AA Interoperability

Functional description and API specification

- Interoperability by federation, not by a central switch
- Two gateway functions inside every AA: DGW and SGW
- Four request hops and four notification hops
- Twelve stitches: discovery, linking, consent and data pull

Sahamati. AA-Interoperability-GW project. Working draft for technical review. September 2026.

All AA, FIP, FIU names and users are illustrative: AA-1 to AA-17, FIP-1 to FIP-7, FIU-4, Priya, Ashwin, Vidya.

The problem

An FIU reaches only its own AA. An AA reaches only the FIPs it is connected to.

17 licensed AAs
unlikely to change

~200 FIPs
sparse, uneven connectivity

~800 FIUs
each contracted with 3 to 4 AAs

Demand side: the FIU's AA is not the user's AA

- Priya holds a forgotten VUA at AA-3
- She approaches FIU-4, which works with AA-4
- Today she must create another VUA at AA-4
- Priya has multiple handles across multiple AAs (proliferation)
- The FIU should discover AA-3 and serve her there

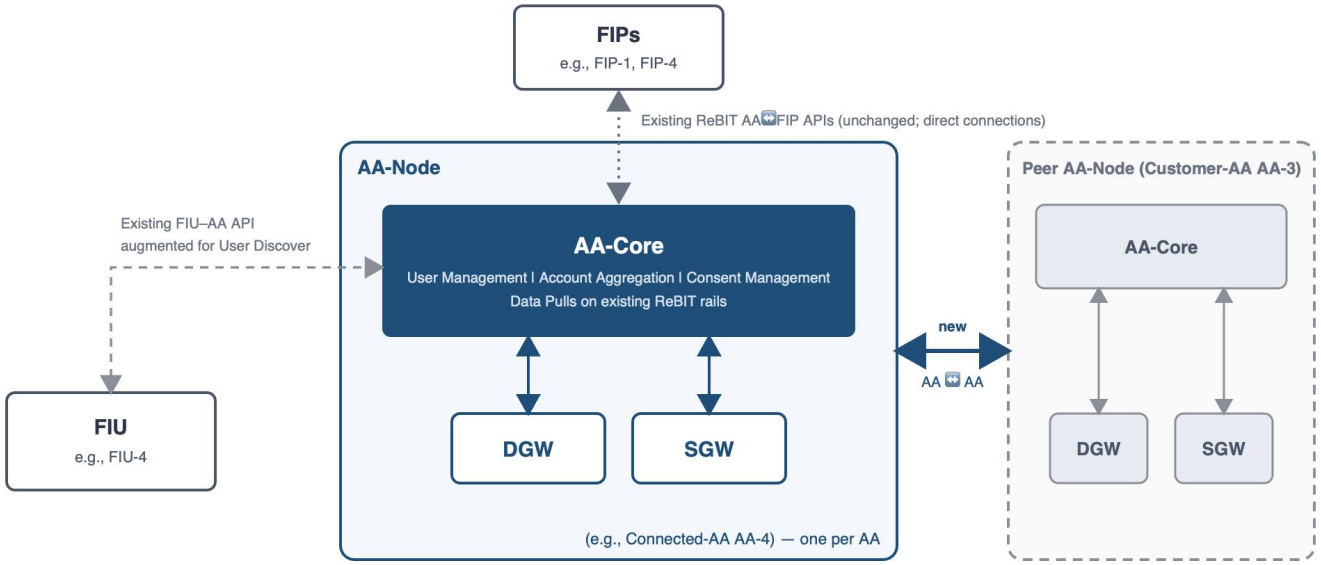
Supply side: the user's AA is not connected to every FIP

- The FIP to AA connectivity matrix is sparse and uneven
- An AA's ability to sell to FIUs depends on its FIP reach
- Early movers connected to the largest FIPs first
- Onboarding with FIP is very cumbersome and time consuming.
- The Solution: must extend reach without asking same FIPs to onboard more AAs

- *Three users bound the problem. Ashwin: VUA at the FIU's own AA, must work exactly as today.*
- *Priya: VUA elsewhere, accounts across FIPs. The interoperability case.*
- *Vidya: no VUA anywhere, routed to fresh onboarding at the FIU's AA.*

High-level statement of interoperability

Interoperability is achieved by federation, not by a central switch.



New

AA-Node ↔ AA-Node — the interoperability interface between two AAs (to be defined)
AA-Core ↔ DGW / SGW — handover inside the AA-Node (implementation-defined)

Existing

FIU ↔ AA-Core — existing FIU–AA API, augmented for User Discover
AA-Core ↔ FIP — ReBIT FIP_2_1_0 + AA-hosted FIP callbacks

Dashed/dotted = existing ReBIT rails. All names illustrative.

Two gateway functions

- Every AA keeps its core functionality and also provides gateway functionality
- DGW is the demand-side gateway role
- SGW is the supply-side gateway role
- These are functional elements, not necessarily components
- The logic may sit inside or outside the AA core application

Which API to call

hop-AA: the Connected-AA or the Inter-AA that sits between the two ends of a leg and forwards the message when the ends are not directly connected.

FIU to AA-Node

- Via the hop-AA: FIU calls /iop/{Request-X}. Otherwise /{Request-X}
  {Request-X} = Consent, Consent/handle, Consent/fetch, FI/request, FI/fetch, User/discover

AA-Node to FIP

- Via the hop-AA: AA calls /iop/{Request-X} on the FIP. Otherwise /{Request-X}
  {Request-X} = Accounts/discover, Accounts/link, Accounts/link/verify, Accounts/delink, Consent, Consent/Notification, FI/request, FI/fetch

FIP to AA-Node (notification, mirror of the above)

- Via the hop-AA: /iop/{Notification-X}. Otherwise /{Notification-X}
  {Notification-X} = FI/Notification, Consent/Notification

AA-Node to FIU (notification, mirror of the above)

- Via the hop-AA: /iop/{Notification-X}. Otherwise /{Notification-X}
  {Notification-X} = Consent/Notification, FI/Notification

Call /{Request-X} or /iop/{Request-X} based on the need to use the hop-AA. Same rule for /{Notification-X}.

Participants and functions

Participants are the regulated entities. Functions are what an AA does internally.

Participants: FIU, AA, FIP

- Regulated entities that hold licences, tokens and signing keys
- Onboarded to each other bilaterally

Functions: AA-Core, DGW, SGW

- What an AA does internally
- No token of its own and no endpoint of its own in this specification

AA-Core

- The licensed AA's existing platform
- User management, account aggregation, consent management, data pulls on existing ReBIT rails
- Consent manager of record for the VUAs it holds
- Behaviour toward FIUs and FIPs is unchanged
- Sets and consumes the gateway routing headers for traffic to or from another AA
- Makes the routing decision per target FIP

DGW (Demand Gateway)

- A minimalistic, stateless conduit
- Sits behind AA-Core. The FIU keeps calling the AA-Core
- Discovery: resolves a user identifier. Runs for every user
- Gateway: a pure conduit between the FIU and the Customer-AA
- Engages only when the VUA lives at another AA
- Connected-AA and Customer-AA are scenario-scoped designations, not separate functions

SGW (Supply Gateway)

- A minimalistic, stateless transit conduit
- Carries an Origin-AA's FIP-bound traffic over the Inter-AA's existing FIP connection
- Single hop only
- Preserves the origin's signature and payload untouched
- It does not look at the payload

- *FIU: contracts unchanged. Learns one new capability, handle discovery. Otherwise calls its contracted AA on the existing endpoints.*
- *FIP: unchanged for direct traffic. On transit traffic it verifies the Origin-AA's token and signature and echoes the routing identifiers.*

Example topology and users

The indicative topology used by every illustration and stitch that follows.

Users

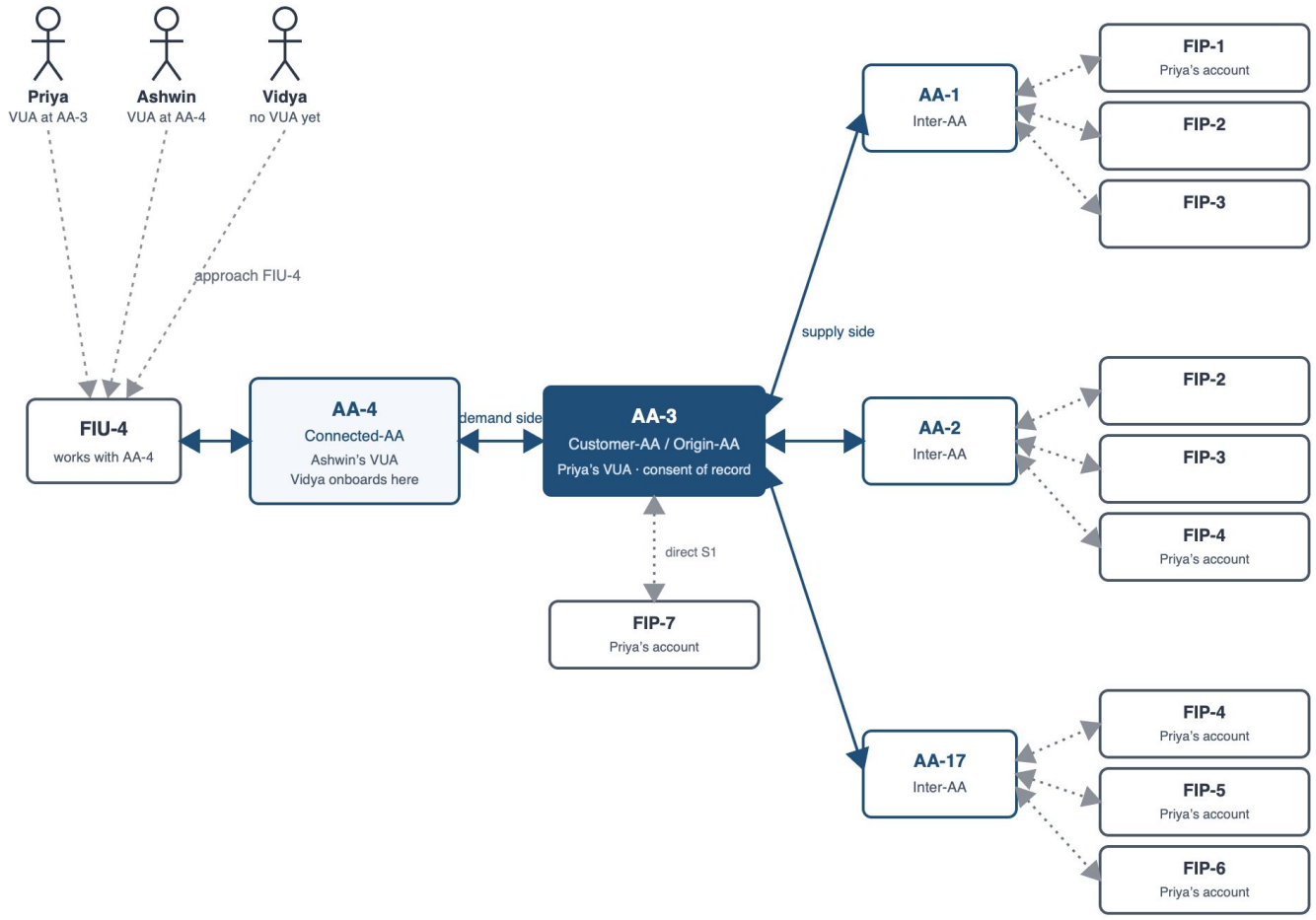
- Vidya has no VUA yet
- Ashwin has a VUA at AA-4
- Priya already has a VUA at AA-3

Role Definition

- **Connected-AA**: the AA the FIU is contracted with; on the demand side it takes the FIU's request as today and forwards it to the Customer-AA, and brings the notification back.
- **Customer-AA**: the AA that holds the user's VUA and consent of record; on the demand side it is where the FIU's request ends and is acted on.
- **Origin-AA**: the same AA (as Customer-AA) as seen from the supply side; it originates every request to the FIPs and is where their notifications from FIPs
- **Inter-AA**: the AA the FIP is contracted with; on the supply side it carries the Origin-AA's request to the FIP as today, and brings the FIP's notification back. come home.

Roles in this example

- AA-3 is the Customer-AA (demand) and the Origin-AA (supply)
- AA-4 is the Connected-AA
- AA-1, AA-2 and AA-17 are Inter-AAs



Demand side: FIU-4 ↔ AA-4 (Connected-AA) ↔ AA-3 (Customer-AA). Supply side: AA-3 (Origin-AA) → FIP-7 (direct) and ↔ Inter-AAs AA-1 / AA-2 / AA-17.

Each FIP is shown once per connected Inter-AA (FIP-2, FIP-3, FIP-4 are reachable via two AAs). Dotted = existing ReBIT connections; solid = gateway paths. All names illustrative.

Terminology

The terms used from here on

Term	Definition
Participants and functions	
AA-Node	The complete AA deployment: the AA-Core plus its two gateway roles. One administrative domain.
AA-Core	The AA as it exists today. May need changes to adapt to interoperability.
GW (gateway)	Generic term for the two gateway roles. Stateless. Never re-signs. Never reads payloads.
DGW	The Demand Gateway. Assists the AA in communicating with other AAs on the demand side.
SGW	The Supply Gateway. Assists the AA in communicating with other AAs on the supply side.
FIU / FIP	The existing ReBIT actors. Unchanged. An FIU works with its own AA only.
Roles: designated per user journey, not fixed properties	
Connected-AA	Demand side. The FIU's own AA, which relays the FIU's requests toward the Customer-AA.
Customer-AA	Demand side. The AA holding the user's VUA. Serves the consent and data flows. Also the Origin-AA on the pull.
Origin-AA	Supply side. The AA that originates and owns an FIP-bound interaction end to end.
Inter-AA	Supply side. The AA carrying another AA's FIP-bound traffic in transit. Exactly one per path.
hop-AA (or hop)	The Connected-AA or the Inter-AA when it sits between the two ends of a leg and forwards the message. Which one is meant follows from the leg being discussed.
Paths and ecosystem infrastructure	
`/iop/*`	Parallel APIs for interoperability.
Sahamati CR	The Central Registry. entityId, endpoints and public keys of every participant.
Sahamati Token Service	Issues the tokens. Each token carries the holder's entityId as a claim.
CR-AIG	AA Interoperability Graph. An enhanced capability of the CR. Which AA connects to which FIPs and AAs.
VUA	Virtual User Address. The user's handle at an AA, for example priya@AA-3.

Design guidelines

The rules the rest of the design follows from. 17 AAs absorb bulk of the change, lesser changes at FIU and FIP

1. **Principle of least change**
   - Least change in code and functionality
   - Continue with interfaces that can be reused
   - Least change in the order of FIU, FIP and AA
   - No new endpoints
2. **Minimalism**
   - The gateway is a conduit
   - It adds only the controls the receiver needs to trust the traffic
   - APIs mirror existing ReBIT call shapes. No new endpoints
3. **Chain of trust**
   - Channel trust is hop-by-hop and for the path
   - No hop is ever implicit
4. **Trust mirroring**
   - Every mechanism on the request path is mirrored on the notification path
   - The same headers and the same two-layer validation, in reverse
5. **Stateless transit and source routing**
   - Routing information travels with the message and is never stored by a GW
   - Any connected transit can carry a notification, so multi-path return is free
6. **Data blindness in transit**
   - Transit gateways cannot read FI data, which stays encrypted FIP to FIU
   - The GW layer adds no customer identifiers to headers or logs
7. **Single-hop transit**
   - Transit depth is exactly one. With 17 AAs a full mesh is trivial
   - Multi-hop buys nothing and complicates trust, latency and liability
8. **Consent of record stays at the Customer-AA**
   - Consent artefacts live and are approved at the AA holding the VUA
   - Sessions bind to the Origin-AA, not to the path, so failover is safe
9. **No central service in the hot path**
   - CR-AIG lookups sit outside discovery, linking, consent and fetch
   - AA-Cores cache the data and refresh at a configured interval
10. **Deterministic error semantics (to be defined)**
    - Distinct codes for channel rejection, origin rejection and misrouting
    - Failures must be debuggable across four parties without leaking data
11. **What the GW does and does not do**
    - Mechanical checks only: loop prevention, route recording, policy
    - Never re-originates, modifies, vouches for or re-signs content
12. **Transit payload encryption**
    - The transit payload is the request or notification body that a hop-AA forwards. FI data is already encrypted FIP to FIU
    - Transit traffic must be encrypted. The hop-AA cannot read what it forwards

High-level flow: the request path

Every AA-Node reaches every other AA-Node through its own gateway functions. FIUs and FIPs stay connected to their own AA, exactly as today.

