MEMEH RTPO

MEMEH RTPO Fintech and Payment Facilitator API Guide

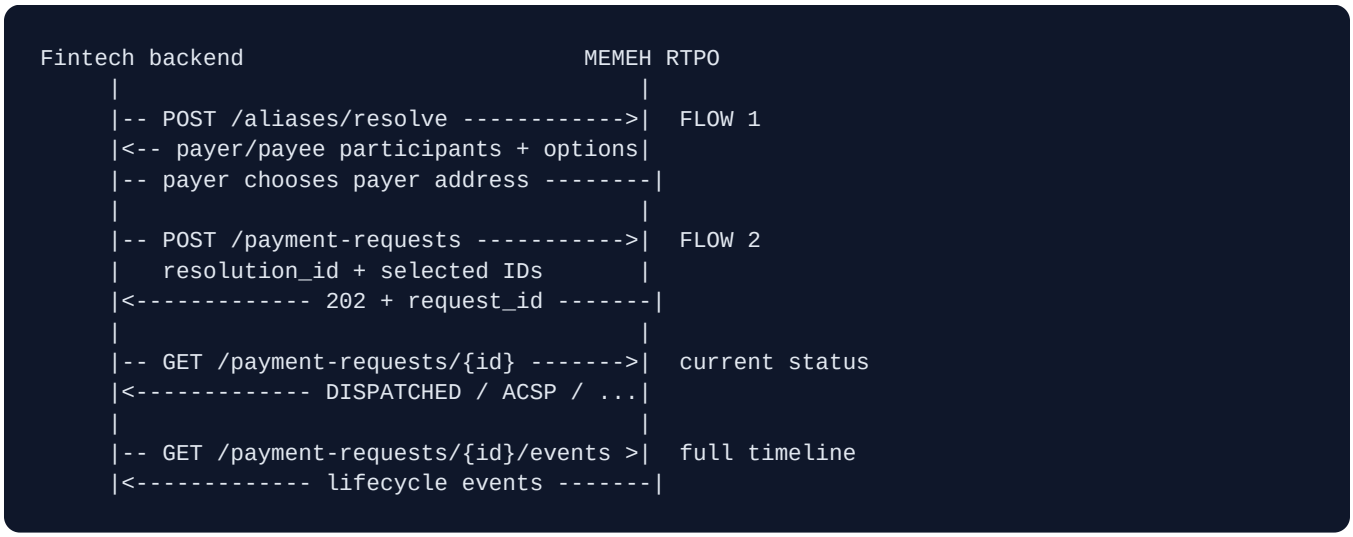
Requests, responses, payment status, lifecycle events, testing, and production integration

Updated 2026-08-08 • Core baseline 7688dff • Enterprise documentation pack

Start Here - Two Flows and Four API Calls

A fintech or payment facilitator first resolves both parties through CAS V2, lets the payer choose an eligible payer financial address, and only then submits the Request-to-Pay. Two read APIs provide the asynchronous status and event history.

Step	Method and path	Purpose
Flow 1	POST /api/v1/aliases/resolve	Resolve payer/payee participants and selectable financial addresses
Flow 2	POST /api/v1/payment-requests	Submit the payer/payee address selections and create RTP
Read 1	GET /api/v1/payment-requests/{request_id}	Read the latest payment status
Read 2	GET /api/v1/payment-requests/{request_id}/events	Read the complete lifecycle history



The fintech does not call /api/v1/participant-status-events. That endpoint is called by the participating provider's adapter. MEMEH RTPO converts each accepted provider update into the status and event history read by the fintech.

Flow 1 and Flow 2 - Resolve, Select, Then Request

The following is the required server-to-server sequence. Replace the endpoint, credential, merchant identity, and approved test aliases with values issued during onboarding.

```
MEMEH_BASE_URL=https://uat-api.memeh.example
MEMEH_API_KEY=<fintech-api-key>

curl -sS -X POST \
  "$MEMEH_BASE_URL/api/v1/aliases/resolve" \
  -H "Authorization: Bearer $MEMEH_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "payer_alias": "+23276000001",
    "payee_alias": "govpay@merchant.sl",
    "amount": "250.00",
    "currency": "SLE",
    "reference": "GOV-2026-000184"
  }'
```

