ALIN LI

OPEN COMPUTING • IOT • ARCHITECTURE • TRAINING

WHITE PAPER

Sovereign Data Exchange

Designing an inter-organisation API exchange on the X-Road model

For
Government digital leaders, CIOs and CTOs in regulated sectors (finance, health, utilities), and enterprise and integration architects

Published
October 2026

Version
1.0

ABSTRACT

In brief

> Organisations that must share data with one another usually do it one bilateral interface at a time. The result is expensive to build, hard to secure and almost impossible to audit as a whole. This paper sets out an alternative based on X-Road, the open-source exchange layer developed by the Nordic Institute for Interoperability Solutions (NIIS), founded by Estonia and Finland. Data moves directly from member to member through security servers at each organisation's edge, every message is signed and time-stamped, and the operator governs trust without holding member data. We cover architecture, trust and identifiers, API management on WSO2, first services and a phased roadmap that leaves the operator in full ownership.

EXECUTIVE SUMMARY

What leaders need to know

- Point-to-point integration between organisations grows roughly with the square of the number of parties, and every connection carries its own contract, security review and failure mode.
- The X-Road model replaces that web with a shared exchange in which each organisation is a registered member, runs its own security server at its edge, and exchanges signed, time-stamped messages directly with other members.
- The central operator governs the member registry, trust services and global configuration, but sits outside the data path: it holds routing and evidence records, and zero copies of member data.
- Stable four-part service identifiers, a single catalogue and per-service consent rules let providers publish once and let consumers subscribe once, with access enforced at the provider's own edge.
- API management on open-source WSO2 complements the exchange by handling developer portals, application keys, rate limits, analytics and mutual TLS for each member's own services.
- The safest path is phased: settle naming and governance first, prove the model in a sandbox, onboard a small group of pilot members with a handful of high-value services such as identity checks and registry lookups, then open production and scale.
- Because the stack is open source and the delivered platform is handed over to the operating organisation, the exchange can sit entirely within national or organisational boundaries, with no licence that can be withdrawn and no vendor between members and their data.

01

Why point-to-point integration does not scale between organisations

Inside a single organisation, integration is a technical problem. Between organisations, it becomes a governance problem as well. When a bank needs to verify a customer's identity with an authority, check a company registration and validate a tax number, it typically negotiates three separate interfaces, each with its own legal agreement, network route, credentials, message format and support arrangement. Every other bank, insurer and enterprise does the same, independently, for the same services.

The arithmetic is unforgiving. If every one of n organisations may need to exchange data with every other, the number of possible bilateral connections approaches n multiplied by n minus one, divided by two. Illustrative: ten organisations can require up to 45 separate connections; fifty organisations can require up to 1,225. Not every pair talks, but the shape of the curve explains why integration budgets keep growing while the integration estate becomes harder to describe, let alone secure.

Each bilateral connection also creates its own trust arrangement. One provider accepts IP allow-lists, another uses shared secrets, a third issues certificates from its own authority. Logs live in different places, in different formats, for different retention periods. When an auditor or a data subject asks who accessed a record and why, the honest answer is often that nobody can say with certainty across the whole estate.

The common response is a central integration hub: everyone connects to one platform that transforms, routes and often caches messages. That reduces the number of connections, but it moves the problem rather than solving it. The hub becomes a single point of failure, a single point of compromise and a single place where, in principle, someone could read everything. Its operator gradually makes decisions that belong to the data owners, such as who may call which service and how long records are kept. Responsibility drifts to the middle, where it is least accountable.

- Cost: the same integration is built many times over by different consumers of the same provider.
- Security: trust is negotiated pair by pair, with inconsistent controls and credentials.
- Evidence: there is no common, signed record of who asked for what, when and under which permission.
- Change: a provider that alters its interface must coordinate with every consumer separately.
- Sovereignty: a central hub concentrates data and control in one operator, often one vendor.

> **The goal is not fewer connections at any price; it is a shared way of connecting that keeps data with its owners and leaves a provable trail.**

02

The X-Road model in plain language

X-Road is an open-source data exchange layer that has underpinned Estonian digital public services, where it is known as X-tee, for many years. It is developed and maintained by the Nordic Institute for Interoperability Solutions (NIIS), a non-profit organisation founded by Estonia and Finland to develop the software jointly. The model has since been adopted in various forms by public and private ecosystems in several countries. This paper uses it as a design pattern rather than a product endorsement: the ideas matter more than any single implementation.

