Digital Nation Pakistan

PAKISTAN DIGITAL AUTHORITY

Standards & Regulatory Publications

D-SERIES: DATA GOVERNANCE, MANAGEMENT & EXCHANGE

DNP-D.300 TS

WASL — Pakistan National Data Exchange Layer: Technical Specification

Summary

This Technical Specification defines the concrete protocol bindings, API endpoint specifications, message envelope formats, token structures, metadata record schemas, consent policy formats, data-classification technical controls, error codes and conformance test requirements for WASL, Pakistan's National Data Exchange Layer. It is the implementer reference for building conformant WASL Client Nodes and platform integrations, subordinate to DNP-D.002 RA.

Document reference	DNP-D.300 TS
Title	WASL — Pakistan National Data Exchange Layer: Technical Specification
Document type	Technical Specification (TS)
Status	Working Draft (WD)
Maturity level	PILOT
Version	0.1
Date	26 April 2026
Classification	PUBLIC
Published by	Pakistan Digital Authority
Persistent URL	https://standards.dnp.gov.pk/DNP-D.300
URN	urn:dnp:D:300:TS:0.1:en
DOI	[To be assigned upon DOI registration]
Gazette reference	N/A

Foreword

The Pakistan Digital Authority (PDA) is the statutory body established under the Digital Nation Pakistan Act, 2025, mandated to issue standards, frameworks, and technical publications in support of the National Digital Masterplan.

WASL — Pakistan's National Data Exchange Layer, is a foundational component of the country's Digital Public Infrastructure. It enables secure, consent-governed, standardized data exchange between federal and provincial entities, regulated private-sector institutions, and authorized platforms, while preserving federated data custody and cryptographic privacy by design.

This technical specification defines the concrete implementation requirements for WASL. It provides the protocol bindings, API specifications, message formats, and conformance test requirements that implementers require to build WASL-conformant systems. It is subordinate to DNP-D.002 RA (WASL Reference Architecture) and shall be read in conjunction with it. It is being issued at PILOT maturity; revision cycles are expected to follow as implementation experience accumulates.

This specification is developed to be approved by the PDA Standards Board under the procedures defined in DNP-X.001 FWK (Nomenclature, Series Structure & Issuance Framework).

Document Control

Version	Date	Description	Approved by
0.1	26 April 2026	Initial publication	

Contact: Pakistan Digital Authority, 4th Floor, 5-A Constitution Avenue, Sector F-5/1, Islamabad 44000, Pakistan. E-mail: standards@pda.gov.pk | https://standards.dnp.gov.pk

Table of Contents

1 SCOPE

This technical specification defines the implementation requirements for WASL, Pakistan's National Data Exchange Layer. It provides the protocol bindings, API endpoint specifications, request and response envelope formats, token structures, metadata record schemas, consent policy formats, data classification technical controls, error codes, and conformance test requirements that together constitute the implementer reference for WASL.

It applies to:

- all entities implementing WASL Client Nodes, whether as data consumers, data providers, or both;
- all vendors and system integrators supplying WASL-conformant software, services, or components to participating entities or to PDA;
- all implementations of the Central WASL Platform, WASL Client Nodes, and the Secure Connectivity Layer intended for use in the WASL production ecosystem or in certified sandbox environments.

This specification does not govern:

- the architectural philosophy, design principles, and component model of WASL, these are defined in DNP-D.002 RA (WASL Reference Architecture), to which this specification is subordinate;
- the internal information systems, databases, APIs, or business logic of participating organizations, except insofar as those systems must conform to published WASL schemas and fulfilment layer requirements;
- the legal, regulatory, and mandatory participation rules governing entities that must transact over WASL, these will be issued separately as a Regulation (REG) under the DNP-D series;
- sector-specific data sharing rules and field-level schema extensions, will be issued as sectoral Profiles (PRF) under the appropriate DNP-D sub-series;

2 REFERENCES

The following references are indispensable for the application of this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document applies.

