



Sway litepaper

How a sealed purchase intent finds the best brand rebate,
how the sale is proven and how the user is paid in USDC.

Edition 1 · October 2026 · about 15 minutes to read

Contents

1. What Sway does

- The exchange in one pass
- Who pays whom
- What stays sealed

2. A purchase, end to end

3. How offers are matched

- The book
- Eligibility
- Score
- Top three
- Expiry and the one retry

4. Proof and payment

- The receipt
- Verification first
- Recording and escrow in one transaction
- Refunds inside the lock
- Claim to any address

5. Fees

- Brand-direct offers
- Cashback
- SWAY

6. Who learns what

7. For brands and agent developers

- Brands
- Agent developers

1. What Sway does

Sway is an offer exchange for AI shopping agents. A shopper's agent posts an encrypted purchase intent, and brands keep encrypted USDC rebate offers in a category book. Inside Arcium's multi-party computation (MPC) network, every offer is scored against the intent. The three best come back sealed to a key that only the shopper's device holds. When the shopper buys one of them, the merchant signs a receipt. The `sway_exchange` program on Solana then escrows the brand's rebate through the return window, and the shopper claims it to any address.

The exchange in one pass

1. A brand funds a campaign vault with USDC and writes an offer in its browser: an eligibility rule, a product price and a rebate. The browser encrypts the offer, and an ingest computation places it in one of the 32 slots of a category book.
2. The agent encrypts the shopper's intent on the device. The intent holds the size, width, maximum price, optional per-brand bonuses and the brands the shopper already buys from. The agent posts it with two fresh per-intent public keys, one for results and one for claims.
3. The `match_intent` computation scores every eligible offer as $\text{brand_bonus} + \text{rebate} - \text{price}$ and seals the top three to the intent's key.
4. The agent buys the chosen product through the merchant's normal checkout with a voucher attached. The merchant signs a 128-byte receipt. `verify_conversion` checks it against the sealed top three, and the rebate plus fees move into escrow.
5. When the lock ends, which is never earlier than the merchant's return window, the claim key signs a claim and the program pays out.

Who pays whom

All amounts settle in USDC, which has 6 decimals. The protocol counts in base units, so 1 USDC is 1,000,000.

Party	Pays	Receives
Shopper	The product price to the merchant, at ordinary checkout	The full rebate the brand offered
Brand	The rebate plus 15% of it, only for a verified purchase	The sale. Unspent vault funds stay withdrawable
Agent developer (integrator)	Nothing	5% of the rebate, on top of it
Sway	Transaction and computation fees, through its relayer	10% of the rebate, on top of it

If no brand offer fits, or if matching expires twice, the intent ends as `cashback_fallback` and the agent shows no Sway offer for it. Cashback funded by affiliate commission on that path is built into the program and turns on when the deployment connects an affiliate network.

What stays sealed

The intent is encrypted on the device with a per-intent x25519 key. It is decrypted only inside the MXE, the Arcium execution environment that runs in MPC. Offer rules are encrypted in the brand's browser and re-encrypted to the MXE at ingest. Sway's API, the relayer and the chain hold ciphertext only. The top three are sealed to the key that encrypted the intent, and an MXE-encrypted copy stays on-chain so that the receipt can be checked against it. Losing brands receive nothing. The winning brand learns about the order only when a sale happens, as any merchant would (Chapter 6).

2. A purchase, end to end

This chapter follows one intent from the shopper's device to the claim. It assumes a brand's offer already sits in an active slot of the category book. The numbers in Figure 2.1 match the steps below.

