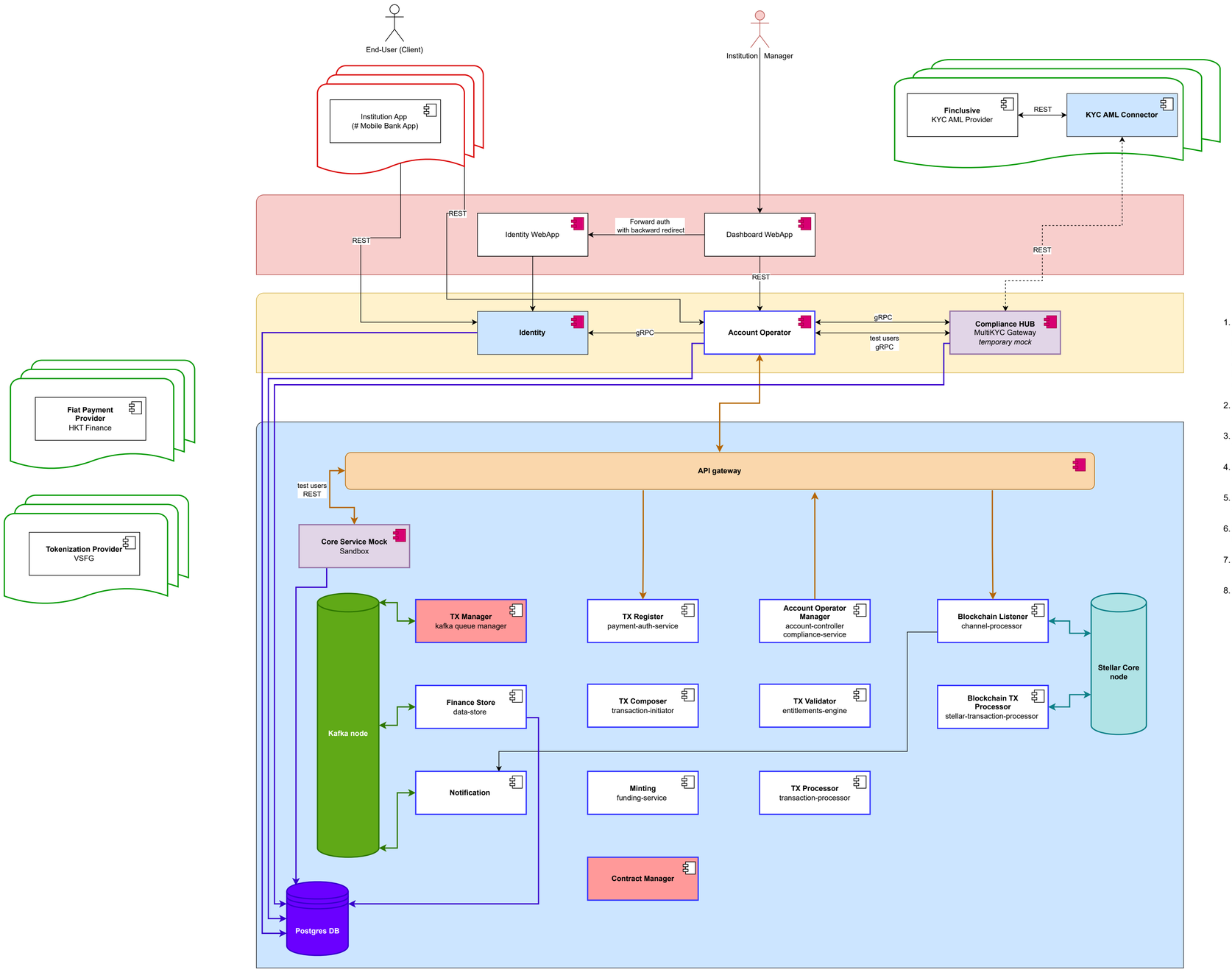


1. **Account Operator (AO Service)**
   Provides public REST API with main PwG functionality: sys.admin functions; user onboarding; user's account and permission management, finance management.
   Proxies queries from institution app through to pw.g, validating authentication and resolving (e.g.) account references
   Maintains (primarily config / preference) data on participants (wallet providers - clients of pw.g) available for use in account management, on/off-ramping and other operations
   Masters data on pw.g accounts, exposed through lifecycle management and querying (i.e. CRUD) APIs on account objects. Includes setting/getting compliance-related restrictions on accounts to participate in transactions.