Outer ring: FIUs and FIPs

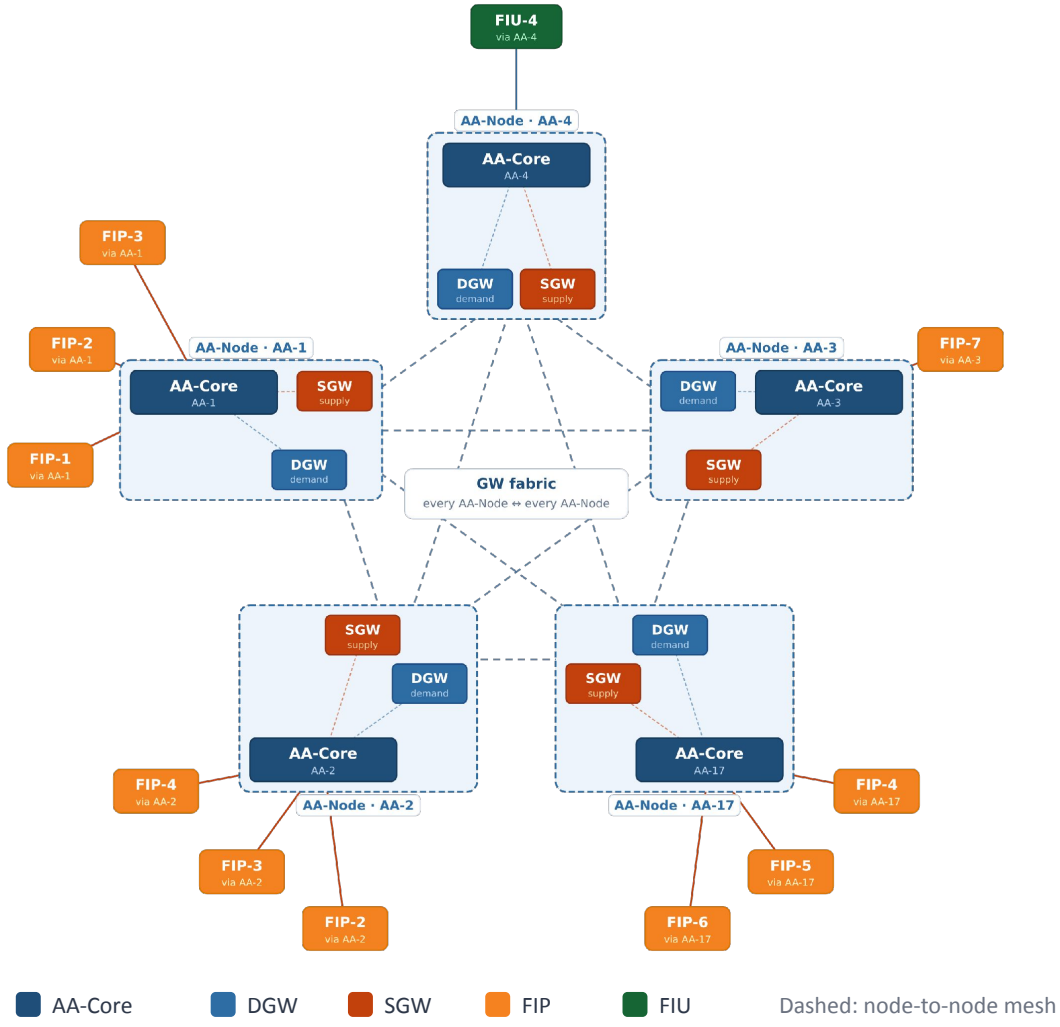
The regulated entities. Each FIU and FIP keeps its existing contract with one AA. The FIP-2, FIP-3 and FIP-4 boxes appear more than once because several AAs serve them, as today.

Middle ring: AA-Core

The AA as it exists now. It is the only component that faces FIUs and FIPs, and the only exit to an FIP. AA-Core is extended minimally, not rebuilt.

Inner ring: DGW and SGW

Two gateway functions inside every AA-Node. The Demand Gateway carries FIU-side traffic between AAs; the Supply Gateway carries FIP-side traffic. Together they form a full mesh: every node reaches every other node in one hop.



The gateway layer is the only new element. Nothing changes for the FIU or the FIP.

Background: the current ReBIT API map

The gateway layer is defined against the existing ReBIT interfaces: AA_2_1_0, FIP_2_1_0 and FIU_2_0_0.

APIs the FIU calls on the AA

Hosted by the AA, per AA_2_1_0

```
POST /Consent
POST /Consent/handle
POST /Consent/fetch
POST /FI/request
POST /FI/fetch
GET  /Heartbeat
```

Authorization key: client_api_key

APIs the AA calls on the FIP

Hosted by the FIP, per FIP_2_1_0

```
POST /Accounts/discover
POST /Accounts/link
POST /Accounts/link/verify
POST /Accounts/delink
POST /Consent
POST /Consent/Notification
POST /FI/request
POST /FI/fetch
GET  /Heartbeat
```

Authorization key: aa_api_key

APIs the FIP calls on the AA

Callback endpoints hosted by the AA, per AA_2_1_0

```
POST /Account/link/Notification
POST /FI/Notification
POST /Consent/Notification
```

Authorization key: fip_api_key

APIs the AA calls on the FIU

The FIU callback API, FIU_2_0_0. Exactly two endpoints

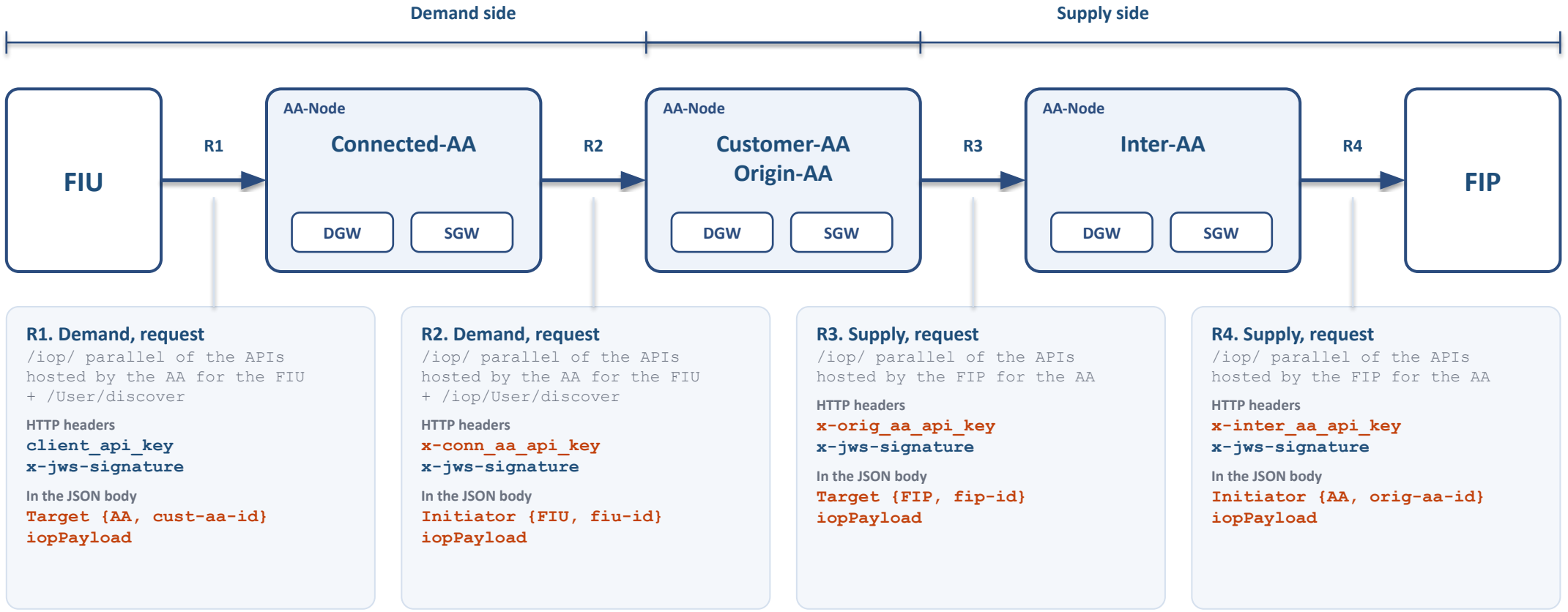
```
POST /Consent/Notification
POST /FI/Notification
```

Authorization key: aa_api_key

Interoperability adds one API to the FIU to AA interface, /User/discover, and an /iop/ parallel of these families on every interface for traffic that goes via a hop-AA. Nothing else on this map changes.

High-level flow: the request path

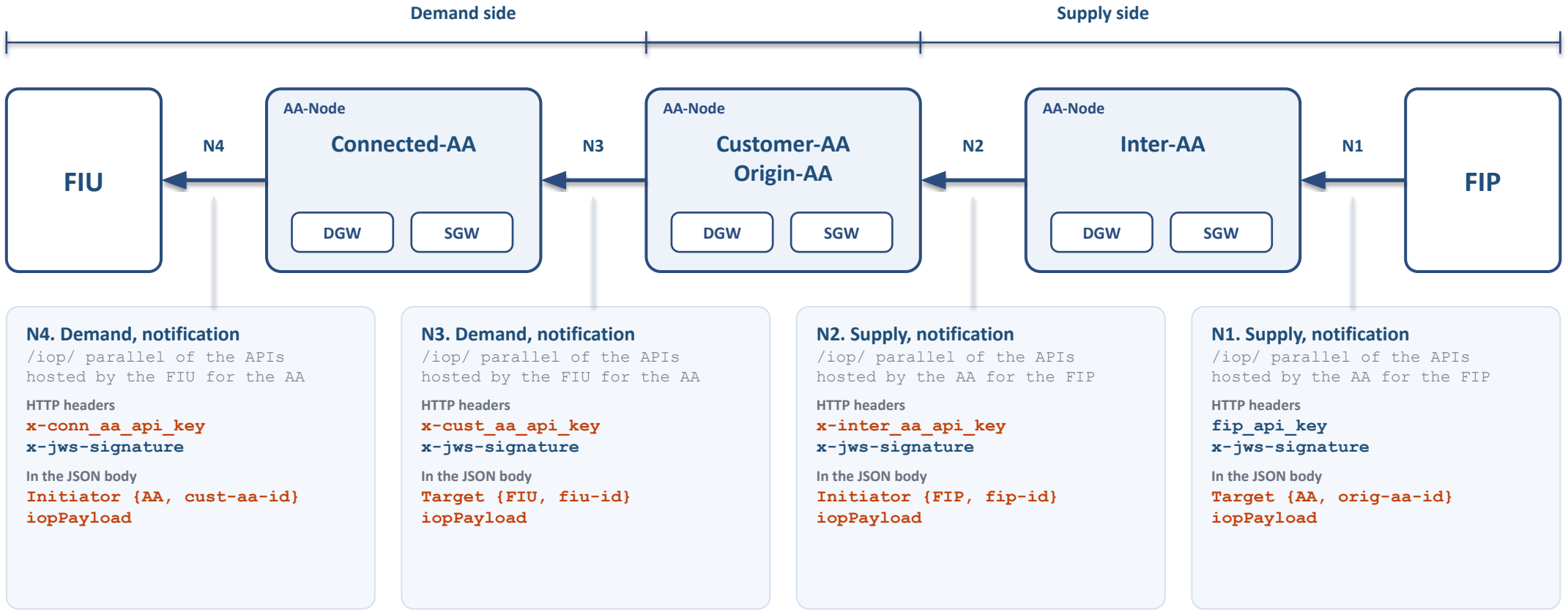
A request goes from the FIU to the FIP in four hops. The AA in the middle is the Customer-AA on the demand side and the Origin-AA on the supply side.



x-jws-signature (redefined): detached JWS of the outer body by the immediate sender of the hop.
Hop check at every receiver: the caller's key and x-jws-signature.
Routing travels in the clear part of the JSON body: Target on the hop into the hop-AA, Initiator on the hop out of it.
iopPayload: the ReBIT body, signed by the originator and encrypted for the far end (JWE). The hop-AA forwards it and cannot read it.

High-level flow: the notification path

The notification path is the request path in reverse. N1 reverses R4 and N4 reverses R1.



x-jws-signature (redefined): detached JWS of the outer body by the immediate sender of the hop.
Hop check at every receiver: the caller's key and x-jws-signature.
Routing travels in the clear part of the JSON body: Target on the hop into the hop-AA, Initiator on the hop out of it.
iopPayload: the ReBIT body, signed by the originator and encrypted for the far end (JWE). The hop-AA forwards it and cannot read it.

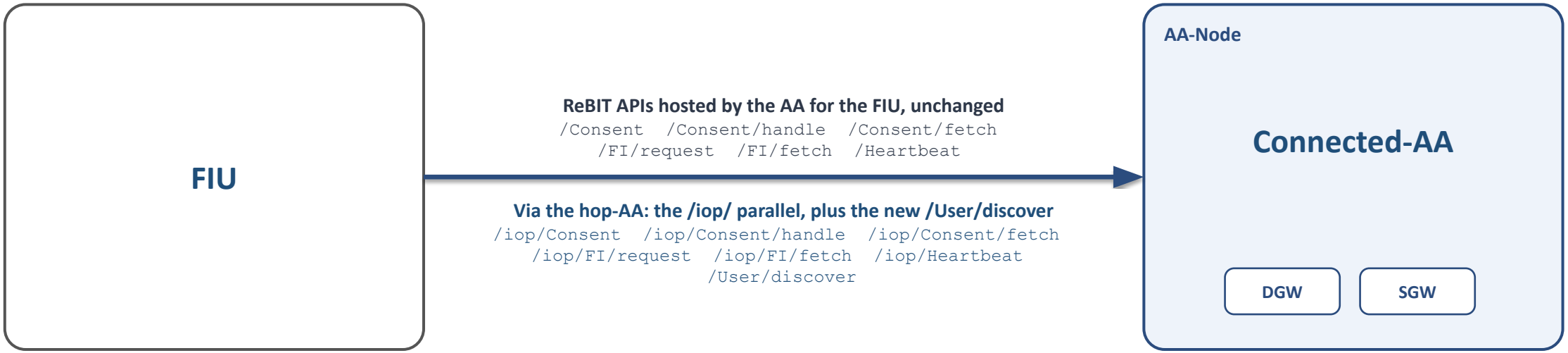
Interface channels

Where the new work sits, and what stays as it is today.

