Workgroup:	Network Working Group
Internet-Draft:	draft-tempo-charge-00
Published:	3 March 2026
Intended Status:	Informational
Expires:	4 September 2026
Authors:	J. Moxey	B. Ryan	T. Meagher
---	---	---	---
	Tempo Labs	*Tempo Labs*	*Tempo Labs*

Tempo charge Intent for HTTP Payment Authentication

Abstract

This document defines the "charge" intent for the "tempo" payment method in the Payment HTTP Authentication Scheme [I-D.httpauth-payment]. It specifies how clients and servers exchange one-time TIP-20 token transfers on the Tempo blockchain.

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

Table of Contents

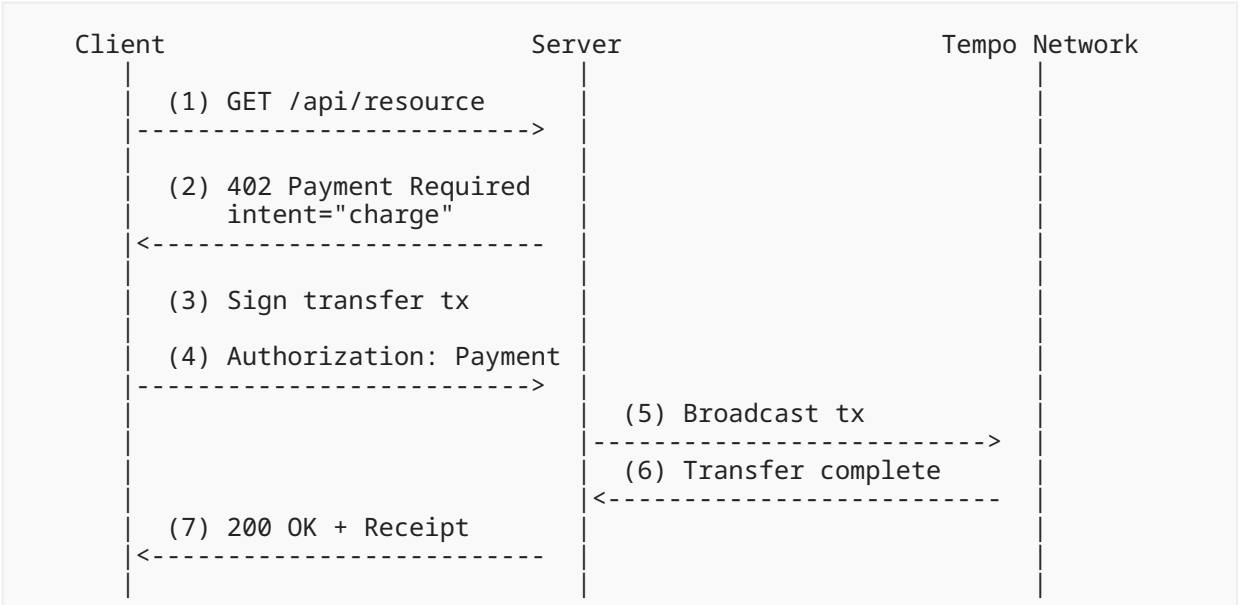
1. Introduction

The `charge` intent represents a one-time payment of a specified amount. The server may submit the signed transaction any time before the `expires` timestamp.

This specification defines the request schema, credential formats, and settlement procedures for charge transactions on Tempo.

1.1. Charge Flow

The following diagram illustrates the Tempo charge 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

TIP-20 Tempo's enshrined token standard, implemented as precompiles rather than smart contracts. TIP-20 tokens use 6 decimal places and provide `transfer`, `transferWithMemo`, `transferFrom`, and `approve` operations.

Tempo Transaction An EIP-2718 transaction with type prefix `0x76`, supporting batched calls, multiple signature types (secp256k1, P256, WebAuthn), 2D nonces, and validity windows.

2D Nonce Tempo's nonce system where each account has multiple independent nonce lanes (`nonce_key`), enabling parallel transaction submission.

Fee Payer An account that pays transaction fees on behalf of another account. Tempo Transactions support fee payment via a separate signature domain (`0x78`), allowing the server to pay for fees while the client only signs the payment authorization.

4. Request Schema

The `request` parameter in the `WWW-Authenticate` challenge contains a base64url-encoded JSON object.

