The Agent Compliance Playbook

7 patterns for adding OFAC sanctions screening to your AI agent’s payment path

Pattern 1: Screen-Before-Sign

The single rule that prevents 100% of sanctioned-wallet payments.

The problem:
Your agent decides to pay, signs the tx, then screens. By then it is too late ? the tx is on-chain.

The pattern:
Call sanctions_check BEFORE the signing step, in the same code path that authorises payment. Treat the screen as a hard gate, not a log line.

Example:

```
 if not sanctions_check(wallet=to_addr).get('clean'):
     abort('blocked: OFAC SDN match')
 sign_tx(to_addr, amount)  # only reached if clean
```

Pattern 2: The 100ms Budget

Compliance that costs more than one network hop gets ripped out in week two.

The problem:
A 900ms screening call blocks the payment path, an engineer removes it, and you are unscreened again.

The pattern:
Enforce a hard latency budget. agentmail returns in <100ms p95; fail closed only if the call exceeds 500ms (configurable). Cache clean results for 60s to coalesce bursts.

Example:

```
 result = sanctions_check(wallet=w, timeout=0.5)
 if result is None:          # timed out
     return 'review'          # do NOT auto-allow on timeout
```

Pattern 3: Risk-Score the Amount, Not Just the Wallet

A clean wallet moving $50k on day one is not the same as one moving $5.

The problem:
You screen the wallet, it is clean, you allow ? and miss that a clean wallet is being used as a mule for a sanctioned actor.

The pattern:
Layer risk_score on top of sanctions_check. It weighs amount anomalies, rail (x402/AP2/ACP), wallet age, and category exposure into a 0-100 score with an explicit allow/review/decline recommendation.

Example:

```
r = risk_score(counterparty=w, amount=amt, rail='x402')
if r['recommendation'] == 'decline':
    return block_with_reason(r['reasons'])
```

Pattern 4: Know Your Agent (KYA)

The counterparty is another agent. Who vouches for it?

The problem:
Two agents transact and neither knows the other's operator. There is no KYC analogue for agents.

The pattern:
Before trusting a counterparty agent, call kya_verify with its wallet age, on-chain history, and domain. It returns a trust score and flags (fresh wallet, no history, sanctioned-adjacent).

Example:

```
k = kya_verify(agent_id=peer, wallet=peer_w,
               wallet_age_days=age, domain=peer_domain)
if k['trust_score'] < 40:
    require_human_approval()
```

Pattern 5: The Audit Trail Is the Product

Regulators do not ask 'did it work.' They ask 'prove it worked, for every tx, for 5 years.'

The problem:
You screened, but you have no timestamped, tamper-evident record. A regulator asks for proof and you cannot produce it.

The pattern:
Log every screen: the wallet, the timestamp, the result, the list version. agentmail's hosted tier writes a tamper-evident audit log you can export. Self-hosters should append to a write-once store.

Example:

```
# every screen becomes an append-only row
audit.append(ts=now, wallet=w, result=r,
             list_version=sdn_version, hash=prev_hash)
```

Pattern 6: Dispute-First, Not Litigate-First

When a payment goes wrong, a 7-day auto-escalation beats a 7-month legal fight.

The problem:
A bad payment happens and you have no structured way to contest, freeze, or recover. It becomes a lawsuit.

The pattern:
Call dispute_open the moment a paid transaction is flagged. It opens a structured dispute with a 7-day auto-escalation timer and a full evidence trail attached.

Example:

```
dispute_open(transaction_id=txid,
             reason='post-payment SDN match')
# auto-escalates at day 7 if unresolved
```

Pattern 7: Fail Closed, Degrade Loudly

Silent failure is worse than no screening ? it gives you false confidence.

The problem:
The screening API goes down, your agent catches the exception and pays anyway, and you do not find out until the fine.

The pattern:
On provider failure: (a) fail CLOSED for high-value txs, (b) fail OPEN only with explicit operator policy, and (c) ALWAYS emit a 'degraded' signal so you know screening is running on stale data. agentmail's /health and /compliance/status report this.

Example:

```
status = compliance_status()
if status.get('degraded'):
    page_oncall('screening on stale cache')
```

Closing

These seven patterns are the difference between 'we have a compliance tool'

and ’we can prove to OFAC that every payment our agent made was screened.’ Pattern 1 alone ? Screen-Before-Sign ? would have prevented every agent-payment sanctions violation on record.

If you want the version that is already wired up, pip install sanctions-mcp or hit https://agentmail-api.fly.dev/sanctions?wallet=0x... right now.

? The agentmail team sanctionsai.dev