Connects	Scope	Status	Description	APIs
Connected-AA to Customer-AA	External	**New**	Demand-side interoperability between AAs. Locates the customer's VUA. Forwards consent and FI interactions, request and notification paths. iopPayload encrypted for the far end; Target or Initiator in the clear.	/iop/User/discover, /iop/Consent, /iop/Consent/handle, /iop/Consent/fetch, /iop/FI/request, /iop/FI/fetch, /iop/Consent/Notification, /iop/FI/Notification, /iop/Heartbeat. Keys: x-conn_aa_api_key, x-cust_aa_api_key.
Origin-AA to Inter-AA	External	**New**	Supply-side transit toward FIPs the Origin-AA is not connected to. Single hop. Stateless source routing. Data-blind: iopPayload encrypted for the far end; Target or Initiator in the clear.	/iop/Account/discover, /iop/Account/link, /iop/Account/link/verify, /iop/Account/delink, /iop/Consent, /iop/FI/request, /iop/FI/fetch, /iop/Consent/Notification, /iop/FI/Notification, /iop/Account/link/Notification, /iop/Heartbeat. Keys: x-orig_aa_api_key, x-inter_aa_api_key.
FIU to AA-Core	External	**Extended**	Augmented with user discovery. Traffic via a hop-AA uses the /iop/{Request-X} parallel APIs with an encrypted iopPayload. Direct traffic unchanged.	ReBIT AA_2_1_0 and FIU_2_0_0 unchanged for direct calls. New: /User/discover; FIU calls /iop/{Request-X} with client_api_key; FIU hosts /iop/Consent/Notification and /iop/FI/Notification, called with x-conn_aa_api_key.
AA-Core to FIP	External	**Extended**	Channel unchanged: IP whitelist, TLS 1.2+. Direct traffic unchanged. Transit traffic arrives on the /iop/ parallel APIs with an encrypted iopPayload and an Initiator naming the Origin-AA.	ReBIT FIP_2_1_0 unchanged for direct calls. New: FIP hosts /iop/{Request-X}, called with x-inter_aa_api_key; FIP calls /iop/{Notification-X} on the Inter-AA with fip_api_key.
Any participant to Sahamati CR and Token Service	External	**Extended**	Identity, keys and tokens. Extended: one encryption public key per participant for iopPayload; interop-enabled flag per participant; the four role keys issued by the Token Service.	No change in APIs. AAs acting in gateway roles register with the CR. Every participant publishes an encryption key.
AA-Node to CR-AIG	External	**Extended**	Path discovery. All AAs reach all AAs, so an AA is never looked up. Two lookups remain: the Origin-AA asks which Inter-AAs reach a target FIP (request path); the Customer-AA asks which Connected-AAs reach a target FIU (notification path). A FIP or FIU reaches any AA through any of its own interop-enabled AAs; no lookup.	getHop(target FIP) ⟶ Inter-AAs. getHop(target FIU) ⟶ Connected-AAs. Caller identity from the token. Response cacheable with a TTL. Being defined.

R1. Demand side, request. FIU to Connected-AA

The FIU calls its Connected-AA. Direct: /{Request-X}. Via the hop-AA, when the customer's VUA is at another AA: /iop/{Request-X}. One API is added: /User/discover.



HTTP headers on this hop

`client_api_key`
Existing. The FIU's token. The Connected-AA identifies its directly onboarded FIU.

`x-jws-signature`
Redefined. Detached JWS of the outer body by the immediate sender, the FIU.

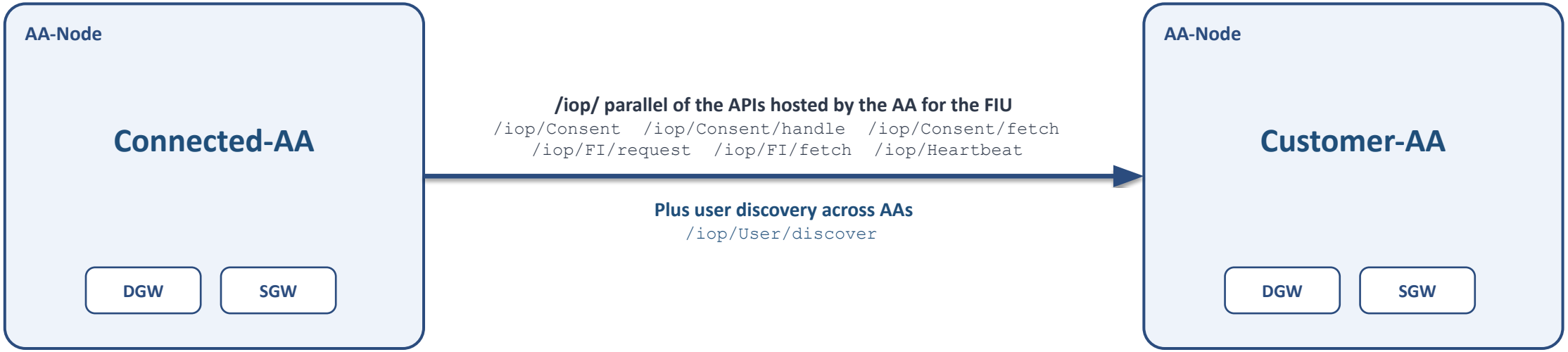
In the JSON body on this hop

`Target {type: AA, id: cust-aa-id}`
New. Names the Customer-AA the request is for. In the clear. Read by the Connected-AA to route. Absent on a direct call.

`iopPayload`
New. The ReBIT body, signed by the originator and encrypted for the far end (JWE). Forwarded unread by the hop-AA. Absent on a direct call.

R2. Demand side, request. Connected-AA to Customer-AA

The Connected-AA forwards the FIU's request to the Customer-AA on the /iop/ parallel APIs. Every AA reaches every AA, so no lookup is needed.



HTTP headers on this hop

`x-conn_aa_api_key`
New. The Connected-AA's role key. The Customer-AA identifies the Connected-AA.

`x-jws-signature`
Redefined. Detached JWS of the outer body by the immediate sender, the Connected-AA.

In the JSON body on this hop

`Initiator {type: FIU, id: fiu-id}`
New. Names the FIU the request came from. In the clear. Replaces Target on the hop out of the hop-AA.

`iopPayload`
New. Forwarded unchanged from the FIU. Decrypted here; the Customer-AA then verifies the FIU's inner signature.

R3. Supply side, request. Origin-AA to Inter-AA

The Origin-AA sends its FIP-bound request to an Inter-AA that is onboarded with the FIP. The Inter-AA comes from CR-AIG: getHop(target FIP).



HTTP headers on this hop

`x-orig_aa_api_key`
New. The Origin-AA's role key. The Inter-AA identifies the Origin-AA.

`x-jws-signature`
Redefined. Detached JWS of the outer body by the immediate sender, the Origin-AA.

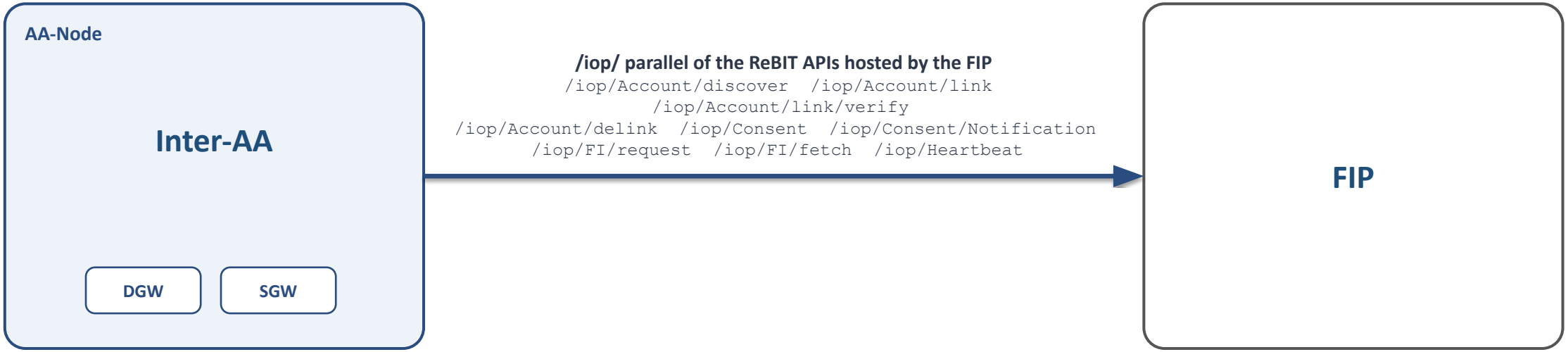
In the JSON body on this hop

`Target {type: FIP, id: fip-id}`
New. Names the FIP the request is for. In the clear. Read by the Inter-AA to route.

`iopPayload`
New. The ReBIT body, signed by the Origin-AA and encrypted for the FIP (JWE). The Inter-AA forwards it and cannot read it.

R4. Supply side, request. Inter-AA to FIP

The Inter-AA calls the FIP on the /iop/ APIs hosted by the FIP. The FIP decrypts, then verifies the Origin-AA's inner signature. The Inter-AA's own traffic stays on /{Request-X}.



HTTP headers on this hop

`x-inter_aa_api_key`
New. The Inter-AA's role key. The FIP identifies the Inter-AA, which it has onboarded.

`x-jws-signature`
Redefined. Detached JWS of the outer body by the immediate sender, the Inter-AA.

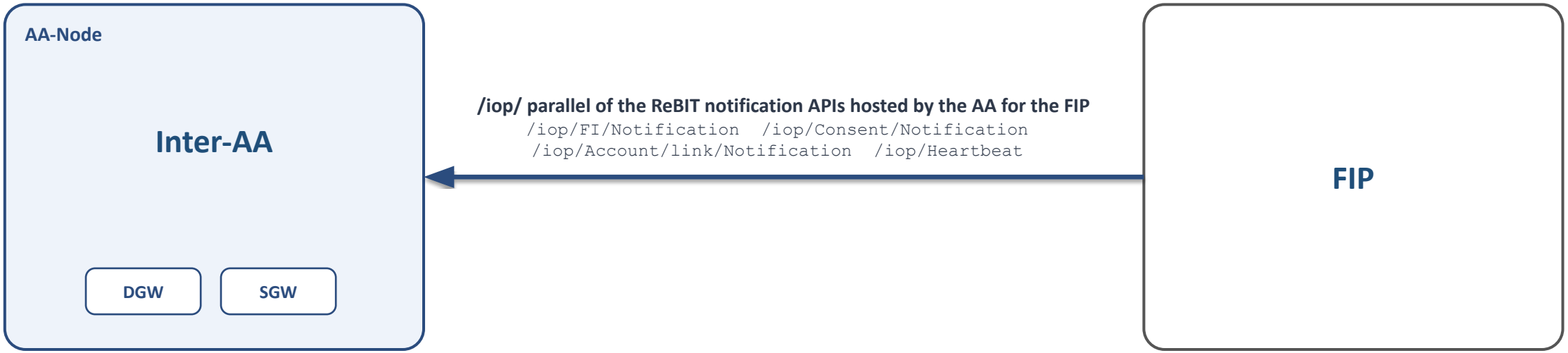
In the JSON body on this hop

`Initiator {type: AA, id: orig-aa-id}`
New. Names the Origin-AA the request came from. In the clear. The FIP uses it to find the Origin-AA's key in the CR.

`iopPayload`
New. Forwarded unchanged from the Origin-AA. Decrypted here by the FIP.

N1. Supply side, notification. FIP to Inter-AA

The FIP notifies through any of its interop-enabled AAs. No lookup: every AA reaches every AA. Direct notifications to its own AAs stay on /{Notification-X}.



HTTP headers on this hop

`fip_api_key`
Existing. The FIP's token. The Inter-AA identifies its directly onboarded FIP.

`x-jws-signature`
Redefined. Detached JWS of the outer body by the immediate sender, the FIP.

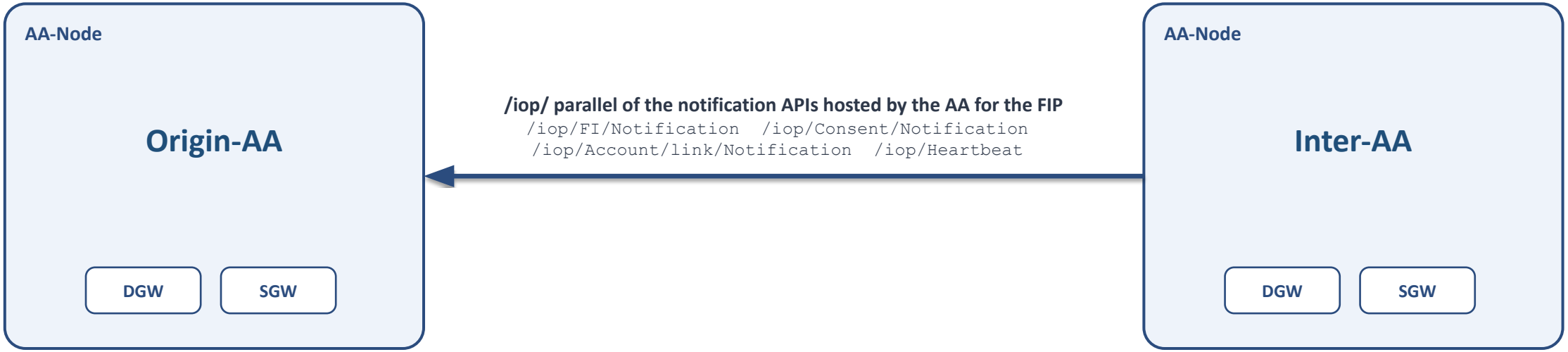
In the JSON body on this hop

`Target {type: AA, id: orig-aa-id}`
New. Names the Origin-AA the notification is for. In the clear. Read by the Inter-AA to route. Absent on a direct call.

`iopPayload`
New. The ReBIT notification body, signed by the FIP and encrypted for the Origin-AA (JWE). Absent on a direct call.

N2. Supply side, notification. Inter-AA to Origin-AA

The Inter-AA forwards the FIP's notification to the Origin-AA on the /iop/ parallel APIs. The Origin-AA decrypts, then verifies the FIP's inner signature.



HTTP headers on this hop

`x-inter_aa_api_key`
New. The Inter-AA's role key. The Origin-AA identifies the Inter-AA.

`x-jws-signature`
Redefined. Detached JWS of the outer body by the immediate sender, the Inter-AA.

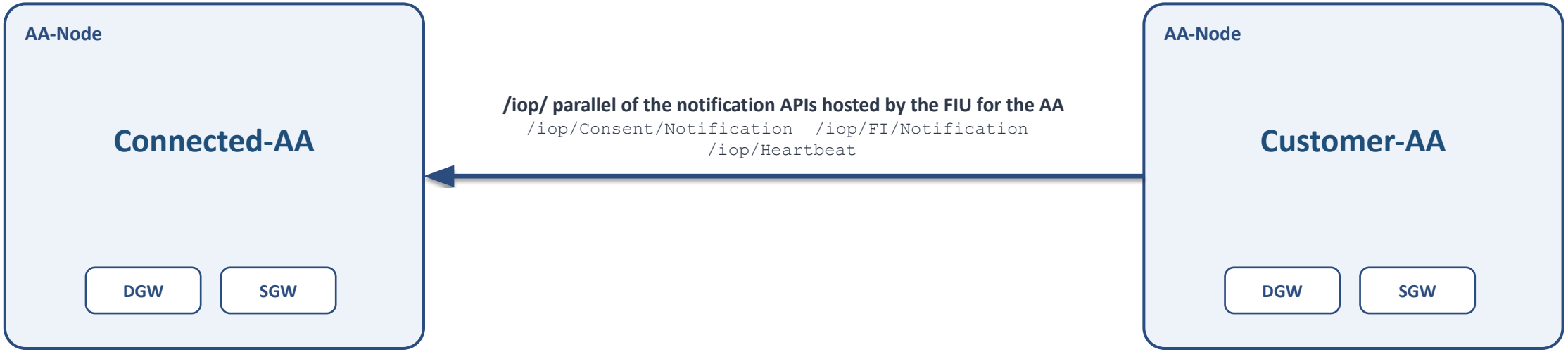
In the JSON body on this hop

`Initiator {type: FIP, id: fip-id}`
New. Names the FIP the notification came from. In the clear. The Origin-AA uses it to find the FIP's key in the CR.

`iopPayload`
New. Forwarded unchanged from the FIP. Decrypted here by the Origin-AA.

N3. Demand side, notification. Customer-AA to Connected-AA

The Customer-AA sends the notification to a Connected-AA that serves the FIU. The Connected-AA comes from CR-AIG: getHop(target FIU).



HTTP headers on this hop

`x-cust_aa_api_key`
New. The Customer-AA's role key. The Connected-AA identifies the Customer-AA.

`x-jws-signature`
Redefined. Detached JWS of the outer body by the immediate sender, the Customer-AA.