4.1. Shared Fields

Field	Type	Required	Description
`amount`	string	**REQUIRED**	Amount in base units (stringified number)
`currency`	string	**REQUIRED**	TIP-20 token address (e.g., `"0x20c0..."`)
`recipient`	string	**REQUIRED**	Recipient address
`expires`	string	**REQUIRED**	Expiry timestamp in [RFC3339] format

Table 1

The `expires` field in the request JSON **MUST** match the `expires` auth-param in the `WWW-Authenticate` header when both are present. Clients **SHOULD** use the auth-param value for expiry checks.

4.2. Method Details

Field	Type	Required	Description
`methodDetails.chainId`	number	**OPTIONAL**	Tempo chain ID (default: 42431)
`methodDetails.feePayer`	boolean	**OPTIONAL**	If `true`, server pays transaction fees (default: `false`)

Table 2

Example:

```
{
  "amount": "1000000",
  "currency": "0x20c0000000000000000000000000000000000000",
  "recipient": "0x742d35Cc6634C0532925a3b844Bc9e7595f8fE00",
  "expires": "2025-01-06T12:00:00Z",
  "methodDetails": {
    "chainId": 42431,
    "feePayer": true
  }
}
```

The client fulfills this by signing a Tempo Transaction with `transfer(recipient, amount)` or `transferWithMemo(recipient, amount, memo)` on the specified `currency` (token address), with `validBefore` set to `expires`. The client **SHOULD** use a dedicated `nonceKey` (2D nonce lane) for payment transactions to avoid blocking other account activity if the transaction is not immediately settled.

If `methodDetails.feePayer` is `true`, the client signs with `fee_payer_signature` set to `0x00` and `fee_token` empty, allowing the server to sponsor fees. If `feePayer` is `false` or omitted, the client **MUST** set `fee_token` and pay fees themselves.

5. Credential Schema

The credential in the `Authorization` header contains a base64url-encoded JSON object per Section 5.2 of [I-D.httpauth-payment].

5.1. Credential Structure

Field	Type	Required	Description
`challenge`	object	**REQUIRED**	Echo of the challenge from the server
`payload`	object	**REQUIRED**	Tempo-specific payload object

Field	Type	Required	Description
`source`	string	**OPTIONAL**	Payer identifier as a DID (e.g., `did:pkh:eip155:42431:0x...`)

Table 3

The `source` field, if present, **SHOULD** use the `did:pkh` method with the Tempo chain ID (42431 for Moderato testnet) and the payer's Ethereum address.

5.2. Transaction Payload (type="transaction")

When `type` is `"transaction"`, `signature` contains the complete signed Tempo Transaction (type 0x76) serialized as RLP and hex-encoded with `0x` prefix. The transaction **MUST** contain a `transfer(recipient, amount)` or `transferWithMemo(recipient, amount, memo)` call on the TIP-20 token.

Field	Type	Required	Description
`signature`	string	**REQUIRED**	Hex-encoded RLP-serialized signed transaction
`type`	string	**REQUIRED**	`"transaction"`

Table 4

Example:

```
{
  "challenge": {
    "id": "kM9xPqWvT2nJrHsY4aDfEb",
    "realm": "api.example.com",
    "method": "tempo",
    "intent": "charge",
    "request": "eyJ...",
    "expires": "2025-02-05T12:05:00Z"
  },
  "payload": {
    "signature": "0x76f901...signed transaction bytes...",
    "type": "transaction"
  },
  "source": "did:pkh:eip155:42431:0x1234567890abcdef1234567890abcdef12345678"
}
```

5.3. Hash Payload (type="hash")

When `type` is `"hash"`, the client has already broadcast the transaction to the Tempo network. The `hash` field contains the transaction hash for the server to verify onchain.

Field	Type	Required	Description
`hash`	string	**REQUIRED**	Transaction hash with `0x` prefix
`type`	string	**REQUIRED**	`"hash"`

Table 5

Example:

```
{
  "challenge": {
    "id": "kM9xPqWvT2nJrHsY4aDfEb",
    "realm": "api.example.com",
    "method": "tempo",
    "intent": "charge",
    "request": "eyJ...",
    "expires": "2025-02-05T12:05:00Z"
  },
  "payload": {
    "hash":
"0x1a2b3c4d5e6f7890abcdef1234567890abcdef1234567890abcdef1234567890",
    "type": "hash"
  },
  "source": "did:pkh:eip155:42431:0x1234567890abcdef1234567890abcdef12345678"
}
```

6. Fee Payment

When a request includes `feePayer: true`, the server commits to paying transaction fees on behalf of the client.