2. **Dashboard WebApp**
   Showcases capabilities from an AO perspective. Exposes AO-specific features, e.g. credentials & permissions management

3. **Identity Service**
   Vends access token to all kind of users (including 3rd party services) in response to authentication flows

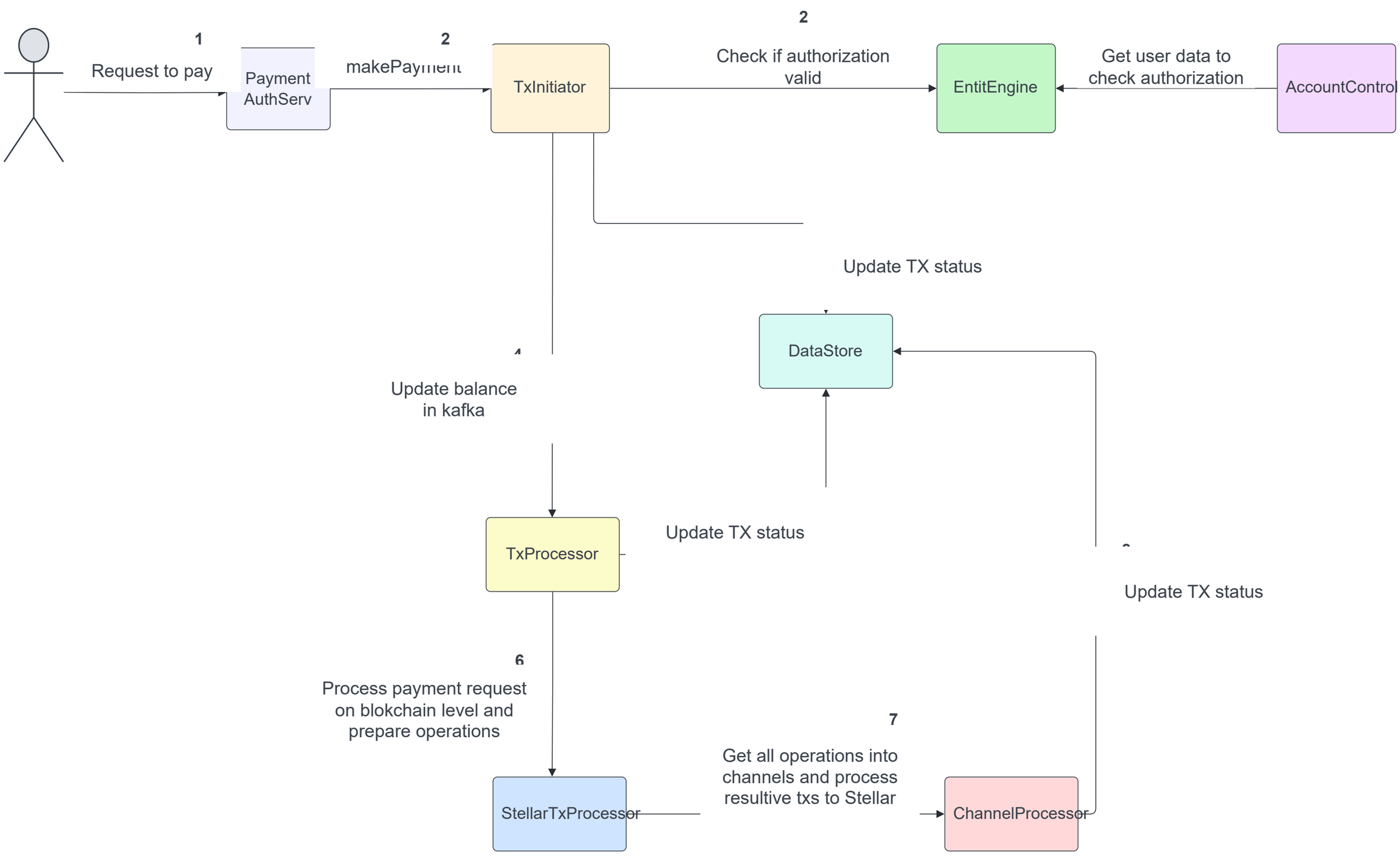
4. **KYC AML Hub**
   Smart gateway to forward and schedule updates for user's KYC from different providers.

