Workgroup:	Network Working Group
Internet-Draft:	draft-payment-intent-subscription-00
Published:	29 July 2026
Intended Status:	Informational
Expires:	30 January 2027
Authors:	J. Moxey	B. Ryan	T. Meagher
---	---	---	---
	Tempo Labs	*Tempo Labs*	*Tempo Labs*

Subscription Intent for HTTP Payment Authentication

Abstract

This document defines the "subscription" payment intent for use with the Payment HTTP Authentication Scheme. The "subscription" intent represents a recurring fixed-amount payment where the payer grants the server permission to charge the same amount once per billing period. It standardizes the recurring payment authorization itself, not the full billing relationship that many application-level systems also call a subscription.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 30 January 2027.

Copyright Notice

Table of Contents

1. Introduction

The "subscription" intent enables recurring fixed-amount payments. A successful subscription activation creates an authorization for the server to collect the same payment amount once per billing period until the payer cancels it or the authorization otherwise becomes invalid.

This intent is useful for recurring API plans, content subscriptions, and other services with a stable price per billing period.

This document intentionally standardizes the payment agreement, not the entire billing system around it. In particular, the shared intent does not define price catalogs, quantities or seat counts, plan swaps, prorations, deferred starts, billing-cycle realignment, invoice state, or other product-management behavior that many billing platforms also associate with a "subscription". Those concerns belong to the application layer or to a narrower payment-method profile.

This is a deliberate trade-off. Using the name "subscription" keeps the user-facing concept familiar, but the interoperable wire contract is intentionally narrower: it means "charge this fixed amount every interval", not "model every behavior of a commercial subscription object".

1.1. Relationship to Payment Methods

Payment methods implement "subscription" using method-specific recurring authorization mechanisms. This document defines the abstract semantics and shared request fields. Payment method specifications define how those semantics are enforced, which request shapes they support, and which requests they reject because they cannot be represented exactly on the underlying payment network.

Payment method specifications **MAY** intentionally define a constrained subset of a richer underlying subscription system. A method **MUST** either preserve the semantics in this document exactly or reject the request; it **MUST NOT** approximate them.

Payment method specifications **MAY** also impose additional constraints that are not part of the shared contract, such as an explicit expiry or recipient requirements, when the underlying payment system cannot safely support the shared intent without them. Such constraints **MUST** be made explicit by the method specification.

2. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

Subscription
: A recurring payment authorization for a fixed amount charged once per billing period.

Billing Period
: A recurrence window, defined by the subscription period fields, during which at most one subscription charge may be collected.

Activation
: The successful initial setup of a subscription, which includes collection of the first billing-period charge.

Renewal
: A later charge that collects the subscription amount for a subsequent billing period.

Cancellation
: The act of ending a subscription, preventing future renewals.

Subscription Identifier
: A server-issued opaque identifier for an activated subscription, used by servers and applications to refer to that subscription in later interactions.