MEMEH RTPO resolves both aliases through CAS V2. Each finaddresses item contains a masked account or wallet plus the CAS servicer identity of the participating bank, mobile money operator, or fintech.

```
HTTP/1.1 200 OK

{
  "verified": true,
  "resolution_id": "cas-resolution-000184",
  "source": "CAS_V2",
  "expires_at": "2026-08-05T10:35:00Z",
  "payer": {
    "alias": "+23276000001",
    "alias_type": "MSISDN",
    "status": "ACTIVE",
    "finaddresses": [
      {
        "id": "fa-payer-bank",
        "label": "BANK1SLFR",
        "masked": "Account ending 0319",
        "default": true,
        "servicer_id": "BANK1SLFR",
        "servicer_id_type": "BICFI",
        "status": "ACTIVE"
      },
      {
        "id": "fa-payer-wallet",
        "label": "MMO1SLFR",
        "masked": "Account ending 4412",
        "default": false,
        "servicer_id": "MMO1SLFR",
        "servicer_id_type": "BICFI",
        "status": "ACTIVE"
      }
    ]
  },
  "payee": {
    "alias": "govpay@merchant.sl",
    "alias_type": "EMAIL",
    "status": "ACTIVE",
    "finaddresses": [
      {
        "id": "fa-payee-govpay",
        "label": "PAYEESLFR",
        "masked": "Account ending 0999",
        "default": true,
        "servicer_id": "PAYEESLFR",
        "servicer_id_type": "BICFI",
        "status": "ACTIVE"
      }
    ]
  }
}
```

- The checkout displays payer.finaddresses as the payer's account/provider choices.
- The payer selects one payer.finaddresses[*].id; the full financial address is never sent to the browser.
- The fintech uses the merchant-selected or default payee.finaddresses[*].id for the destination.
- The fintech submits the same aliases, amount, currency, and reference in Flow 2.

Example Flow 2 request after the payer chooses the mobile-money address:

```
curl -sS -X POST \
  "$MEMEH_BASE_URL/api/v1/payment-requests" \
  -H "Authorization: Bearer $MEMEH_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "idempotency_key": "govpay-order-2026-000184-attempt-1",
    "resolution_id": "cas-resolution-000184",
    "payer_finaddress_id": "fa-payer-wallet",
    "payee_finaddress_id": "fa-payee-govpay",
    "payer_alias": "+23276000001",
    "payee_alias": "govpay@merchant.sl",
    "merchant_id": "GOVPAY-SL",
    "amount": "250.00",
    "currency": "SLE",
    "reference": "GOV-2026-000184",
    "description": "Government service fee"
  }'
```

Core resolves CAS again, confirms the resolution and selected IDs are still valid, and only then dispatches. MEMEH RTPO returns HTTP 202 after successful handoff. The fintech stores request_id for every later status call.

```
HTTP/1.1 202 Accepted

{
  "request_id": "<request-uuid>",
  "status": "DISPATCHED",
  "normalized_aliases": {
    "payer": "+23276000001",
    "payee": "govpay@merchant.sl"
  },
  "inferred_types": {
    "payer": "MSISDN",
    "payee": "EMAIL"
  },
  "dispatch": {
    "status": "DISPATCHED",
    "http_status": 202
  }
}
```

```
REQUEST_ID=<request-uuid>

curl -sS \
  -H "Authorization: Bearer $MEMEH_API_KEY" \
  "$MEMEH_BASE_URL/api/v1/payment-requests/$REQUEST_ID"

curl -sS \
  -H "Authorization: Bearer $MEMEH_API_KEY" \
  "$MEMEH_BASE_URL/api/v1/payment-requests/$REQUEST_ID/events"
```

DISPATCHED is not payment completion. Continue reading status until an approved terminal result is reached. ACSC is the successful payment state; RJCT, CANC, FAILED, and RETURNED are terminal non-success or post-completion outcomes.

Postman Collection - Import and Run

The enterprise documentation package includes an executable Postman Collection v2.1 contract and a local environment template.

