Remittance Flow

Low-Level Design: Remittance Flow

Version: 1.0 **Date:** 2026-02-21 **Author:** Banking Architecture Team **Status:** Approved **Applies to:** Drop — Cross-Border Money Transfer (PISP)

---

1. Overview

Remittance is Drop's core feature — sending money from a Norwegian bank account to a recipient abroad. The flow has 4 user-facing steps:

1. **Select recipient** (or add new)
2. **Enter amount** (see FX rate + fee in real-time)
3. **Review** (PSD2 Art. 45 pre-payment disclosure)
4. **Confirm** (SCA via BankID at user's bank)

Drop uses **PISP** (Payment Initiation Service) to initiate the transfer directly from the user's bank account. Drop never touches the money.

API endpoint: `POST /api/transactions/remittance` **Fee:** 0.5% of send amount **Amount range:** 100 - 50,000 NOK **KYC required:** Yes (`kyc_status = 'approved'`) **Supported corridors:** Serbia (RSD), Bosnia (BAM), Poland (PLN), Pakistan (PKR), Turkey (TRY), EU (EUR)

---

2. End-to-End Remittance Flow

```
sequenceDiagram
    participant U as User
    participant UI as Drop App<br/>(/send)
    participant API as Drop API
    participant DB as Drop DB
```

```
    participant ASPSP as User's Bank
    participant RB as Recipient Bank

    Note over U,RB: Step 1 — Select Recipient
    U->>UI: Navigate to Send Money
    UI->>API: GET /api/recipients?page=1&limit=20
    API->>DB: SELECT * FROM recipients<br/>WHERE user_id = ? ORDER BY created_at DESC
    DB-->>API: [{id: "rec_1", name: "Marko Petrovic",<br/>country: "Serbia", currency: "RSD"}]
    API-->>UI: Recipient list (bank accounts masked)
    U->>UI: Select "Marko Petrovic"

    Note over U,RB: Step 2 — Enter Amount
    UI->>API: GET /api/rates/RSD
    API->>DB: SELECT rate FROM exchange_rates<br/>WHERE to_currency = 'RSD'
    DB-->>API: {rate: 10.17}
    API-->>UI: {rate: 10.17, fee: 0.005}
    U->>UI: Enter 2000 NOK
    UI->>UI: Live calculation:<br/>Send: 2,000 NOK<br/>Fee: 10 NOK (0.5%)<br/>Rate: 1 NOK =
10.17 RSD<br/>Receives: 20,340 RSD

    Note over U,RB: Step 3 — Review (PSD2 Art. 45 Disclosure)
    U->>UI: Tap "Neste" (Next)
    UI->>API: POST /api/transactions/disclosure<br/>{type: "remittance", amount:
2000,<br/>recipientId: "rec_1"}
    API->>DB: Lookup recipient currency, exchange rate
    API->>API: Calculate fee (2000 * 0.005 = 10)<br/>Calculate receive (2000 * 10.17 =
20340)<br/>Determine delivery (non-EEA: 2-4 days)
    API-->>UI: {amount: 2000, fee: 10, feePercentage: 0.5,<br/>exchangeRate: 10.17,
receiveAmount: 20340,<br/>receiveCurrency: "RSD",<br/>estimatedDelivery: "2-4 business
days",<br/>totalCost: 2010}
    UI->>UI: Display disclosure screen:<br/>"Du sender 2 000 kr til Marko Petrovic<br/>Gebyr:
10 kr (0,5%)<br/>Vekslingskurs: 1 NOK = 10,17 RSD<br/>Marko mottar: 20 340 RSD<br/>Total
kostnad: 2 010 kr<br/>Estimert levering: 2-4 virkedager"

    Note over U,RB: Step 4 — Confirm & Payment Initiation
    U->>UI: Tap "Bekreft og send" (Confirm and send)
    UI->>API: POST /api/transactions/remittance<br/>{recipientId: "rec_1", amount:
2000,<br/>bankAccountId: "ba_1"}
    API->>DB: Verify KYC: kyc_status = 'approved'
    API->>DB: Verify recipient belongs to user
```

```
    API->>DB: Verify bank account exists + balance >= 2010
    API->>DB: Lookup exchange rate for RSD
    API->>DB: Generate idempotency_key<br/>BEGIN TRANSACTION<br/>UPDATE bank_accounts SET balance = balance - 201000<br/>INSERT INTO transactions (status: 'processing')
    API->>DB: COMMIT

    API->>ASPSP: POST /v1/payments/cross-border-credit-transfers<br/>{debtorAccount: {iban: user_iban},<br/>instructedAmount: {currency: "NOK", amount: "2010.00"},<br/>creditorName: "Marko Petrovic",<br/>creditorAccount: {bban: "265-1234567-89"},<br/>remittanceInformationUnstructured: "Drop remittance tx_rem_xxx"}
    ASPSP-->>API: {paymentId: "pay_xyz",<br/>transactionStatus: "RCVD",<br/>scaRedirect: "https://dnb.no/sca/pay/..."}
    API-->>UI: {transactionId: "tx_rem_xxx",<br/>scaRedirect: "https://dnb.no/sca/pay/..."}

    Note over U,RB: SCA at Bank (Dynamic Linking)
    UI->>U: Redirect to bank SCA page
    U->>ASPSP: BankID authentication<br/>(sees: "2 010 NOK to Marko Petrovic")
    ASPSP-->>U: Redirect to Drop callback

    U->>API: GET /api/payments/callback?paymentId=pay_xyz
    API->>ASPSP: GET /v1/payments/pay_xyz/status
    ASPSP-->>API: {transactionStatus: "ACCP"}
    API->>DB: UPDATE transactions<br/>SET status = 'completed',<br/>completed_at = now<br/>WHERE id = 'tx_rem_xxx'
    API->>DB: INSERT INTO audit_log<br/>(action: 'payment.completed')
    API->>DB: INSERT INTO notifications<br/>(title: 'Overfoering sendt',<br/>body: '2 000 kr sendt til Marko Petrovic')
    API-->>UI: {status: "completed"}
    UI-->>U: Success screen:<br/>"2 000 kr sendt til Marko Petrovic!<br/>Estimert levering: 2-4 virkedager"

    Note over ASPSP,RB: Settlement (Drop is not involved)
    ASPSP->>RB: SWIFT gpi / correspondent banking<br/>NOK converted to RSD
    RB->>RB: Credit Marko's account: 20,340 RSD
```