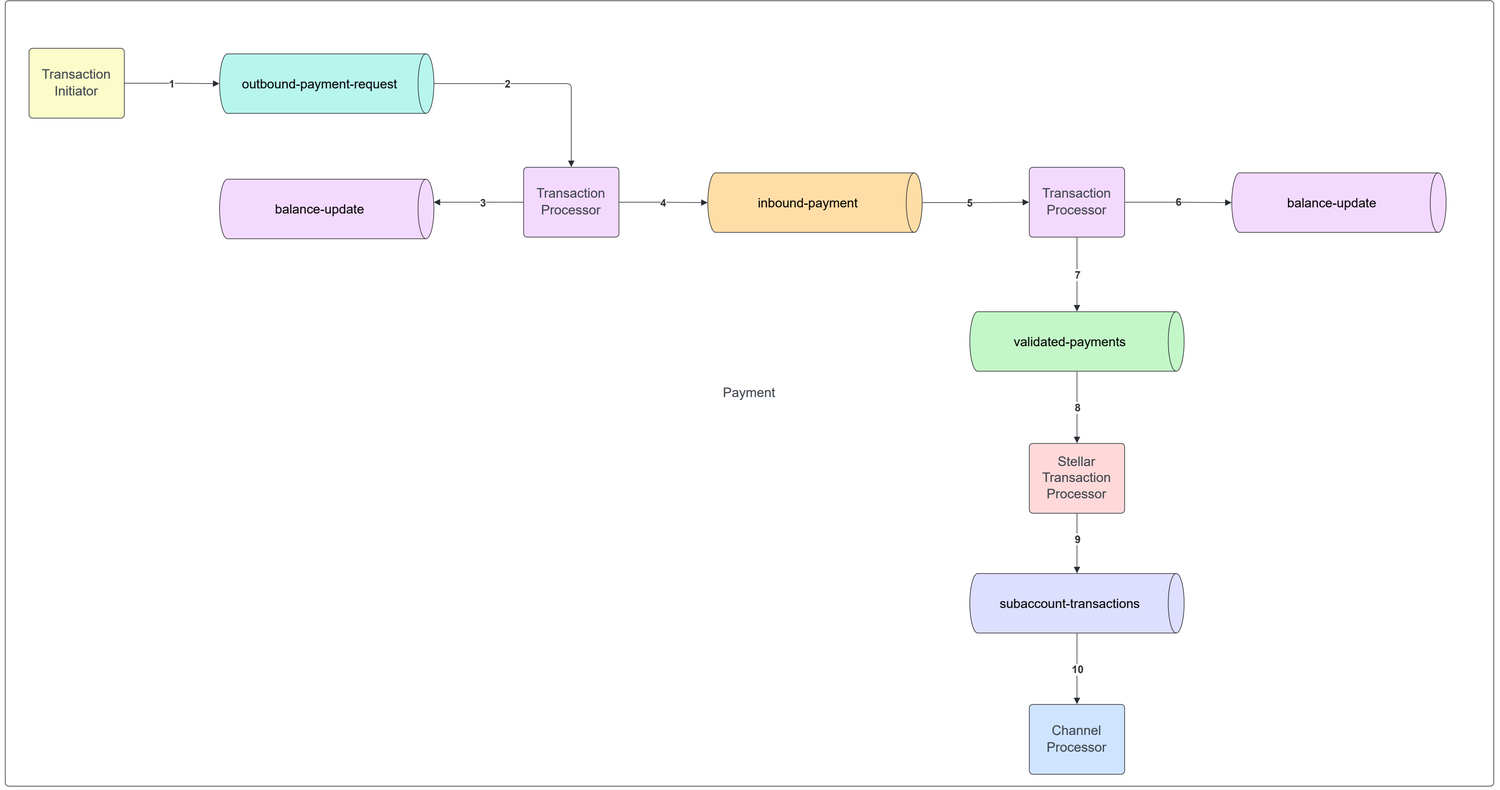
5. **KYC Connector (Gateway)**
   Service for integration KYC provider to PwG ecosystem. Transforms all standardized requests from KYC HUB to KYC provider REST API.

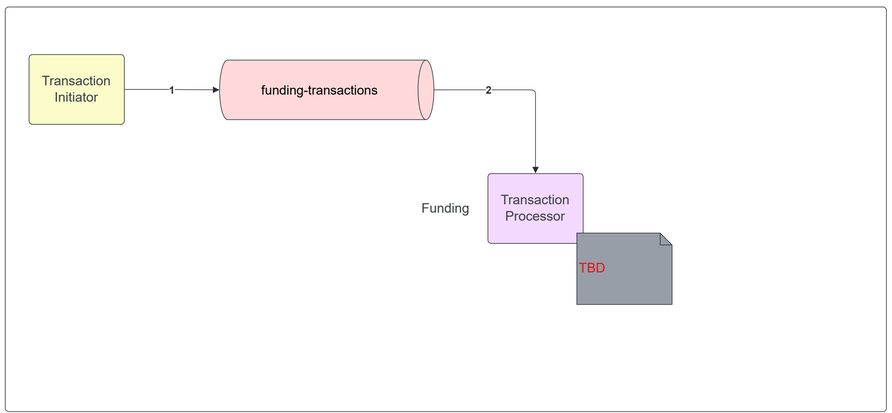
6. **Finclusive**
   Main KYC Parntner. Provides due diligence on individuals and legal entities. Used to source data for asserting KYC compliance over end-users