Artifact	Purpose
MEMEH-Fintech-PayFac-API.postman_collection.json	Health, required Flow 1, required Flow 2, current status, and lifecycle events
MEMEH-Local.postman_environment.json	Local base URL, server-side test credential, merchant identity, aliases, amount, currency, and description

- Import both JSON files into Postman and select the MEMEH RTPO Local environment.
- Replace memeh_api_key and any test identities with values approved for the target environment.
- Run the folders in numeric order or run the full collection.
- Flow 1 generates a fresh reference and idempotency key, verifies both parties, and stores resolution_id plus the default active payer/payee option IDs.
- To test a different payer bank, wallet, mobile money operator, or fintech, run Flow 1, copy the desired payer.finaddresses item ID into the payer_finaddress_id collection variable, then run Flow 2 manually.
- Flow 2 refuses to run when the resolution or either selected address ID is missing and stores request_id after HTTP 202.
- The status and lifecycle requests use the stored request_id and assert the public response contract.

Do not place production credentials in an exported environment file. Use a Postman secret variable or the organization's approved secrets workflow and remove current values before sharing an environment export.

Framework Samples and Next.js Simulator

The developer starter suite contains small server-side reference implementations that all use the current two-flow contract. These samples are implementation accelerators, not separately supported or versioned vendor SDK products.

Artifact	What it demonstrates
Express.js	Node.js server-side client, validation, timeout, error propagation, and checkout façade
Go Fiber	Typed request models, bounded upstream reads, fixed MEMEH RTPO routing, and façade endpoints
PHP Laravel	Service container client, validated controller methods, Sanctum-protected routes, and throttling
ASP.NET Core	Typed HttpClient registration, request-size limit, validation, cancellation, and upstream forwarding
Python FastAPI	Pydantic validation, shared async HTTP client, timeout handling, and lifecycle reads
Java Servlet	Jakarta Servlet endpoint, Java HttpClient, JSON validation, and upstream response propagation
Next.js RTP Simulator	GovPay resolve/select/request checkout, server-only credentials, status polling, events, and optional provider-consent demonstration

- Configure MEMEH_BASE_URL, MEMEH_API_KEY, and the sample's timeout setting on the server only.
- Protect every local checkout façade with the integrating application's authentication and authorization before deployment.
- Use the Next.js simulator for sandbox and UAT learning; it is not a production payment application.
- Verify each downloaded ZIP against SHA256SUMS.txt before internal distribution.

Status Response - What the Fintech Reads

GET /api/v1/payment-requests/{request_id} returns the latest state only. The same response shape is used as the request progresses; the status and updated_at values change.

```
HTTP/1.1 200 OK

{
  "request_id": "<request-uuid>",
  "status": "ACSP",
  "amount": "250.00",
  "currency": "SLE",
  "reference": "GOV-2026-000184",
  "resolution_id": "<cas-resolution-id>",
  "participant": "PAYEE_PROVIDER",
  "dispatch_http_status": 202,
  "dispatch_message": "DISPATCHED",
  "created_at": "2026-08-05T10:30:00Z",
  "updated_at": "2026-08-05T10:31:15Z"
}
```

Status returned	Fintech interpretation	Action
ROUTABLE	Trusted resolution and routing completed	Keep pending
DISPATCHED	Provider adapter accepted handoff	Keep pending
RCVD / PDNG / ACSP	Provider processing is active	Keep pending and continue polling
ACSC	Provider reports payment completed	Apply approved successful-payment fulfilment
RJCT / CANC / FAILED	Request did not complete	Stop normal polling and show the applicable result
RETURNED	A completed payment was later returned	Apply the approved return/reversal workflow

Lifecycle Events - Request, Provider Update, Fintech View

Lifecycle updates originate from the participating provider, not from the fintech. The provider adapter posts one canonical event to MEMEH RTPO. MEMEH RTPO validates it, updates the latest request status, and appends a public event-history item.

```
Provider RTP platform
  -> Participant Adapter
  -> POST /api/v1/participant-status-events
  -> MEMEH RTPO updates current status and appends an event
  -> Fintech GET /payment-requests/{request_id}/events
```

Reference only - this provider event is not sent by the fintech:

```
POST /api/v1/participant-status-events
Provider adapter authentication and signing apply

{
  "schema_version": "1.0",
  "event_id": "evt-provider-000184-002",
  "instruction_id": "<instruction-id>",
  "request_id": "<request-uuid>",
  "rtp_id": "GOV-2026-000184",
  "participant_code": "PAYEE_PROVIDER",
  "source": "PAYEE_PARTICIPANT",
  "message_type": "pain.014.001.10",
  "status": "ACSP",
  "reason": "Customer approved payment",
  "occurred_at": "2026-08-05T10:31:15Z"
}
```