The core idea is simple. Instead of routing data through a central platform, every participating organisation becomes a member of the exchange and runs its own security server at its edge. When one member's system needs data from another, the request travels from the consumer's security server directly to the provider's security server. On the way it is authenticated with certificates, signed, encrypted in transit and time-stamped. Both security servers log the exchange, so both parties hold the same signed evidence.

There is still a central function, but its role is deliberately narrow. It maintains the register of members and their subsystems, distributes the global configuration, and defines which trust services are recognised: the certificate authority, certificate status checking and time-stamping. It decides who belongs to the ecosystem. It does not carry or see the payloads. That separation, between governing trust and carrying data, is the heart of the model.

For leaders, the practical consequence is that sovereignty becomes concrete. Each provider decides which consumers may call each of its services, and that decision is enforced at the provider's own security server, not at a hub run by someone else. Each member keeps its own logs. Because there is no third party on the data path, no third party can quietly read, copy or alter what passes between two members.

The model also supports federation. Separate exchange instances, for example in two countries or two sectors, can agree to recognise each other's trust configuration so that a member of one can consume a service in the other under a bilateral agreement, without either side rebuilding its platform. Estonia and Finland have publicly demonstrated this between their national exchanges. For an ecosystem that may later need to connect across a border or a regional grouping, designing for federation from the start costs little and preserves the option.

Characteristic	Bilateral interfaces	Central integration hub	X-Road member model
Where data travels	Directly, over many bespoke routes	Through the hub, often stored or cached	Directly, member to member, through security servers
Who enforces access	Each provider, inconsistently	The hub operator	Each provider, at its own edge, with common rules
Trust mechanism	Negotiated per connection	Hub credentials	Shared trust services: certificates and time-stamps
Evidence	Scattered logs, varied formats	Hub logs, held by the operator	Signed, time-stamped logs at both ends
Number of connections per member	One per counterpart	One, to the hub	One, to the exchange
Single point of compromise	No, but many weak points	Yes, the hub	No data concentration in the middle

03

Members, not a hub: the architecture

An exchange on this model has a small number of building blocks, and it pays to name them plainly before any procurement or build begins. The operator runs central services: the member registry, the global configuration and the trust services, or contracts with recognised providers for them. Each member runs one or more security servers in its own environment, under its own control. Behind each security server sit the member's information systems, organised into subsystems, which either consume services, provide them, or both.

Members are organised into member classes, which group participants by legal nature. A simple and widely used pattern is to distinguish government entities (GOV) from commercial entities (COM), with further classes added only if governance genuinely requires them. Within a member, subsystems represent distinct systems or business units. Access is granted to a specific subsystem rather than to a whole organisation, which keeps permissions narrow and auditable.

A typical exchange of data follows the sequence shown in Figure 1. The consumer's system makes an ordinary API call to its own security server. The security server identifies the target service by its stable identifier, looks up the provider's security server in the global configuration, establishes a mutually authenticated connection, signs the message and obtains a time-stamp. The provider's security server verifies the signature and the consumer's certificate, checks that this consumer

subsystem is permitted to call this service, logs the request and passes it to the provider's system. The answer returns along the same path, signed and logged at both ends.

Two architectural choices deserve emphasis. First, security servers should be placed at the true edge of each member's network, with the member's own firewall and monitoring around them, so that the member remains in control of what enters and leaves. Second, the security server should be the only route by which exchange traffic reaches internal systems. If side doors exist, the evidence trail is incomplete and the trust model is weakened.

Many providers do not offer clean REST or SOAP interfaces. Behind the security server, an adapter layer can translate to whatever the provider actually runs, whether that is a message queue, a file drop or a legacy service, without exposing that complexity to consumers. Where consumers need to know about changes rather than ask repeatedly, an event pattern can push signed notifications to subscribers who remain permitted, removing much of the polling that otherwise loads provider systems.