4. Intent Semantics

4.1. Definition

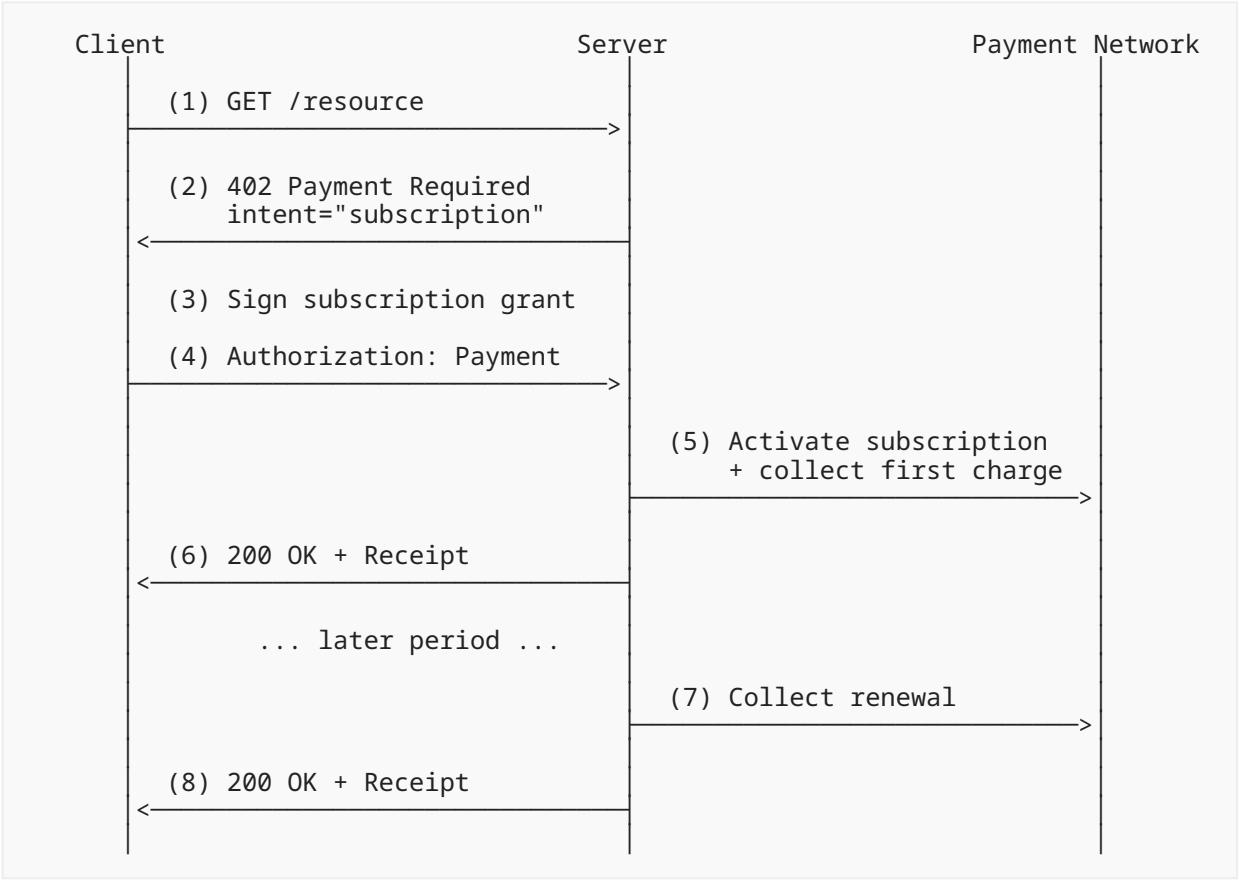
The "subscription" intent represents a request for a recurring fixed-amount payment of `amount`, charged once per billing period until explicit cancellation or until the recurring authorization otherwise becomes invalid.

4.2. Properties

Property	Value
Intent Identifier	`subscription`
Payment Timing	Recurring (initial charge at activation, then once per period)
Idempotency	Credential single-use; subscription grant reusable across billing periods
Reversibility	Cancellable

Table 1

4.3. Flow



5. Request Schema

The `request` parameter for a "subscription" intent is a JSON object with shared fields defined by this specification and optional method-specific extensions in the `methodDetails` field. The `request` JSON **MUST** be serialized using JSON Canonicalization Scheme (JCS) [RFC8785] and base64url-encoded without padding per [I-D.httpauth-payment].

5.1. Shared Fields

All payment methods implementing the "subscription" intent **MUST** support the required shared fields below. Method specifications **MAY** support, forbid, or elevate optional shared fields to **REQUIRED**, but **MUST** document those choices explicitly.

5.1.1. Required Fields

Field	Type	Description
`amount`	string	Fixed payment amount per billing period in base units
`currency`	string	Currency or asset identifier (see Section 5.2)
`periodUnit`	string	Billing period unit. The value **MUST** be `day`, `week`, or `month`
`periodCount`	string	Positive integer count of `periodUnit` values per billing period

Table 2

The `amount` value **MUST** be a string representation of a positive integer in base 10 with no sign, decimal point, exponent, or surrounding whitespace. Leading zeros **MUST NOT** be used.

The `periodCount` value **MUST** be a string representation of a positive integer in base 10 with no sign, decimal point, exponent, or surrounding whitespace. Leading zeros **MUST NOT** be used.

The `periodUnit` value defines how billing-period boundaries are computed:

- `day`: fixed elapsed-time periods of `periodCount * 86400` seconds
- `week`: fixed elapsed-time periods of `periodCount * 604800` seconds
- `month`: calendar-month periods anchored at activation

For `periodUnit="month"`, billing-period boundaries are computed by adding `N * periodCount` calendar months to the original activation anchor using UTC calendar fields. If the target month does not contain the activation anchor's day-of-month, the boundary uses the last valid day of the target month while preserving the activation time-of-day. Boundaries **MUST** be computed from the original activation anchor, not by adding months to the previous boundary.

Payment methods **MUST** reject request objects whose `periodUnit` or `periodCount` values they cannot represent exactly. They **MUST NOT** approximate the requested period by rounding, truncating, or substituting a nearby network-native cadence.