2.1 Normative References — National

- Digital Nation Pakistan Act, 2025 (Act No. I of 2025).
- Electronic Transactions Ordinance, 2002 (Ordinance No. LI of 2002).
- Prevention of Electronic Crimes Act, 2016 (Act No. XL of 2016), as amended.
- Pakistan Security Standards for Cryptographic and Information Technology Security Devices.
- The National Database and Registration Authority (NADRA) Ordinance, 2000
- DNP-X.001 FWK — Standards & Regulatory Publications: Nomenclature, Series Structure & Issuance Framework (03/2026), Pakistan Digital Authority.
- DNP-D.001 FWK — National Data Governance Framework, Pakistan Digital Authority.
- DNP-D.002 RA — WASL Reference Architecture, Pakistan Digital Authority. [This specification is subordinate to and shall be read in conjunction with DNP-D.002 RA.]

2.2 Normative References — International

- IETF RFC 2119 — Key words for use in RFCs to Indicate Requirement Levels.
- IETF RFC 8174 — Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words.
- IETF RFC 6749 — The OAuth 2.0 Authorization Framework.
- IETF RFC 7519 — JSON Web Token (JWT).
- IETF RFC 7517 — JSON Web Key (JWK).
- IETF RFC 7518 — JSON Web Algorithms (JWA).
- IETF RFC 8446 — The Transport Layer Security (TLS) Protocol Version 1.3.
- IETF RFC 8705 — OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens.
- IETF RFC 9110 — HTTP Semantics.
- OpenID Foundation — OpenID Connect Core 1.0.
- OAuth 2.1 Authorization Framework (draft-ietf-oauth-v2-1, latest).
- OpenAPI Specification v3.0.
- JSON Schema Specification (Draft 2020-12).
- ISO/IEC 27001 — Information security management systems — Requirements.
- ISO/IEC 27018 — Code of practice for protection of PII in public clouds.
- ISO/IEC 29100 — Privacy framework.
- NIST SP 800-207 — Zero Trust Architecture.
- FIPS 186-5 — Digital Signature Standard (DSS), NIST.
- FIPS 140-2/3 — Security Requirements for Cryptographic Modules, NIST.
- Protocol Buffers Language Guide (proto3), Google — applicable where gRPC transport is used.

2.3 Informative References

- Universal DPI Safeguards Framework, UNDP / Office of the UN Secretary-General's Envoy on Technology, 2023.
- G20 Framework for Digital Public Infrastructure, 2023.
- Estonia X-Road Technical Specification (reference implementation).
- European Interoperability Framework (EIF), European Commission.

3 DEFINITIONS

For the purposes of this specification, the following definitions apply. Definitions are listed alphabetically.

Term	Definition
3.1 Access Token	A short-lived JWT credential issued by the WASL IAM Token Endpoint to an authenticated Client Node. Encodes the requesting entity's identity, permitted scopes, and declared purpose.
3.2 Central WASL Platform	The shared national control-plane and routing-plane operated by NADRA and owned by the Pakistan Digital Authority, providing identity management, metadata discovery, consent orchestration, API governance, routing, audit, and transaction proof recording.
3.3 Client Node	A containerized software component deployed within a participating entity's infrastructure that