6.1. Server-Paid Fees

When `feePayer: true`:

1. **Client signs with placeholder**: The client signs the Tempo Transaction with `fee_payer_signature` set to a placeholder value (`0x00`) and `fee_token` left empty. The client uses signature domain `0x76`.
2. **Server receives credential**: The server extracts the client-signed transaction from the credential payload.
3. **Server adds fee payment signature**: The server selects a `fee_token` (any USD-denominated TIP-20 stablecoin) and signs the transaction using signature domain `0x78`. This signature commits to the transaction including the `fee_token` and client's address.
4. **Server broadcasts**: The final transaction contains both signatures:
   - Client's signature (authorizing the payment)
   - Server's `fee_payer_signature` (committing to pay fees)

6.2. Client-Paid Fees

When `feePayer: false` or omitted, the client **MUST** set `fee_token` to a valid USD TIP-20 token address and pay fees themselves. The server broadcasts the transaction as-is without adding a fee payer signature.

6.3. Server Requirements

When acting as fee payer, servers:

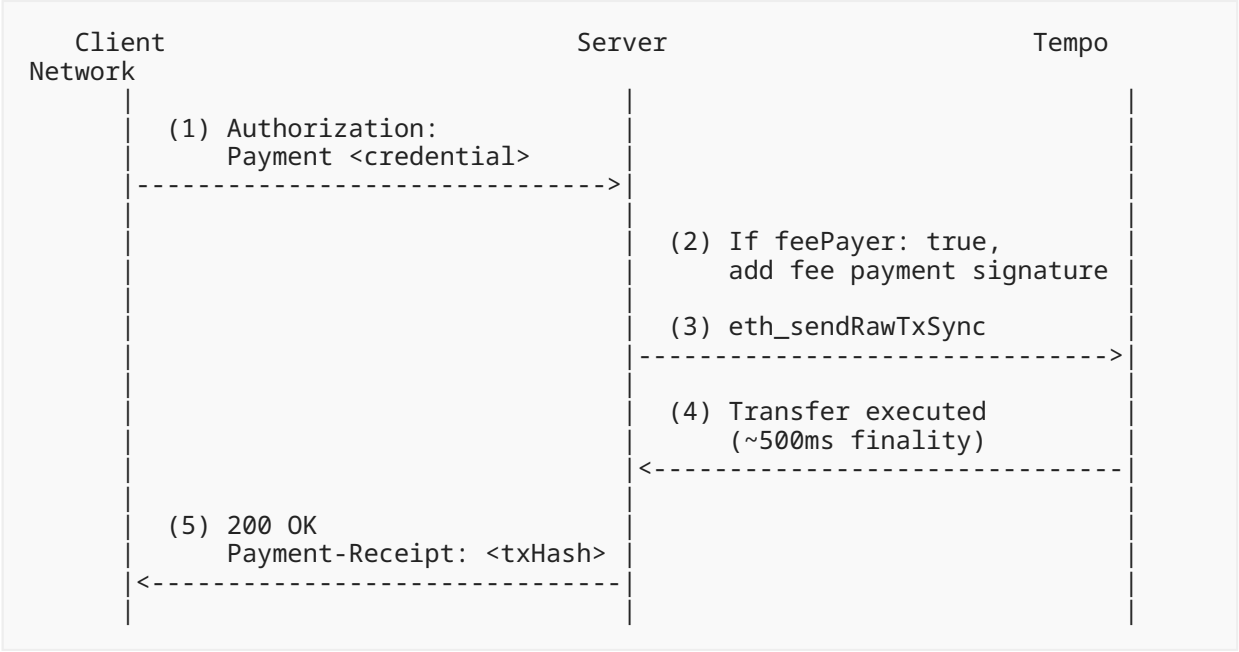
- **MUST** maintain sufficient balance of a USD TIP-20 token to pay transaction fees
- **MAY** use any USD-denominated TIP-20 token with sufficient AMM liquidity as the fee token
- **MAY** recover fee costs through pricing or other business logic

