Workgroup:	Network Working Group
Internet-Draft:	draft-stripe-charge-00
Published:	3 March 2026
Intended Status:	Informational
Expires:	4 September 2026
Authors:	B. Ryan	S. Kaliski
---	---	---
	Tempo Labs	*Stripe*

Stripe charge Intent for HTTP Payment Authentication

Abstract

This document defines the "charge" intent for the Stripe payment method within the Payment HTTP Authentication Scheme [I-D.httpauth-payment]. It specifies how clients and servers exchange one-time payments using Stripe Payment Tokens (SPTs).

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 4 September 2026.

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.

Table of Contents

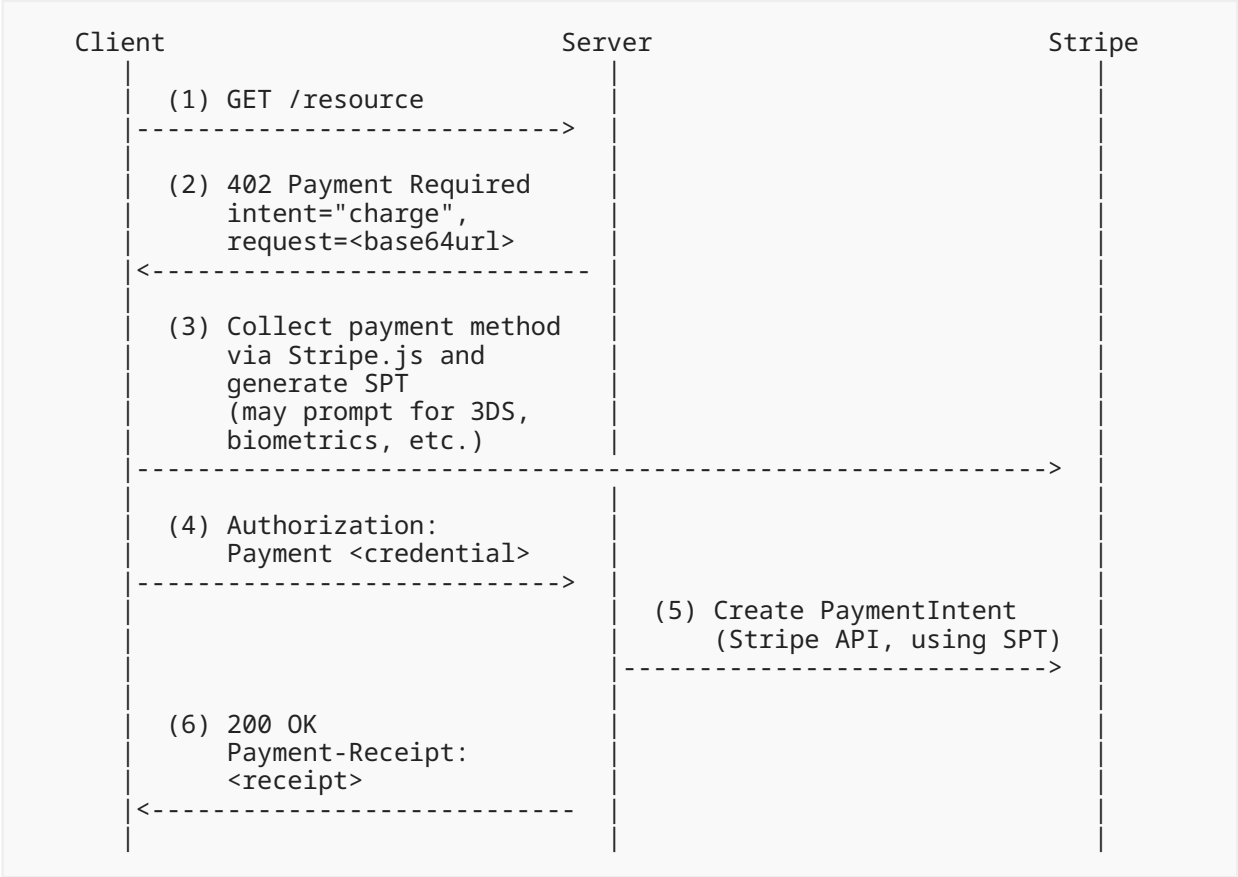
1. Introduction

This specification defines the "charge" intent for use with the Stripe payment method in the Payment HTTP Authentication Scheme [I-D.httpauth-payment]. The charge intent enables one-time payments where the server processes the payment immediately upon receiving a Stripe Payment Token (SPT).