> **The operator's dashboard can honestly show zero copies of member data, because routing and evidence records are all it ever holds.**

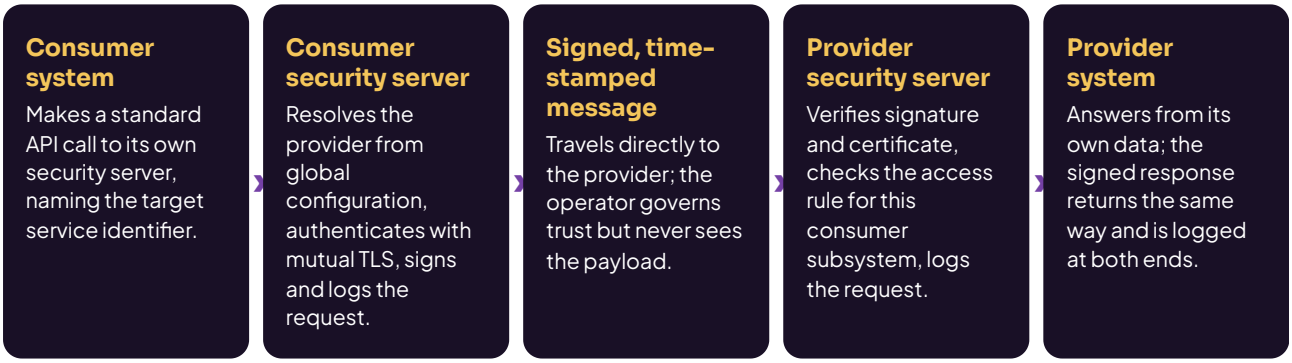


Figure 1. Figure 1. One request across the exchange: data moves member to member, and nothing pools in the middle.

04

Trust services, certificates and identifiers

Trust in the exchange rests on three services that the operator either runs or recognises. A certificate authority issues authentication certificates to security servers and signing certificates to members. A certificate status service lets any security server check, at the time of an exchange, that a certificate has not been revoked. A time-stamping service provides independent proof of when a message was signed. Together they let any member verify any other without a prior bilateral arrangement.

The choice of trust service providers is a sovereignty decision as much as a technical one. Where national or sector rules recognise qualified trust service providers, the exchange should align with them, so that signed evidence carries weight beyond the exchange itself. Where the ecosystem is private, the operator may run its own certificate authority, but it should then publish clear policies on issuance, revocation, key protection and audit. In both cases, private keys for security servers should be generated and held under each member's control, ideally in hardware security modules for higher-risk services.

Sandbox and production environments should be separated by trust, not only by network. A sandbox can admit developers with an ordinary portal account and test certificates, so that teams can build and experiment quickly. Production should require a client certificate issued to the member's security server under the production trust configuration, after the member has passed an onboarding and certification process. This makes it structurally impossible for a sandbox credential to reach production data.

Every service on the exchange is addressed by a stable identifier rather than a host name or network address. The identifier is hierarchical, as shown in Table 1, and it stays the same for the life of the service. That stability is what lets a consumer depend on a service, what lets a provider move infrastructure without notifying every subscriber, and what lets an auditor trace an exchange years later. Naming conventions for members, subsystems and services should therefore be settled before the first connection, because they become very hard to change once consumers depend on them.

Cryptography ages. The certificate and signing profiles chosen today should be documented, configurable and reviewed periodically, with a migration path for stronger algorithms, including post-quantum options as standards and supported implementations mature. An exchange designed to be crypto-agile can rotate algorithms through configuration and certificate renewal rather than through a rebuild.

> **A full identifier such as EXAMPLE-EX/GOV/10012345/identity-checks/verifyPerson.v1 (Illustrative) is the contract between consumer and provider; host names are an implementation detail.**

Part	Meaning	Illustrative example	Who assigns it
INSTANCE	The exchange ecosystem, for example a national or sector instance	EXAMPLE-EX	Fixed when the exchange is established
CLASS	The member class, grouping members by legal nature	GOV or COM	Defined in governance rules
MEMBER	The registered organisation, usually by an official registration code	10012345	Operator, on approval of membership
SUBSYSTEM	A specific system or unit within the member	identity-checks	Member, within agreed naming conventions
SERVICE	The individual service, optionally with a version	verifyPerson.v1	Provider, on publication

05

Catalogue, consent and evidence

An exchange is only useful if services can be found. The catalogue is the shared directory of everything published: each service with its identifier, owner, description, sector, consent rule, environment status and a machine-readable interface description, preferably in an open standard such as OpenAPI. Consumers should be able to search by name, sector, provider or endpoint, read the documentation, test in the sandbox and request a subscription without a meeting. Providers publish once and reach every eligible subscriber.