---

3. Transaction States

```
stateDiagram-v2
    [*] --> Draft: User on review screen<br/>(disclosure shown, not yet confirmed)

    Draft --> Initiated: User taps "Bekreft og send"<br/>Transaction record
created<br/>(status: processing)
    Draft --> Abandoned: User navigates away<br/>(no record created)

    Initiated --> ScaPending: ASPSP returns scaRedirect<br/>User redirected to bank
    Initiated --> Failed: ASPSP rejects initiation<br/>(invalid IBAN, bank error)

    ScaPending --> Completed: User completes BankID SCA<br/>ASPSP status: ACCP/ACSC
    ScaPending --> Failed: SCA timeout (5 min)
    ScaPending --> Failed: User cancels SCA
    ScaPending --> Failed: Bank rejects payment<br/>(insufficient funds at bank)

    Completed --> Settled: Funds credited to recipient<br/>(tracked via ASPSP status polling)
    Completed --> RefundPending: Settlement failed<br/>(correspondent bank error)

    Failed --> [*]: User sees error,<br/>can retry from Step 1

    RefundPending --> Refunded: Refund processed<br/>Balance restored
    Refunded --> [*]
    Settled --> [*]
    Abandoned --> [*]
```

Database status values (`transactions.status` CHECK constraint):

- `processing` — Transaction created, awaiting SCA or settlement
- `completed` — ASPSP accepted payment, settlement in progress or done
- `failed` — Payment rejected, SCA failed, or settlement error

---

4. Pre-Payment Disclosure (PSD2 Art. 45)

Before the user confirms a remittance, Drop must show a **complete cost breakdown**. This is a legal requirement under PSD2 Art. 45 (Betalingstjenesteloven in Norwegian law).

4.1 Disclosure Checklist

Item	PSD2 Reference	Drop Field	Example
Amount to be transferred	Art. 45(1)(a)	`amount`	2,000 NOK
All fees/charges	Art. 45(1)(b)	`fee`, `feePercentage`	10 NOK (0.5%)
Exchange rate used	Art. 45(1)(c)	`exchangeRate`	1 NOK = 10.17 RSD
Amount after conversion	Art. 45(1)(d)	`receiveAmount`, `receiveCurrency`	20,340 RSD
Total cost to payer	Art. 45(1)(e)	`totalCost`	2,010 NOK
Estimated delivery time	Art. 45(1)(f)	`estimatedDelivery`	2-4 business days
Currency of debit	Implicit	Send currency	NOK
Currency of credit	Implicit	Receive currency	RSD