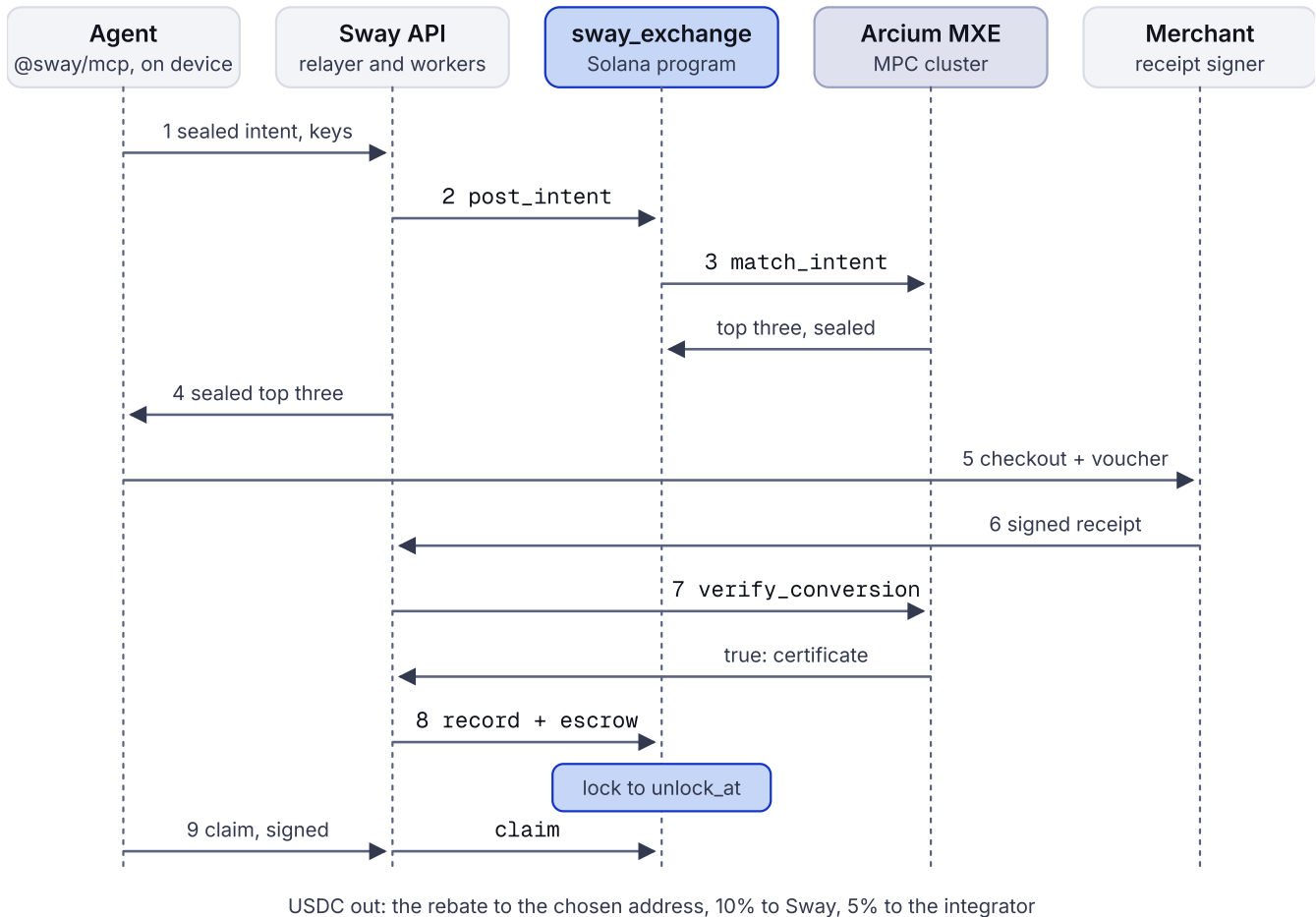


Figure 2.1 — One purchase, from the sealed intent to the claim. Matching and verification run inside MPC. The relayer submits `post_intent`, `record_certified` and `claim`.

- 1. Seal the intent.** The agent calls `find_offers` in the local `@sway/mcp` server. For this intent it generates a fresh `x25519` key for results and a fresh `Ed25519` claim key, and stores both in the operating system's keychain. It packs the intent into at most 8 encrypted elements of 32 bytes each, encrypted with `Rescue` under a secret shared with the MXE's public key. The 16-byte nonce holds the purchase deadline in its first 8 bytes (little-endian), followed by 8 random bytes. Only the ciphertext, the public keys and the public category shard leave the device.
- 2. `post_intent`.** The relayer submits the intent. The program checks that the input key equals the result key, that the deadline falls inside the configured purchase window, and that the shard matches the book. It then creates the `Intent` account. It also copies the book's campaign map and active mask into an `IntentBook` account, records an expiry 180 slots ahead, and stores an input commitment. That commitment is a `SHA-256` digest over the intent address, the whole

book, the ciphertext, the claim key and the integrator. A one-use guard over the (x25519 key, nonce) pair blocks replays.