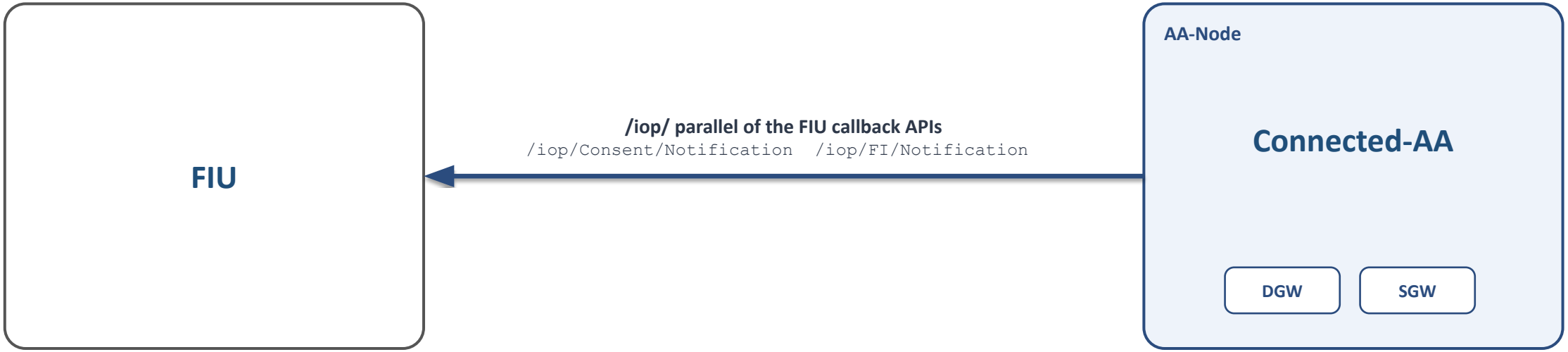
In the JSON body on this hop

`Target {type: FIU, id: fiu-id}`
New. Names the FIU the notification is for. In the clear. Read by the Connected-AA to route.

`iopPayload`
New. The ReBIT notification body, signed by the Customer-AA and encrypted for the FIU (JWE). The Connected-AA forwards it and cannot read it.

N4. Demand side, notification. Connected-AA to FIU

The Connected-AA delivers the notification to its FIU on the /iop/ callback APIs. The FIU decrypts, then verifies the Customer-AA's inner signature. Direct notifications stay on /{Notification-X}.



HTTP headers on this hop

x-conn_aa_api_key
New. The Connected-AA's role key. The FIU identifies its own AA acting as the hop.

x-jws-signature
Redefined. Detached JWS of the outer body by the immediate sender, the Connected-AA.

In the JSON body on this hop

Initiator {type: AA, id: cust-aa-id}
New. Names the Customer-AA the notification came from. In the clear. The FIU uses it to find the Customer-AA's key in the CR.

iopPayload
New. Forwarded unchanged from the Customer-AA. Decrypted here by the FIU.

Hop-by-hop trust and path trust

Hop validation per hop, unchanged. Path encryption and path validation end to end, inside iopPayload.

Hop validation

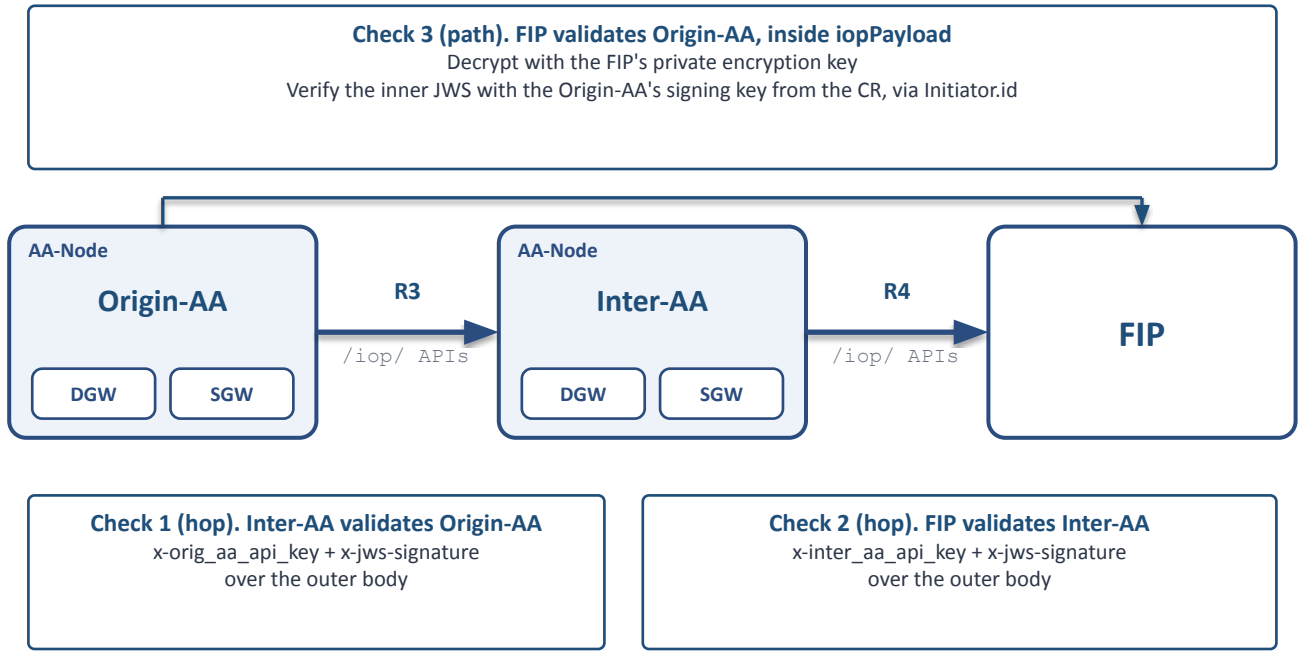
- Every receiver checks the caller's key and the caller's x-jws-signature over the outer body
- Keys: client / fip / aa_api_key on direct hops; the four role keys on /iop/ hops
- Identity is read from the validated token claim, never from a header

Path encryption

- The originator encrypts the signed ReBIT body for the far end: JWE with the far end's public encryption key from the CR
- The hop-AA forwards iopPayload unchanged and cannot read it
- Every entity publishes a second public key, for encryption, in the CR

Path validation

- The far end decrypts, then verifies the originator's inner JWS with the originator's signing key from the CR, found via Initiator.id
- Initiator.id must match the inner signer, else reject
- The originator's token is no longer forwarded; the hop-AA checked it on the previous hop



Supply-side request path shown. Demand side mirrors it: FIU encrypts for the Customer-AA; the Customer-AA decrypts and verifies the FIU's inner signature.

The originator signs once, then encrypts. The hop-AA adds its own signature over the outer body. Nothing is re-signed, modified or read in transit.
Two public keys per entity in the CR: one for signing, one for encryption. Encryption alone proves nothing about the sender; the inner signature does.

Stitching it all together

Twelve stitches on the example topology. Each ladder shows the calls, the responses, and the headers and body on every arrow. New headers and body fields in red.

Demand side

Stitch 1	Vidya. /User/discover, demand side
Stitch 2	Ashwin. /User/discover, demand side
Stitch 3	Priya. /User/discover, demand side
Stitch 4	/Account/discover, supply side
Stitch 5	/Account/link and /Account/link/verify, supply side
Stitch 6	/Consent, supply side

Supply side and the return journey

Stitch 7	/FI/request, demand side
Stitch 8	/FI/request, supply side
Stitch 9	/FI/Notification, supply side
Stitch 10	/FI/Notification, demand side
Stitch 11	/FI/fetch, demand side
Stitch 12	/FI/fetch, supply side

- FIU-4 is connected to AA-4. Priya's VUA is at AA-3.
- AA-3 reaches FIP-7 directly, FIP-1 through AA-1, and FIP-5 and FIP-6 through AA-17.
- Every supply-side request opens with step 0: AA-3 takes the Inter-AAs for each target FIP from CR-AIG, getHop(target FIP), cached.
- A FIP needs no lookup: it notifies through any of its interop-enabled AAs, naming the Origin-AA as Target.
- On every /iop/ hop the sender toward the hop-AA adds Target; the hop-AA replaces it with Initiator. iopPayload is signed by the originator and encrypted for the far end.

Stitch 1. Vidya. /User/discover, demand side

No AA holds a VUA for Vidya. AA-4 broadcasts, every AA answers not found, and the FIU creates the user at AA-4.

In this stitch

- AA-4 checks locally first and finds nothing
- AA-4 broadcasts /iop/User/discover to every AA, adding Initiator {FIU, FIU-4}
- Every AA answers not found
- The FIU onboards Vidya at AA-4
- User discovery is not wrapped in iopPayload (v0.5): identifiers travel in the clear



Stitch 2. Ashwin. /User/discover, demand side

Ashwin's VUA is at AA-4, the FIU's own AA. No broadcast. The FIU continues at AA-4.



AA-4 checks locally and finds Ashwin's handle

No broadcast is sent, so no other AA is involved

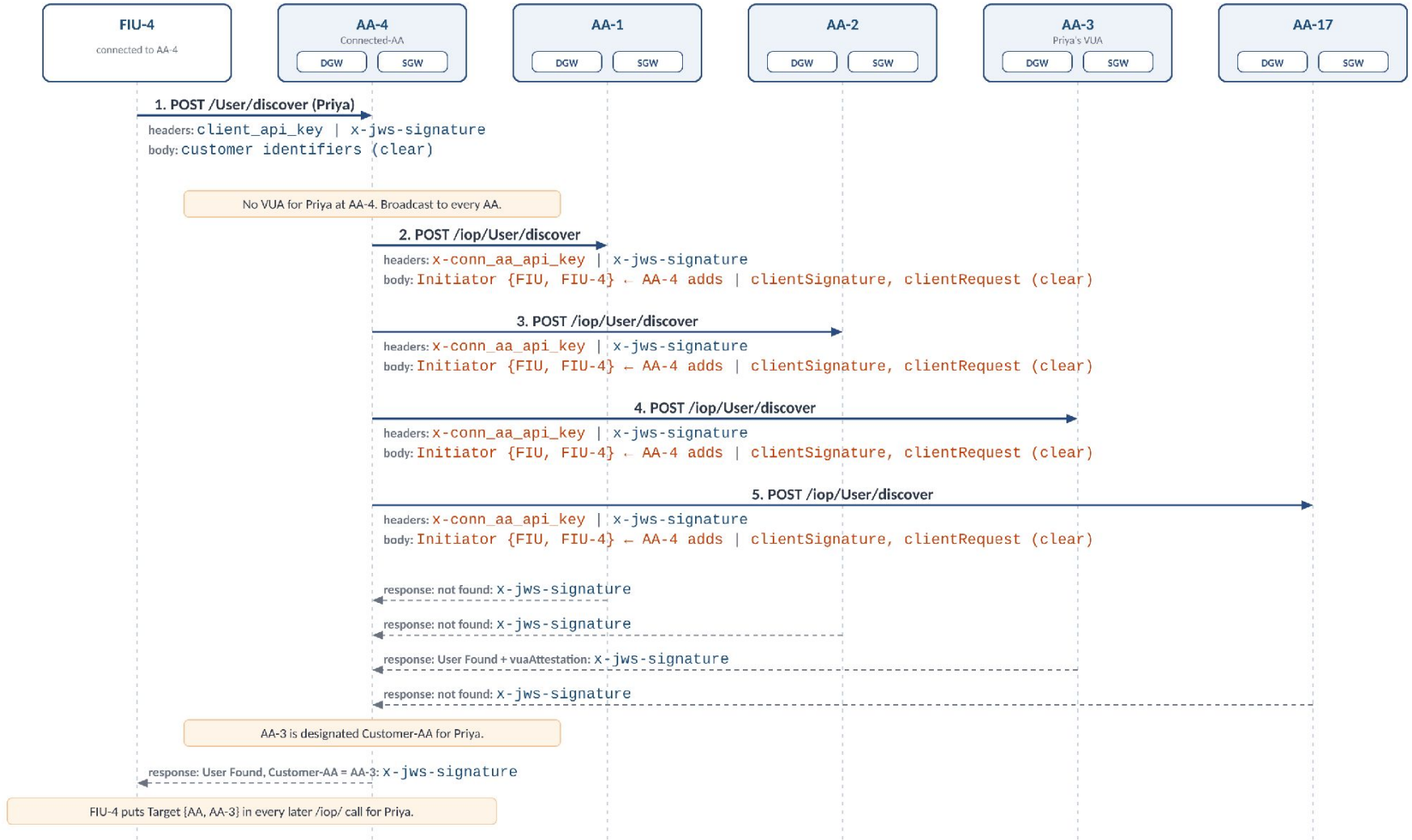
Everything from here works exactly as it does today

Stitch 3. Priya. /User/discover, demand side

Priya's VUA is at AA-3. AA-4 broadcasts, AA-3 answers found with a VUA attestation and becomes the Customer-AA.

In this stitch

- AA-4 checks locally and finds nothing
- AA-4 broadcasts /iop/User/discover
- AA-3 answers found and becomes the Customer-AA
- AA-4 becomes the Connected-AA
- The FIU puts Target {AA, AA-3} in every later /iop/ call for Priya



Stitch 4. /Account/discover, supply side

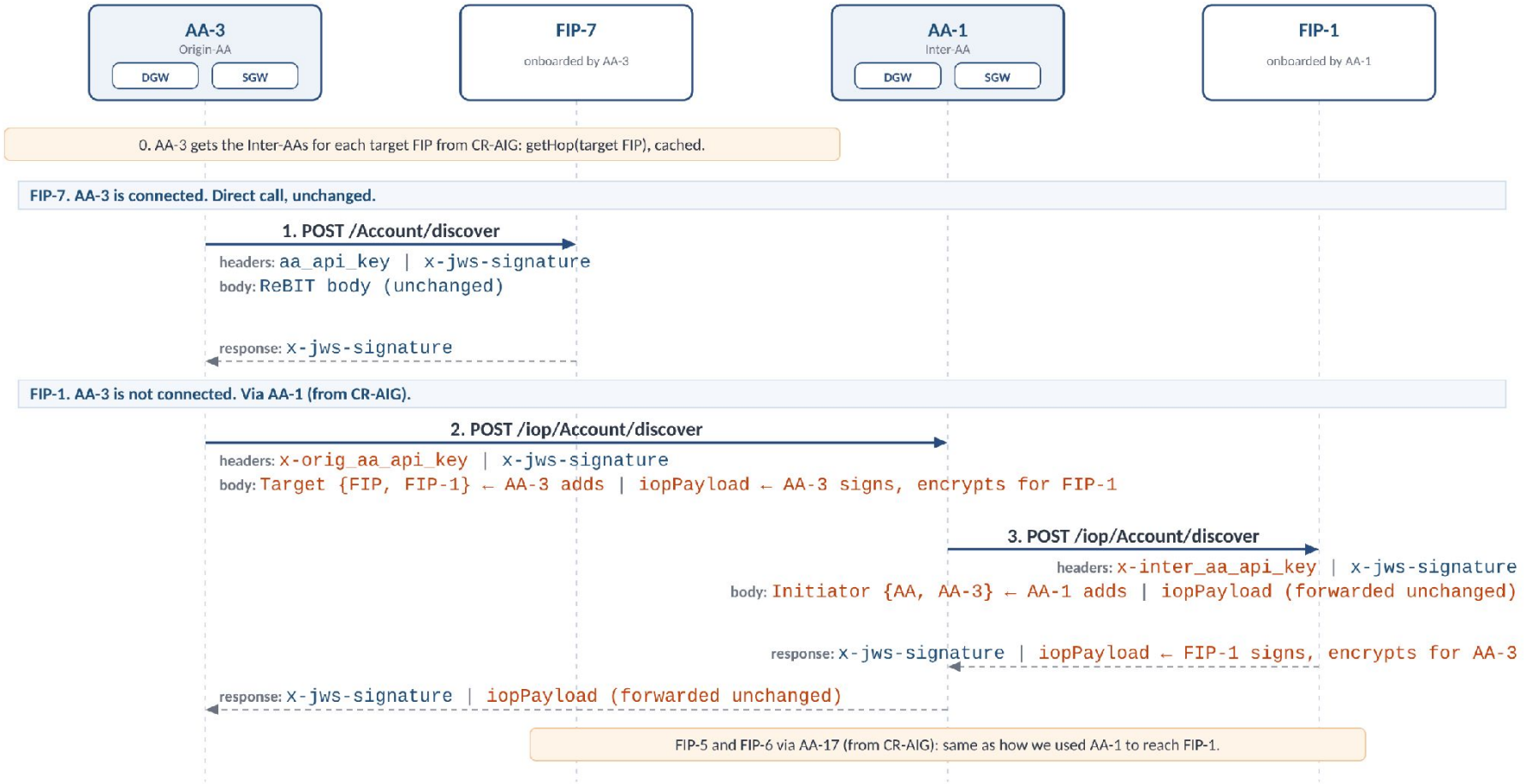
AA-3 reaches FIP-7 directly and the other FIPs through the Inter-AAs CR-AIG gives it.