Publication itself should be an owner-controlled act. A service enters the catalogue because the providing member has requested it, with its own approval recorded, not because the operator has switched it on. The operator's role is to check that the request is complete and conforms to the rules, then to publish it. The same principle applies to subscriptions: the operator may facilitate, but the provider decides who may call its service, and that decision is enforced at the provider's security server.

Every service should carry an explicit consent rule describing the legal basis on which it may be called. Some services are open to any authenticated member for a defined purpose; some require a contractual relationship; some require the documented consent of the data subject; some are restricted to a named list of consumers. Writing the rule into the catalogue entry makes it visible to consumers before they build, and makes it testable by auditors afterwards. Where consent is the basis, revocation should take effect on the next call, so that a withdrawn permission fails closed rather than lingering in caches.

Evidence is the quiet strength of the model. Because every message is signed and time-stamped, and logged at both ends, either party can later prove exactly what was asked and answered, and neither can credibly deny it. These logs belong to the members. The operator keeps routing and operational records, such as which service was called, by which subsystem, when, and with what outcome, but not the payloads.

That evidence can also be turned outward. Estonia's public services include a data tracker that lets individuals see which institutions have accessed their data. An exchange that records the purpose and the requesting subsystem for each call can support a similar view, where governance and law allow it: a person or a business can see who looked, why and when. Few features do more for public trust in a data-sharing programme.

Data protection and data residency deserve early, specialist attention. Rules on lawful basis, purpose limitation, retention, cross-border transfer and the rights of data subjects differ between jurisdictions and sectors, and this paper does not offer legal advice. The architecture helps by keeping data at source, minimising what crosses the exchange, and producing evidence by default; it does not replace a proper legal assessment for each service.

- Catalogue entry: identifier, owner, purpose, consent rule, interface description, environments, support contact, version history.
- Consent rule types: open to authenticated members, contract-based, data-subject consent, named consumers only.
- Evidence at members: signed request and response records, time-stamps, retained under each member's policy.
- Evidence at the operator: routing and operational metadata, with payload values masked or absent.

06

API management alongside the exchange

An exchange on the X-Road model governs trust between organisations. It is not, by itself, a complete API management platform. Members still need to design, version, document, secure and monitor their own services, and consumers still need a developer experience that makes building against those services straightforward. That is the role of API management, and in Alin Li's designs it is delivered on the open-source WSO2 stack.

The two layers complement each other. The security server handles member-to-member trust, signing, time-stamping and evidence. The API gateway, sitting behind a provider's security server or in front of a consumer's internal systems, handles application-level concerns: routing to the right back end, enforcing quotas and rate limits, transforming payloads, applying versioning policy and collecting analytics. The same API management platform can also serve system-to-system integration inside a single organisation, where an exchange is unnecessary.

Keys and credentials should be scoped tightly. Application keys are issued per application and per environment, so that a sandbox key can never call production, and a revoked key stops working on the next call. Mutual TLS between gateways and back ends closes the internal path. Each consuming organisation should see only its own applications, subscriptions, calls, latency and service-level performance, and nothing belonging to anyone else.

Support access needs the same care. Operator and provider support teams must be able to diagnose incidents without reading personal or commercial data. Masked support views, which show the shape, timing and outcome of a call while hiding payload values, let teams find the fault without widening access to the data.

Before a member receives production credentials, it should pass a certification suite: a defined set of functional, security and conformance tests run in the sandbox. The same suite should be used on the provider side, so that both ends are tested against one standard and onboarding becomes a documented, repeatable process rather than a series of favours.