3. **match_intent.** The MXE scores all 32 slots against the intent (Chapter 3). The callback writes two encrypted copies of the top three to the intent account: one sealed to the intent's key and one encrypted to the MXE. It also writes a snapshot hash of the MXE copy. The program accepts the callback only if the attempt number and the input commitment match and the 180-slot window has not passed.
4. **Read the top three.** The agent decrypts the result locally and checks its intent and category hashes. It then maps the campaign and brand hashes to names using the captured public book map from `GET /v1/intents/:id/book`. The API never learns which slots won.
5. **Voucher and checkout.** `get_voucher` returns `sway:<intentId>:<slot>` and the intent's account address. The API issues a voucher only for a matched intent whose purchase deadline has not passed. The agent attaches the voucher at the merchant's checkout (browser, ACP, UCP or AP2) and pays the offer price.
6. **Receipt.** The merchant's signer signs a 128-byte `SWAYRCP1` receipt and posts it to `POST /v1/receipts`. The receipt holds the intent account, the slot, a SHA-256 hash of the order, the amount paid, the purchase time, the signer epoch and the program id. Separately, the agent calls `report_purchase` with the order hash and an evidence hash, and keeps the checkout receipt on the device.
7. **verify_conversion.** Inside MPC, the receipt's slot and price and the merchant's proposed rebate, campaign and brand are compared with the MXE copy of the top three, and a single boolean comes back. A true result becomes a certificate in Sway's Arcium coordinator program. A false one writes nothing, and the merchant can resend the receipt with the right rebate.
8. **Record and escrow.** The relay sends an Ed25519 precompile instruction immediately followed by the coordinator's `record_certified`. In that one transaction, `record_conversion` checks the signature, the receipt fields, the captured slot binding and the caps and creates the `Conversion` account, and the rebate and fees move from the campaign vault to the claims vault. `unlock_at` is set to the current time plus `lock_days × 86,400` seconds (Chapter 4).
9. **Claim.** Once `unlock_at` has passed, `claimable` shows the conversion as unlocked. `claim` signs a destination-bound `SWAYCLM1` message with the claim key. The relay pays the transaction fee, and the program pays the rebate to the chosen address and the two fees to Sway's treasury and to the integrator.

If the merchant refunds the order before `unlock_at`, a signed refund notice returns the whole debit to the campaign vault, and nothing is claimable.

3. How offers are matched

The book

Every category shard, such as running footwear, has one `OfferBook` account with 32 slots. A brand fills a slot with `submit_offer`. The `ingest_offer` computation then checks, inside MPC, that the offer is well formed, that its rebate is within the campaign's per-conversion cap, and that its brand and category match the campaign. A valid offer is re-encrypted to the MXE, written into the slot, and marked in the book's public 32-bit active mask. At matching time a slot counts as active only when its offer was accepted, its merchant is active, and its campaign vault holds at least the configured floor.

`post_intent` takes a snapshot of the book's campaign map and active mask and binds the whole book into the intent's input commitment. Matching runs against that snapshot.

Eligibility

An offer is eligible only if every check passes. The rules live in `evaluateOffer`, and the circuit is generated from that same function.

Check	Rule	Reason when it fails
Active	The slot's bit is set in the active mask	<code>inactive-slot</code>
Category	The offer's category equals the intent's	<code>category-mismatch</code>
Size	$\text{min} \leq \text{size} \leq \text{max}$, inclusive, in half-size steps	<code>size-outside-range</code>
Width	The intent's width is among the offer's widths	<code>width-not-supported</code>
Region	The intent's region is among the offer's (US in v0)	<code>region-not-supported</code>
New to brand	A new-customers-only offer excludes the brand's existing customers	<code>existing-brand-customer</code>
Price	$\text{price} \leq \text{max_price}$, inclusive	<code>price-over-maximum</code>

Score

```
score = brand_bonus + rebate - price
```

All terms are USDC base units. `brand_bonus` is what the shopper says a brand is worth to them, and it is 0 for brands they did not name. An intent carries up to 10 brand preferences. The circuit uses signed 128-bit arithmetic, so scores can be negative. Payment is first-price: the winning brand pays exactly the rebate it offered.

For example, offer A sells a 120 USDC shoe with a 15 USDC rebate and scores -105 . Offer B sells a 130 USDC shoe with a 20 USDC rebate, and the shopper gave brand B a 10 USDC bonus, so B scores $10 + 20 - 130 = -100$. B ranks above A.

