Workgroup:	Network Working Group
Internet-Draft:	draft-stripe-subscription-00
Published:	29 July 2026
Intended Status:	Informational
Expires:	30 January 2027
Authors:	B. Ryan	S. Kaliski
---	---	---
	Tempo Labs	*Stripe*

Stripe Subscription Intent for HTTP Payment Authentication

Abstract

This document defines the `subscription` intent for the `stripe` payment method within the Payment HTTP Authentication Scheme [I-D.httpauth-payment]. It specifies a constrained Stripe Billing profile for fixed-price recurring subscriptions whose activation succeeds only when the first invoice is paid synchronously.

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

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction

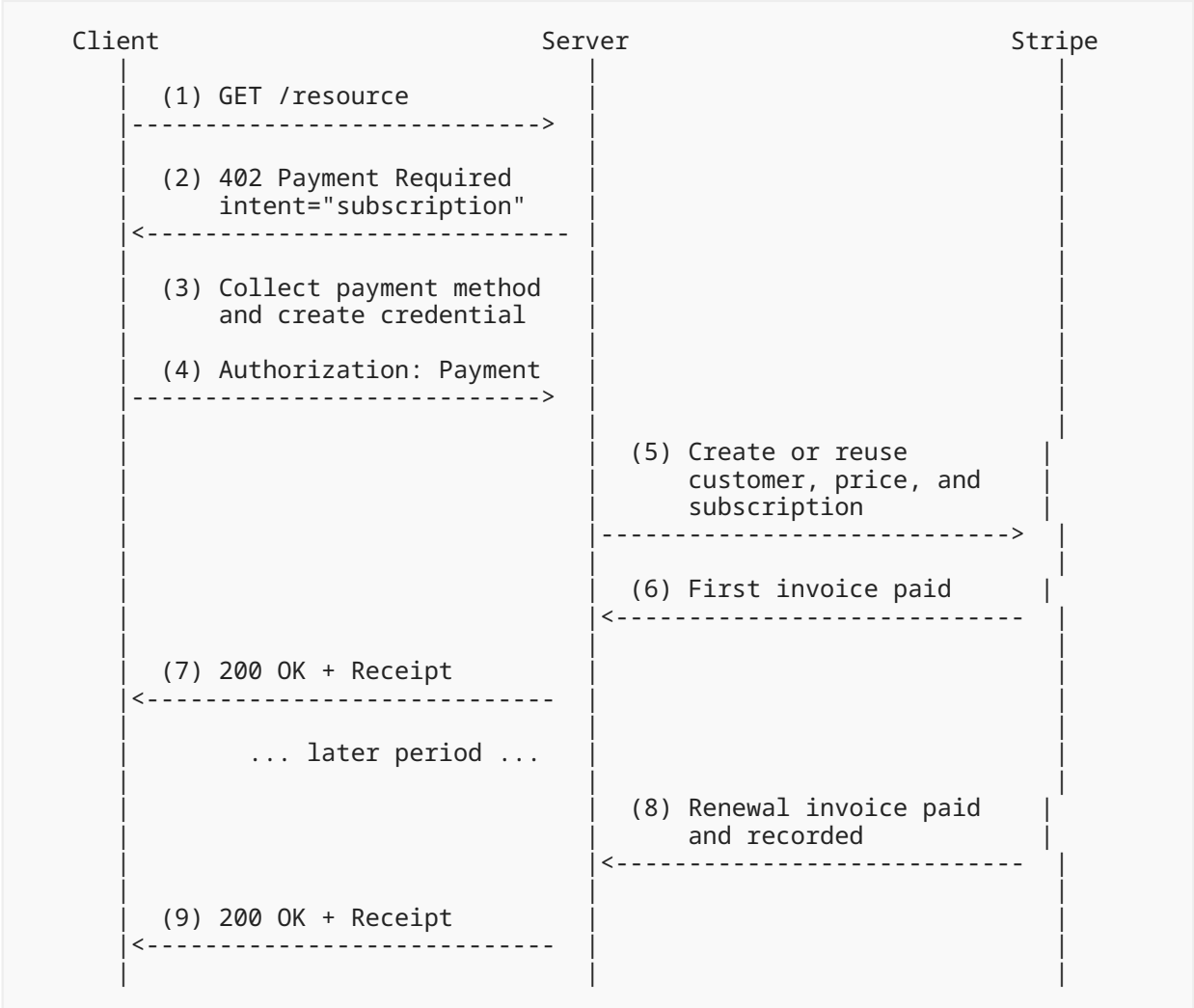
This specification defines the `subscription` intent for use with the `stripe` payment method in the Payment HTTP Authentication Scheme [I-D.httpauth-payment]. It profiles Stripe Billing as a narrow, canonical mapping of the shared `subscription` intent defined in [I-D.payment-intent-subscription].