6.4. Client Requirements

- When `feePayer: true`: Clients **MUST** sign with `fee_payer_signature` set to `0x00` and `fee_token` empty or `0x80` (RLP null)
- When `feePayer: false` or omitted: Clients **MUST** set `fee_token` to a valid USD TIP-20 token and have sufficient balance to pay fees

7. Settlement Procedure

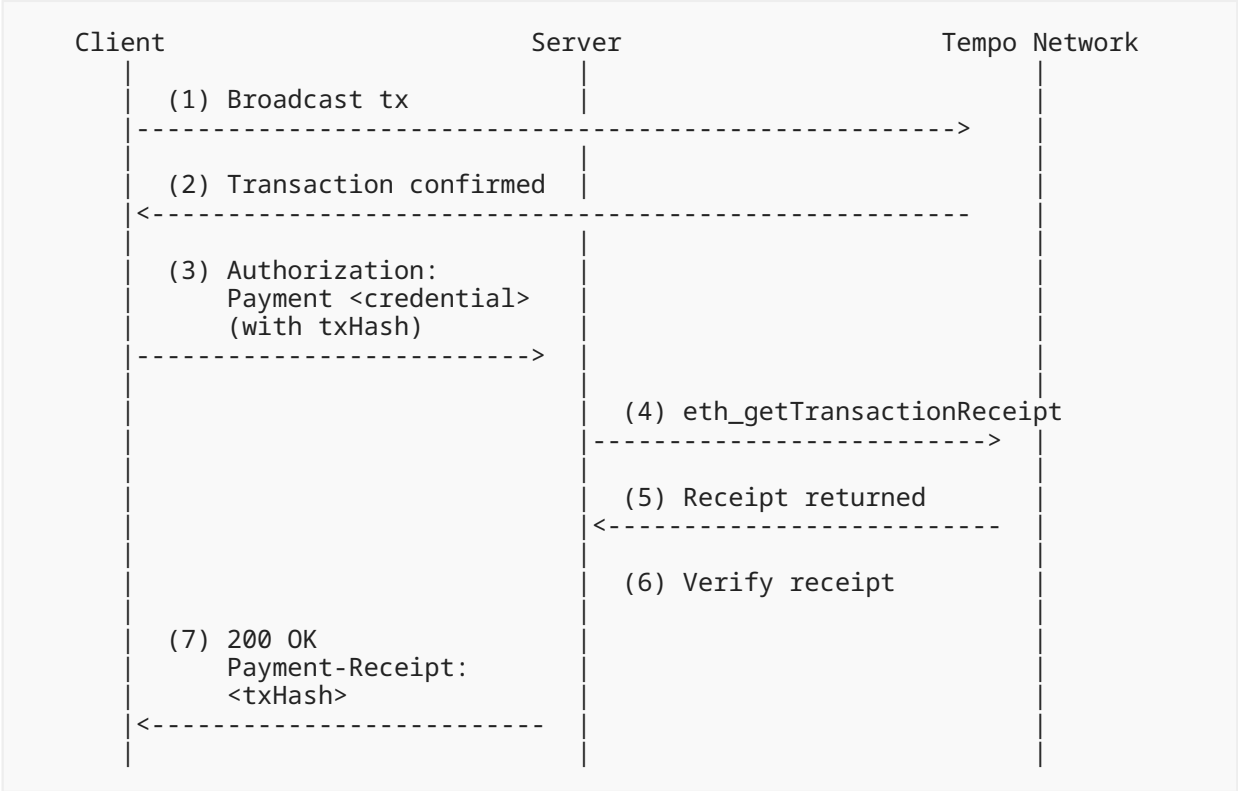
For `intent="charge"` fulfilled via transaction, the client signs a transaction containing a `transfer` or `transferWithMemo` call. If `feePayer: true`, the server adds its fee payer signature before broadcasting:



1. Client submits credential containing signed `transfer` or `transferWithMemo` transaction
2. If `feePayer: true`, server adds fee sponsorship (signs with `0x78` domain)
3. Server broadcasts transaction to Tempo
4. Transaction included in block with immediate finality (~500ms)
5. Server returns receipt with the transaction digest

7.1. Hash Settlement

For credentials with `type="hash"`, the client has already broadcast the transaction. The server verifies the transaction onchain:



Limitations:

- Cannot be used with `feePayer: true` (client must pay their own fees)
- Server cannot modify or enhance the transaction

7.2. 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 Tempo charge:

Field	Type	Description
`method`	string	`"tempo"`
`reference`	string	Transaction hash of the settlement transaction
`status`	string	`"success"`
`timestamp`	string	[RFC3339] settlement time

Table 6

8. Security Considerations

8.1. Transaction Replay

Tempo Transactions include chain ID, nonce, and optional `validBefore`/ `validAfter` timestamps that prevent replay attacks:

- Chain ID binding prevents cross-chain replay
- Nonce consumption prevents same-chain replay
- Validity windows limit temporal replay windows

8.2. Amount Verification

Clients **MUST** parse and verify the `request` payload before signing:

1. Verify `amount` is reasonable for the service
2. Verify `currency` is the expected token address
3. Verify `recipient` is controlled by the expected party

8.3. Server-Paid Fees

Servers acting as fee payers accept financial risk in exchange for providing a seamless payment experience.

Denial of Service: Malicious clients could submit valid-looking credentials that fail onchain, causing the server to pay fees without receiving payment. Servers **SHOULD** implement rate limiting and **MAY** require client authentication before accepting payment credentials.

Fee Token Exhaustion: Servers **MUST** monitor their fee token balance and reject new payment requests when balance is insufficient.

9. IANA Considerations

9.1. Payment Method Registration

This document registers the following payment method in the "HTTP Payment Methods" registry established by [I-D.httpauth-payment]:

Method Identifier	Description	Reference
`tempo`	Tempo blockchain TIP-20 token transfer	This document

Table 7

Contact: Tempo Labs (contact@tempo.xyz)

9.2. Payment Intent Registration

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

Intent	Applicable Methods	Description	Reference
`charge`	`tempo`	One-time TIP-20 transfer	This document

Table 8

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>.

[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-ietf-httpauth-payment/>.

10.2. Informative References

[EIP-2718] Zoltu, M., "Typed Transaction Envelope", October 2020, <https://eips.ethereum.org/EIPS/eip-2718>.

[EIP-55] Buterin, V., "Mixed-case checksum address encoding", January 2016, <https://eips.ethereum.org/EIPS/eip-55>.

[TEMPO-TX-SPEC] Tempo Labs, "Tempo Transaction Specification", n.d., <https://docs.tempo.xyz/protocol/transactions/spec-tempo-transaction>.

Appendix A. Example

Challenge:

```
HTTP/1.1 402 Payment Required
WWW-Authenticate: Payment id="kM9xPqWvT2nJrHsY4aDfEb",
  realm="api.example.com",
  method="tempo",
  intent="charge",

request="eyJhbW91bnQiOiIxMDAwMDAwIiwiY3VycmVuY3kiOiIweDIwYzAwMDAwMDAwMDAwMDAw
MDAwMDAwMDAwMDAwMDAwMDAwMDEiLCJyZWNpcGllbnQiOiIweDc0MmQzNUNjNjYzNEMwNTMyO
TI1YTNiODQ0QmM5ZTc1OTVmOGZFMDAiLCJleHBpcmVzIjoiMjAyNS0wMS0wNlQxMjowMDowMFoiLC
JtZXRob2REZXRhaWxzIjp7ImNoYWluSWQiOjQyNDMxfX0"
```

The `request` decodes to:

```
{
  "amount": "1000000",
  "currency": "0x20c0000000000000000000000000000000000000",
  "recipient": "0x742d35Cc6634C0532925a3b844Bc9e7595f8fE00",
  "expires": "2025-01-06T12:00:00Z",
  "methodDetails": {
    "chainId": 42431
  }
}
```

This requests a transfer of 1.00 pathUSD (1000000 base units).

Credential:

```
GET /api/resource HTTP/1.1
Host: api.example.com
Authorization: Payment
eyJjaGFsbGVuZ2UiOnsiaWQiOiJrTTl4UHFXdlQybkpySHNZNGFEZkViIn0sInBheWxvYWQiOnsic
2lnbmF0dXJlIjoiMHg3NmY5MDEuLi4iLCJ0eXBlIjoidHJhbnNhY3Rpb24ifSwic291cmNlIjoiZG
lkOnBraDplaXAxNTU6NDI0MzE6MHgxMjM0NTY3ODkwYWJjZGVmMTIzNDU2Nzg5MGFiY2RlZjEyMzQ
1Njc4In0
```

Appendix B. Acknowledgements

The authors thank the Tempo 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