5.1.2. Optional Fields

Field	Type	Description
`recipient`	string	Payment recipient in method-native format
`subscriptionExpires`	string	Optional recurring-authorization expiry timestamp in [RFC3339] format
`description`	string	Human-readable subscription description
`externalId`	string	Merchant's reference for the subscription

Field	Type	Description
`methodDetails`	object	Method-specific extension data

Table 3

When present, `subscriptionExpires` bounds the reusable lifetime of the subscription authorization. Once that timestamp is reached, the server **MUST** stop treating the subscription as authority for future billing periods. Payment methods **MAY** require this field and **MAY** impose additional constraints, such as billing-period boundary alignment or network-native representation limits.

Payment methods **MUST** place all method-specific request parameters in `methodDetails`. They **MAY** require or forbid shared optional fields, but **MUST NOT** define additional top-level request fields.

Servers issuing `intent="subscription"` challenges **SHOULD** include the `expires` auth-param in `WWW-Authenticate` per [I-D.httpauth-payment], using [RFC3339] format. Request objects **MUST NOT** duplicate the challenge expiry value.

The first billing period begins immediately when the subscription is activated. Payment methods **MAY** define additional activation controls in `methodDetails`, but **MUST** define exact activation semantics if they do so.

The billing anchor for a subscription is the time activation succeeds, or an equivalent network-native timestamp defined by the payment method specification. Billing periods are contiguous windows derived from that anchor according to `periodUnit` and `periodCount`.

This shared intent does not define deferred starts or merchant-selected billing anchors. A payment method that needs a more specific anchor rule **MUST** document it explicitly.

The shared fields in this section are the canonical subscription contract. Payment method specifications **MUST** document how they map `amount`, `periodUnit`, `periodCount`, and activation to the underlying payment system. If a payment method cannot represent those fields or semantics exactly, it **MUST** reject the request rather than approximate it.

5.2. Currency Formats

The `currency` field supports multiple formats to accommodate different payment networks:

Format	Example	Description
ISO 4217	`"usd"`, `"eur"`	Fiat currencies (lowercase)
Token address	`"0x20c0..."`	On-chain token contract address
Method-defined	(varies)	Payment method-specific currency identifiers

Table 4

Payment method specifications **MUST** document which currency formats they support and how to interpret amounts for each format.

5.3. Method Extensions

Payment methods **MAY** define additional fields only in the `methodDetails` object. Shared top-level fields retain the meanings defined in this document.

5.4. Implementor Guidance

This section is non-normative.

Payment method authors should treat the shared `subscription` intent as the canonical interoperable contract between clients and servers. A method specification may intentionally define a narrower profile of its underlying payment system, but it should do so explicitly and fail closed.

In particular:

- Methods should support only request shapes they can represent exactly.
- Methods should document the supported and rejected ranges or values of `periodUnit` and `periodCount`, any additional bounded-lifetime or expiry rules they impose, and what conditions make activation succeed.
- Activation should not be reported as successful until both subscription setup and the first billing-period charge have succeeded.
- Methods should preserve the shared invariants of one successful charge per billing period, no automatic accumulation of missed periods, and no renewals after cancellation or any method-specific expiry.
- Richer network-native features such as trials, prorations, discounts, metered billing, pause or resume controls, quantity changes, plan changes, or open-ended renewals should be disabled or rejected unless the method specification defines an exact mapping that preserves the shared semantics.
- Implementations should maintain durable server state sufficient to prevent duplicate charges across retries, concurrent requests, and out-of-band network events.

5.5. Examples

5.5.1. Traditional Payment Processor

```
{
  "amount": "9900",
  "currency": "usd",
  "periodUnit": "month",
  "periodCount": "1",
  "description": "Pro plan"
}
```

5.5.2. Blockchain Payment (Tempo)

```
{
  "amount": "10000000",
  "currency": "0x20c0000000000000000000000000000000000001",
  "periodUnit": "day",
  "periodCount": "30",
  "subscriptionExpires": "2026-07-14T12:00:00Z",
  "recipient": "0x742d35cc6634c0532925a3b844bc9e7595f8fe00",
  "methodDetails": {
    "accessKey": {
      "accessKeyAddress": "0x1111111111111111111111111111111111111111",
      "keyType": "p256"
    },
    "chainId": 42431
  }
}
```

6. Credential Requirements

6.1. Payload

The credential `payload` for a "subscription" intent contains the subscription authorization grant. The format is method-specific:

Authorization Type	Description	Example Methods
Periodic key auth	Delegated key with per-period limits	Tempo
Subscription setup	Processor-managed recurring payment setup	Stripe
Signed mandate	Recurring debit mandate	ACH, SEPA

Table 5

6.2. Single-Use

Each "subscription" credential **MUST** be usable only once per challenge. Servers **MUST** reject replayed credentials.