This document is intentionally not a specification for all Stripe subscription features. Stripe Billing supports richer behaviors such as trials, prorations, discounts, usage-based billing, and flexible schedule changes. This method supports only the subset that preserves the shared subscription semantics exactly. Servers **MUST** reject request objects or Stripe configurations that would broaden those semantics.

This profile models the recurring payment agreement, not the full Stripe Billing object surface. Quantities or seat counts, plan schedules, prorations, billing-anchor resets, and other commercial-policy behavior remain out of scope even though Stripe can support them.

1.1. Stripe Subscription Flow

The following diagram illustrates the Stripe subscription flow:



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

Stripe Customer A Stripe object representing the payer for a subscription.

Stripe Price A Stripe object that defines the fixed recurring amount, currency, and cadence for a subscription item.

Stripe Subscription
: A Stripe Billing object representing the recurring commercial relationship. In this profile it **MUST** contain exactly one fixed-price recurring item.

First Invoice
: The initial Stripe invoice created for the subscription at activation time. Activation succeeds only after this invoice is paid.

4. Request Schema

The `request` parameter in the `WWW-Authenticate` challenge contains a base64url-encoded JSON object. The `request` JSON **MUST** be serialized using JSON Canonicalization Scheme (JCS) [RFC8785] and base64url-encoded without padding per [I-D.httpauth-payment].

4.1. Request Fields

The Stripe `subscription` profile uses the shared `amount`, `currency`, `periodUnit`, `periodCount`, `description`, and `externalId` fields from [I-D.payment-intent-subscription]. It additionally defines the following request constraints:

Field	Type	Required	Description
`amount`	string	**REQUIRED**	Fixed payment amount per billing period in the currency's smallest unit
`currency`	string	**REQUIRED**	Lowercase ISO 4217 currency code
`periodUnit`	string	**REQUIRED**	Billing period unit. The value **MUST** be `day`, `week`, or `month`
`periodCount`	string	**REQUIRED**	Positive integer count of `periodUnit` values per billing period
`description`	string	**OPTIONAL**	Human-readable subscription description
`externalId`	string	**OPTIONAL**	Merchant's reference for the subscription
`recipient`	string	**MUST NOT**	This profile identifies the merchant by the challenged Stripe account and `methodDetails.networkId`, not by a request-native recipient field

Table 1

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.

Servers **MUST** reject request objects that include `recipient` or `subscriptionExpires`.

4.2. Method Details

Field	Type	Required	Description
`methodDetails.networkId`	string	**REQUIRED**	Stripe Business Network Profile ID for the challenged merchant
`methodDetails.paymentMethodTypes`	[]string	**REQUIRED**	Stripe payment method types accepted for synchronous activation and future off-session recurring invoices
`methodDetails.metadata`	object	**OPTIONAL**	Stripe metadata as a string key/value map

Table 2

Servers **MUST** include only payment method types that can complete this profile's activation flow synchronously and can also be reused for future off-session recurring charges under the challenged account. Servers **MUST** reject payment method types that require an asynchronous first-invoice settlement path or customer action after the credential is submitted.

If `methodDetails.metadata` is present, every key and value **MUST** be a JSON string and the object **MUST** satisfy Stripe metadata limits [STRIPE-METADATA]. Metadata **MUST NOT** affect payment authorization, amount, period, recipient, invoice validation, cancellation, or access-control decisions.

Example:

```
{
  "amount": "5000",
  "currency": "usd",
  "periodUnit": "week",
  "periodCount": "1",
  "description": "Weekly Pro plan",
  "externalId": "sub_12345",
  "methodDetails": {
    "networkId": "profile_1MqDcVKA5fEO2tZvKQm9g8Yj",
    "paymentMethodTypes": ["card", "link"],
    "metadata": {
      "plan": "weekly-pro"
    }
  }
}
```

4.3. Constrained Stripe Billing Profile

This method defines a constrained profile of Stripe Billing. Servers **MUST** either implement this profile exactly or reject the request.

Servers **MUST** create or reuse exactly one Stripe Customer and exactly one Stripe Subscription containing exactly one recurring Stripe Price. The Price **MUST** have a fixed `unit_amount`, fixed `currency`, and fixed recurring cadence for the full life of the subscription.