	performs all cryptographic operations (signing, verification, encryption, decryption) and mediates the entity's participation in WASL data exchange.
3.4 Conformance Test	A defined procedure that verifies whether a specific implementation satisfies one or more normative requirements of this specification. Conformance tests are expressed as observable, repeatable, and automatable where possible.
3.5 Consent	An explicit, verifiable, revocable authorization granted by a natural person through an approved consent channel for the exchange of personal data concerning that person, for a specified purpose and duration.
3.6 Consent Artifact	A signed JSON object produced by the WASL Consent Layer upon citizen approval of a data sharing request, containing the consent identifier, subject identifier, consumer and provider identifiers, dataset and field scope, declared purpose, validity window, channel, and a WASL platform digital signature.
3.7 Consumer	A participating entity that requests data through WASL for a defined and authorized purpose.
3.8 Data Blindness	The architectural property whereby the Central WASL Platform handles only encrypted payloads and is cryptographically prevented from reading the content of exchanged data, while retaining cryptographic proof that an exchange occurred.
3.9 Data Exchange Plane	The functional layer of WASL responsible for secure routing, protocol mediation, orchestration, and transaction lifecycle management between Consumer and Provider Client Nodes.
3.10 Federated Data Ownership	The principle whereby each participating entity retains custody, control, and serving responsibility for its own data and private keys, with WASL performing routing and control-plane functions only.
3.11 Fulfilment Layer	The sub-component of the Provider Client Node that receives inbound getData requests, performs all verification steps, retrieves data from internal backend systems, validates responses against the published schema, signs the response, and submits the provider-side TPL proof record.
3.12 Metadata Repository	The authoritative control-plane registry maintaining machine-readable dataset definitions, schemas, provider bindings, consent policies, and classification rules for the WASL ecosystem.
3.13 mTLS	Mutual Transport Layer Security. A mode of TLS in which both the client and server present certificates for mutual authentication. Required for all WASL Client Node to platform connections.
3.14 PAK-ID	Pakistan's national digital identity platform operated by or in coordination with NADRA, used by WASL for citizen authentication and consent acquisition.
3.15 Participating Entity	A federal, provincial, or regulated private-sector organization formally onboarded to WASL as a Consumer, Provider, or both, under a duly executed participation agreement.
3.16 Provider	A participating entity that holds authoritative source data in its systems and serves that data through

	WASL in response to authorized and consent-verified requests.
3.17 Request Envelope	The standardized JSON structure that wraps all WASL getData requests, containing the transaction identifier, dataset identifier, schema version, subject identifier, declared purpose, consent reference, requester identity, timestamp, and consumer digital signature.
3.18 Response Envelope	The standardized JSON structure that wraps all WASL getData responses, containing the transaction identifier, status code, provider identifier, data payload, timestamp, provider digital signature, and TPL proof reference.
3.19 Secure Connectivity Layer	The network-level component of the Central WASL Platform responsible for encrypted routing of payloads between Consumer and Provider Client Nodes.
3.20 Token Endpoint	The IAM-hosted endpoint to which a WASL Client Node submits credentials to obtain an OAuth 2.1 access token for use in WASL API calls.
3.21 Transaction Proof Layer (TPL)	The component of the Central WASL Platform that records cryptographic proofs of each transaction (hashes, signatures, timestamps) without recording transaction content.
3.22 Verifiable Credential (VC)	A tamper-evident, cryptographically signed digital credential conforming to the W3C Verifiable Credentials Data Model; issuance and lifecycle are governed under the DNP-U series and are outside the scope of this specification.
3.23 WASL	Pakistan's National Data Exchange Layer, the subject of this specification. The name is rendered in Urdu as وصل, meaning connection or linkage.
3.24 Zero-Knowledge Proof (ZKP)	A cryptographic method by which one party can prove to another that a statement is true without revealing any information beyond the validity of the statement itself. 3.25 AI Agent: A software system that uses an artificial intelligence model to autonomously plan, decide, or execute actions, including actions that invoke WASL APIs. An AI Agent may operate on behalf of an institution or, under a delegated consent artefact (v2 capability), on behalf of a citizen. For WASL authentication purposes, an AI Agent is an authorized Consumer application subject to the additional identity binding requirements in Section 24.1.
3.26 Client Category	A risk-based classification assigned to each Participating Entity at onboarding determining private key protection requirements. Four categories are defined in section
3.27 Cryptographic Agility	The architectural property whereby cryptographic algorithms, key sizes, and protocol parameters can be updated or replaced without redesigning the system. Implemented in WASL through the Cryptographic Parameter Registry and per-session algorithm negotiation per Section 14.3a.
3.28 Cryptographic Parameter Registry	The sub-component of the Metadata Repository publishing permitted cryptographic algorithm suites, minimum key sizes, and deprecation schedules. Client Nodes retrieve and cache these parameters

	for per-session algorithm negotiation and PQC migration.
3.29 Model Context Protocol (MCP)	An open protocol standardizing how AI applications connect to external data sources and tools. WASL exposes an MCP server surface through the Developer Portal (Section 24.3).
3.30 Post-Quantum Cryptography (PQC)	Cryptographic algorithms designed to remain secure against adversaries with access to cryptographically-relevant quantum computers. WASL mandates cryptographic agility to enable migration to NIST-standardized PQC algorithms (ML-KEM / FIPS 203, ML-DSA / FIPS 204, SLH-DSA / FIPS 205) per Section 14.3a.