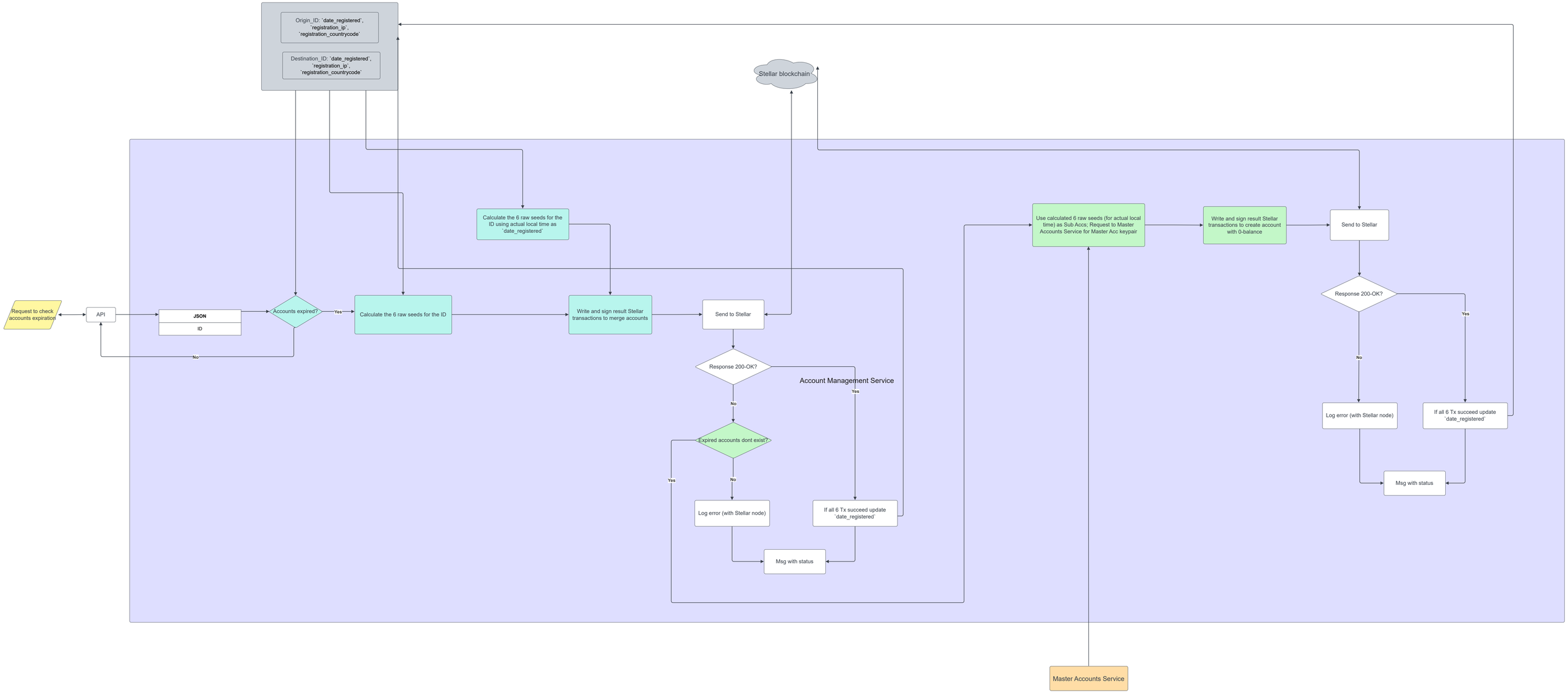
7. **API Gateway (Request Router)**
   Routes inbound requests to the relevant core service for a given data

8. **Core Service Mock (Sandbox)**
   Emulates all Core service functionality for test user's account for finance management and transaction processing without use of real components like blockchain