In this stitch

- Step 0: AA-3 takes the Inter-AAs from CR-AIG, cached
- FIP-7 is called directly, unchanged
- FIP-1 is reached through AA-1 on /iop/Account/discover

Continued

- FIP-5 and FIP-6 are reached through AA-17
- AA-3 adds Target {FIP}; the Inter-AA replaces it with Initiator {AA, AA-3}
- iopPayload is encrypted for the FIP; the Inter-AA cannot read it



Stitch 5. /Account/link and /Account/link/verify, supply side

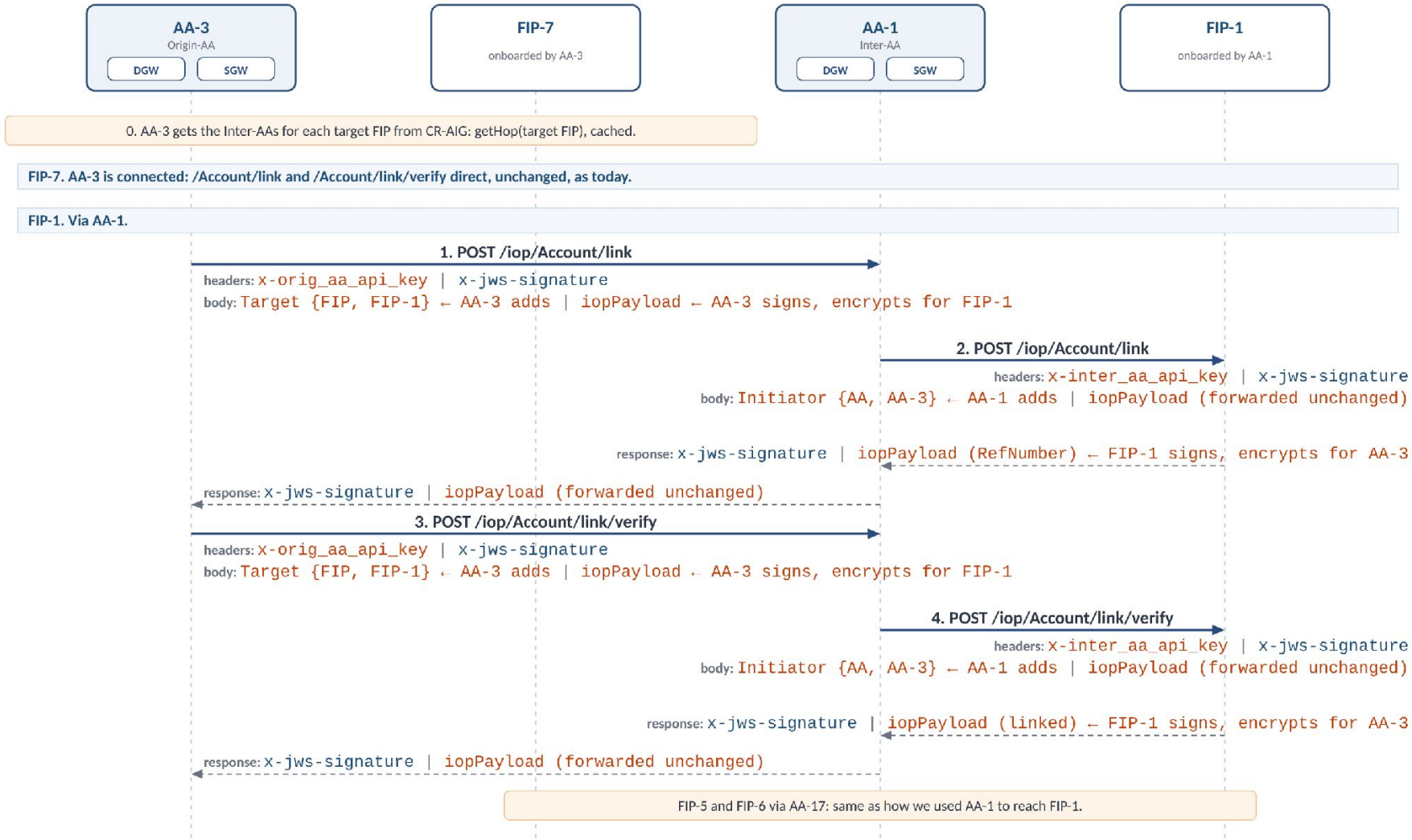
Linking follows the same paths. The OTP between the FIP and Priya is outside the flow.

In this stitch

- Same routing decision as the discovery stitch
- Link and verify travel on the same direct and transit paths
- The OTP goes inside iopPayload; the Inter-AA cannot read it

Continued

- The OTP exchange between the FIP and Priya sits outside this flow
- Link notifications (/iop/Account/link/Notification) follow Stitch 9's pattern



Stitch 6. /Consent, supply side

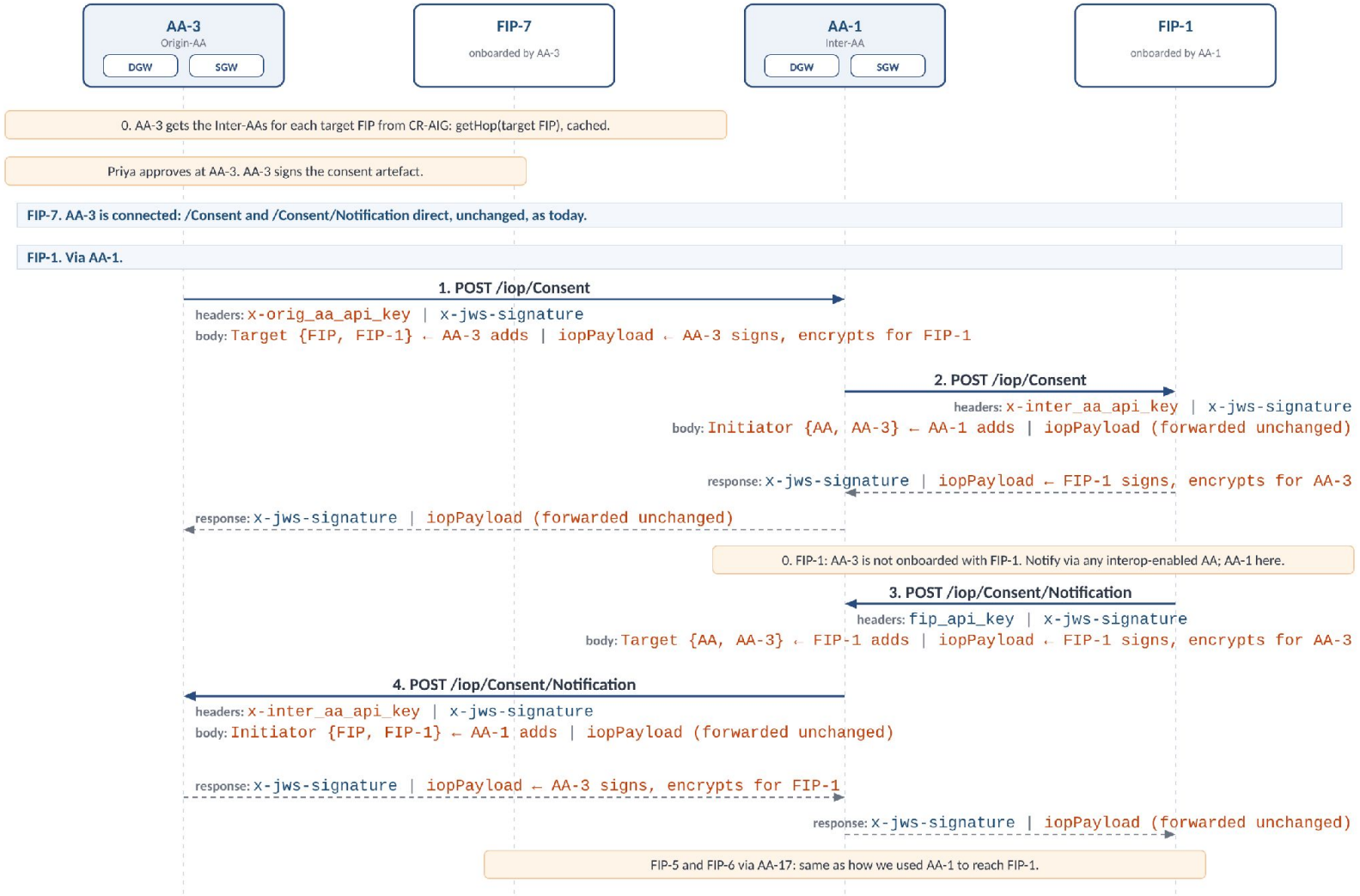
AA-3 delivers the consent artefact to each FIP. Each FIP notifies back, directly or through any of its interop-enabled AAs.

In this stitch

- The consent of record is created and approved at AA-3
- The artefact travels inside iopPayload, encrypted for each FIP
- FIP-7 is served directly, the rest through their Inter-AAs

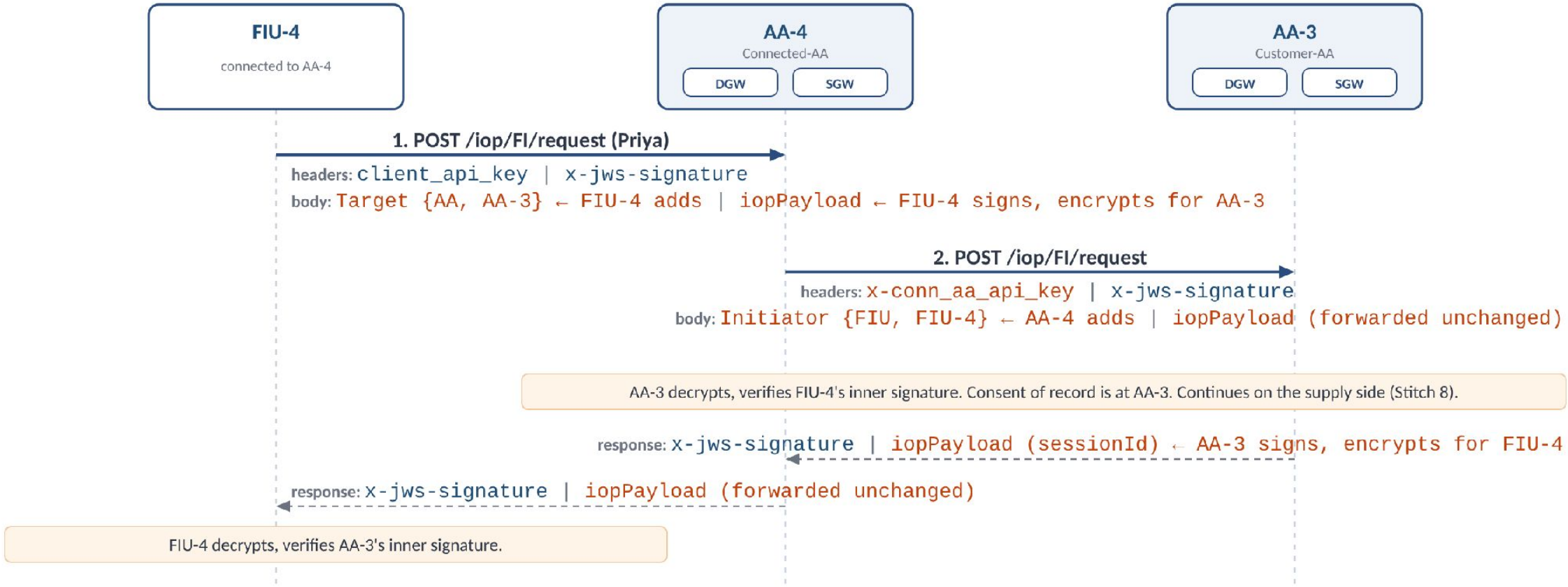
Continued

- Each FIP checks whether AA-3 is onboarded with it
- If not, it notifies via any of its interop-enabled AAs, adding Target {AA, AA-3}
- The Inter-AA replaces Target with Initiator {FIP}



Stitch 7. /FI/request, demand side

The FIU asks its Connected-AA. AA-4 forwards to AA-3, where the consent of record is.



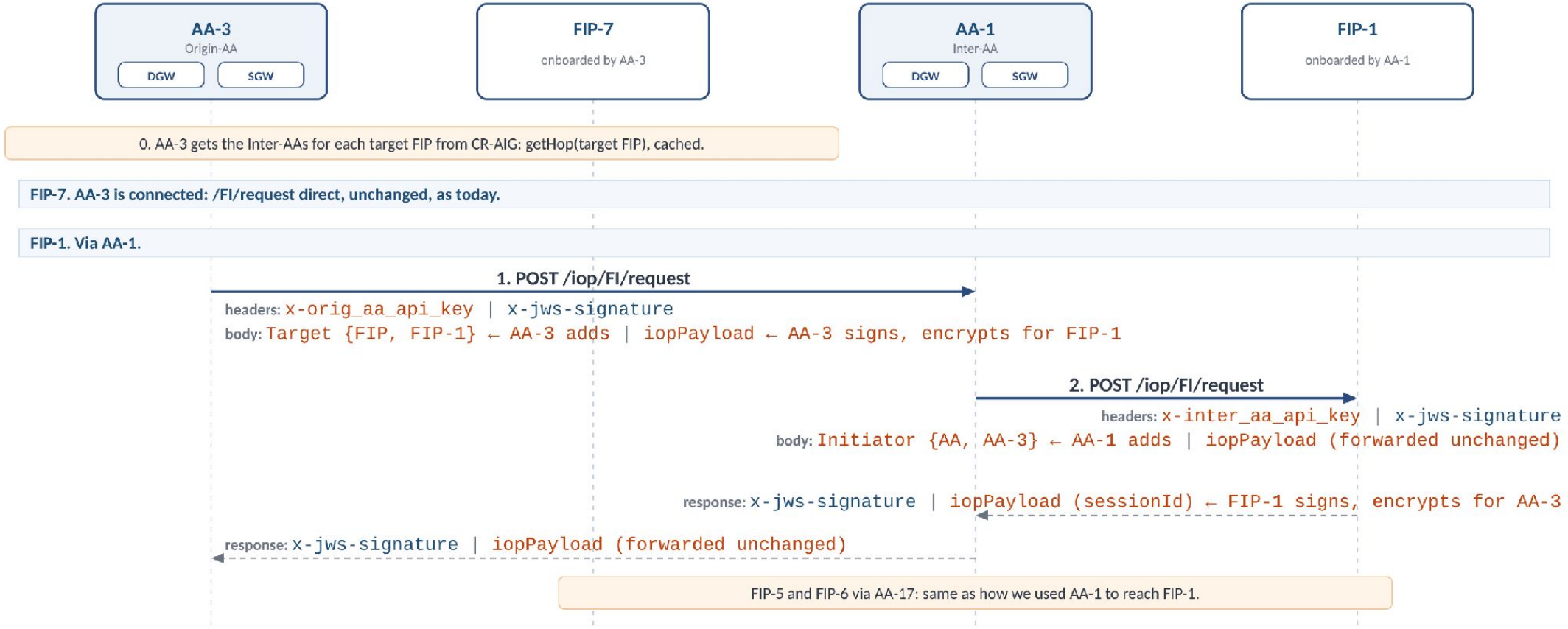
FIU-4 calls /iop/FI/request on AA-4, adding Target {AA, AA-3} and iopPayload encrypted for AA-3

AA-4 forwards on /iop/FI/request with x-conn_aa_api_key, replacing Target with Initiator {FIU, FIU-4}

AA-3 decrypts, verifies FIU-4's inner signature, and continues on the supply side with the consent of record

Stitch 8. /FI/request, supply side

AA-3 places the request with each FIP, directly or through an Inter-AA.