After MEMEH RTPO accepts that provider event, the fintech sees it through GET /events:

```
HTTP/1.1 200 OK

{
  "events": [
    {
      "id": "<event-uuid-1>",
      "request_id": "<request-uuid>",
      "instruction_id": "<instruction-id>",
      "event_type": "CREATED",
      "status": "ROUTABLE",
      "message": "<human-readable routing message>",
      "created_at": "2026-08-05T10:30:00Z"
    },
    {
      "id": "<event-uuid-2>",
      "request_id": "<request-uuid>",
      "instruction_id": "<instruction-id>",
      "event_type": "DISPATCH_RESULT",
      "status": "DISPATCHED",
      "message": "DISPATCHED",
      "created_at": "2026-08-05T10:30:01Z"
    },
    {
      "id": "<event-uuid-3>",
      "request_id": "<request-uuid>",
      "instruction_id": "<instruction-id>",
      "event_type": "PARTICIPANT_STATUS",
      "status": "ACSP",
      "message": "<human-readable lifecycle message>",
      "created_at": "2026-08-05T10:31:15Z"
    }
  ]
}
```

Field	Meaning
event_type	How the event entered MEMEH RTPO: request creation, dispatch result, or participant status
status	The payment lifecycle state at that point
instruction_id	The exact MEMEH RTPO instruction correlated with the provider update
created_at	When MEMEH RTPO stored the event; events remain append-only

The message field is descriptive support text and may evolve. Use event_type and status for application logic; do not parse message text.

The current fintech contract is polling. MEMEH RTPO does not currently send a webhook to the fintech. The fintech backend calls GET status for the latest result and GET events for the full audit timeline.

Document Control

Item	Value
Document owner	MEMEH RTPO Platform Product and Integration Team
Intended audience	Fintechs, payment facilitators, government-service platforms, billers, merchant platforms, architects, developers, QA, operations, security, and compliance
Integration boundary	Request-submitter backend to MEMEH RTPO
Current API version	v1
Classification	Enterprise guidance for approved integration teams
Authoritative rule	The deployed API contract and signed onboarding profile take precedence over examples in this guide

This is the implementation handoff for teams that originate a commercial Request-to-Pay through MEMEH RTPO. It explains how to connect, submit safely, understand asynchronous outcomes, test, operate, and prepare for production certification.

1. Executive Summary

MEMEH RTPO gives an approved request submitter one canonical API for asking a payer to approve a payment through a participating bank, mobile money operator, or fintech. The request submitter does not connect separately to every provider and does not generate IPS ISO 20022 messages.

MEMEH RTPO resolves payer and payee aliases through the Central Addressing Service (CAS), selects the trusted route to the payee provider, dispatches a canonical instruction, and records lifecycle events reported by that provider. The payee provider remains the RTP Initiator in IPS and owns customer consent, scheme messaging, account processing, and payment execution.

- Fintech responsibility: checkout experience, customer-entered aliases, merchant context, safe API submission, status presentation, retries, support correlation, and business reconciliation.
- MEMEH RTPO responsibility: authentication, validation, alias normalization, trusted CAS resolution, governed routing, canonical dispatch, lifecycle correlation, and event history.
- Provider responsibility: RTP initiation in IPS, payer interaction through provider-controlled channels, financial controls, payment execution, and authoritative lifecycle reporting.

2. Scope and Non-Scope

In scope for this guide	Outside this integration boundary
Fintech technical onboarding and credentials	Direct CAS V2 credentials or direct CAS queries by the fintech
Required payer/payee resolution and masked financial-address selection	Creating pain.013, pain.014, pacs.008, pacs.002, or other IPS messages
Payment-request submission and idempotency	Overriding the participant identity or full financial address returned by CAS V2
Polling current status and immutable events	Holding funds, operating a wallet, clearing, or settlement
Error handling, observability, reconciliation, and UAT	Payer authentication inside a bank, mobile money, or fintech channel

A MEMEH RTPO acceptance response confirms orchestration and handoff progress. It is not proof that funds moved. Only an authoritative terminal lifecycle event closes the business journey.

