

One2One Payment, Support Agent, and Resolution Workflow

Purpose: This document captures the proposed payment and issue-resolution process for One2One before implementation. It is a study draft, not final code behavior.

Main Goal

Keep the process simple for both clients and hosts:

- Clients should feel protected when they pay.
 - Hosts should be paid fairly when they attend and deliver the session.
 - Admins and support agents should have clear tools to resolve disputes.
 - The system should avoid automatic unfair payouts when a session fails, is missed, or ends too early.
-

Recommended Principle

Payment should confirm booking or access, but payout should depend on service completion.

Recommended flow:

1. Client pays.
 2. Amount is held by One2One.
 3. Session attendance and duration are tracked.
 4. If the session is valid, the host payout becomes eligible.
 5. If there is a no-show, early ending, or technical issue, admin/support agent can review.
 6. Money is either paid to host, returned to client balance, refunded to account, or used to book another session.
-

New Role: Support Agent

Add a new role called `agent`.

Support agents are staff members created by admin.

Agent permissions:

- View users, appointments, payments, and disputes.
- Join or manage support chat with clients.

- Review raised concerns.
- Resolve issues using predefined actions.
- Edit/update support-related records.

Agent should not have full platform control like admin.

Admin permissions:

- Everything agent can do.
- Create/suspend agents.
- Manage platform settings.
- Override payment/refund decisions.
- View financial reports.

Payment States

Suggested payment lifecycle:

- `unpaid` : no payment received.
- `paid_pending_service` : payment received, waiting for session completion.
- `eligible_for_payout` : session completed successfully.
- `settled_to_host` : amount transferred or marked payable to host.
- `refund_due` : amount should be returned to client.
- `refunded` : refund completed.
- `balance_credited` : amount added to client One2One Balance.
- `disputed` : issue raised and pending review.
- `failed` : payment failed.

Service States

Suggested service lifecycle:

- `scheduled` : appointment booked.
- `in_progress` : meeting started.
- `completed` : session finished successfully.
- `host_no_show` : host did not attend.
- `client_no_show` : client did not attend.
- `ended_early` : session ended too quickly.
- `technical_issue` : connection/system issue reported.
- `under_review` : agent/admin is reviewing.

- `resolved` : issue decision completed.

One2One Balance

Use the name `One2One Balance` instead of `wallet` for now.

Reason:

- “Wallet” may create extra financial/legal expectations.
- “Balance” is easier to explain as platform credit.

Client can use One2One Balance to:

- Book another session.
- Pay for memberships.
- Request withdrawal/refund to account if allowed.

Single Session Rules

Host Absent

If host does not join:

- Payment should not transfer to host.
- Client amount should be credited to One2One Balance or refunded.
- Client can also choose a new slot using the same paid amount.

Suggested grace period:

- Host should join within 5 minutes of session start.
- If host does not join within grace period, mark as `host_no_show`.

Client Absent

If client does not join:

- If client did not cancel/reschedule before the allowed deadline, amount can go to host.

Suggested cancellation deadline:

- Client must cancel or reschedule at least 12 hours before session start.

Wrong Booking Protection

If client accidentally books the wrong appointment:

- Allow edit/cancel for 10 minutes after booking.

- Show a countdown timer: “Edit/cancel allowed for X minutes.”
- After 10 minutes, normal cancellation rules apply.
- If the session starts in less than 12 hours, edit/cancel should lock after the 10-minute grace period.

Valid Session Duration

We need to decide what counts as a valid session.

Recommended starting rule:

- A session is valid if the host and client were both present for at least 5 minutes, or at least 20% of the booked duration, whichever is lower.

Examples:

- 10-minute session: valid after 2 minutes.
- 30-minute session: valid after 5 minutes.
- 60-minute session: valid after 5 minutes.

This can be adjusted later.

Early Ended Sessions

If the session ends too soon:

- Mark as `ended_early`.
- Do not automatically release host payout.
- Allow client or host to raise a concern.
- Agent/admin reviews attendance logs, duration, and notes.

Possible resolutions:

- Pay host fully.
- Refund client fully.
- Partial refund.
- Add amount to One2One Balance.
- Rebook another slot using same payment.

Technical Issue Flow

If either party reports technical problems:

1. Client or host clicks “Raise Concern.”

2. System creates a support ticket/dispute.
3. Payment moves to `disputed`.
4. Agent/admin reviews meeting logs.
5. Agent/admin chooses resolution.

Important logs to track:

- Who joined.
 - Join time.
 - Leave time.
 - Duration.
 - Network/signaling failure if available.
 - Whether host joined.
 - Whether client joined.
 - Whether video/audio was enabled.
-

Support Chat System

Need a live support chat between client and support agents.

Suggested behavior:

- Client clicks “Need Help.”
- Available agents see highlighted request.
- Any available agent can click “Connect.”
- Once connected, that agent handles the chat.
- Chat history should be saved with the support ticket.

Agent availability:

- `available`
- `busy`
- `offline`

Future enhancement:

- Auto-assign to least busy available agent.
-

Dispute System

A dispute should store:

- Client
- Host

- Appointment/session
- Payment
- Reason
- Description
- Evidence/logs
- Status
- Assigned agent
- Resolution
- Final financial action

Suggested dispute statuses:

- open
- assigned
- waiting_for_client
- waiting_for_host
- under_review
- resolved
- rejected

Suggested reasons:

- Host absent
- Client absent
- Technical issue
- Session ended early
- Wrong booking
- Unsatisfied client
- Payment issue
- Other

Membership Rules

Membership is different from single sessions because client pays for multiple sessions.

Host Absent in Membership Session

Recommended rule:

- Do not deduct the client's session allowance.
- Add one makeup session.

- Notify affected clients.