Top three

The circuit evaluates all 32 slots every time, so every computation has the same shape, whatever the book holds. Eligible offers pass through a fixed three-rank insertion network. Higher scores rank first, and equal scores go to the lower slot (`lower-slot-first`). For each winner the result holds the slot, price, rebate, campaign hash and brand hash, and it carries the category and intent hashes as well. It is sealed to the intent's key, and a second copy is encrypted to the MXE for verification. If nothing is eligible, the intent ends as `cashback_fallback` (`NO_ELIGIBLE_OFFERS`).

Expiry and the one retry

A queued computation must return within 180 slots, and a later callback is rejected as stale. The agent may then call `retry_intent` once. This opens attempt 2 with a fresh 180-slot window over the same ciphertext. If that attempt also expires, the intent ends as `cashback_fallback` (`COMPUTATION_EXPIRED_TWICE`). Retries must come before the purchase deadline. See the whitepaper, "Matching inside the MXE".

4. Proof and payment

The receipt

A brand-direct sale is proven by a merchant receipt (tier T2). The receipt is a fixed 128-byte message signed with Ed25519 by the receipt signer the merchant registered on-chain. Each field ties the receipt to exactly one purchase.

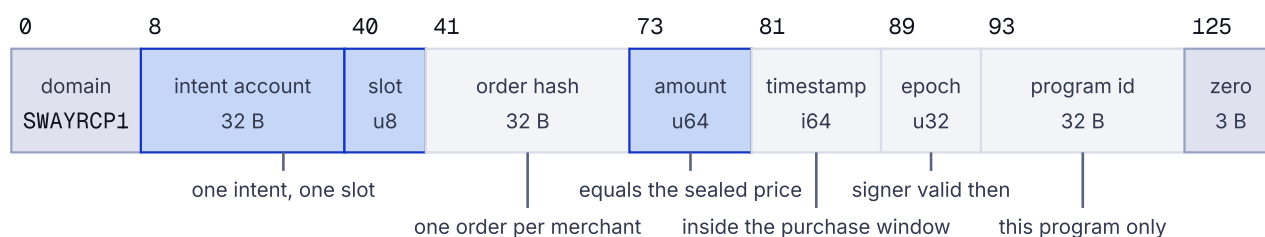


Figure 4.1 — The T2 receipt, with byte offsets. Integers are little-endian. Bytes 125–127 must be zero.

Verification first

The merchant sends the receipt with the rebate its offer carries. Before anything is recorded, `verify_conversion` runs inside MPC against the MXE copy of the top three. It looks for a rank whose slot, price, rebate, campaign hash and brand hash equal the receipt's values and the proposed rebate, under matching category and intent hashes. Only a boolean is revealed, and Sway's Arcium coordinator accepts it only after checking the cluster's signature over the output. The receipt amount is the purchase price, so a sale counts only at the sealed offer price. A true result becomes a certificate. A false result writes nothing to the financial program, so a wrong proposal never uses up the receipt or the slot.

Recording and escrow in one transaction

The relay then submits the receipt as an Ed25519 precompile instruction immediately followed by the coordinator's `record_certified`, which calls `record_conversion` and the verification callback in the same transaction. `record_conversion` reads the preceding instruction through the `instructions` sysvar. It requires the exact 128 bytes and the signer key of the epoch named in the receipt. It then checks the rest of the purchase. The intent must hold a match and still be inside its settlement deadline. The captured book must show the slot as active and bound to this campaign. The receipt must name this intent, this slot and this program. The timestamp must fall between the intent's creation and its purchase deadline, and inside the signer epoch's validity. The rebate must be positive and within both the campaign's per-conversion cap and the protocol cap of 50 USDC, and the debit must fit the vault balance, the funding and the daily cap. It creates the `Conversion` account at seeds `["conversion", intent, slot]` and an order guard at `["order", merchant, order_hash]`, so an order can back only one conversion.

The callback then moves the rebate and both fees from the campaign vault into the claims vault, which the program's Config account controls, and sets `unlock_at = now + lock_days × 86,400` seconds. `create_campaign` already rejects a lock shorter than the merchant's registered return window (`LockTooShort`). If any check fails, the whole transaction fails and nothing moves.

Refunds inside the lock

If the order comes back before `unlock_at`, the receipt signer signs a 128-byte `SWAYREF1` notice. The notice binds the conversion address, the SHA-256 of the original receipt, the program id and the signer epoch. `refund_conversion` accepts only those exact bytes. It returns the full debit, meaning the rebate and both fees, to the campaign vault and marks the conversion as refunded. After `unlock_at`, refunds are closed (`RefundWindowClosed`).