4 ABBREVIATIONS

Abbreviation	Definition
API	Application Programming Interface
CA	Certificate Authority
CNIC	Computerized National Identity Card
CRL	Certificate Revocation List
DCS	Department of Communications Security
DNP	Digital Nation Pakistan
DPI	Digital Public Infrastructure
FIPS	Federal Information Processing Standard
gRPC	Google Remote Procedure Call
HMAC	Hash-based Message Authentication Code
HSM	Hardware Security Module
IAM	Identity and Access Management
IPSec	Internet Protocol Security
JWA	JSON Web Algorithms
JWK	JSON Web Key
JWS	JSON Web Signature
JWT	JSON Web Token
mTLS	Mutual Transport Layer Security
NADRA	National Database and Registration Authority
NDEL	National Data Exchange Layer
NIST	National Institute of Standards and Technology
NTN	National Tax Number
OCSP	Online Certificate Status Protocol
OIDC	OpenID Connect
PDA	Pakistan Digital Authority
PEP	Policy Enforcement Point
PKI	Public Key Infrastructure
RBAC	Role-Based Access Control
REST	Representational State Transfer
RFC	Request for Comments
TLS	Transport Layer Security
TPL	Transaction Proof Layer
TSA	Timestamping Authority
VC	Verifiable Credential
VPN	Virtual Private Network

Abbreviation	Definition
WASL	Pakistan's National Data Exchange Layer
WAF	Web Application Firewall
ZKP	Zero-Knowledge Proof

5 PROTOCOL BINDINGS

This section specifies the concrete protocol bindings that all WASL-conformant implementations **SHALL** support. Key words (**SHALL**, **MUST**, **SHOULD**, MAY) are used in accordance with IETF RFC 2119 / RFC 8174.

5.1 Transport Security: TLS 1.3

All communications within the WASL ecosystem **SHALL** use TLS 1.3 as defined in IETF RFC 8446. TLS 1.2 MAY be used only where TLS 1.3 is not supported by existing infrastructure and only with explicit PDA exemption. TLS 1.1 and earlier are prohibited.

5.1.1 Mandatory TLS 1.3 Cipher Suites

Cipher Suite	IANA Code	Requirement
TLS_AES_256_GCM_SHA384	0x1302	REQUIRED
TLS_CHACHA20_POLY1305_SHA256	0x1303	REQUIRED
TLS_AES_128_GCM_SHA256	0x1301	RECOMMENDED

5.1.2 Certificate Requirements

All TLS certificates used in WASL **SHALL**: be issued by the WASL-recognized PKI Certificate Authority; conform to X.509 version 3; use RSA-4096 or ECDSA P-256/P-384 key pairs; have a maximum validity period of 24 months for entity signing certificates and for VPN device certificates; include appropriate Key Usage and Extended Key Usage extensions; and support OCSP stapling.

5.2 Mutual TLS (mTLS) Binding

All WASL Client Node to Central Platform connections **SHALL** use mTLS as specified in IETF RFC 8705. Both client and server **MUST** present valid X.509 certificates issued by the WASL PKI. Certificate validation **SHALL** include: chain verification to the WASL Root CA; revocation checking via OCSP (preferred) or CRL; hostname verification against the registered entity endpoint; and certificate expiry validation.

NOTE: mTLS is enforced at the application transport layer above the IPSec VPN tunnel, providing defence in depth. Failure of mTLS validation **MUST** result in connection termination with no data exchanged.

5.2.1 mTLS Client Certificate Binding to OAuth Token

When mTLS client authentication is used with OAuth 2.1, the access token **SHALL** be bound to the client certificate per RFC 8705 Section 3. The token endpoint **SHALL** include the SHA-256 thumbprint (x5t#S256) of the client certificate in issued tokens. The API Gateway **SHALL** verify that the presented client certificate matches the thumbprint in the access token.

```
// Certificate thumbprint binding in JWT (cnf claim)
{
  "sub": "org:punjab_education_dept",
  "cnf": {"x5t#S256": "bwDkHGl3JzRrVF1AkOhQdS7Pc_q3EjFEUKuLb2xKH88"},
  "scope": "data.fetch consent.read",
  "exp": 1746000000
}
```

5.3 OAuth 2.1 Binding

WASL IAM implements OAuth 2.1 (draft-ietf-oauth-v2-1). The client credentials grant with private_key_jwt client authentication is the mandatory grant for all automated Client Node to platform interactions.