A successfully activated subscription may be reused for later billing periods until:

- The payer explicitly cancels it
- The payment method revokes or invalidates the authorization
- The `subscriptionExpires` timestamp, if present, is reached

7. Subscription Lifecycle

7.1. Activation

When the server receives a "subscription" credential, it **MUST**:

1. Verify the subscription authorization proof
2. Perform any method-specific subscription setup and collect the first billing-period charge
3. Initialize durable subscription state for later renewals
4. Return success (200) with a `Payment-Receipt` for the first charge, including a `subscriptionId`

The subscription becomes active only after these steps succeed.

7.2. Renewal

For each later billing period, the server **MAY** collect one renewal charge for `amount` using the method-specific recurring authorization flow.

If the server grants access for a later billing period, it **MUST** ensure that the renewal charge for that period has been collected before, or atomically with, delivering the corresponding service.

Servers **MUST NOT** collect more than one renewal charge for the same billing period.

If one or more billing periods elapse without a successful renewal charge, the subscription intent authorizes at most one charge for the then-current billing period. Servers **MUST NOT** treat missed billing periods as automatically accumulated authority for additional charges.

Payment method specifications define the concrete renewal, retry, recovery, and cancellation mechanisms, but they **MUST** preserve the invariants in this section.

7.3. Subscription Identifier

After successful activation, the server **MUST** return a `subscriptionId` in the `Payment-Receipt`. The value **MUST** be a base64url [RFC4648] string without padding and **MUST** be unique within the server's subscription namespace.

This specification does not define a dedicated request header or parameter for selecting an existing subscription. Selecting an existing subscription is an application-layer concern. Applications **MAY** use authenticated session state, account identity, resource scope, an application-defined selector, or other context to associate a later request with an existing subscription.

Clients **MAY** retain the `subscriptionId` as application data when referring to the active subscription in later interactions, but the `subscriptionId` is only a receipt identifier unless an application explicitly assigns it additional application-layer meaning.

Servers **MUST** authenticate or otherwise authorize the client's use of the identified subscription before granting access or collecting a renewal charge. Possession or presentation of a `subscriptionId` alone is insufficient.

7.4. Server Accounting and Idempotency

Servers **MUST** maintain durable subscription state sufficient to enforce per-period charging rules across retries and concurrent requests.

At minimum, servers **MUST** track:

- Subscription identifier
- Billing anchor or equivalent current billing-period start time
- Last successfully charged billing-period index, or whether the current billing period has been charged
- Any method-specific expiry or bounded-lifetime state
- Cancellation or revocation status

For non-idempotent requests, clients **SHOULD** send an `Idempotency-Key` header per [I-D.ietf-httpapi-idempotency-key-header]. Servers **MUST NOT** collect the same activation or renewal charge more than once for a duplicate idempotent request.

7.5. Cancellation

Payers **SHOULD** be able to cancel subscriptions before any applicable method-specific expiry. Cancellation mechanisms, effective-time rules, and any continued access for already-paid service are method-specific and **MUST** be documented by the payment method or application profile.

Servers **MUST NOT** collect renewal charges for billing periods after cancellation takes effect.

7.6. Error Responses

When a subscription cannot be used to fulfill a request, the server **MUST** return an appropriate HTTP status code:

Condition	Status Code	Behavior
Method-specific expiry reached	402 Payment Required	Issue new challenge
Cancellation effective or authorization revoked	402 Payment Required	Issue new challenge
Current billing period unpaid or renewal failed	402 Payment Required	Issue new challenge

Condition	Status Code	Behavior
Invalid credential	402 Payment Required	Issue new challenge

Table 6

For all 402 responses, the server **MUST** include a `WWW-Authenticate` header with a fresh challenge. Clients receiving a 402 after a previously valid subscription **SHOULD** treat the subscription as no longer usable and initiate a new subscription flow.

8. Illustrative Lifecycle Examples

This section is non-normative.