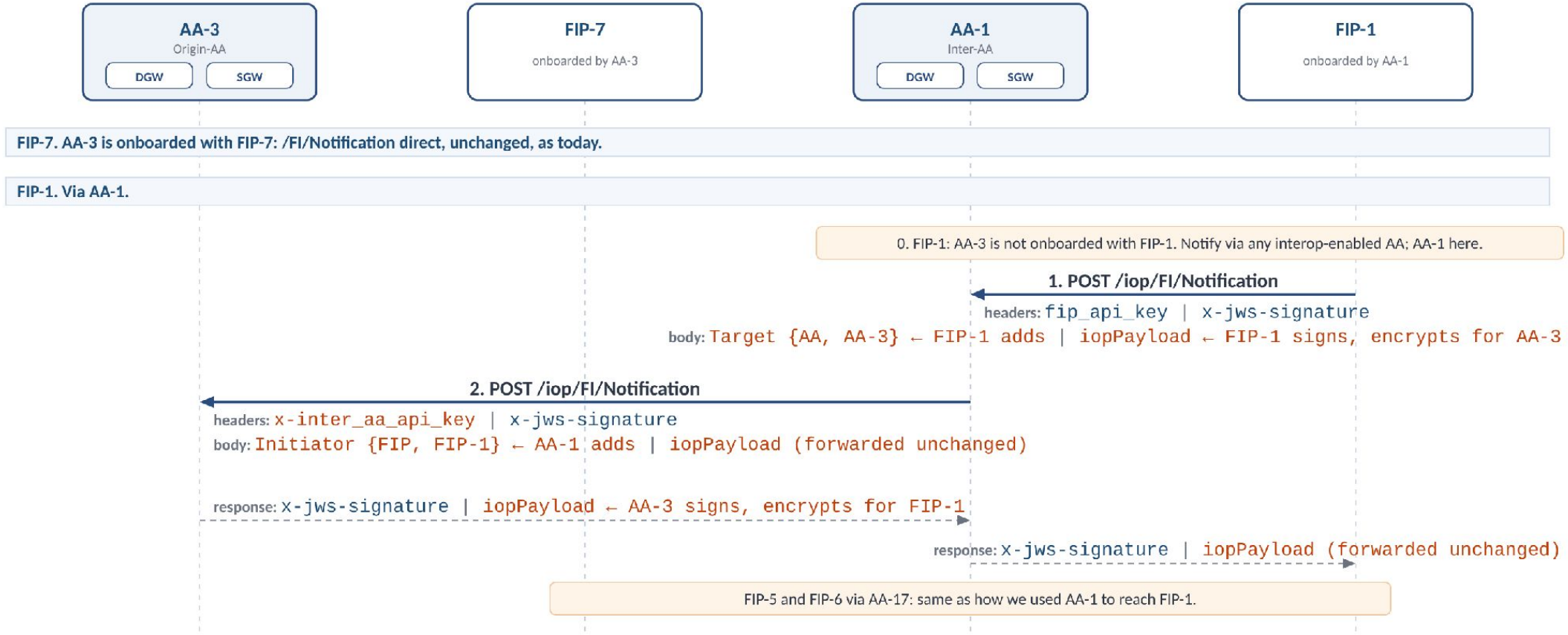
Step 0: the routing decision is made again from CR-AIG, cached

Direct to FIP-7. Through AA-1 to FIP-1. Through AA-17 to FIP-5 and FIP-6

iopPayload is encrypted for each FIP. The FIP decrypts and verifies AA-3's inner signature

Stitch 9. /FI/Notification, supply side

Each FIP tells AA-3 the FI is ready. FIP-1 does so through AA-1, adding Target {AA, AA-3}.



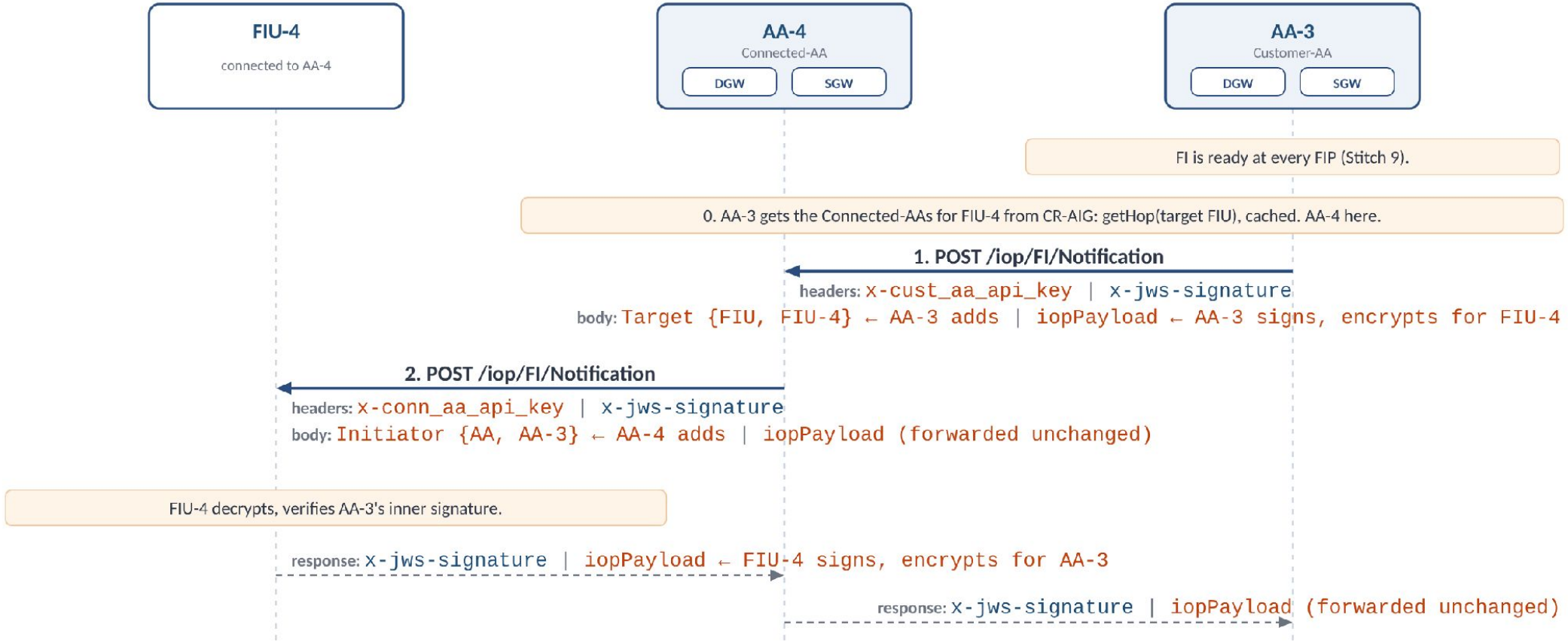
Each FIP checks whether AA-3 is onboarded with it

FIP-7 notifies AA-3 directly. FIP-1 notifies through any of its interop-enabled AAs; AA-1 here

AA-1 forwards on /iop/FI/Notification, replacing Target with Initiator {FIP, FIP-1}

Stitch 10. /FI/Notification, demand side

AA-3 tells AA-4, and AA-4 tells the FIU. The notification body is encrypted for the FIU.



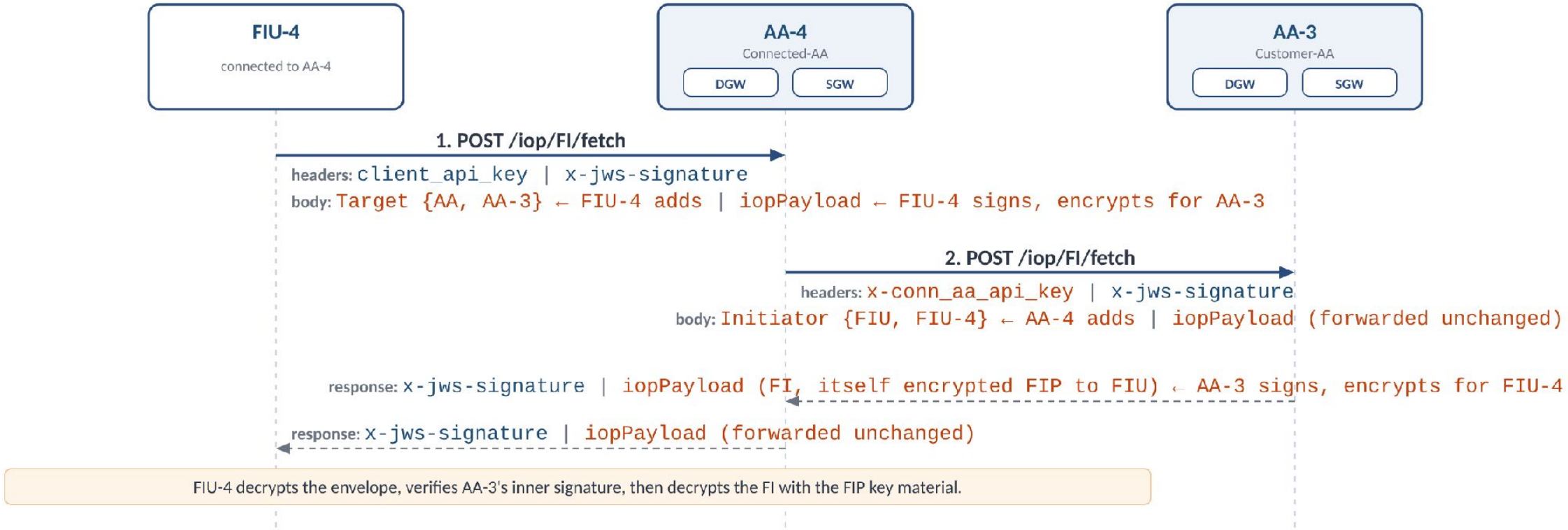
Step 0: AA-3 takes the Connected-AAs for FIU-4 from CR-AIG, getHop(target FIU). It calls /iop/FI/Notification on AA-4 with Target {FIU, FIU-4}

AA-4 calls /iop/FI/Notification on FIU-4 with x-conn_aa_api_key and Initiator {AA, AA-3}

The FIU decrypts and verifies AA-3's inner signature

Stitch 11. /FI/fetch, demand side

The FIU fetches through AA-4. The FI stays encrypted FIP to FIU, inside an envelope encrypted for the FIU.



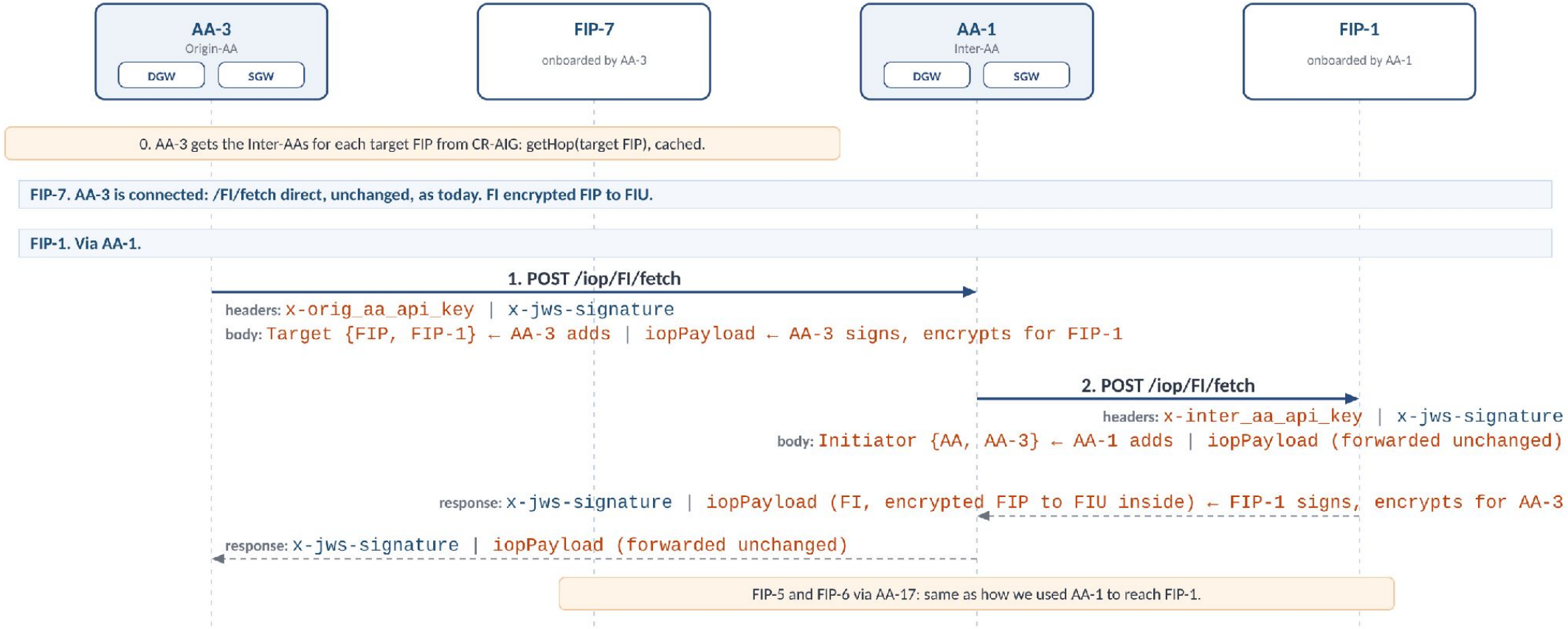
FIU-4 calls /iop/FI/fetch on AA-4 with Target {AA, AA-3}

AA-4 forwards on /iop/FI/fetch with Initiator {FIU, FIU-4}

The FI is encrypted FIP to FIU as today, and the whole response is wrapped in iopPayload for FIU-4. No AA can read it

Stitch 12. /FI/fetch, supply side

AA-3 fetches from each FIP, directly or through an Inter-AA.



The routing decision is rebuilt per message, so path failover is free

Direct to FIP-7, through AA-1 and AA-17 for the rest

The FI ciphertext travels inside iopPayload through the Inter-AA, untouched and unread

Changes to be done by FIUs

One new capability, the /iop/ parallel APIs toward an unconnected Customer-AA, and end-to-end encryption with it. Commercial contracts are unchanged.