5.3.1 Token Request

```
POST /oauth2/token HTTP/1.1
Host: iam.wasl.gov.pk
Content-Type: application/x-www-form-urlencoded
grant_type=client_credentials
&client_assertion_type=urn:ietf:params:oauth:client-assertion-
type:jwt-bearer
&client_assertion=<signed-JWT>
&scope=data.fetch+consent.read
```

5.3.2 Token Endpoint Parameters

Parameter	Type	Required	Description
grant_type	string	REQUIRED	**MUST** be client_credentials
client_assertion_type	string	REQUIRED	**MUST** be urn:ietf:params:oauth:client-assertion-type:jwt-bearer
client_assertion	string	REQUIRED	Signed JWT asserting client identity
scope	string	REQUIRED	Space-separated list of requested scopes

5.3.3 Supported Scopes

Scope	Permits	Restriction
data.fetch	Invoke getData API	Requires dataset subscription and consent where applicable
data.schema.read	Invoke getSchema and listDatasets APIs	All registered entities
consent.read	Invoke getConsentInfo API	Consumer and subject entities only
consent.write	Submit consent decisions; invoke revokeConsent	Consent Layer and authorised entities
tpl.query	Invoke getTransactionTPLStatus API	Parties to the transaction only

metadata.admin	Publish/update metadata records	PDA and approved domain custodians only

5.4 OpenID Connect (OIDC) Binding

WASL IAM supports OIDC 1.0 for citizen-facing authentication and consent flows via PAK-ID integration. The authorization_code flow with PKCE **MUST** be used for all user-facing interactions. The implicit flow is prohibited. The OIDC discovery document is published at: https://iam.wasl.gov.pk/.well-known/openid-configuration

5.5 IPSec VPN Binding

All participating entities **SHALL** establish site-to-site IPSec VPN tunnels to the WASL Secure Connectivity Layer prior to production participation.

Parameter	Required Value
IKE Version	IKEv2 (RFC 7296)
Authentication Method	Certificate-based (X.509 issued by WASL PKI)
Encryption Algorithm	AES-256-GCM
Integrity Algorithm	SHA-384 or SHA-512
Diffie-Hellman Group	Group 19 (ECDH P-256) or Group 20 (ECDH P-384) — minimum
Perfect Forward Secrecy	Required — rekey interval not to exceed 8 hours
Tunnel Mode	Site-to-site (not transport mode)
Dead Peer Detection	Required — interval: 30s, timeout: 120s

6 IDENTITY AND ACCESS MANAGEMENT: TECHNICAL SPECIFICATION

The Identity and Access Management (IAM) component is the trust anchor of the WASL ecosystem, responsible for establishing and enforcing identity, authentication, and authorization across all participating entities. IAM enables secure machine-to-machine and user-mediated interactions by issuing verifiable access tokens, enforcing access policies, and integrating with national identity systems such as PAK-ID. It ensures that every request entering the WASL ecosystem is attributable, authorized, and cryptographically verifiable.