8.1. Monthly Billing Example

Suppose a server offers a plan with these request fields:

- `amount = "9900"`
- `currency = "usd"`
- `periodUnit = "month"`
- `periodCount = "1"`

If activation succeeds at `2026-01-15T12:03:10Z`, that time becomes the billing anchor. The resulting billing periods are:

- Period 0: `[2026-01-15T12:03:10Z, 2026-02-15T12:03:10Z)`
- Period 1: `[2026-02-15T12:03:10Z, 2026-03-15T12:03:10Z)`
- Period 2: `[2026-03-15T12:03:10Z, 2026-04-15T12:03:10Z)`

Activation collects the Period 0 charge. Requests during Period 0 do not require another renewal charge. When Period 1 begins, the server may collect one renewal charge for Period 1 before, or atomically with, granting access for that period. After that renewal succeeds, additional requests during Period 1 do not permit another charge for Period 1.

If the same monthly plan activates at `2026-01-31T12:03:10Z`, the resulting boundaries are computed from the January 31 anchor:

- Period 0: `[2026-01-31T12:03:10Z, 2026-02-28T12:03:10Z)`
- Period 1: `[2026-02-28T12:03:10Z, 2026-03-31T12:03:10Z)`
- Period 2: `[2026-03-31T12:03:10Z, 2026-04-30T12:03:10Z)`

8.2. Cancellation Example

Suppose the subscription above has already been charged through Period 2 and the payer cancels on `2026-03-20T09:00:00Z`.

Cancellation takes effect at the end of the current paid billing period, which is `2026-04-15T12:03:10Z` in this example. The server continues honoring access through that time. The server does not collect a renewal charge for Period 3. A request after `2026-04-15T12:03:10Z` receives `402 Payment Required` with a fresh challenge.

8.3. Failed Renewal Example

Suppose Period 3 begins and the server attempts the renewal charge for that period, but the method-specific payment step fails.

The server does not grant access for the unpaid period and returns `402 Payment Required` with a fresh challenge. If a later retry during Period 3 succeeds, the server may then grant access for Period 3.

If Period 4 begins before any successful charge occurs, the subscription intent authorizes at most one charge for Period 4. The missed Period 3 charge does not automatically accumulate into authority to collect both Period 3 and Period 4.

8.4. Expiry Example

The shared subscription intent defines `subscriptionExpires` as an optional top-level field. Some payment methods require it and others leave it optional.

Once `subscriptionExpires` is reached, the server stops treating the subscription as reusable for future billing periods. Requests after that time receive `402 Payment Required` with a fresh challenge.

9. Security Considerations

9.1. Recurring Charge Awareness

Clients **MUST** clearly communicate that a subscription authorizes future recurring charges without requiring a new user action for each billing period.

9.2. Amount and Period Verification

Clients **MUST** verify before activating a subscription:

1. `amount` is acceptable for the service
2. `currency` is expected
3. `periodUnit` and `periodCount` match the expected billing interval
4. Any `subscriptionExpires` value and method-specific constraints are understood and acceptable

Clients **MUST NOT** rely on the `description` field for payment verification.

9.3. Duplicate Charge Prevention

Servers **MUST** prevent duplicate activation and renewal charges caused by retries, parallel requests, or races between charging and service delivery.

9.4. Server Accountability

Servers operating subscriptions are responsible for:

- Secure storage of subscription authorization data
- Not charging more than once per billing period
- Honoring cancellation and revocation
- Providing transaction or billing records to payers

9.5. Caching

Responses to subscription challenges (402 Payment Required) **MUST** include `Cache-Control: no-store` to prevent sensitive payment data from being cached by intermediaries.

Responses containing `Payment-Receipt` headers **MUST** include `Cache-Control: private` to prevent shared caches from storing payment receipts.

10. IANA Considerations

10.1. Payment Intent Registration

This document registers the "subscription" intent in the "HTTP Payment Intents" registry established by [I-D.httpauth-payment]:

Intent	Description	Reference
`subscription`	Recurring fixed-amount payment	This document

Table 7

Contact: Tempo Labs (contact@tempo.xyz)

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/info/rfc8785>.

[I-D.httpauth-payment] Moxey, J., "The 'Payment' HTTP Authentication Scheme", January 2026, <https://datatracker.ietf.org/doc/draft-ietf-httpauth-payment/>.

[I-D.ietf-httpapi-idempotency-key-header] Jena, J., "The Idempotency-Key HTTP Header Field", June 2024, <https://datatracker.ietf.org/doc/draft-ietf-httpapi-idempotency-key-header/>.

Appendix A. Acknowledgements

The authors thank the MPP community for their feedback on this specification.

Authors' Addresses

Jake Moxey
Tempo Labs
Email: jake@tempo.xyz

Brendan Ryan
Tempo Labs
Email: brendan@tempo.xyz

Tom Meagher
Tempo Labs
Email: thomas@tempo.xyz