1
Request to pay
Payment AuthServ
2
makePayment
TxInitiator
2
Check if authorization valid
EntitEngine
Get user data to check authorization
AccountControl
Update TX status
DataStore
4
Update balance in kafka
TxProcessor
Update TX status
Update TX status
6
Process payment request on blokchain level and prepare operations
7
Get all operations into channels and process resultive txs to Stellar
StellarTxProcessor
ChannelProcessor


Transaction Initiator
1
outbound-payment-request
2
balance-update
3
Transaction Processor
4
inbound-payment
5
Transaction Processor
6
balance-update
7
validated-payments
Payment
8
Stellar Transaction Processor
9
subaccount-transactions
10
Channel Processor


Transaction Initiator
1
funding-transactions
2
Funding
Transaction Processor
TBD


Origin_ID: `date_registered`, `registration_ip`, `registration_countrycode`
Destination_ID: `date_registered`, `registration_ip`, `registration_countrycode`
Stellar blockchain
Request to check accounts expiration
API
JSON
ID
Accounts expired?
Yes
No
Calculate the 6 raw seeds for the ID
Calculate the 6 raw seeds for the ID using actual local time as `date_registered`
Write and sign result Stellar transactions to merge accounts
Send to Stellar
Response 200-OK?
Yes
No
Account Management Service
Expired accounts dont exist?
Yes
No
Log error (with Stellar node)
If all 6 Tx succeed update `date_registered`
Msg with status
Use calculated 6 raw seeds (for actual local time) as Sub Accs; Request to Master Accounts Service for Master Acc keypair
Write and sign result Stellar transactions to create account with 0-balance
Send to Stellar
Response 200-OK?
Yes
No
Log error (with Stellar node)
If all 6 Tx succeed update `date_registered`
Msg with status
Master Accounts Service