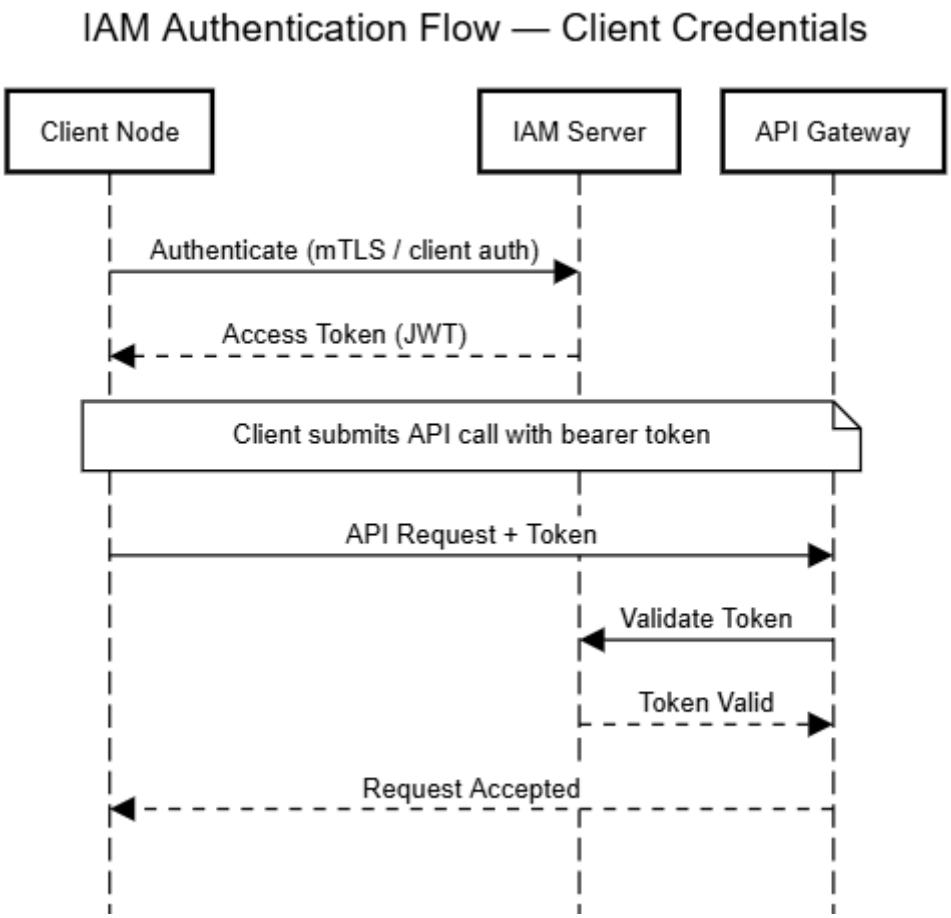

Figure 6-1: IAM Authentication Flow — OAuth 2.1 client credentials with mTLS client authentication

The IAM component performs the following core functions:

- Registration and identity lifecycle management for participating organizations.
- Client application registration and credential issuance.
- OAuth 2.1 token issuance and validation using the client_credentials flow with private_key_jwt client assertions.
- OpenID Connect (OIDC) supports citizen identity federation via PAK-ID.
- Role-based and attribute-based access control enforcement (consumer, provider, administrator).
- Token introspection, revocation, and expiry enforcement including JTI replay prevention.
- Secure binding of organizational identities to PKI-issued cryptographic credentials.

WASL IAM follows a zero-trust, token-based security model where every request must be authenticated and authorized independently. Authentication uses Mutual TLS (mTLS) for client-node authentication combined with the OAuth 2.1 client credentials flow for machine-to-machine communication. Authorization is enforced using OAuth scopes, role-based access control (RBAC), and attribute-based policies derived from the Metadata Repository and Consent Layer. No implicit trust exists between participants; all interactions are explicitly verified before any data exchange proceeds.

6.1 IAM Endpoints

IAM Endpoint	URI	Protocol

Token Endpoint	https://iam.wasl.gov.pk/oauth2/token	OAuth 2.1
JWKS URI	https://iam.wasl.gov.pk/.well-known/jwks.json	JWK (RFC 7517)
Introspection Endpoint	https://iam.wasl.gov.pk/oauth2/introspect	RFC 7662
Revocation Endpoint	https://iam.wasl.gov.pk/oauth2/revoke	RFC 7009
OIDC Discovery	https://iam.wasl.gov.pk/.well-known/openid-configuration	OIDC 1.0
Client Registration	https://iam.wasl.gov.pk/register	RFC 7591

6.2 JWT Access Token Structure

WASL IAM issues JSON Web Tokens as access tokens. All tokens **SHALL** be signed using ES256 (ECDSA with P-256 and SHA-256) or RS256. The signing key **MUST** be stored in an HSM conforming to FIPS 140-2/3 Level 3 or above.