#	Change	Purpose	What has to be built
1	**User handle discovery**	Identify existing user handles in the AA ecosystem	Call /User/discover on the Connected-AA. Handle found, not found and multiple handles. Keep the aaId and vuaAttestation of the chosen handle.
2	**Call the /iop/ parallel APIs**	Reach a Customer-AA the FIU is not connected to, through its Connected-AA	When the handle is at another AA, call /iop/{Request-X} instead of /{Request-X}, with client_api_key and x-jws-signature over the outer body. Put Target {AA, cust-aa-id} in the clear part of the body.
3	**Encrypt the request**	Keep the request readable only by the Customer-AA	Sign the ReBIT body (JWS), then encrypt the signed JWS for the Customer-AA with its public encryption key from the CR (JWE, RSA-OAEP-256 / A256GCM). Send it as iopPayload.
4	**Host the /iop/ callbacks**	Receive notifications that come through a Connected-AA	Host /iop/Consent/Notification and /iop/FI/Notification. Validate x-conn_aa_api_key (the Connected-AA's role key) and x-jws-signature. Read Initiator {AA, cust-aa-id}.
5	**Decrypt and verify**	Trust the Customer-AA end to end	Decrypt iopPayload in responses and notifications with the FIU's private encryption key. Verify the Customer-AA's inner signature with its signing key from the CR, via Initiator.id. Reject on mismatch.
6	**Publish an encryption key**	Let Customer-AAs encrypt for the FIU	Generate a second key pair, for encryption. Register the public key in the CR. Rotate per policy.
7	**Carry correlationId**	End-to-end traceability of a flow	Generate a correlationId at the first call of each flow and carry it on every call of that flow. Expect it back in every response and in every notification of that flow. Handle IopUpstreamError and the encrypted iopError.
8	**Multiple handles**	Let the user pick a handle from a list	Accept one or more handles belonging to different AAs. Display them and let the customer choose a preferred handle.
9	**Redirection to an unconnected AA**	Complete the consent journey on an existing handle at an unconnected AA	Implement the consent-journey redirection flow to an unconnected Customer-AA.
10	**Selecting a Connected-AA**	Decide which Connected-AA carries a request to a Customer-AA	Implement selection logic for user discovery and for other calls to a Customer-AA.

Changes to be done by FIPs

The /iop/ parallel APIs for transit traffic, end-to-end encryption with the Origin-AA, and one routing choice. No new FIP state. A non-upgraded FIP keeps working for direct traffic.

#	Change	Purpose	What has to be built
1	**Host the /iop/ parallel APIs**	Receive requests from Origin-AAs the FIP has not onboarded, through an Inter-AA it has	Host /iop/{Request-X} for Account/discover, Account/link, Account/link/verify, Account/delink, Consent, Consent/Notification, FI/request, FI/fetch, Heartbeat. Validate x-inter_aa_api_key (the Inter-AA's role key) and x-jws-signature over the outer body. Read Initiator {AA, orig-aa-id}. Direct traffic stays on /{Request-X}.
2	**Decrypt and verify**	Trust the Origin-AA end to end, without a forwarded key	Decrypt iopPayload with the FIP's private encryption key. Verify the Origin-AA's inner signature with its signing key from the CR, via Initiator.id. Reject on mismatch. Apply the FIP's own opt-in list of Origin-AAs here.
3	**Encrypt what goes back**	Keep responses and notifications readable only by the Origin-AA	Sign the ReBIT body, then encrypt the signed JWS for the Origin-AA with its public encryption key from the CR. Send it as iopPayload in responses and in /iop/{Notification-X} calls.
4	**Call the /iop/ notification APIs**	Notify an Origin-AA the FIP has not onboarded	For a consent that arrived through an Inter-AA, call /iop/FI/Notification, /iop/Consent/Notification or /iop/Account/link/Notification on an Inter-AA with fip_api_key and x-jws-signature. Put Target {AA, orig-aa-id} in the clear part of the body. Direct notifications stay on /{Notification-X}.
5	**Selecting an Inter-AA**	Decide which Inter-AA carries a notification to an Origin-AA	Any of the FIP's interop-enabled AAs will do, since every AA reaches every AA. No CR-AIG lookup. Returning through the Inter-AA that called is the natural default.
6	**Publish an encryption key**	Let Origin-AAs encrypt for the FIP	Generate a second key pair, for encryption. Register the public key in the CR. Rotate per policy.
7	**Carry correlationId**	End-to-end traceability of a flow	Carry the correlationId received in every response and in every notification of that flow, unchanged. Return far-end errors as IopUpstreamError with an encrypted iopError.

Changes to be done by AAs (1 of 2)

Discovery and the /iop/ surface in all four roles. The hop-AA forwards; the far end encrypts and decrypts.

#	Change	Purpose	What has to be built
1	**Implement the user-discover API**	Let FIUs discover existing user handles	Expose /User/discover for the FIU. Return the handle, aaId and vuaAttestation when one exists. Call Sahamati's attestation service at VUA creation.
2	**Discover handles at other AAs**	Find handles held elsewhere	On a /User/discover with no local handle, send /iop/User/discover to the other AAs with x-conn_aa_api_key and Initiator {FIU}. Consolidate and return the response to the FIU. Not wrapped in iopPayload (v0.5).
3	**Implement /iop/User/discover**	Answer AAs querying on behalf of FIUs	Validate x-conn_aa_api_key. Return the handle with its vuaAttestation if one exists; not found otherwise.
4	**Host and call /iop/* as a Connected-AA**	Demand-side connectivity between an onboarded FIU and an unconnected Customer-AA	Host /iop/{Request-X} for the FIU; call it on the Customer-AA with x-conn_aa_api_key. Host /iop/{Notification-X} for the Customer-AA; call it on the FIU with x-conn_aa_api_key. Forward iopPayload unchanged; never read it.
5	**Host and call /iop/* as a Customer-AA**	Demand-side connectivity with an unconnected FIU through a Connected-AA	Host /iop/{Request-X} for Connected-AAs. Decrypt iopPayload; verify the FIU's inner signature via Initiator.id. Sign and encrypt responses and /iop/{Notification-X} bodies for the FIU; call the Connected-AA with x-cust_aa_api_key.
6	**Host and call /iop/* as an Origin-AA**	Supply-side connectivity with an unconnected FIP through an Inter-AA	Sign and encrypt request bodies for the FIP; call /iop/{Request-X} on the Inter-AA with x-orig_aa_api_key. Host /iop/{Notification-X} for Inter-AAs; decrypt and verify the FIP's inner signature via Initiator.id.
7	**Host and call /iop/* as an Inter-AA**	Supply-side connectivity between an FIP and an unconnected Origin-AA	Host /iop/{Request-X} for Origin-AAs; call it on the FIP with x-inter_aa_api_key. Host /iop/{Notification-X} for FIPs; call it on the Origin-AA with x-inter_aa_api_key. Forward iopPayload unchanged; never read it.

Changes to be done by AAs (2 of 2)

Routing in the body, encryption keys, CR-AIG lookups and traceability.

#	Change	Purpose	What has to be built
8	**Read Target, write Initiator**	Route on the clear part of the body and tell the far end who sent it	As a hop-AA (Connected-AA or Inter-AA): read Target {type, id} on the way in, remove it, and add Initiator {type, id} naming the sender on the way out. Exactly one of the two on every /iop/ call; reject otherwise.
9	**Sign every hop**	Hop validation at every receiver	Add x-jws-signature over the outer body on every /iop/ call and response the AA sends, in any role. Validate the caller's key and x-jws-signature on every /iop/ call received.
10	**Encrypt and decrypt as the far end**	Path encryption and path validation	As a Customer-AA or Origin-AA: sign the ReBIT body, then encrypt for the far end with its public encryption key from the CR (iopPayload). Decrypt incoming iopPayload with the AA's private key; verify the originator's inner signature via Initiator.id.
11	**Publish an encryption key**	Let FIUs, FIPs and AAs encrypt for this AA	Generate a second key pair, for encryption. Register the public key in the CR. Rotate per policy.
12	**Use the role keys**	Identify the AA in the role it is playing	Obtain x-conn_aa_api_key, x-cust_aa_api_key, x-orig_aa_api_key and x-inter_aa_api_key from the Token Service; send the one that matches the role on each /iop/ hop. Validate the peer's role key on receipt.
13	**Look up CR-AIG**	Make routing decisions toward unconnected FIPs and FIUs	As an Origin-AA: getHop(target FIP) for the Inter-AAs that reach the FIP. As a Customer-AA: getHop(target FIU) for the Connected-AAs that reach the FIU. Cache with the TTL. No lookup for AA-to-AA: every AA reaches every AA.
14	**Carry correlationId**	End-to-end traceability of a flow	Carry the correlationId on every call of a flow, in every role. Return it in every response and in every notification of that flow, unchanged. Forward far-end errors as IopUpstreamError with the encrypted iopError.

Changes to be done by Sahamati

Registry, tokens, encryption keys, VUA attestation and the interoperability graph. Nothing sits in the hot path.

#	Change	Purpose	What has to be built
1	**CR-AIG: getHop API**	Let Origin-AAs and Customer-AAs find a hop-AA toward the FIPs and FIUs they need	getHop(target FIP) returns the Inter-AAs onboarded with that FIP; getHop(target FIU) returns the Connected-AAs of that FIU. Caller identity from the token. Response carries interop-enabled status, asOf and a TTL; cacheable. Errors: UnknownTarget, TargetNotInteropEnabled, NoHopAvailable.
2	**CR-AIG: build and maintain the graph**	Keep the AA-to-FIP and AA-to-FIU edges current	Seed from SNA link data; refresh at a configured interval; accept onboarding and tear-down reports from REs. AA-to-AA edges are not needed: every AA reaches every AA.
3	**CR: encryption public keys**	Let every participant encrypt iopPayload for the far end	Accept and publish a second public key per participant, for encryption, with a key id and rotation. Keep the signing key separate. Lookup by entityId, as for signing keys.
4	**CR: interop-enabled flag and base URLs**	Let an AA know which peers and participants can take /iop/ traffic	Expose interop-enabled status per participant. Expose base URL and keys for AA-to-AA lookup, so a Connected-AA and Customer-AA, or an Origin-AA and Inter-AA, can reach each other.
5	**Token Service: the four role keys**	Identify an AA in the role it is playing on each /iop/ hop	Issue x-conn_aa_api_key, x-cust_aa_api_key, x-orig_aa_api_key and x-inter_aa_api_key as role-scoped tokens that cannot be used on the direct ReBIT APIs. Claims: entityId and role; short lifetime.
6	**VUA attestation service**	Make a found handle verifiable by the Connected-AA	Stateless POST /vua/attest at VUA creation: sign the hashed (AA, mobile) and return a JWS. Publish the service key in the CR. One-time backfill of existing VUAs. Nothing stored at Sahamati.
7	**Conformance and telemetry**	Certify the external interfaces only	Test FIU-AA, AA-AA and AA-FIP on the /iop/ APIs: hop validation, Target/Initiator rule, encryption and decryption, correlationId returned in every response and notification. Collect per-flow telemetry keyed on correlationId via SNA.

Interconnectivity and telemetry

The AIG answers questions about who is connected to whom. It does not carry traffic.

AA Interconnectivity Graph (AIG)

- The Sahamati-maintained view of who is connected to whom, and with what capabilities
- Reporting of connectivity
- Reporting of readiness
- Lookup of connectivity, exercised by the Origin-AA's AA-Core on every routing decision
- Direct or transit, and which Inter-AA

What the AIG is not

- Not a central switch. It answers questions, it does not carry traffic
- Not in the hot path of discovery, linking, consent or data fetch
- AA-Cores cache the graph and refresh at a configured interval
- FIUs do not call the AIG. The demand side assumes AA to AA connectivity
- FIPs do not need it. An FIP returns through the Inter-AA that called it. The supply side assumes AA to AA connectivity.

Telemetry support for AA interoperability

- To be defined. Interop-enabled participants report operational telemetry through the SNA. Governance uses it for intervention.
- Discovery-misuse telemetry: gateway functions report /User/discover volumes and patterns per FIU
- Abuse that survives the technical controls (possession OTP, HMAC, rate caps) surfaces here for Governance intervention
- Telemetry reported by each gateway function, DGW and SGW, still to be added and decided upon

End-to-end traceability. Three identifiers

Every journey and every user must be traceable for support and telemetry. The v0.5 spec names the journey identifier correlationId.

Transaction ID

- Identifies one API request and its response.
- Exists today.
- Not sufficient to trace journeys

Session ID

- Identifies one leg of a data pull.
- One for FIU to AA. One for each leg between AA and FIP.
- Exists today.
- Not sufficient to trace journeys

correlationId (new)

- One ID per user per flow, generated at the first call.
- Carried on every call of the flow by every participant.
- Returned in every response and every notification of that flow, unchanged.
- In the clear part of the /iop/ envelope: the hop-AA can log it without reading the payload.

Why the two existing identifiers are not enough

- A journey spans requests, responses and notifications across up to four hops.
- A journey spans several sessions and several transactions.
- Neither ID ties them to one user and one journey. The session ID now sits inside the encrypted payload, invisible to a hop-AA.

Flows the correlationId covers (per the v0.5 spec)

- User discovery
- Account discovery
- Account linking
- Consent creation and approval
- Data pull (all its APIs)