The period fields **MUST** map exactly to a Stripe recurring cadence using one of the following forms:

- `periodUnit="day"`, where `periodCount` maps to `interval_count` and Stripe `recurring.interval` is `day`
- `periodUnit="week"`, where `periodCount` maps to `interval_count` and Stripe `recurring.interval` is `week`
- `periodUnit="month"`, where `periodCount` maps to `interval_count` and Stripe `recurring.interval` is `month`

Servers **MUST** reject any period fields that would require approximation, calendar-year interpretation, or an unsupported Stripe interval count.

This profile supports only a fixed quantity of 1 for the single subscription item. Servers **MUST** reject any request or server-side configuration that would vary quantity during the active lifetime of the subscription.

When creating a Stripe Subscription for this profile, servers **MUST** use the following create Subscription parameters [STRIPE-SUBSCRIPTIONS-API]:

- `collection_method=charge_automatically`
- `payment_behavior=error_if_incomplete`
- `proration_behavior=none`
- exactly one subscription item with `quantity=1`
- no `add_invoice_items`
- no `billing_cycle_anchor` other than immediate activation
- no `backdate_start_date`
- no `cancel_at` or `cancel_at_period_end` at activation
- no `pending_invoice_item_interval`
- no subscription schedule

Servers **MUST** create the subscription using an idempotency key bound to the challenge ID, payer, payment method, amount, currency, and `periodUnit` and `periodCount`. If an idempotent retry returns an existing Stripe Subscription, the server **MUST** verify that the existing object still matches this profile before treating the retry as successful.

4.4. Unsupported Stripe Billing Features

Servers implementing this profile **MUST** disable or reject the following features:

- free trials
- paid trials
- prorations
- discounts or coupons
- automatic tax
- additional invoice items
- pending invoice items
- usage-based billing
- metered add-ons
- mid-cycle plan changes
- quantity changes during an active subscription
- pause or resume controls
- asynchronous first-invoice settlement
- customer-action-required first-invoice flows
- manual invoice collection

5. Credential Schema

The Payment credential is a base64url-encoded JSON object containing `challenge` and `payload` fields per [I-D.httpauth-payment]. For Stripe subscription, the `payload` object contains the following fields:

Field	Type	Required	Description
`paymentMethod`	string	**REQUIRED**	Stripe PaymentMethod ID to use for the first invoice and future recurring charges
`customer`	string	**OPTIONAL**	Existing Stripe Customer ID if the merchant already has one for the payer

Table 3

The `paymentMethod` **MUST** reference a Stripe PaymentMethod whose type is included in `methodDetails.paymentMethodTypes` and which is suitable for future off-session recurring charges under the challenged Stripe account.

Before submitting a credential, the client or Stripe-native collection flow **MUST** have obtained any authorization, mandate, or setup required by Stripe for future off-session recurring charges [STRIPE-SETUP-FUTURE]. Servers **MUST** reject PaymentMethods that are not reusable for the challenged merchant and subscription terms.

Example:

```
{
  "paymentMethod": "pm_1Qabc32eZvKYlo2C7b8H1234",
  "customer": "cus_S7x1Pq5R9n2Lm4"
}
```

6. Verification Procedure

Servers **MUST** verify Payment credentials for Stripe subscription intent:

1. Verify the challenge ID matches the one issued
2. Verify the challenge has not expired
3. Decode the request object and verify it matches this constrained profile, including exact support for `periodUnit` and `periodCount`
4. Extract the `paymentMethod` and optional `customer` from the credential payload
5. Verify the Stripe PaymentMethod exists, is reusable by the challenged merchant, has a type allowed by the challenge, and can support both the profile's synchronous first-invoice activation flow and future off-session recurring charges
6. Verify the credential has not been replayed for the same challenge

Servers **MUST** complete challenge validation before creating or mutating Stripe objects.

7. Settlement Procedure

7.1. Activation and First-Period Charge

For `intent="subscription"`, the server **MUST**:

1. Create or reuse a Stripe Customer for the payer
2. Attach or select the challenged `paymentMethod` for that Customer
3. Create or reuse a Stripe Price whose amount, currency, and recurring cadence exactly match the request
4. Create a Stripe Subscription with exactly one recurring item, quantity 1, the creation parameters defined above, and no unsupported features
5. Verify the first invoice and its PaymentIntent completed synchronously
6. Treat activation as successful only after the first invoice for that subscription is paid and validated
7. Initialize durable local subscription state for later renewals

8. Return success (200) with a `Payment-Receipt` for the first invoice, including a `subscriptionId`

Servers **MUST NOT** treat the subscription as active, grant access, or return a success receipt while the first invoice is unpaid, requires additional customer action, or remains incomplete.