> **The exchange decides who may talk to whom; API management makes each conversation easy to build, safe to run and simple to measure.**

WSO2 role	What it does on the exchange	Who uses it
API gateway	Routes, throttles and secures calls behind each member's security server; applies policies and transformations	Provider and consumer integration teams
Developer portal	Publishes documentation, interface descriptions and sandbox access; manages applications and subscriptions	Developers at consuming members
Key and identity management	Issues and revokes application keys and tokens per app and per environment; integrates with enterprise identity	Member security and platform teams
Rate limits and quotas	Protects provider systems and enforces agreed consumption tiers	Providers and the operator
Analytics	Reports calls, latency, errors and service levels per tenant, with masking	Members, operator service management
Mutual TLS and policy enforcement	Secures internal hops and applies security policies consistently	Platform and security teams
Integration and adapters	Connects to legacy protocols such as SOAP, message queues and files behind the security server	Provider integration teams

07

Typical first services and how to sequence them

The first services on an exchange shape its reputation. They should be valuable to many consumers, technically straightforward for providers, and carry a clear legal basis. Read-only verification services usually meet all three tests. They answer a narrow question, such as whether a record exists or whether two attributes match, rather than returning a full dataset, which also keeps data minimisation natural.

A sensible sequence starts with verification, moves to lookups that return limited attributes, and only then introduces transactional services and events. Each stage builds confidence in the trust model, the onboarding process and the evidence trail before the stakes rise. Open banking and cross-sector system-to-system integration typically follow once the exchange has a track record, because they involve more parties, more regulation and higher transaction volumes.

Cost conversations are easier with a simple shared model. Integration money is already being spent across the ecosystem, bilaterally, by every consumer separately. The question is whether it buys one shared fabric or the same connection many times over. A transparent model, with the assumptions set by the ecosystem's own participants and every output labelled Illustrative until it has been validated, helps decision-makers compare the two without relying on a vendor's figures.

- Start with verification services that return the minimum data needed to answer the question.
- Choose providers whose systems are stable and whose owners are committed to publishing.
- Prefer services with a clear, documented legal basis and a simple consent rule.
- Add events only once request-and-response services are proven and monitored.
- Label every volume, saving or cost estimate Illustrative until measured in pilot.

Service	Typical provider	Typical consumers	Why it is a good early candidate
Identity and KYC checks	Identity authority or trusted identity provider	Banks, insurers, telecoms, public services	High demand, narrow yes or no answers, strong regulatory driver
Commercial registration lookups	Company registry	Banks, procurement, licensing, enterprises	Public or semi-public data, clear identifiers, many consumers
Tax number validation	Tax authority	Banks, suppliers, procurement, invoicing platforms	Simple validation, reduces fraud and manual checks
Email and mobile verification	Telecom operators or verification providers	Onboarding journeys across sectors	Fast to implement, immediate user-facing benefit
Open banking	Banks and payment institutions	Licensed third parties, fintechs, enterprises	High value once trust, consent and certification are proven
System-to-system integration	Any member's internal systems	Other units of the same organisation or partners	Reuses the same gateway, keys and evidence patterns internally

08

Implementation roadmap

An exchange is a programme, not a single release. The roadmap below separates decisions that are expensive to reverse, such as naming, trust and governance, from those that can evolve, such as which services come next. Each phase has explicit exit criteria, so that progress is measured by outcomes rather than by elapsed time. Durations depend heavily on the number of members, the readiness of providers and the legal groundwork; any timeline quoted for a specific programme should be treated as Illustrative until it has been planned with the participants.

Throughout, Alin Li works on the principle that the client owns what is delivered. Source code, configuration, infrastructure definitions, runbooks and documentation are handed over to the operating organisation, which runs the platform. Alin Li can continue to support it in production, but there is no proprietary component and no licence that would prevent the operator from changing provider, bringing operations fully in-house or extending the platform itself.