Claim to any address

Once `unlock_at` has passed, the per-intent claim key signs a 128-byte `SWAYCLM1` authorization. It covers the program id, the conversion, the destination wallet, an expiry and a 16-byte nonce. `@sway/mcp` sets the expiry 120 seconds ahead by default. The program rebuilds the message from the actual destination account's owner, checks the signature against the intent's `claim_key`, and pays the rebate to that destination, the Sway fee to the treasury and the integrator fee to the integrator's account. A copied claim cannot be redirected, and a claimed conversion cannot be claimed again (`AlreadyConsumed`). See the whitepaper, "Escrow, lock, refunds and claims".

5. Fees

Brand-direct offers

Sway charges a brand nothing to list an offer or to take part in matching. The brand is charged only when a purchase is verified. The fee basis is the rebate, and both fees are added on top of it, so the shopper receives the full rebate the brand offered.

Line	Rate	Paid by	Received by
Rebate	As offered	Brand	Shopper, at the address chosen at claim
Sway fee	1,000 bps (10%)	Brand	Sway treasury
Integrator fee	500 bps (5%)	Brand	Integrator's payout wallet

```
sway_fee      = floor(rebate × 1000 / 10000)
integrator_fee = floor(rebate × 500 / 10000)
total_debit   = rebate + sway_fee + integrator_fee
```

Fees round down to the base unit. For example, take a 120 USDC pair of shoes with a 15 USDC rebate. The rebate is 15,000,000 base units, the Sway fee 1,500,000 and the integrator fee 750,000. The campaign vault is debited 17,250,000. So the brand pays 17.25 USDC, the shopper receives 15.00, Sway 1.50 and the agent developer 0.75. The 120 USDC price goes to the merchant through ordinary checkout, and Sway takes nothing from it.

A rebate can be at most 50 USDC per conversion, and a campaign can hold at most 5,000 USDC of funding. Each campaign also sets its own per-conversion and daily caps within those limits.

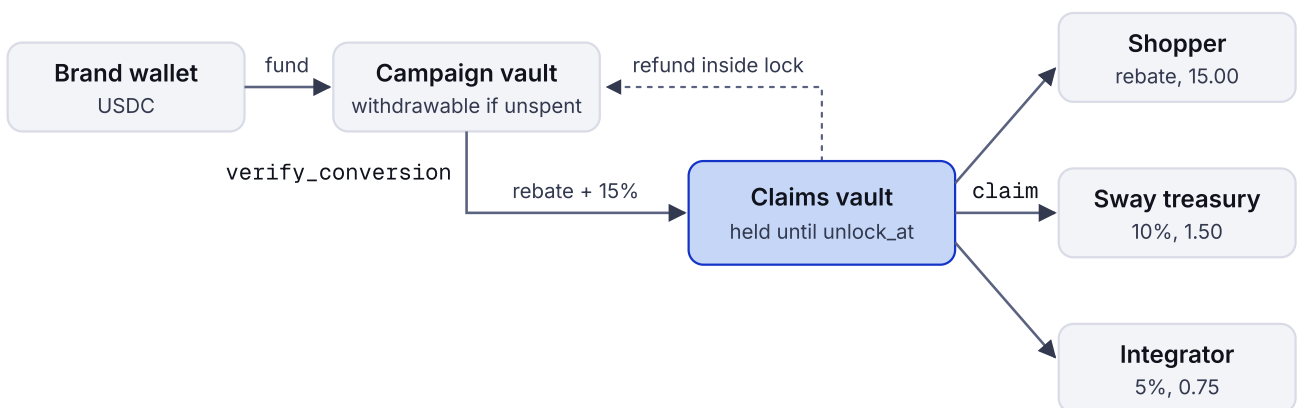


Figure 5.1 — Money flow for the 15 USDC example.

Cashback

Cashback from affiliate commissions (tier T3) is built into the program and turns on when the deployment connects an affiliate network. See the whitepaper, "Cashback fallback and the affiliate oracle".