3. Team Responsibilities

Team	Primary responsibility	Required output
Product/business	Define checkout journey, references, customer messages, and fulfilment policy	Approved journey and acceptance criteria
Backend engineering	Implement server-to-server client, idempotency, polling, timeouts, and secret use	Versioned integration and automated tests
Frontend/mobile	Collect payer intent and display clear states without exposing MEMEH RTPO credentials	Accessible checkout and status experience
QA/UAT	Execute positive, negative, duplicate, timeout, and lifecycle tests	Signed evidence pack and defect disposition
Security	Review network, TLS, secrets, logging, dependency risk, and incident procedures	Security approval and residual-risk record
Operations/support	Monitor, investigate by request ID, reconcile, and escalate	Runbook, dashboards, contacts, and support readiness

4. Current End-to-End Journey

```
Customer browser or mobile app
  -> Fintech backend: create checkout and collect payer alias
  -> MEMEH RTPO: POST /api/v1/aliases/resolve
  -> CAS Resolver -> CAS V2: resolve payer and payee aliases
  <- Fintech backend: payer/payee participants and masked address choices
  <- Customer: select one payer financial-address option
  -> MEMEH RTPO: POST /api/v1/payment-requests with resolution and selected IDs
  -> CAS Resolver -> CAS V2: re-resolve and validate both selections
  -> MEMEH RTPO runtime route: select CAS-resolved payee provider
  -> Payee provider adapter: canonical RTP initiation instruction
  -> Provider RTP platform -> IPS: participant-owned Request-to-Pay flow
  -> Payer provider channel: customer reviews and approves or rejects
  -> Provider adapter -> MEMEH RTPO: lifecycle events
  <- Fintech backend: GET status and GET event history
  <- Customer: pending, completed, rejected, cancelled, failed, or returned
```

The current fintech contract uses polling for status and event history. A fintech webhook is not part of the current MEMEH RTPO API surface and must not be assumed.

5. Onboarding Prerequisites

Prerequisite	Owner	Acceptance evidence
Approved organization and use case	MEMEH RTPO onboarding and business owner	Approval reference and named service
Merchant identity	MEMEH RTPO onboarding	Issued merchant_id and authorized payee relationship
Environment endpoints	MEMEH RTPO integration team	UAT and production base URLs
Client credential	MEMEH RTPO security administrator	Environment-specific bearer credential delivered securely
Network access	Both infrastructure teams	VPN/private route or approved source ranges and firewall evidence
TLS trust	Both security teams	Trusted CA chain, hostname validation, and certificate ownership
Test data	MEMEH RTPO UAT team	Approved aliases, participant routes, and expected outcomes
Contacts	Both organizations	Technical, security, operations, and incident contacts

- Use different credentials for development/UAT and production.
- Do not use merchant_id as an authentication secret.
- Do not start UAT until the merchant identity is authorized for the intended payee context.

6. Environment and Connectivity Model

Environment	Purpose	Data rule	Connectivity
Local/development	Client development against approved mocks or sandbox	Synthetic data only	Developer-controlled network
UAT/certification	End-to-end contract and lifecycle certification	Approved test identities only	Approved BSL network, VPN, or controlled integration network
Production	Live Request-to-Pay processing	Live data under approved handling rules	Private/controlled route with production TLS and monitoring

Credentials, endpoints, trust stores, and test identities must never be reused across environments.

7. Security Baseline

- Call MEMEH RTPO only from the fintech backend. Never embed the API key in browser JavaScript, a mobile binary, public source code, logs, screenshots, or support tickets.
- Send Authorization: Bearer <MEMEH_CORE_API_KEY> on protected fintech endpoints.
- Use HTTPS with hostname and certificate validation. Production must not use TLS_SKIP_VERIFY or an equivalent bypass.
- Store secrets in an approved secrets manager, restrict read access by workload identity, rotate under dual control, and maintain revocation procedures.
- Apply egress allowlisting and participant VPN/private routing where required by the approved design.
- Do not log bearer credentials, complete financial addresses, full request bodies, OTPs, or unnecessary personal data.
- Use application authentication, transport security, network restrictions, and monitoring as layered controls.

The current fintech API contract uses bearer authentication. HMAC headers used on selected internal MEMEH RTPO boundaries are not part of the fintech-facing contract unless an approved deployment profile explicitly adds them.