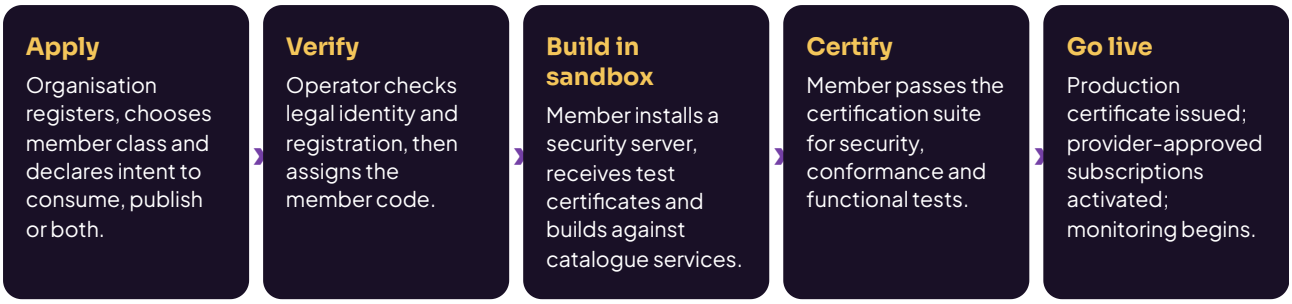


Figure 2. Figure 2. Onboarding a new member, the repeatable path from application to production.

Phase	Scope	Outcomes	Exit criteria
1. Design	Governance model, member classes, naming conventions, trust service choices, security and data-protection assessment, target architecture	Agreed rulebook, identifier scheme, architecture and operating model	Rulebook and naming conventions approved by the governing body; trust providers selected
2. Sandbox	Central services and trust configuration for test, reference security servers, WSO2 gateway and developer portal, catalogue, certification suite	Working end-to-end exchange with test certificates and sample services	A test consumer calls a test provider by identifier, with signed evidence at both ends and certification suite passing
3. Pilot members	A small group of committed providers and consumers; two to four verification services; onboarding process run for real	First real services in a controlled environment; documented onboarding	Pilot members certified; services meet agreed service levels; evidence reviewed by audit or risk functions
4. Production	Production trust configuration and client certificates, monitoring and support model, incident and change processes	Live exchange with production members and published services	Operational acceptance, disaster-recovery test passed, support and escalation running, handover documents accepted
5. Scale	Further members and services, lookups and transactional services, events, federation where needed	Growing catalogue and membership under stable governance	Onboarding repeatable without project intervention; operator running the platform independently

09

Governance, operating model and risks

Technology rarely causes an exchange to fail. Governance does. The operating model must make clear who decides membership, who approves publication, who sets consent rules, who funds the central services, who resolves disputes and who is accountable when something goes wrong. A short, published rulebook, agreed by a governing body that represents both providers and consumers, is worth more than any technical feature.

The operator's powers should be designed as carefully as its duties. A well-designed operator console administers membership, publication queues, quotas and audit logs, and every action it takes is recorded. Equally important is what it cannot do: read a member's payload, export another tenant's data, or approve a service on an owner's behalf. Making those limits technical rather than merely contractual is what allows members to trust an operator they do not control.

Operationally, the central services and each member's security servers need the same discipline as any critical system: high availability across zones, tested disaster recovery, monitoring with masked metrics, security operations, signed software releases and policy managed as code. Where data residency matters, all of this can run on infrastructure within the relevant national or organisational boundary, with no dependency on a public cloud, because every component is open source and can be self-hosted.

Funding should reflect the shape of the costs. Central services are built and run once and shared by the whole ecosystem; member connections scale with membership. Models that combine a shared contribution to the central function with transparent per-member or per-tier charges are common, but the right answer depends on the ecosystem and should be decided in governance, not by the platform supplier.

> **Make the operator's limits technical, not just contractual: members trust what the system cannot do.**

Risk	What it looks like	Mitigation
Weak governance	Unclear authority over membership, publication or disputes; decisions drift to the operator	Published rulebook, representative governing body, owner-signed publication
Naming churn	Identifiers change after consumers depend on them	Settle conventions in the design phase; version services rather than rename them
Side doors	Systems exchange data outside the security servers, leaving gaps in evidence	Policy that exchange traffic uses security servers only; network controls and periodic review
Provider readiness	Providers lack stable interfaces or capacity	Adapter layer, rate limits, start with simple verification services
Over-collection	Services return more data than consumers need	Design verification-first, data minimisation reviews, consent rules in the catalogue
Key compromise	Certificate or private key exposed	Hardware key protection, revocation and status checking, rotation runbooks
Operator overreach	Central team gains sight of payloads	Technical separation of routing from data, masked support views, audited operator actions
Vendor lock-in	Platform cannot be changed or run without one supplier	Open-source stack, full handover of code, configuration and runbooks to the operator
Cryptographic ageing	Algorithms or key sizes become inadequate	Documented crypto profiles, configurable algorithms, planned migration path