4.2 Disclosure Screen Content (Norwegian)

```
Bekreft overfoering

Til:           Marko Petrovic
Land:          Serbia
Bankkonto:     *****567-89 (Banca Intesa)

Du sender:     2 000,00 kr
Gebyr (0,5%):     10,00 kr
Totalt belop:  2 010,00 kr

Vekslingskurs: 1 NOK = 10,17 RSD
Marko mottar:  20 340,00 RSD

Estimert levering: 2-4 virkedager
Pengene trekkes fra: DNB Brukskonto

[Bekreft og send]    [Avbryt]
```

---

5. FX Rate Lock & Expiry

5.1 Rate Lock Window

Event	Time	Action
User views disclosure	T+0	Rate displayed from `exchange_rates` table
User confirms payment	T+0 to T+15min	Rate locked in `transactions.exchange_rate`
SCA completes	T+0 to T+15min	Locked rate applies to settlement
SCA not completed	T+15min	Rate expires, transaction fails, user must re-quote

5.2 Rate Lock Implementation

1. When `POST /api/transactions/remittance` is called, the current rate is read from `exchange_rates`
2. The rate is stored in `transactions.exchange_rate` at insert time
3. If the SCA takes longer than 15 minutes, the reconciliation job detects the stale transaction and marks it `failed`
4. User is notified to retry (with a new, current rate)

---

6. Validation Rules

6.1 Pre-Flight Checks (Before Transaction Creation)

Check	Source	Error if Failed
User authenticated	JWT from cookie/header	401 `unauthorized`
KYC approved	`users.kyc_status = 'approved'`	403 `kyc_required`
Recipient exists	`recipients.id` WHERE `user_id = ?`	404 `not_found`
Recipient belongs to user	`recipients.user_id = jwt.userId`	404 `not_found`
Bank account exists	`bank_accounts.id` WHERE `user_id = ?`	400 `no_bank_account`
Amount in range	100 to 50,000 NOK	422 `validation_error`
Amount valid	`Number.isFinite()`, max 2 decimals	422 `validation_error`
Balance sufficient	`bank_accounts.balance >= amount + fee`	402 `insufficient_balance`

Check	Source	Error if Failed
Currency corridor supported	`exchange_rates.to_currency` exists	422 `validation_error`
Not duplicate	`idempotency_key` unique	Return existing transaction
Rate limit	< 10 requests/min per IP	429 `rate_limited`

6.2 Amount Validation

```
Minimum: 100 NOK (protect against micro-transaction abuse)
Maximum: 50,000 NOK (regulatory limit for simplified CDD)
Decimals: max 2 (validated by validateAmount())
Type: Number.isFinite() (prevents NaN, Infinity injection)
```

---

7. Error Scenarios & User Messages

Scenario	API Response	User Message (Norwegian)	Next Step
KYC not approved	403 `kyc_required`	"Du maa fullfoere identitetsverifisering for aa sende penger."	Redirect to KYC flow
No linked bank account	400 `no_bank_account`	"Du har ingen tilkoblet bankkonto. Koble til en bank foerst."	Redirect to /accounts
Insufficient balance	402 `insufficient_balance`	"Ikke nok penger paa kontoen. Saldo: 1 200 kr, totalt belop: 2 010 kr."	Show balance, suggest lower amount
Unsupported corridor	422 `validation_error`	"Vi stoetter ikke overfoering til dette landet ennaa."	Show supported countries
Amount too low	422 `validation_error`	"Minimumsbelopet er 100 kr."	Adjust amount
Amount too high	422 `validation_error`	"Maksimumsbelopet er 50 000 kr."	Adjust amount
SCA timeout	(callback timeout)	"BankID-sesjonen utlop. Overforingen ble ikke gjennomfoert."	Retry button
SCA cancelled	(callback cancelled)	"Du avbrot betalingen. Ingen penger er trukket."	Retry button
Bank rejected	ASPSP `RJCT`	"Banken avviste overforingen. Kontakt banken din."	Show bank support info

Scenario	API Response	User Message (Norwegian)	Next Step
Rate expired	(rate > 15min old)	"Vekslingskursen har utlopt. Vennligst bekreft ny kurs."	Re-show disclosure with new rate
Network error	502/503	"Teknisk feil. Proev igjen om noen minutter."	Retry after 30s
Duplicate detected	200 (existing tx)	"Denne overforingen er allerede registrert."	Show existing transaction

---

8. Post-Transaction

8.1 Confirmation Screen

After successful SCA:

```
Overfoering sendt!

2 000 kr sendt til Marko Petrovic
Marko mottar 20 340 RSD

Referanse: tx_rem_a1b2c3d4...
Status: Under behandling
Estimert levering: 2-4 virkedager

[Se detaljer]    [Send til en annen]
```

8.2 Transaction Tracking

Users can track their remittance in the Transaction History (`/transactions`):

Status	Display	Icon
`processing`	"Under behandling"	Spinner
`completed`	"Fullfoert"	Green checkmark
`failed`	"Feilet"	Red X

8.3 Transaction Summary

`GET /api/transactions/summary` returns aggregated transaction statistics for the authenticated user (total sent, total fees, transaction count, breakdown by corridor).