If Stripe cannot pay the first invoice synchronously, including because the invoice requires customer action, remains incomplete, enters processing, or depends on asynchronous settlement, the server **MUST** treat activation as failed and return `402 Payment Required` with a fresh challenge. The server **MUST NOT** expose a protocol continuation state for that incomplete Stripe Subscription.

The canonical billing anchor for this profile is the start timestamp of the first paid Stripe invoice period. Servers **MUST** use that anchor when mapping later Stripe invoices to the shared billing periods.

Before activating a subscription or recording a renewal, servers **MUST** validate the paid Stripe invoice. The invoice **MUST**:

- belong to the expected Stripe Subscription and Customer
- have status `paid`
- contain exactly one subscription line item
- have no invoice items outside the subscription item
- have no discounts, tax, credits, or prorations that change the amount
- match the challenged `amount` and `currency`
- map to exactly one canonical billing period derived from the billing anchor, `periodUnit`, and `periodCount`
- not have already been recorded for another billing period or subscription

7.2. Renewal

Later billing periods are fulfilled by Stripe renewal invoices. Servers **MUST** use durable local state to map Stripe invoices and webhook events onto canonical billing periods derived from the activation anchor and the period fields.

Servers **MUST** treat a later billing period as paid only after they observe a successful paid Stripe invoice for that subscription and record that canonical billing period durably.

Servers **MUST NOT** grant more than one newly paid billing period because of duplicate webhooks, retries, concurrent requests, or later collection of older unpaid invoices. If a Stripe recovery or retry flow cannot be mapped exactly to the shared one-charge-per-period invariant, servers **MUST** disable that flow or reject the request.

Servers **MUST** prevent Stripe from collecting invoices for canonical billing periods that are no longer payable under the shared subscription intent. If a renewal invoice remains unpaid when a later canonical billing period begins, the server **MUST** void it, mark it uncollectible, cancel

automatic collection for it, or configure Stripe retry and recovery behavior so the stale invoice cannot later collect payment for that closed period. A later paid invoice for an older canonical billing period **MUST NOT** be recorded as a successful subscription charge.

Implementations **MUST** process Stripe invoice events idempotently by recording the Stripe event ID, invoice ID, subscription ID, and canonical billing-period index. A duplicate webhook or API retry **MUST** return the previously recorded result without creating a second local payment record or granting another billing period.

Servers **MUST NOT** rely on `invoice.created` delivery or acknowledgement for access decisions. Access can be granted only after a validated paid invoice has been durably recorded. If webhook delivery, invoice finalization, or automatic collection is delayed, the corresponding billing period remains unpaid until a later validated paid invoice is recorded.

7.3. Cancellation

Payers **MUST** be able to cancel Stripe subscriptions. For this profile, the default cancellation effective time is the end of the current paid canonical billing period.

When a payer cancels, the server **MUST** set the Stripe Subscription to cancel at the period end corresponding to the last paid canonical billing period, and **MUST** record that cancellation effective time in durable local state. The server **MAY** cancel immediately only if the application separately handles any already-paid access period without creating an additional charge.

Servers **MUST** treat `customer.subscription.deleted` and equivalent Stripe cancellation state as revocation for future renewals. Servers **MUST NOT** collect or record renewal invoices for billing periods whose start time is at or after the cancellation effective time.

Servers **MUST** prevent pending invoice items from being collected after cancellation. If any pending invoice item, proration, credit, tax, or other non-profile invoice component exists for the Customer or Subscription, the server **MUST** remove it before cancellation or reject the subscription as no longer conforming to this profile.

7.4. Receipt Generation

Upon successful activation or renewal, servers **MUST** return a `Payment-Receipt` header per [I-D.httpauth-payment]. Servers **MUST NOT** include a `Payment-Receipt` header on error responses.

The receipt payload for Stripe subscription:

Field	Type	Description
`method`	string	`"stripe"`
`reference`	string	Stripe invoice ID whose successful payment activated or renewed the subscription
`status`	string	`"success"`

Field	Type	Description
`subscriptionId`	string	Server-issued opaque identifier for the subscription
`stripeSubscription`	string	Stripe subscription ID
`timestamp`	string	[RFC3339] time the invoice was recorded as paid
`externalId`	string	**OPTIONAL**. Echoed from the challenge request

Table 4

8. Security Considerations

8.1. Reject Unsupported Features

Stripe Billing supports features whose semantics are broader than the shared `subscription` intent. Servers **MUST** reject or disable those features rather than silently approximating the requested subscription.