validated-payments

payment transactions consumer

Service

Payments

Seeding

Stellar Client

Cycling

Payments

Shuffling

Producer

Data Store Client

Funding Service Client

Account Controller

Funding Service

Stellar

subaccount-transactions

Data Store

Funding Service Client

Seeding

Stellar Client

Cycling

Payments

Shuffling

Data Store Client

Fundings

validated-fundings

funding transactions consumer

Service

Account Controller

Stellar

Data Store

Seeding

Stellar Client

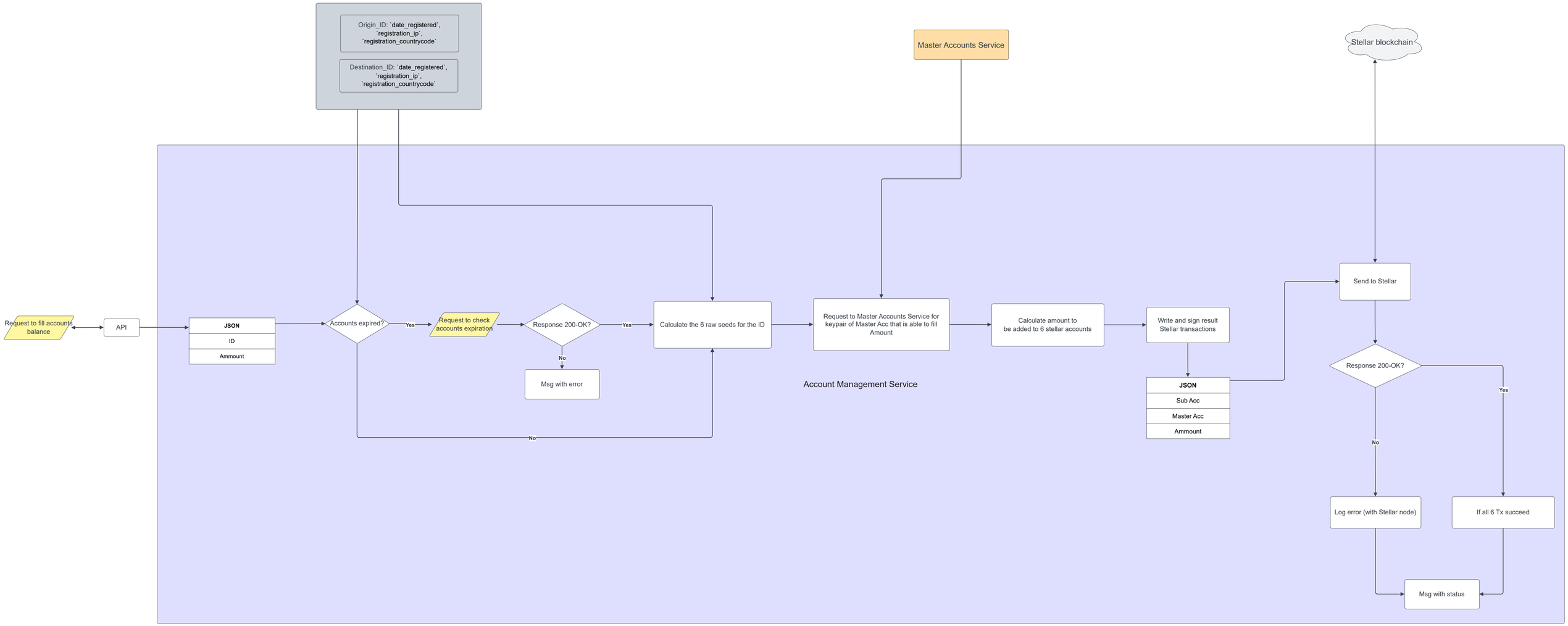
Data Store Client

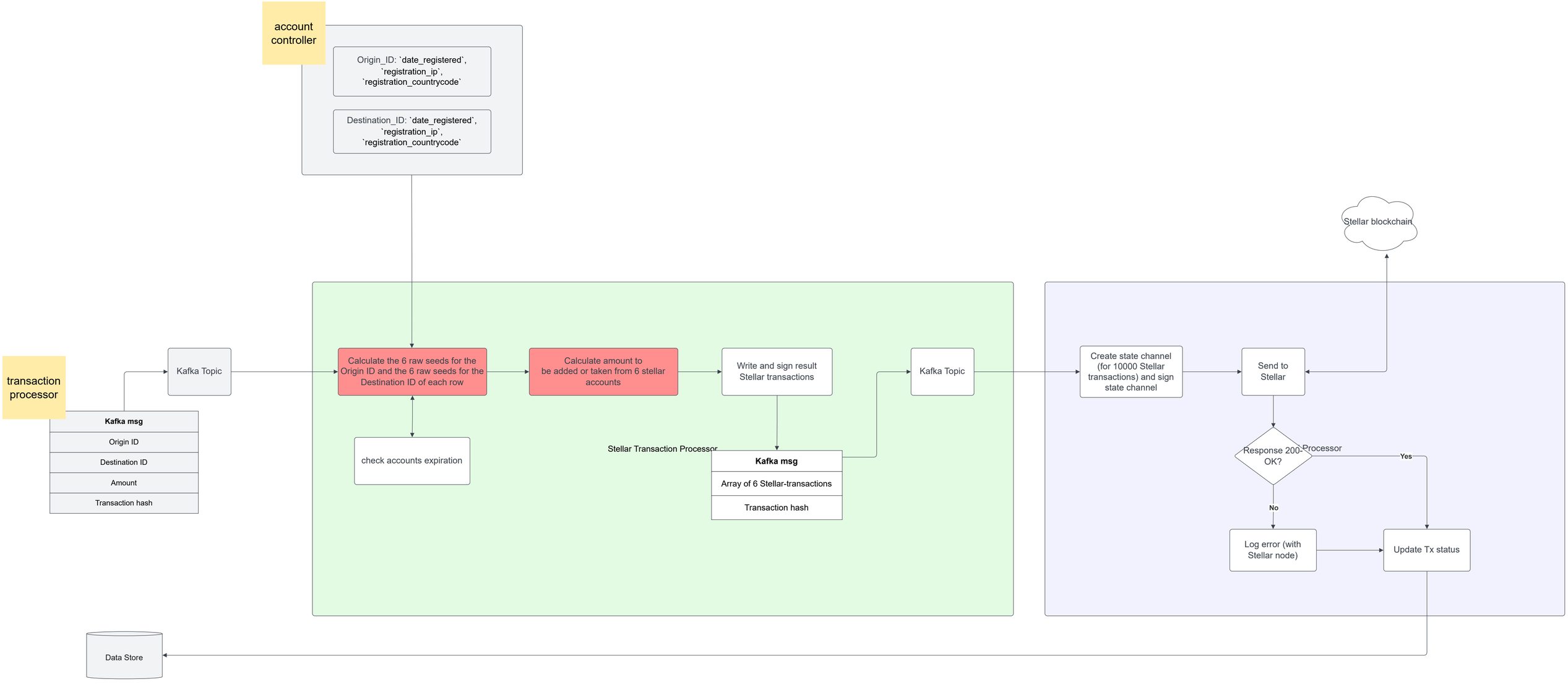
Trustlines

accounts-trustlines

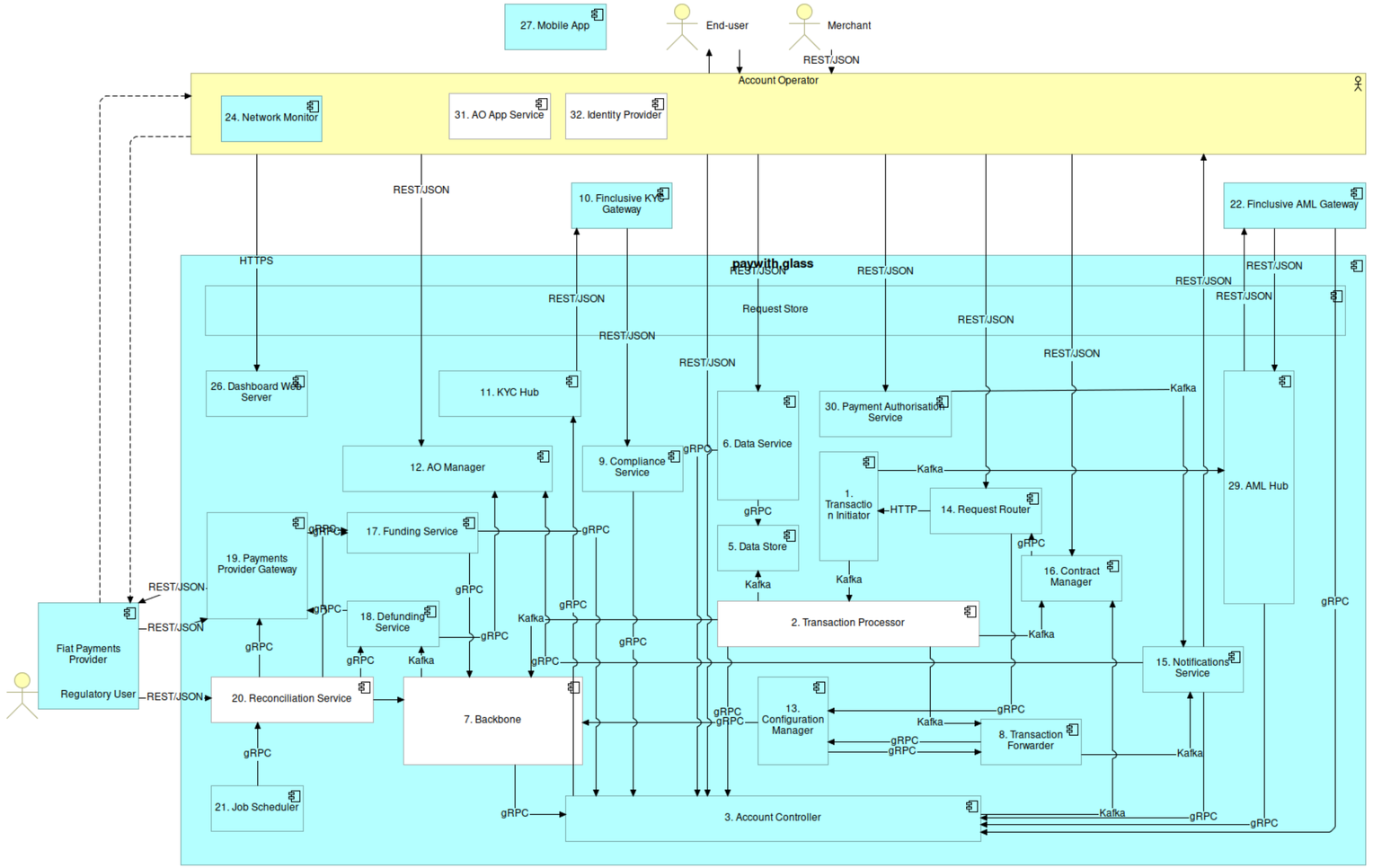
trustlines consumer

Service


Origin_ID: `date_registered`, `registration_ip`, `registration_countrycode`
Destination_ID: `date_registered`, `registration_ip`, `registration_countrycode`
Master Accounts Service
Stellar blockchain
Request to fill accounts balance
API
JSON
ID
Ammount
Accounts expired?
Yes
Request to check accounts expiration
Response 200-OK?
Yes
No
Msg with error
Calculate the 6 raw seeds for the ID
No
Request to Master Accounts Service for keypair of Master Acc that is able to fill Amount
Calculate amount to be added to 6 stellar accounts
Write and sign result Stellar transactions
JSON
Sub Acc
Master Acc
Ammount
Account Management Service
Send to Stellar
Response 200-OK?
Yes
No
Log error (with Stellar node)
If all 6 Tx succeed
Msg with status