Stripe provides payment processing through SPTs, which are single-use tokens that represent payment authorization. SPTs abstract away the complexity of payment method details (cards, bank accounts, wallets) and provide a unified interface for payment acceptance.

1.1. Stripe Charge Flow

The following diagram illustrates the Stripe charge payment flow:



1.2. Relationship to the Payment Scheme

This document is a payment method intent specification as defined in Section 10.1 of [I-D.httpauth-payment]. It defines the `request` and `payload` structures for the `charge` intent of the `stripe` payment method, along with verification and settlement procedures.

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 Payment Token (SPT) A single-use token (prefixed with `spt_`) that represents authorization to charge a payment method. SPTs are created by clients using the Stripe API and consumed by servers to process payments. Both the Client and Server require a Stripe

account. In the Stripe API, SPTs are referenced as `shared_payment_granted_token` on PaymentIntent creation. Learn more: https://docs.stripe.com/agentic-commerce/concepts/shared-payment-tokens

Business Network Profile A Stripe profile is a business's public identity on Stripe. With a Stripe profile, businesses can find, verify, and connect with each other on Stripe. Learn more: https://docs.stripe.com/get-started/account/profile

Payment Intent A Stripe API object that tracks the lifecycle of a customer payment, from creation through settlement. Not to be confused with the HTTP Payment Auth protocol's "payment intent" parameter. Learn more: https://docs.stripe.com/payments/payment-intents

4. Intent Identifier

This specification defines the following intent for the `stripe` payment method:

```
charge
```

The intent identifier is case-sensitive and **MUST** be lowercase.

5. Intent: "charge"

A one-time payment of the specified amount. The server processes the payment immediately upon receiving the SPT.

Fulfillment mechanism:

1. **Stripe Payment Token (SPT)**: The payer creates an SPT using the Stripe API, which the server uses to create a PaymentIntent via Stripe.

6. Request Schema

The `request` parameter in the `WWW-Authenticate` challenge contains a base64url-encoded JSON object with the following fields. The JSON **MUST** be serialized using JSON Canonicalization Scheme (JCS) [RFC8785] before base64url encoding, per Section 5.1.2 of [I-D.httpauth-payment].

6.1. Shared Fields

Field	Type	Required	Description
`amount`	string	**REQUIRED**	Amount in smallest currency unit (e.g., cents), encoded as a string
`currency`	string	**REQUIRED**	Three-letter ISO currency code (e.g., `"usd"`)

Field	Type	Required	Description
`description`	string	**OPTIONAL**	Human-readable payment description
`externalId`	string	**OPTIONAL**	Merchant's identifier (e.g., order ID, cart ID)
`expires`	string	**OPTIONAL**	Expiry timestamp in [RFC3339] format
`recipient`	string	**OPTIONAL**	Payment recipient identifier

Table 1

6.2. Method Details

Field	Type	Required	Description
`methodDetails.networkId`	string	**REQUIRED**	Stripe Business Network Profile ID
`methodDetails.paymentMethodTypes`	[]string	**REQUIRED**	The list of payment method types that the seller can process.
`methodDetails.metadata`	object	**OPTIONAL**	Key-value pairs for additional context

Table 2

Example:

```
{
  "amount": "5000",
  "currency": "usd",
  "description": "Premium API access for 1 month",
  "externalId": "order_12345",
  "methodDetails": {
    "networkId": "profile_1MqDcVKA5fEO2tZvKQm9g8Yj",
    "paymentMethodTypes": ["card", "link"]
  }
}
```

The client fulfills this by creating an SPT using Stripe:

```
const spt = await stripe.sharedPayment.issuedTokens.create({
  payment_method: 'pm_123',
  usage_limits: {
    currency: 'usd',
    max_amount: 5000,
    expires_at: Timestamp
  },
  seller_details: {
    networkId: 'profile_123'
  }
});
// Returns: { id: 'spt_1N...' }
```

7. Credential Schema

The Payment credential is a base64url-encoded JSON object containing `challenge` and `payload` fields per Section 5.2 of [I-D.httpauth-payment]. For Stripe charge, the `payload` object contains the following fields:

Field	Type	Required	Description
`spt`	string	**REQUIRED**	Stripe Payment Token ID (starts with `spt_`)
`externalId`	string	**OPTIONAL**	Client's reference ID

Table 3

Example:

```
{
  "spt": "spt_1N4Zv32eZvKYlo2CPhVPkJlW",
  "externalId": "client_order_789"
}
```

8. Verification Procedure

Servers **MUST** verify Payment credentials for charge intent:

1. Extract the `spt` from the credential payload
2. Verify the challenge ID matches the one issued
3. Verify the challenge has not expired
4. Verify the SPT has not been previously used (replay protection)
5. Validate the SPT exists and is valid via Stripe API (optional pre-check)

8.1. Challenge Binding

Servers **MUST** verify that the credential corresponds to the exact challenge issued. This includes validating:

- Challenge ID
- Amount (if specified in request)
- Currency
- Business Network (if specified)
- Any custom metadata

9. Settlement Procedure

Synchronous settlement:

1. Server receives and verifies the credential (Section 8)
2. Server creates a Stripe PaymentIntent with `confirm: true` and the SPT as `shared_payment_granted_token`:

```
const paymentIntent = await stripe.paymentIntents.create({
  amount: Number(request.amount),
  currency: request.currency,
  shared_payment_granted_token: credential.spt,
  confirm: true,
  automatic_payment_methods: {
    enabled: true,
    allow_redirects: 'never'
  },
  metadata: { challenge_id: challenge.id }
}, {
  idempotencyKey: `${challenge.id}_${credential.spt}`
});
```

1. Server **MUST** verify the PaymentIntent `status` is `"succeeded"` before returning 200 with `Payment-Receipt` header
2. If the PaymentIntent fails or requires additional action, server returns 402 with a new challenge

Idempotency:

Servers **SHOULD** include an idempotency key derived from the challenge ID and SPT when creating PaymentIntents. This prevents duplicate charges if the client retries a request.

Settlement timing:

Stripe processes fund transfers asynchronously. Servers **SHOULD** return 200 immediately after PaymentIntent confirmation (status `"succeeded"`), even if final fund settlement to the merchant is pending.

9.1. Receipt Generation

Upon successful settlement, servers **MUST** return a `Payment-Receipt` header per Section 5.3 of [I-D.httpauth-payment]. Servers **MUST NOT** include a `Payment-Receipt` header on error responses; failures are communicated via HTTP status codes and Problem Details.

The receipt payload for Stripe charge:

Field	Type	Description
`method`	string	`"stripe"`
`reference`	string	Stripe PaymentIntent ID (e.g., `"pi_1N4..."`)
`status`	string	`"success"`
`timestamp`	string	[RFC3339] confirmation time
`externalId`	string	**OPTIONAL**. Echoed from credential payload

Table 4

10. Security Considerations

10.1. SPT Single-Use Constraint

SPTs are single-use tokens. Stripe automatically prevents SPT reuse at the API level, and idempotency keys (Section 9) prevent duplicate PaymentIntent creation. Servers **MUST** enforce single-use challenge IDs per Section 5.1.3 of [I-D.httpauth-payment] and **SHOULD** use Stripe idempotency keys to prevent repeated charges. Servers **MAY** additionally maintain a local replay cache of consumed challenge IDs.

10.2. Amount Verification

Clients **MUST** verify the payment amount in the challenge matches their expectation before creating an SPT. The SPT itself does not encode the amount, so clients must trust the challenge parameters.

Verification checklist:

1. Verify the `amount` matches the expected cost
2. Verify the `currency` matches the expected currency
3. Verify the `description` matches the expected service
4. Verify the challenge hasn't expired

5. Verify the server's identity (TLS certificate validation)

10.3. PCI DSS Compliance

Stripe's SPT model ensures clients never handle raw payment method details, significantly reducing PCI DSS compliance scope.

10.4. HTTPS Requirement

All communication **MUST** use TLS 1.2 or higher. Stripe Payment Tokens **MUST** only be transmitted over HTTPS connections.

11. IANA Considerations

11.1. Payment Intent Registration

This specification registers the "charge" intent for the "stripe" payment method in the Payment Intent Registry per Section 13.4 of [I-D.httpauth-payment]:

- **Intent**: charge
- **Method**: stripe
- **Specification**: [this document]

Contact: Stripe (stevekaliski@stripe.com) and Tempo Labs (brendan@tempo.xyz)

12. References

12.1. 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>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/info/rfc7235>.