8. API Conventions

Convention	Requirement
Base path	/api/v1
Media type	Content-Type: application/json for POST requests
Authentication	Authorization: Bearer <credential>
Amount	Positive decimal string with at most two fraction digits, for example 250.00
Currency	Three uppercase letters; examples use SLE
Identifiers	Treat returned identifiers as opaque strings
Time	Store service timestamps as UTC-aware values
Error body	Structured envelope with error.code and error.message
Uncertain timeout	Retry with the same idempotency key; absence of a response is not proof of failure

9. Flow 1 - Resolve Both Parties and Select Addresses

Alias resolution is required before payment creation. MEMEH RTPO resolves both payer and payee aliases through CAS V2 and returns every eligible financial-address option with its participating financial service provider. The payer chooses one payer option. The fintech selects the authorized payee option for the merchant, normally the configured default.

```
POST /api/v1/aliases/resolve
Authorization: Bearer <credential>
Content-Type: application/json

{
  "payer_alias": "+23276000001",
  "payee_alias": "govpay@merchant.sl",
  "amount": "250.00",
  "currency": "SLE",
  "reference": "GOV-2026-000184"
}
```

```
HTTP/1.1 200 OK

{
  "verified": true,
  "resolution_id": "<opaque-resolution-id>",
  "source": "CAS_V2",
  "expires_at": "2026-08-05T10:35:00Z",
  "payer": {
    "alias": "+23276000001",
    "alias_type": "MSISDN",
    "status": "ACTIVE",
    "finaddresses": [
      {
        "id": "<payer-bank-choice-id>",
        "label": "BANK1SLFR",
        "masked": "Account ending 0319",
        "default": true,
        "servicer_id": "BANK1SLFR",
        "servicer_id_type": "BICFI",
        "status": "ACTIVE"
      },
      {
        "id": "<payer-wallet-choice-id>",
        "label": "MMO1SLFR",
        "masked": "Account ending 4412",
        "default": false,
        "servicer_id": "MMO1SLFR",
        "servicer_id_type": "BICFI",
        "status": "ACTIVE"
      }
    ]
  },
  "payee": {
    "alias": "govpay@merchant.sl",
    "alias_type": "EMAIL",
    "status": "ACTIVE",
    "finaddresses": [{
      "id": "<payee-choice-id>",
      "label": "PAYEESLFR",
      "masked": "Account ending 0999",
      "default": true,
      "servicer_id": "PAYEESLFR",
      "servicer_id_type": "BICFI",
      "status": "ACTIVE"
    }]
  }
}
```

- Display payer.finaddresses so the payer can choose the bank, mobile money operator, fintech, account, or wallet represented by that option.
- Use payee.finaddresses to confirm the merchant destination; do not let the payer redirect the payment to an unrelated payee option.
- Store resolution_id and the selected payer/payee option IDs only for this checkout attempt.
- Never reconstruct or store the full financial address from the masked display value.
- If either party is unverified, inactive, expired, or has no eligible option, stop checkout and resolve again.

10. Flow 2 - Create the Payment Request

```
POST /api/v1/payment-requests
Authorization: Bearer <credential>
Content-Type: application/json

{
  "idempotency_key": "govpay-order-2026-000184-attempt-1",
  "resolution_id": "<opaque-resolution-id>",
  "payer_finaddress_id": "<payer-wallet-choice-id>",
  "payee_finaddress_id": "<payee-choice-id>",
  "payer_alias": "+23276000001",
  "payee_alias": "govpay@merchant.sl",
  "merchant_id": "GOVPAY-SL",
  "amount": "250.00",
  "currency": "SLE",
  "reference": "GOV-2026-000184",
  "description": "Government service fee"
}
```

Field	Required	Rule
idempotency_key	Yes	Unique per logical payment; max 100 characters; reuse only for an exact retry
resolution_id	Yes	Opaque ID returned by Flow 1; Core rejects a stale or changed resolution
payer_finaddress_id	Yes	Opaque payer option chosen from payer.finaddresses
payee_finaddress_id	Yes	Opaque authorized merchant option chosen from payee.finaddresses
payer_alias	Yes	Customer alias in a supported canonicalizable form
payee_alias	Yes	Approved merchant/payee alias
merchant_id	Yes	Onboarded identity; cannot override CAS routing
amount	Yes	Positive decimal string, maximum 18 integer digits and 2 decimal places
currency	Yes	Uppercase three-letter code
reference	Yes	Business reference for customer service and reconciliation
description	No	Short purpose; no secrets or unnecessary personal data

Core repeats the CAS V2 lookup during Flow 2. The request is dispatched only when the aliases, resolution, and selected opaque IDs still match active CAS V2 records. A client cannot submit a raw financial address or participant override.