account controller
Origin_ID: `date_registered`, `registration_ip`, `registration_countrycode`
Destination_ID: `date_registered`, `registration_ip`, `registration_countrycode`
Stellar blockchain
transaction processor
Kafka Topic
Kafka msg
Origin ID
Destination ID
Amount
Transaction hash
Calculate the 6 raw seeds for the Origin ID and the 6 raw seeds for the Destination ID of each row
check accounts expiration
Calculate amount to be added or taken from 6 stellar accounts
Write and sign result Stellar transactions
Stellar Transaction Processor
Kafka msg
Array of 6 Stellar-transactions
Transaction hash
Kafka Topic
Create state channel (for 10000 Stellar transactions) and sign state channel
Send to Stellar
Response 200-OK?
Processor
Yes
No
Log error (with Stellar node)
Update Tx status
Data Store

The platform comprises a number of discrete, loosely-coupled applications/services interconnected using standard technologies.
All services will have high-availability support with state data replicated between host nodes. Each service is designed to be independently deployable, hence supporting a range of deployment models with varying degrees of centralisation.



1. Transaction Initiator: Passes transactions to request queue and listens for outcomes to respond to user

2. Transaction Processor: Performs checks to validate inbound transaction request

3. Account Controller: Masters data on pw.g accounts, exposed through lifecycle management and querying (i.e. CRUD) APIs on account objects. Includes setting/getting compliance-related restrictions on accounts to participate in

6. Data Service: Provides querying operations on historic transactions and account balances

7. Backbone: Receives and processes settlement (ensuring finality) of transactions

8. Transaction Forwarder: Forwards transaction messages to topic(s) partitioned by the payee to enable timely refresh of local balances.

9. Compliance Service: Retains record of compliance assertions provided for each account. Monitors compliance assertions against defined policy (minimum KYC requirements) and applies account restrictions (via Account Manager) when

10. Finclusive KYC Gateway: Sources compliance assertions from Finclusive when needed to maintain compliance for accounts

11. KYC Hub: Directory of KYC providers. Also handles pass-through authentication

12. AO Manager: Maintains (primarily config / preference) data on participants (wallet providers - clients of pw.g) available for use in account management, on/off-ramping and other operations

13. Configuration Manager: Maintains / serves mappings of accounts to individual Kafka queues to enable accurate routing

14. Request Router: Routes inbound transaction requests to the relevant instance for a given account

15. Notifications Service: Provides push notifications to the wallet providers for key events (inbound payments, account lifecycle events, etc.)

16. Contract Manager: Allows definition and re-use of custom logic governing transactions - scheduled events, testing external conditions etc.

17. Funding Service: Listens to inbound payments to dFMI and triggers corresponding issuances of dSterling onto the ledger

18. Defunding Service: Listens for transactions which reflect off-ramping of dSterling and executes corresponding off-ledger payments

19. Payments Provider Gateway: Provides format translation to generalise / connect Modulr-specific interfaces to pw.g-generic services.

20. Reconciliation Service: Sources balance information from external payment processors, backbone and funding/defunding services to produce reconciliation reports demonstrating equality of assets on ledger vs. in backing reserves

21. Job Scheduler: Executes arbitrary jobs at pre-scheduled intervals; reports on execution failures

22. Finclusive AML Gateway:
Sources compliance assertions from Finclusive when needed to maintain compliance for accounts

24. Network Monitor: Monitors network latency between AO locations and pw.g nodes, sending feedback to the configuration manager to inform account re-homing decisions

27. Mobile App: Showcases capabilities from an end-user perspective

26. Dashboard Web Server: Showcases capabilities from an AO perspective. Exposes AO-specific features, e.g. credentials & permissions management

29. AML Hub: Routes transactions through to the relevant AML gateway for external transaction monitoring

30. Payment Authorisation Service: Caches payment requests to be processed following acquisition & approval by an account

31. AO App Service: Proxies queries from mobile app through to pw.g, validating authentication and resolving (e.g.) account references

32. Identity Provider: Vends session tokens to mobile app clients in response to authentication flows

Finclusive: Provides due diligence on individuals and legal entities. Used to source data for asserting KYC compliance over end-users

Fiat Payments Provider: Provides account management and inbound/outbound payments for traditional fiat currencies. Used to enable on-/off-ramping of dSterling vs. GBP