CHECKLIST

Readiness checklist for an inter-organisation exchange

- [ ] A governing body that represents providers and consumers has been named and has authority to approve the rulebook.
- [ ] Member classes, member codes, subsystem and service naming conventions are agreed and documented.
- [ ] Trust service providers for certificates, status checking and time-stamping are selected, with published policies.
- [ ] Sandbox and production are separated by trust configuration, and production requires a client certificate per security server.
- [ ] Every catalogue entry carries an owner, a purpose, a consent rule and a machine-readable interface description.
- [ ] Publication and subscription approvals are made by the providing member and recorded, not switched on by the operator.
- [ ] Security servers are the only route for exchange traffic into member systems, and this is verified periodically.
- [ ] Signed, time-stamped logs are retained at both ends under each member's documented retention policy.
- [ ] Support and monitoring views mask payload values by default.
- [ ] A certification suite runs before any member receives production credentials.
- [ ] Data-protection, lawful-basis and data-residency assessments have been completed for each initial service with appropriate specialists.
- [ ] Source code, configuration, infrastructure definitions and runbooks are held by the operating organisation, which can run the platform without the original supplier.

CONCLUSION

Where to start

Organisations will keep sharing data; the question is whether they do it through a growing tangle of bilateral interfaces, a central hub that concentrates risk, or a shared exchange that keeps data with its owners. The X-Road model, proven in public service and maintained as open source by NIIS, shows that the third path is practical: members connect once, providers decide who may call

their services, every message is signed and time-stamped, and the operator governs trust without holding the data.

Success depends less on software than on early decisions: governance, naming, trust services, consent rules and an onboarding process that can be repeated without heroics. Combined with open-source API management on WSO2 and a phased roadmap that starts with a few high-value verification services, those decisions turn a national or sector exchange from an ambition into an operating service.

Alin Li designs and builds such platforms on open-source software, hands them over to the organisation that runs them, and supports them in production across Europe and the Gulf. There is no lock-in: the client owns what is delivered, and every figure we share is labelled Verified or Illustrative.

GLOSSARY

Terms used

X-Road	Open-source data exchange layer developed by the Nordic Institute for Interoperability Solutions; known in Estonia as X-tee.
NIIS	Nordic Institute for Interoperability Solutions, the non-profit founded by Estonia and Finland to develop X-Road.
Member	A registered organisation that participates in the exchange as a consumer, a provider or both.
Member class	A grouping of members by legal nature, for example GOV for government entities and COM for commercial entities.
Subsystem	A specific system or unit within a member, to which access rights are granted.
Security server	The component each member runs at its edge to authenticate, sign, time-stamp and log exchange traffic.
Operator	The organisation that runs central services: member registry, global configuration and trust services.
Global configuration	The signed, centrally distributed description of members, security servers and recognised trust services.
Service identifier	The stable address of a service: INSTANCE/CLASS/MEMBER/SUBSYSTEM/SERVICE.
Catalogue	The shared directory of published services, with owners, consent rules and interface descriptions.
Consent rule	The documented legal basis and conditions under which a service may be called.
Time-stamping	An independent service proving when a message was signed.
Mutual TLS	Transport security in which both client and server authenticate with certificates.
Federation	An agreement between two exchange instances to recognise each other's trust, allowing cross-instance services.
Certification suite	The set of tests a member must pass in the sandbox before receiving production credentials.

About Alin Li

Alin Li OÜ builds platforms on open-source software, hands them over to the organisation that runs them, and supports them in production, with engineers in Europe and the Gulf, working in English and Arabic.

- **No lock-in**, including to Alin Li.
- **You own what is delivered**: configuration, code, rules, documentation and data.
- **Every figure is labelled** Verified or Illustrative.

Alin Li OÜ · Registry code 17184688 · Ahtri tn 12, 10151 Tallinn, Estonia
Info@AlinLi.com · +372 5705 1771 · alinli.com

This white paper is for general information. It is not legal, regulatory or financial advice. Figures marked Verified were measured on one estate and are not a guarantee for any other; figures marked Illustrative are hypothetical. Product names belong to their owners. © 2026 Alin Li OÜ. You may share this document unchanged.