8.2. Invoice Status Versus Access

Servers **MUST NOT** grant access based only on a Stripe subscription's high-level status. Stripe can report an `active` subscription while other invoices remain open or while retry logic is still in progress [STRIPE-BILLING-OVERVIEW]. Access decisions **MUST** use the canonical per-period accounting required by [I-D.payment-intent-subscription] together with successfully paid invoices.

8.3. Webhook Authenticity and Ordering

Implementations using Stripe webhooks **MUST** verify webhook authenticity, handle duplicate deliveries safely, and tolerate out-of-order event arrival [STRIPE-BILLING-WEBHOOKS].

8.4. Duplicate Charge Prevention

Stripe invoices and webhooks do not by themselves guarantee that the same HTTP billing period will be applied only once. Servers **MUST** keep durable local state sufficient to prevent duplicate activation or renewal accounting across retries, concurrent requests, and webhook replays.

9. IANA Considerations

The `subscription` payment intent is registered by [I-D.payment-intent-subscription]. This document does not register it again.

10. References

10.1. 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>.

[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>.

[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.payment-intent-subscription] Moxey, J., "Subscription Intent for HTTP Payment Authentication", April 2026, <https://datatracker.ietf.org/doc/draft-payment-intent-subscription/>.

10.2. Informative References

[STRIPE-BILLING-OVERVIEW] Stripe, Inc., "How subscriptions work", n.d., <https://docs.stripe.com/billing/subscriptions/overview>.

[STRIPE-BILLING-CANCEL] Stripe, Inc., "Cancel subscriptions", n.d., <https://docs.stripe.com/billing/subscriptions/cancel>.

[STRIPE-BILLING-WEBHOOKS] Stripe, Inc., "Using webhooks with subscriptions", n.d., <https://docs.stripe.com/billing/subscriptions/webhooks>.

[STRIPE-SUBSCRIPTIONS-API] Stripe, Inc., "Create a subscription", n.d., <https://docs.stripe.com/api/subscriptions/create>.

[STRIPE-METADATA] Stripe, Inc., "Metadata", n.d., <https://docs.stripe.com/api/metadata>.

[STRIPE-SETUP-FUTURE] Stripe, Inc., "Set up future payments", n.d., <https://docs.stripe.com/payments/setup-intents>.

Appendix A. Examples

This section is non-normative.

A.1. Activation

Challenge:

```
HTTP/1.1 402 Payment Required
Cache-Control: no-store
WWW-Authenticate: Payment id="qT8wErYuI3oPlKjH6gFdSa",
  realm="api.example.com",
  method="stripe",
  intent="subscription",
  expires="2026-01-15T12:05:00Z",
  request="<base64url-encoded JSON below>"
```

The `request` decodes to:

```
{
  "amount": "5000",
  "currency": "usd",
  "periodUnit": "week",
  "periodCount": "1",
  "description": "Weekly Pro plan",
  "methodDetails": {
    "networkId": "profile_1MqDcVKA5fEO2tZvKQm9g8Yj",
    "paymentMethodTypes": ["card", "link"]
  }
}
```

Credential payload:

```
{
  "paymentMethod": "pm_1Qabc32eZvKYlo2C7b8H1234",
  "customer": "cus_S7x1Pq5R9n2Lm4"
}
```

The server creates or reuses a Stripe Customer, creates or reuses a weekly fixed-price Stripe Price, creates a Stripe Subscription, and waits for the first invoice to be paid. Once Stripe reports the first invoice as paid, the `Payment-Receipt` payload decodes to:

```
{
  "method": "stripe",
  "reference": "in_1QabdK2eZvKYlo2C0L9n4321",
  "status": "success",
  "subscriptionId": "c3ViX3N0cmlwZV8wMQ",
  "stripeSubscription": "sub_1Qabd52eZvKYlo2CgP0Lm789",
  "timestamp": "2026-01-15T12:03:10Z"
}
```

A.2. Rejected Unsupported Cadence

If a request uses period fields that cannot be represented by the shared subscription contract and Stripe recurring cadence, the server rejects it rather than approximating. For example, the following request is invalid for this profile because `year` is not a supported `periodUnit`:

```
{
  "amount": "5000",
  "currency": "usd",
  "periodUnit": "year",
  "periodCount": "1",
  "methodDetails": {
    "networkId": "profile_1MqDcVKA5fEO2tZvKQm9g8Yj",
    "paymentMethodTypes": ["card"]
  }
}
```

Appendix B. Acknowledgements

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

Authors' Addresses

Brendan Ryan
Tempo Labs
Email: brendan@tempo.xyz

Steve Kaliski
Stripe
Email: stevekaliski@stripe.com