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

12.2. Informative References

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

Appendix A. ABNF Collected

```
stripe-charge-challenge = "Payment" 1*SP
  "id=" quoted-string ","
  "realm=" quoted-string ","
  "method=" DQUOTE "stripe" DQUOTE ","
  "intent=" DQUOTE "charge" DQUOTE ","
  "request=" base64url-nopad

stripe-charge-credential = "Payment" 1*SP base64url-nopad

; Base64url encoding without padding per RFC 4648 Section 5
base64url-nopad = 1*( ALPHA / DIGIT / "-" / "_" )
```

Appendix B. Examples

B.1. Charge Example (HTTP Transport)

Step 1: Client requests resource

```
GET /api/generate HTTP/1.1
Host: api.example.com
```

Step 2: Server issues payment challenge

```
HTTP/1.1 402 Payment Required
WWW-Authenticate: Payment id="ch_1a2b3c4d5e",
  realm="api.example.com",
  method="stripe",
  intent="charge",

request="eyJhbW91bnQiOiI1MDAwIiwiY3VycmVuY3kiOiJ1c2QiLCJkZXNjcmlwdGlvbiI6IkFJ
IGdlbmVyYXRpb24ifQ"
Cache-Control: no-store
Content-Type: application/json

{
  "type": "https://paymentauth.org/problems/payment-required",
  "title": "Payment Required",
  "status": 402,
  "detail": "This resource requires payment"
}
```

Decoded request: ~~~ json { "amount": "5000", "currency": "usd", "description": "AI generation" } ~~~

Step 3: Client creates SPT and submits credential

```
GET /api/generate HTTP/1.1
Host: api.example.com
Authorization: Payment
eyJjaGFsbGVuZ2UiOnsiaWQiOiJjaF8xYTJiM2M0ZDVlIiwicmVhbG0iOiJhcGkuZXhhbXBsZS5jb
20iLCJtZXRob2QiOiJzdHJpcGUiLCJpbnRlbnQiOiJjaGFyZ2UiLCJyZXF1ZXN0IjoiZXlKaGJXOT
FiblFpT2lJMU1EQXdJaXdpWTNWeWNtVnVZM2tpT2lKMWMyUWlMQ0prWlhOamNtbHdkR2x2YmlJNkl
rRkpJR2RsYm1WeVlYUnBiMjRpZlEiLCJleHBpcmVzIjoiMjAyNS0wMS0xNVQxMjowNTowMFoifSwi
cGF5bG9hZCI6eyJzcHQiOiJzcHRfMU40WnYzMmVadktZbG8yQ1BoVlBrSmxXIn19
```

Decoded credential: ~~~ json { "challenge": { "id": "ch_1a2b3c4d5e", "realm": "api.example.com", "method": "stripe", "intent": "charge", "request": "eyJhbW91bnQiOiI1MDAwIiwiY3VycmVuY3kiOiJ1c2QiLCJkZXNjcmlwdGlvbiI6IkFJIGdlbmVyYXRpb24ifQ", "expires": "2025-01-15T12:05:00Z" }, "payload": { "spt": "spt_1N4Zv32eZvKYlo2CPhVPkJlW" } } ~~~

Step 4: Server processes payment and returns resource

```
HTTP/1.1 200 OK
Payment-Receipt:
eyJtZXRob2QiOiJzdHJpcGUiLCJyZWZlcmVuY2UiOiJwaV8xTjRadjMyZVp2S1lsbzJDUGhWUGtKb
FciLCJzdGF0dXMiOiJzdWNjZXNzIiwidGltZXN0YW1wIjoiMjAyNS0wMS0xNVQxMjowNDozMloifQ
Cache-Control: private
Content-Type: text/plain

Here is your generated content...
```

Decoded receipt: ~~~ json { "method": "stripe", "reference": "pi_1N4Zv32eZvKYlo2CPhVPkJlW", "status": "success", "timestamp": "2025-01-15T12:04:32Z" } ~~~

Appendix C. Acknowledgements

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

Authors' Addresses

Brendan Ryan
Tempo Labs
Email: brendan@tempo.xyz

Steve Kaliski
Stripe
Email: stevekaliski@stripe.com