SWAY

\$SWAY is Sway's token: 1,000,000,000 fixed supply, sold entirely in the public launch on equal terms, with no presale, team, investor or airdrop allocation and nothing vesting. Sway's whole share of every fee, the 10% Sway fee of each verified brand-direct rebate, lands in the treasury at claim and buys SWAY on the market; Sway's 20% of each cashback commission joins it once cashback runs. One transaction moves the treasury's USDC to the buyback key with `buyback_begin`, swaps it through a Jupiter route, burns the quoted minimum of SWAY and ends with `buyback_finish`, which fails the whole transaction unless the USDC was spent and the supply fell by that minimum. Any SWAY above the minimum is burned right after, so the supply only goes down. In the example above, the 15 USDC conversion sends 1.50 USDC to the burn. Every payment inside Sway stays in USDC.

6. Who learns what

Party	Learns	Never learns
Losing brands	Nothing	That the intent existed
Winning brand	The order, at purchase, as any merchant would	The shopper's other preferences, rival offers, the shopper's wallet
Sway	Category, timing, agent key, conversion amounts	Intent contents, offer rules, which brands made a top three
Public chain	Campaign spend per conversion	Who received a rebate, unless claims are linked

Each row follows from two things: where plaintext exists, and which key each output is sealed to.

Losing brands. An offer exists as plaintext only in the brand's browser and inside MPC. Matching writes its result to the intent account and nowhere else, so no brand is notified when an intent is posted or matched, and an offer that was scored but not chosen leaves its campaign unchanged.

The winning brand. The brand learns of the sale through its own checkout, like any other order. The voucher carries an intent id and a slot, and the receipt binds the order hash and the amount. Neither carries the shopper's size, maximum price, brand bonuses or the rival offers. The claim destination is chosen later and has no connection to the checkout.

Sway. The API and the relay store and forward ciphertext. In plaintext they see only the category shard, the timing, the agent key that signed each request and, at settlement, the conversion amounts. The top three are sealed to the intent's own x25519 key. The device maps hashes to campaigns and brands using the full captured book map. That map is the same for every intent posted against a given book revision, so fetching it reveals nothing about which slots won.

The public chain. The book shows which campaign occupies each slot and whether the slot is active. Each verified conversion appears as a debit of the rebate plus fees from a campaign vault. Every intent has its own claim key, and every claim names a destination chosen at claim time. Claiming several intents to one address links them, at the shopper's choice.

Outside MPC, the intent and its top three exist in plaintext only on the agent's device, where the per-intent keys stay in the operating system's keychain. See the whitepaper, "Privacy model".

7. For brands and agent developers

Brands

A brand works in the brand console with a browser wallet. For each on-chain action, the Sway API prepares the transaction, the wallet signs exactly those bytes, and Sway submits it.

1. **Sign in.** Sign a one-use challenge with the wallet.
2. **Register.** Enter the brand name, the return window in days and the public key of an Ed25519 receipt signer. The brand generates and keeps that key itself.
3. **Bond.** Post a USDC bond of at least the configured minimum. Sway activates the merchant once the bond meets it. If a dispute is upheld, the shopper is compensated from the bond.
4. **Campaign.** Set a per-conversion cap of up to 50 USDC, a daily cap, and a lock no shorter than the return window. Fund it with up to 5,000 USDC. Funds still in the vault stay withdrawable.
5. **Offers.** Set the category, size range, widths, region, new-to-brand flag, price and rebate. The browser encrypts the rule before anything is sent.
6. **Receipts.** Sign the 128-byte receipt for each order that carries a Sway voucher, and a refund notice for any return inside the lock.

Agent developers

An agent uses the `@sway/mcp` package, a local MCP server that runs over stdio in the agent's MCP host. `keygen-authority` creates an integrator key in the operating system's keychain. `register-integrator` records a name and a USDC payout wallet. After Sway approves the authority, `register-agent` adds a scoped agent key. The relayer pays the agent's transaction fees.

Tool	What it does
<code>find_offers</code>	Encrypts and posts the intent, retries once, returns up to three decrypted offers, or the intent's status with its <code>fallbackReason</code>
<code>get_voucher</code>	Returns the <code>sway:<intentId>:<slot></code> voucher for checkout
<code>report_purchase</code>	Sends order and evidence hashes. The receipt stays on the device
<code>claimable</code>	Lists amounts, unlock times and states
<code>claim</code>	Signs a destination-bound claim for one unlocked conversion

The integrator earns 5% of each brand rebate, paid to its payout wallet at claim. See the whitepaper, "Off-chain services".