11. Acceptance Response

```
HTTP/1.1 202 Accepted

{
  "request_id": "<uuid>",
  "status": "DISPATCHED",
  "normalized_aliases": {
    "payer": "+23276000001",
    "payee": "govpay@merchant.sl"
  },
  "inferred_types": {
    "payer": "MSISDN",
    "payee": "EMAIL"
  },
  "dispatch": {"status": "DISPATCHED", "http_status": 202}
}
```

- Persist request_id immediately and link it to the order, merchant, amount, currency, reference, and idempotency key.
- Treat HTTP 202 and DISPATCHED as successful handoff, not customer payment completion.
- Do not mark an invoice paid, issue value, or fulfil a service until the approved terminal success state is observed.

12. Idempotency and Retry Rules

Situation	Required action
No response or client timeout	Retry identical JSON with the identical idempotency_key
Connection closed after request write	Treat outcome as uncertain and retry unchanged
Same key and same logical payload	Accept the original logical result; do not create another local order
409 IDEMPOTENCY_CONFLICT	Stop automatic retry and investigate changed payload or key reuse
Customer intentionally starts a new payment	Generate a new key and preserve the relationship to the original order

Generate the key before the first network attempt and persist it transactionally with the local order. Retry workers must load the original stored payload instead of rebuilding it from mutable data.

13. Read Status and Events

```
GET /api/v1/payment-requests/{request_id}
Authorization: Bearer <credential>

GET /api/v1/payment-requests/{request_id}/events
Authorization: Bearer <credential>
```

Endpoint	Use
GET payment request	Latest status, amount, currency, reference, resolution, participant, dispatch result, and timestamps
GET payment request events	Append-only sequence for support, audit, lifecycle reconstruction, and reconciliation

Poll with bounded backoff. Start with a short interval for an active checkout, then slow while pending. Stop normal polling at a terminal state and retain operational reconciliation for late or corrected events.

14. Lifecycle Interpretation

Status	Meaning	Terminal
RECEIVED	MEMEH RTPO accepted the request into initial processing	No
ROUTABLE	Trusted resolution and routing succeeded	No
DISPATCHED	Participant adapter accepted the instruction handoff	No
PDNG	Participant processing is pending	No
RCVD	Participant observed receipt	No
ACSP	Accepted for processing; not completed	No
ACSC	Provider reports successful payment completion	Business success; may later become RETURNED
RETURNED	A completed payment was later returned	Yes
RJCT	Request or linked payment rejected	Yes
CANC	Request cancelled	Yes
FAILED	Technical processing failed	Yes

Use plain customer language such as Awaiting approval, Processing, Paid, Rejected, Cancelled, Failed, or Returned. Keep canonical codes in logs and support views.

15. Error Handling

HTTP/code	Meaning	Client action
400 INVALID_JSON	Malformed JSON	Correct request; do not retry unchanged
401	Credential missing or invalid	Stop, alert, and verify environment/credential
404 ALIAS_NOT_FOUND	No matching active alias	Ask customer to correct or use an eligible alias
409 RESOLUTION_STALE	CAS V2 resolution changed after Flow 1	Resolve again and ask the payer to confirm the new options
409 IDEMPOTENCY_CONFLICT	Key reused with different data	Treat as an integrity exception
415	Content-Type is not JSON	Correct headers
422 VALIDATION_FAILED	Field or format failed	Correct client data
422 FINANCIAL_ADDRESS_NOT_FOUND	Selected opaque address ID is not an active option	Resolve again; never replace it with a raw address
422 NO_ROUTE	No active governed route	Stop checkout and escalate to MEMEH RTPO operations
422 MERCHANT_PAYEE_MISMATCH	Merchant not authorized for resolved payee	Escalate onboarding/configuration
503 CAS_UNAVAILABLE	Trusted alias service unavailable	Bounded retry with same idempotency key
503 ADAPTER_UNAVAILABLE	Participant handoff unavailable	Retry only under agreed policy with same key
502 contract/configuration error	Downstream response or runtime route invalid	Stop high-rate retry and escalate

Use error.code for program logic and retain error.message for support. Do not make business decisions from free text alone.