Support for Design
and
Implementation of Interoperability

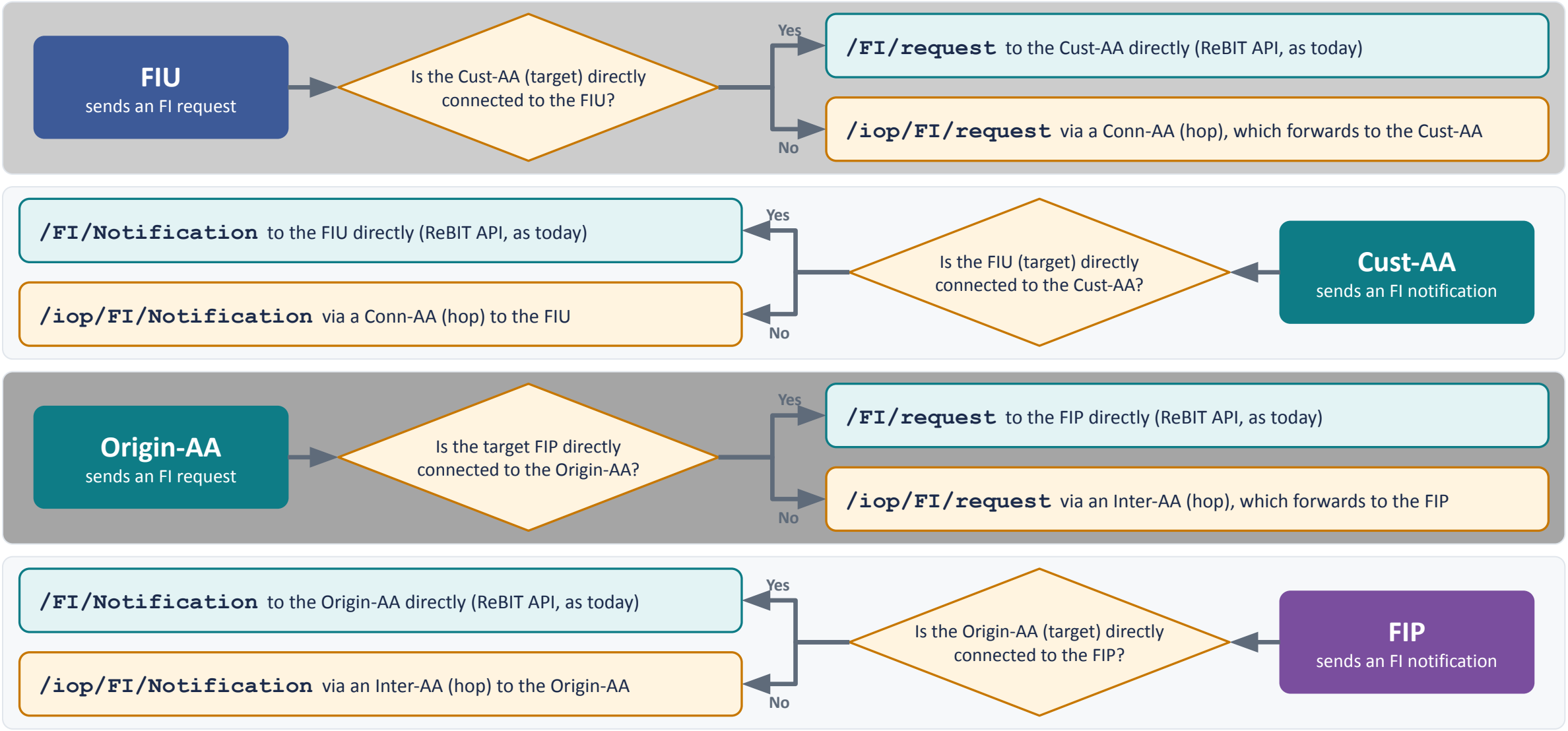
Current Flow: FIU to AA

FIU sends the FI request directly to its own Account Aggregator



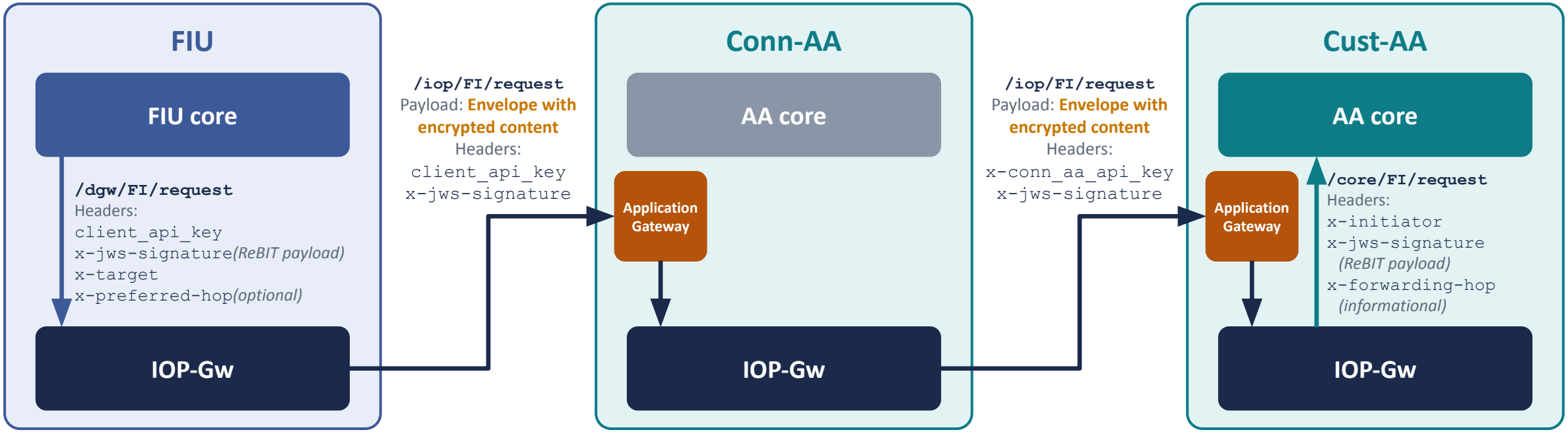
Routing Decisions at the Originator

Each originator (demand and supply side) decides whether to call the ReBIT API directly or its /iop/ parallel via a hop AA



Interoperable Flow: FIU → Conn-AA → Cust-AA

IOP-Gws carry the FIU request across AAs while preserving the original signature



FIU IOP-Gw

- Select a Conn-AA
- Generate IopPayload field: iopPayload = Encrypt (signature + payload)
- Add envelope payload with target
- Generate x-jws-signature for the new Iop request payload JSON

Conn-AA IOP-Gw

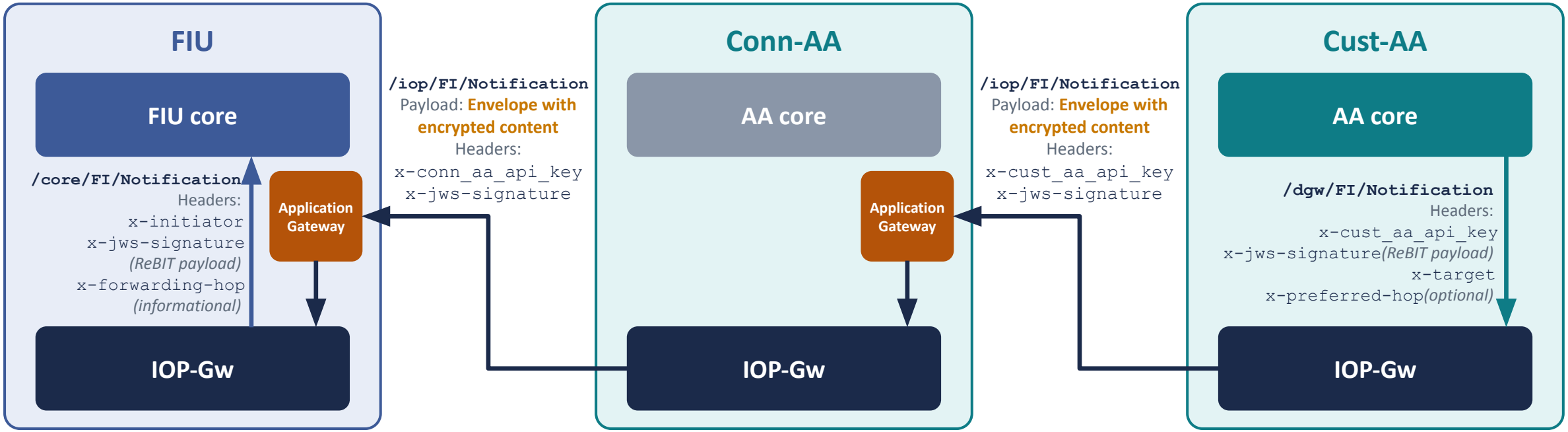
- Add envelope payload with initiator
- Generate x-jws-signature for the forwarded Iop request payload JSON
- Read the target and forward to the recipient

Cust-AA IOP-Gw

- Identify immediate sender from x-conn_aa_api_key
- Verify x-jws-signature
- Decrypt the iopPayload, extract the ReBIT payload and the x-jws-signature for that payload
- Extract the initiator and pass it to the core

Notification Flow: Cust-AA → Conn-AA → FIU

IOP-Gws carry the Cust-AA notification back to the FIU through the Conn-AA



FIU IOP-Gw

- Identify immediate sender from x-conn_aa_api_key
- Verify x-jws-signature
- Decrypt the iopPayload, extract the ReBIT payload and the x-jws-signature for that payload
- Extract the initiator and pass it to the core

Conn-AA IOP-Gw

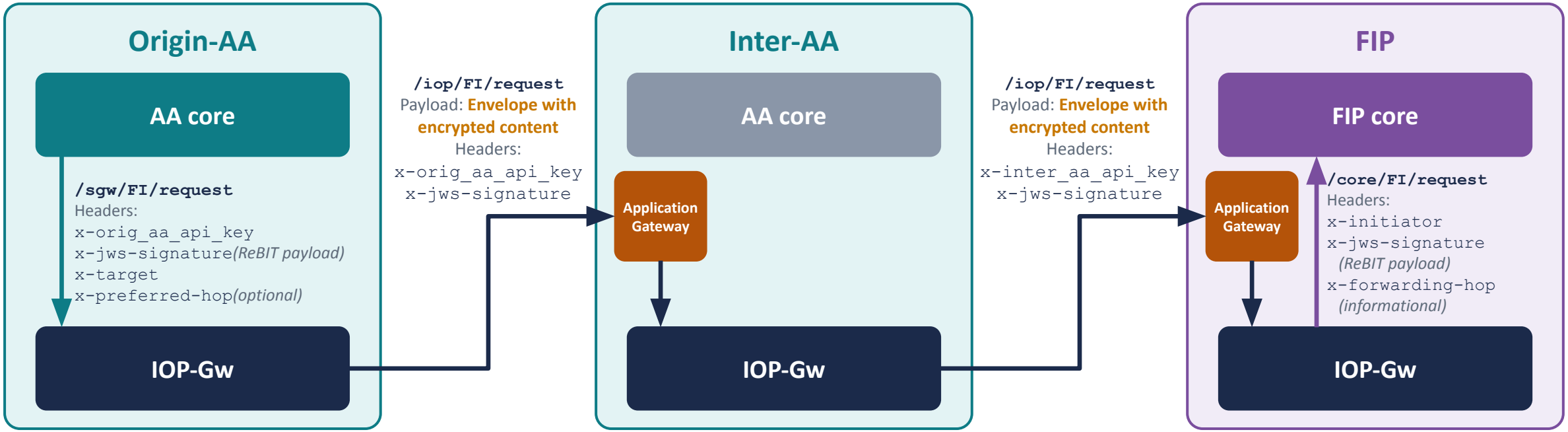
- Add envelope payload with initiator
- Generate x-jws-signature for the forwarded Iop request payload JSON
- Read the target and forward to the recipient

Cust-AA IOP-Gw

- Lookup AIG and identify hop
- Generate IopPayload field: iopPayload = Encrypt (signature + payload)
- Add envelope payload with target
- Generate x-jws-signature for the new Iop request payload JSON

Interoperable Flow: Origin-AA → Inter-AA → FIP

IOP-Gws carry the AA request across AAs to the FIP while preserving the original signature



Origin-AA IOP-Gw

- Lookup AIG and identify hop
- Generate IopPayload field: iopPayload = Encrypt (signature + payload)
- Add envelope payload with target
- Generate x-jws-signature for the new Iop request payload JSON

Inter-AA IOP-Gw

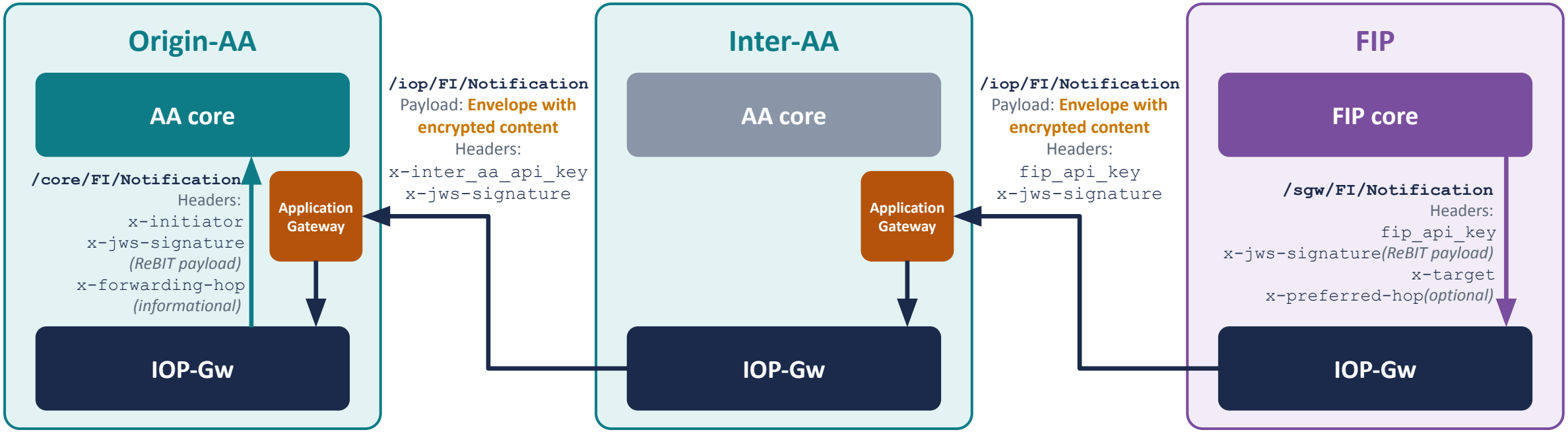
- Add envelope payload with initiator
- Generate x-jws-signature for the forwarded Iop request payload JSON
- Read the target and forward to the recipient

FIP IOP-Gw

- Identify immediate sender from x-inter_aa_api_key
- Verify x-jws-signature
- Decrypt the iopPayload, extract the ReBIT payload and the x-jws-signature for that payload
- Extract the initiator and pass it to the core

Notification Flow: FIP → Inter-AA → Origin-AA

IOP-Gws carry the FIP notification back to the AA that originated the request



Origin-AA IOP-Gw

- Identify immediate sender from x-inter_aa_api_key
- Verify x-jws-signature
- Decrypt the iopPayload, extract the ReBIT payload and the x-jws-signature for that payload
- Extract the initiator and pass it to the core

Inter-AA IOP-Gw

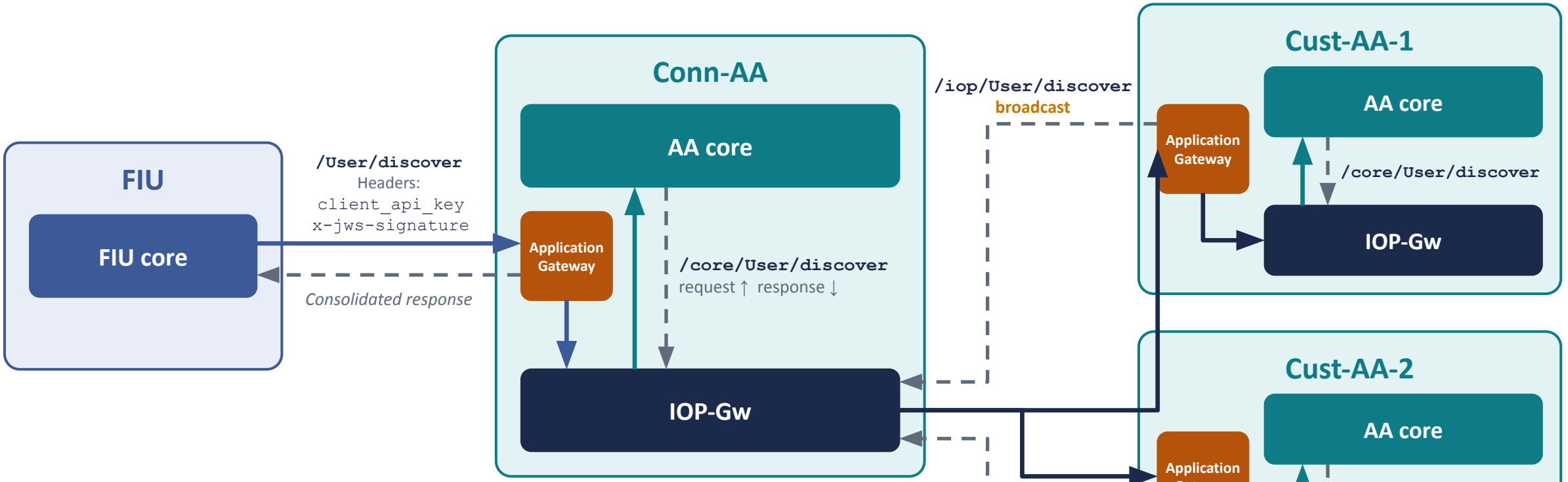
- Add envelope payload with initiator
- Generate x-jws-signature for the forwarded Iop request payload JSON
- Read the target and forward to the recipient

FIP IOP-Gw

- Select an Inter-AA
- Generate IopPayload field: iopPayload = Encrypt (signature + payload)
- Add envelope payload with target
- Generate x-jws-signature for the new Iop request payload JSON

User Discovery: /User/discover

Conn-AA checks its own core first, then broadcasts to other AAs and consolidates the replies



Conn-AA IOP-Gw

- Verify the client (FIU) x-jws-signature
- Look up the user in own AA core first
- User not found → broadcast to Cust-AAs (client's payload + client's signature + envelope payload)
- Consolidate the responses and reply to the FIU

Federation, not a central switch

Two gateway functions inside every AA. Eight hops. One set of headers.

The FIU

- Same AA, same contracts, same callbacks
- Learns one new capability: handle discovery
- Sends one target header, reads two forwarded ones

The FIP

- Unchanged for direct traffic
- On transit traffic, validates the Origin-AA's key and signature
- Names the Origin-AA on the way back. No new FIP state

The AA

- Two stateless gateway roles, DGW and SGW
- A /gw/* mirror of the existing API families
- Discovery, the routing decision and the forwarding relays

- Change lands where there are seventeen parties, not a thousand.
- Existing endpoints, paths, bodies and signatures are kept. The design is additive and backwards-compatible.
- The origin signs once and the signature travels unmodified. The transit never gains authority.

Reference: AA Interoperability Functional Description and API Spec (file01). Supply-side and demand-side stitching (file02, file03).