Workgroup:	Network Working Group
Internet-Draft:	draft-payment-intent-charge-00
Published:	17 April 2026
Intended Status:	Informational
Expires:	19 October 2026
Authors:	J. Moxey	B. Ryan	T. Meagher
---	---	---	---
	Tempo Labs	*Tempo Labs*	*Tempo Labs*

Charge Intent for HTTP Payment Authentication

Abstract

This document defines the "charge" payment intent for use with the Payment HTTP Authentication Scheme [I-D.httpauth-payment]. The "charge" intent represents a one-time payment where the payer provides proof of payment immediately in exchange for resource access.

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 19 October 2026.

Copyright Notice

Table of Contents

1. Introduction

The "charge" intent is the most fundamental payment pattern: a one-time exchange of payment for resource access. The payer provides proof of payment (or a signed authorization to collect payment), and the server grants access to the requested resource.

This intent applies to any payment method that supports immediate payment verification, including:

- Invoice-based systems (preimage revelation)
- Signed transaction authorization
- Token-based payment confirmation
- Traditional payment processor confirmation

1.1. Relationship to Payment Methods

This document defines the abstract semantics of the "charge" intent. Payment method specifications define how to implement this intent using their specific payment infrastructure.

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

Charge A one-time payment where the payer provides proof of payment immediately in exchange for resource access.

Base Units The smallest denomination of a currency or asset. For USD, this is cents (1/100). For tokens, this is the smallest transferable unit defined by the token's decimal precision.

4. Intent Semantics

4.1. Definition

The "charge" intent represents a request for immediate, one-time payment of a specified amount in exchange for resource access.

4.2. Properties

Property	Value
Intent Identifier	`charge`
Payment Timing	Immediate (before or with request)
Idempotency	Single-use per challenge
Reversibility	Method-dependent

Table 1

4.3. Flow

1. Server issues a 402 response with `intent="charge"`
2. Client fulfills the payment (method-specific)
3. Client submits credential with proof of payment
4. Server verifies payment and grants access
5. Server returns `Payment-Receipt` header

4.4. Atomicity

The "charge" intent implies atomic exchange: the server **SHOULD NOT** provide partial access if payment verification fails. Either the full resource is provided (payment succeeded) or access is denied (payment failed).

5. Request Schema

The `request` parameter for a "charge" 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) and base64url-encoded without padding per [I-D.httpauth-payment].

5.1. Shared Fields

All payment methods implementing the "charge" intent **MUST** support these shared fields, enabling clients to parse and display payment requests consistently across methods. Payment methods **MAY** elevate **OPTIONAL** fields to **REQUIRED** in their method specification (e.g., `recipient` and `expires` are **REQUIRED** for blockchain methods).

5.1.1. Required Fields

Field	Type	Description
`amount`	string	Payment amount in base units (smallest denomination)
`currency`	string	Currency or asset identifier (see Section 5.2)

Table 2

5.1.2. Optional Fields

Field	Type	Description
`recipient`	string	Payment recipient in method-native format
`description`	string	Human-readable payment description
`externalId`	string	Merchant's reference (order ID, invoice number, etc.)
`methodDetails`	object	Method-specific extension data

Table 3

Challenge expiry is conveyed by the `expires` auth-param in `WWW-Authenticate` per [I-D.httpauth-payment], using [RFC3339] format. Request objects **MUST NOT** duplicate the expiry value.

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 in the `methodDetails` object. These fields are method-specific and **MUST** be documented in the payment method specification. Clients that do not recognize a payment method **SHOULD** ignore `methodDetails` but **MUST** still be able to display the shared fields to users.

5.4. Examples

5.4.1. Traditional Payment Processor (Stripe)

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

5.4.2. Blockchain Payment (Tempo)

```
{
  "amount": "1000000",
  "currency": "0x20c0000000000000000000000000000000000000",
  "recipient": "0x742d35Cc6634C0532925a3b844Bc9e7595f8fE00",
  "methodDetails": {
    "chainId": 42431,
    "feePayer": true
  }
}
```

5.4.3. Lightning Network

```
{
  "amount": "100000",
  "currency": "sat",
  "methodDetails": {
    "invoice": "lnbc1000n1pj9..."
  }
}
```

Payment method specifications define the complete `methodDetails` schema for their implementation of the "charge" intent.

6. Credential Requirements

6.1. Payload

The credential structure follows [I-D.httpauth-payment], containing `challenge`, `payload`, and an optional `source` field identifying the payer. The `payload` for a "charge" intent **MUST** contain proof that payment has been made or authorized. The proof type is method-specific:

Proof Type	Description	Example Methods
Preimage	Hash preimage proving invoice payment	Lightning
Signature	Signed transaction authorization	Tempo, EVM
Confirmation	Payment processor confirmation identifier	Stripe
Ledger transaction	Transaction hash on public ledger	Bitcoin, Ethereum

Table 5

6.2. Single-Use

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

7. Verification

7.1. Server Responsibilities

Servers verifying a "charge" credential **MUST**:

1. Verify the `id` matches an outstanding challenge
2. Verify the challenge has not expired
3. Verify the payment proof using method-specific procedures
4. Verify the payment amount matches the request
5. Verify the payment recipient matches the request

7.2. Settlement

Settlement semantics differ by method:

- **Immediate settlement**: Payment is final upon verification (e.g., Lightning preimage, confirmed blockchain transaction)
- **Deferred settlement**: Server submits payment after verification (e.g., signed authorization submitted to chain)

- **Processor settlement**: External processor handles settlement (e.g., Stripe PaymentIntent)

8. Security Considerations

8.1. Amount Verification

Clients **MUST** verify the requested amount is appropriate for the resource before authorizing payment. Malicious servers could request excessive amounts.

8.2. Recipient Verification

Clients **SHOULD** verify the payment recipient when possible. Not all payment methods expose an explicit recipient (e.g., processor-based methods like Stripe route payments internally). For methods that do expose a recipient (e.g., blockchain addresses), clients **SHOULD** warn users about unknown recipients.

8.3. Replay Protection

Servers **MUST** implement replay protection. Each challenge `id` **MUST** be single-use. Servers **MUST NOT** accept the same credential twice.

8.4. Finality

The finality of a "charge" payment depends on the payment method:

- Some methods provide instant finality (Lightning)
- Some methods may have delayed finality (blockchain confirmations)
- Some methods may be reversible (card chargebacks)

Servers **SHOULD** understand the finality guarantees of their accepted payment methods and adjust resource access accordingly.

8.5. Transport Security

All Payment authentication flows **MUST** use TLS 1.2 or later per [I-D.httpauth-payment]. Payment credentials contain sensitive authorization data that could result in financial loss if intercepted.

8.6. Currency Verification

Clients **MUST** verify the `currency` field matches their expectation before authorizing payment. Malicious servers could request payment in a different currency or token than expected.

9. IANA Considerations

9.1. Payment Intent Registration

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

Intent	Description	Reference
`charge`	One-time immediate payment	This document

Table 6

Contact: Tempo Labs (contact@tempo.xyz)

10. 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>.

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

Authors' Addresses

Jake Moxey
Tempo Labs
Email: jake@tempo.xyz

Brendan Ryan
Tempo Labs
Email: brendan@tempo.xyz

Tom Meagher
Tempo Labs
Email: tom@tempo.xyz