For group memberships:

- If host is absent, all enrolled clients should get a makeup session.

For one-to-one memberships:

- Only the affected client gets the makeup session.

Client Absent in Membership Session

Options:

1. Deduct allowance if client misses without notice.
2. Do not deduct allowance if host allows flexible attendance.
3. Let host choose the missed-session policy when creating membership.

Current recommendation:

- Keep the existing missed-session policy setting.
- Add clear explanation in UI.

Whole-Day Availability Rules

Whole-day availability should behave like a live single session.

If client books/connects:

- Host status becomes **busy**.
- Session attendance starts.

When client leaves:

- If one-to-one, host becomes available again.
- Time attended is saved in live stream history.

When host ends session:

- Current session ends for client(s).
- Attendance is saved.
- Host can become available again if whole-day availability remains active.

If host forgot to end previous session:

- When host toggles whole-day availability again, system should detect active/busy session.
- Host should be offered:
 - End current live session and become available again.
 - Keep current session active.

Notifications

Clients should receive notifications for:

- Payment successful.
- Appointment confirmed.
- Edit/cancel lock timer.
- Cancellation deadline.
- Host joined.
- Host absent/no-show.
- Session completed.
- Concern raised.
- Refund/balance decision.

Hosts should receive notifications for:

- New booking.
- Payment pending service.
- Client cancellation.
- Client no-show.
- Dispute raised.
- Payout eligible.
- Payout settled.

Agents/admin should receive notifications for:

- New dispute.
- New support chat request.
- High-priority payment issue.
- Long unresolved disputes.

Suggested Database Additions

Users

Add role:

- `agent`

Optional fields:

- `support_status`
- `last_seen_at`

Payments

If no dedicated payment table exists yet, create one.

Fields:

- `id`
- `user_id`
- `host_id`
- `appointment_id`
- `amount`
- `currency`
- `gateway`
- `gateway_payment_id`
- `payment_status`
- `settlement_status`
- `paid_at`
- `eligible_for_payout_at`
- `settled_at`
- `refunded_at`

Account Balances

Fields:

- `id`
- `user_id`
- `balance`
- `currency`

Balance Transactions

Fields:

- `id`
- `user_id`
- `payment_id`
- `appointment_id`
- `type`
- `amount`
- `description`
- `created_by`

Types:

- `credit_refund`
- `debit_booking`
- `withdrawal_requested`
- `withdrawal_completed`
- `adjustment`

Support Tickets / Disputes

Fields:

- `id`
- `client_id`
- `host_id`
- `agent_id`
- `appointment_id`
- `payment_id`
- `reason`
- `description`
- `status`
- `resolution`
- `financial_action`
- `resolved_by`
- `resolved_at`

Support Messages

Fields:

- `id`
- `ticket_id`
- `sender_id`
- `message`
- `read_at`
- `created_at`

Attendance Logs

Fields:

- `id`

- `appointment_id`
 - `user_id`
 - `role`
 - `joined_at`
 - `left_at`
 - `duration_seconds`
 - `leave_reason`
-

Suggested Services

CancellationPolicyService

Handles:

- 10-minute booking correction window.
- 12-hour cancellation deadline.
- Whether cancellation/edit is allowed.

SettlementService

Handles:

- Host payout eligibility.
- Client no-show.
- Host no-show.
- Early-ended sessions.
- Refund/balance decision.

SupportTicketService

Handles:

- Creating disputes.
- Assigning agents.
- Updating statuses.
- Saving resolution.

BalanceService

Handles:

- Credit client balance.
- Debit balance for bookings.

- Transaction ledger.

Simple User Experience

Client should see simple messages:

- “Your payment is safe until the session is completed.”
- “You can edit or cancel this booking for 10 minutes.”
- “Free cancellation/reschedule is available until 12 hours before the session.”
- “If there is a problem, raise a concern and our support team will review.”

Host should see simple messages:

- “Payout is released after a valid completed session.”
- “If client does not attend and cancellation window has passed, payout remains eligible.”
- “If there is a dispute, payout will be held until review.”

Admin/agent should see detailed controls:

- Pay host.
- Refund to client balance.
- Refund to account.
- Rebook session.
- Add makeup session.
- Reject dispute.
- Add internal note.

Open Questions Before Implementation

1. What is the minimum duration for a valid session?
2. Should host join grace be 5 minutes?
3. Should client join grace be 10 minutes?
4. Should dispute deadline be 24 hours or 48 hours after session?
5. Should partial refunds be supported immediately or later?
6. Should One2One Balance withdrawals be available at launch?
7. Should support agents be able to approve refunds, or only recommend them?
8. For memberships, how long should makeup sessions remain valid?
9. Should unsatisfied client claims always create a dispute, or only after rating below a threshold?

Recommended Implementation Phases

Phase 1: Core Protection

- Add `agent` role.
- Add payment hold statuses.
- Track attendance logs.
- Add host/client no-show rules.
- Add edit/cancel lock timer.
- Add basic admin/agent resolution panel.

Phase 2: Balance and Disputes

- Add One2One Balance.
- Add support tickets/disputes.
- Add support chat.
- Add refund/balance/rebook actions.
- Add notifications.

Phase 3: Membership and Advanced Settlement

- Add membership makeup automation.
- Add partial refunds.
- Add reports and analytics.
- Add agent workload dashboard.
- Add region/location analytics for client requests.

Draft Workflow Diagram



```
I --> J["Agent/admin reviews"]
```

```
J --> K{"Resolution"}
```

```
K -- "Pay host" --> F
```

```
K -- "Refund client" --> G
```

```
K -- "Rebook" --> L["Create replacement session"]
```

```
K -- "Membership makeup" --> M["Add makeup session"]
```