16. Recommended Checkout Implementation

- Create a local order and immutable business reference before calling MEMEH RTPO.
- Generate and persist the idempotency key on the server.
- Resolve payer and payee aliases, then show the payer every eligible masked payer financial-address option and provider.
- Keep the payee selection bound to the merchant's authorized destination and display it as confirmation, not a free-form redirect control.
- Submit resolution_id plus the selected payer_finaddress_id and payee_finaddress_id in the payment request.
- Require explicit customer confirmation before submitting the request.
- Persist request_id and acceptance response before returning success to the browser.
- Display pending after 202 and poll through the fintech backend.
- Fulfil only after ACSC under the approved business policy.
- Apply the approved reversal or suspension workflow when a later RETURNED event applies.

17. Reconciliation and Audit

Fintech record	MEMEH RTPO evidence	Control
Local order ID	reference and merchant_id	Documented one-to-one or one-to-many mapping
Submission attempt	idempotency_key and request_id	No duplicate fulfilment
Amount/currency	Status response	Exact comparison; no floating-point rounding
Current business state	Status plus immutable events	Approved lifecycle policy
Customer fulfilment	ACSC evidence	Timestamp and actor retained
Return handling	RETURNED after ACSC	Exception/reversal linked to original order

- Run daily reconciliation even when real-time polling succeeds.

- Investigate requests that remain non-terminal beyond the agreed threshold.
- Do not use a reusable business reference as the only technical correlation key.

18. Observability and Support

- Log local order ID, request_id, idempotency-key fingerprint, HTTP status, error code, latency, and environment.
- Measure create success, dependency errors, time to terminal state, duplicate retries, and aged pending requests.
- Alert on authentication failures, sustained 5xx/503, unusual idempotency conflicts, and polling backlog growth.
- Use request_id as the primary MEMEH RTPO support reference with timestamp, environment, endpoint, HTTP status, and error code.
- Redact aliases and never include credentials or full sensitive payloads in tickets.

19. UAT and Certification

Test group	Minimum scenarios
Connectivity	Approved path, valid TLS, invalid certificate, invalid source network
Authentication	Valid, missing, invalid, and rotated key
Validation	Missing fields, invalid amount, excess precision, invalid currency, malformed alias, non-JSON
Idempotency	Same key/same payload, same key/changed payload, timeout then replay
Resolution/routing	Valid aliases, not found, inactive, no route, merchant-payee mismatch
Lifecycle	DISPATCHED through progress to ACSC, rejection, cancellation, failure, and return
Operations	Aged pending, reconciliation, credential failure, support trace
Performance	Agreed sustained rate, burst, timeout, and no duplicate dispatch

Evidence includes sanitized requests/responses, timestamps, event histories, expected versus actual results, defects, retests, and sign-off from business, engineering, operations, and security.

20. Production Readiness

- Production merchant_id, endpoint, credentials, route, and certificate chain are approved.
- Credentials are outside source code and rotation/revocation are tested.
- Exact idempotency replay is automated and proven under timeout.
- Checkout never treats 202, DISPATCHED, RCVD, PDNG, or ACSP as paid.
- Polling, terminal handling, RETURNED handling, and daily reconciliation are operational.
- Logs and dashboards expose correlation without leaking secrets or full financial addresses.
- Support, incident, security, and business contacts are tested.
- UAT defects are closed or accepted and required signatories approve go-live.

21. First-Line Runbook

Symptom	First checks	Escalation
401 from MEMEH RTPO	Environment, secret version, Authorization header	Fintech security and MEMEH RTPO access administrator
Alias cannot resolve	Input normalization, test identity, CAS availability	MEMEH RTPO/CAS operations
NO_ROUTE	Request time, payee alias, merchant_id, error code	MEMEH RTPO Control Plane operations
ADAPTER_UNAVAILABLE	Retry count, request_id, participant	MEMEH RTPO and participant operations
Request remains pending	Current status, full events, last update versus SLA	MEMEH RTPO lifecycle and participant support
Possible duplicate	Order, idempotency key, request_id, amount, reference	Fintech reconciliation and MEMEH RTPO support
Paid then returned	Confirm ACSC followed by RETURNED	Business operations and finance

22. Required Deliverables

- Architecture and data-flow diagram.
- Environment matrix containing secret references, not secret values.
- API client and automated contract tests.
- Idempotency, retry, polling, lifecycle, and reconciliation design.
- Security review and vulnerability disposition.
- UAT evidence and signed readiness checklist.
- Runbook, dashboard, alert ownership, and incident contacts.