keys to unauthorized servers, Clients shall authenticate the Server identity by validating the Server's TLS certificate, either by:

(a) Using the standard system trust store for certificate validation that might have been updated by a web browser that previously accepted the certificate, or
(b) Allowing configuration of a Server certificate in PEM format as an explicit trust anchor configured out-of-band as described in Clause XX.3.1, or
(c) Some other means as a local matter.

All Client implementations shall support option (b), often called "certificate pinning". The certificate file for this option shall be in PEM format containing server's certificate, with the presence of other certificates, i.e. a chain, as a local matter. All Servers shall provide a mechanism for Users to obtain this file.

XX.4.2 Client Authentication

Clients shall authenticate using the "Authorization" HTTP Header as defined in RFC 9110, with "Bearer" as the authentication scheme and the API Key as the credential. For example,

```
Authorization: Bearer <api-key>
```

XX.4.3 API Keys

API Keys are opaque to the Client. Since the Server is the entity that both generates and consumes the API Key, there is no expected interoperability between Servers. Therefore, the format and structure of API Keys are a local matter, subject to the following constraints.

API Keys shall be unpredictable and non-forgeable. Examples include cryptographically random numbers with a minimum of 128 bits entropy, OAuth 2.0 access tokens, or other tokens meeting equivalent security strength.

API Keys shall only contain characters valid for the token68 production as defined in RFC 9110 for the Authorization header.

The distribution of API Keys to Clients is a local matter. Users shall use secure out-of-band channels such as:

(a) Authenticated HTTPS web interface with user login
(b) Encrypted email or secure messaging
(c) Physical media or in-person exchange

API Keys shall be treated as private data and protected against unauthorized disclosure. It is recommended that Client tools do not store this API Key in plain text on permanent storage and require a User log-in to decrypt it into memory for use.

XX.5 Policies

The API Key authenticates the Client to the Server but does not imply unlimited authorization. The server can maintain policies that limit the scope of what the API Key is allowed to do. These policies are a local matter, but could contain restrictions such as:

(a) Device instance ranges
(b) Beginning and ending dates
(c) Restrictions on issuance of “router” and “hub” certificates
(d) Whether to immediately reject or queue for review

If the policy indicates a review is necessary, then the asynchronous flow described in Clause XX.3.4 will be followed until such review can be completed.

XX.6 Examples

XX.6.1 Successful Synchronous Flow

```
POST /bacme/request HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Accept: application/zip
Content-Type: application/zip
Content-Length: 4123

[Binary CARI file data - 4123 bytes]
```

```
HTTP/1.1 200 OK
Date: Sat, 18 Jan 2026 14:30:15 GMT
Content-Type: application/zip
Content-Length: 4534
```

```
[Binary CARI file with signed certificates - 4534 bytes]
```

XX.6.2 Asynchronous Flow with Polling

Step 1: Certificate Request (Requires Review)

```
POST /bacme/request HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Accept: application/zip
Content-Type: application/zip
Content-Length: 4123

[Binary CARI file data - 4123 bytes]
```

```
HTTP/1.1 201 Created
Date: Sat, 18 Jan 2026 14:30:15 GMT
Location: https://example.com/bacme/response/a3f2b891
Content-Type: text/plain
Content-Length: 63

Certificate request accepted for review. Poll 'Location' header URL.
```

Step 2: First Poll (Still Pending)

```
GET /bacme/response/a3f2b891 HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Accept: application/zip
```

```
HTTP/1.1 202 Accepted
Date: Sat, 18 Jan 2026 14:35:15 GMT
Retry-After: 600
Content-Type: text/plain
Content-Length: 30

Request still under review.
```

Step 3: Second Poll (Completed)

```
GET /bacme/response/a3f2b891 HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Accept: application/zip
```

```
HTTP/1.1 200 OK
Date: Sat, 18 Jan 2026 14:45:30 GMT
Content-Type: application/zip
Content-Length: 4534

[Binary CARI file with signed certificates - 4534 bytes]
```

XX.6.3 Authentication Failure (invalid)

```
POST /bacme/request HTTP/1.1
```

```
Host: example.com
Authorization: Bearer invalid_token_xyz123
Content-Type: application/zip
Accept: application/zip
Content-Length: 4123

[Binary CARI file data]
```

```
HTTP/1.1 401 Unauthorized
Date: Sat, 18 Jan 2026 14:30:15 GMT
WWW-Authenticate: Bearer
Content-Type: text/plain
Content-Length: 42

Invalid or expired API Key
```

XX.6.4 Authentication Failure (missing)

```
GET /bacme/response/a3f2b891 HTTP/1.1
Host: example.com
Accept: application/zip
```

```
HTTP/1.1 401 Unauthorized
Date: Sat, 18 Jan 2026 14:30:15 GMT
WWW-Authenticate: Bearer
Content-Type: text/plain
Content-Length: 42

Missing Authorization header with API Key
```

XX.6.5 Authentication Failure (scheme)

```
GET /bacme/response/a3f2b891 HTTP/1.1
Host: example.com
Authorization: Basic foobar
Accept: application/zip
```

```
HTTP/1.1 401 Unauthorized
Date: Sat, 18 Jan 2026 14:30:15 GMT
WWW-Authenticate: Bearer
Content-Type: text/plain
Content-Length: 42

Invalid Authorization scheme
```

XX.6.6 Malformed CARI File

```
POST /bacme/request HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/zip
Accept: application/zip
Content-Length: 256

[Invalid/corrupt CARI data]
```

```
HTTP/1.1 400 Bad Request
Date: Sat, 18 Jan 2026 14:30:15 GMT
Content-Type: text/plain
Content-Length: 78

The uploaded CARI file is malformed
```