8.4 Receipt

`GET /api/transactions/{id}/receipt` returns a detailed receipt:

```
{
  "transactionId": "tx_rem_xxx",
  "date": "2026-02-21T14:30:00Z",
  "type": "remittance",
  "amount": 2000,
  "currency": "NOK",
  "fee": 10,
  "exchangeRate": 10.17,
  "receiveAmount": 20340,
  "receiveCurrency": "RSD",
  "recipient": {"name": "Marko Petrovic", "country": "RS"},
  "reference": "tx_rem_xxx",
  "status": "completed",
  "completedAt": "2026-02-21T14:35:00Z"
}
```

8.5 Notifications

On completion/failure, a notification is created:

Event	Notification Title	Notification Body
Payment sent	"Overfoering sendt"	"2 000 kr sendt til Marko Petrovic"
Payment completed	"Overfoering fullfoert"	"20 340 RSD mottatt av Marko Petrovic"
Payment failed	"Overfoering feilet"	"Overfoering til Marko Petrovic ble avvist. Kontakt oss for hjelp."

---

9. Refund Handling

If a remittance fails after funds were debited (e.g., correspondent bank rejects, recipient IBAN invalid):

Step	Action	Timeline
1	ASPSP reports `RJCT` or `CANC` status	1-5 business days
2	Drop detects via reconciliation job	Within 1 hour of status change
3	Drop creates refund record in audit_log	Immediate
4	ASPSP reverses the debit (automatic for SEPA)	1-3 business days
5	Drop updates `bank_accounts.balance` on next AISP sync	Next balance refresh
6	User notified via push notification	Immediate

Note: For SWIFT transfers, refund timing depends on correspondent banks and may take 5-10 business days. Drop sends a notification with estimated refund timeline.

---

10. AML/Compliance Checks

Each remittance triggers compliance checks before PISP initiation:

Check	Implementation	Action on Trigger
Velocity limit	> 5 remittances/hour or > 20/day	`aml_alerts` record (medium severity), continue
Structuring detection	Multiple amounts just below 25,000 NOK	`aml_alerts` record (high severity), review queue
High-risk corridor	FATF grey/black list country	Enhanced due diligence flag
Single large transfer	> 25,000 NOK	Enhanced monitoring
Total daily volume	> 100,000 NOK cumulative	`aml_alerts` record, may require manual approval
Sanctions screening	Recipient name vs sanctions lists	Block if match, `screening_results` record

---

11. Database Impact

11.1 Tables Written

Table	Operation	When

`transactions`	INSERT	Payment initiated (Step 4)
`transactions`	UPDATE	Status change (processing to completed/failed)
`bank_accounts`	UPDATE (balance)	Atomic debit during transaction creation
`audit_log`	INSERT	Every payment action
`notifications`	INSERT	Payment sent / completed / failed
`aml_alerts`	INSERT	If AML rule triggered

11.2 Tables Read

Table	Operation	When
`users`	SELECT (kyc_status)	Pre-flight KYC check
`recipients`	SELECT	Recipient lookup (Step 1)
`bank_accounts`	SELECT (balance)	Balance check (Step 4)
`exchange_rates`	SELECT (rate)	FX rate lookup (Step 2, 3, 4)
`transactions`	SELECT (idempotency_key)	Duplicate detection

---

12. Cross-References

- **Payment Processing:** ../integration/payment-processing.md — SEPA/SWIFT settlement, FX, fees
- **Open Banking AISP/PISP:** ../integration/open-banking-aisp-pisp.md — PISP API details
- **BankID OIDC:** ../integration/bankid-oidc-integration.md — Drop authentication
- **Security Architecture:** ../hld/security-architecture.md — Fraud detection, AML pipeline
- **Bank Account Linking:** flow-bank-account-linking.md — Prerequisite: linked bank account
- **Database Schema:** ../../backend/DATABASE-SCHEMA.md — `transactions`, `recipients`, `exchange_rates` tables
- **API Reference:** ../../backend/API-REFERENCE.md — Remittance, disclosure, receipt, rate endpoints

---

Revision #5
Created 2026-02-21 05:58:53 UTC by John
Updated 2026-05-23 10:51:59 UTC by John