6.2.1 JWT Header

```
{
  "alg": "ES256",
  "typ": "JWT",
  "kid": "wasl-iam-signing-key-2026-01"
}
```

6.2.2 JWT Payload — Complete Claim Set

```
{
  "iss": "https://iam.wasl.gov.pk",
  "sub": "org:bank_alfalah",
  "aud": ["https://api.wasl.gov.pk"],
  "scope": "data.fetch consent.read",
  "org_id": "bank_alfalah",
  "org_type": "financial_institution",
  "client_id": "loan_app_01",
  "purpose": "credit_assessment",
  "roles": ["consumer"],
  "dataset_subscriptions": ["income.tax", "civil.registry.basic"],
  "iat": 1746000000,
  "exp": 1746000900,
  "nbf": 1746000000,
  "jti": "7f3d4a21-8b9c-4e1f-a2d3-5c6e7f8a9b0c",
  "cnf": {"x5t#S256": "bwDkHGl3JzRrVF1AkOhQdS7Pc_q3EjFEUKuLb2xKH88"}
}
```

6.2.3 JWT Claim Definitions

Claim	Type	Required	Description
iss	string (URI)	REQUIRED	Issuer. **MUST** be https://iam.wasl.gov.pk

sub	string	REQUIRED	Subject. Format: org:<entity_id> for entity tokens
aud	string[]	REQUIRED	Audience. **MUST** include https://api.wasl.gov.pk
scope	string	REQUIRED	Space-separated list of granted scopes
org_id	string	REQUIRED	Registered WASL entity identifier
org_type	string	REQUIRED	government | financial_institution | regulated_private | platform
client_id	string	REQUIRED	Registered client application identifier
purpose	string	REQUIRED	Declared purpose code from the Purpose Code Registry
roles	string[]	REQUIRED	WASL roles: consumer | provider | admin
dataset_subscriptions	string[]	REQUIRED	List of dataset IDs the entity is subscribed to
iat	integer (epoch)	REQUIRED	Issued-at time
exp	integer (epoch)	REQUIRED	Expiry time. Maximum: iat + 900 seconds (15 minutes)
nbf	integer (epoch)	REQUIRED	Not-before time. **SHOULD** equal iat
jti	string (UUID v4)	REQUIRED	Unique token ID for replay prevention
cnf.x5t#S256	string	REQUIRED	SHA-256 thumbprint of mTLS client certificate (RFC 8705). When the token is issued to an AI-driven application, the following additional claims **SHALL** be present: ai_model_id (REQUIRED for AI agents) — registered model identifier from the DNP-A model registry, format: <registry>:<model_id>:<version>; ai_invoking_principal (REQUIRED for AI agents), the human operator, service account, or orchestrating system that initiated the AI action; ai_model_version (REQUIRED for AI agents), model version string for forensic traceability. These claims are omitted (null) for standard non-AI tokens.
delegated_by	CONDITIONAL	REQUIRED when token_type=delegated_channel. The org_id of the WASL Participating Entity that issued the API key/secret to this delegated application. Absent for standard Participating Entity tokens. Used by the API Gateway to enforce scope and resolve the sponsoring entity's authorisations (Section 26.2).	
delegation_scope	CONDITIONAL	REQUIRED when token_type=delegated_channel. Space-separated list of dataset IDs and API scopes the sponsoring entity has explicitly	

		permitted the delegated application to access. The API Gateway **SHALL** reject any request where the requested operation or dataset falls outside this list, even if the sponsoring entity itself has broader subscriptions. Absence of this claim on a delegated_channel token is a token validation failure (Section 26.3).	

6.2.4 Token Validation Requirements

Resource servers **SHALL** validate access tokens by: verifying JWT signature against the WASL IAM JWKS public key identified by kid; verifying iss is exactly https://iam.wasl.gov.pk; verifying aud includes the resource server's registered audience URI; verifying exp is in the future (clock skew tolerance: 30 seconds); verifying jti has not been seen before (maintain JTI cache for token_lifetime + 30s); for mTLS-bound tokens, verifying the presented client certificate SHA-256 thumbprint matches cnf.x5t#S256; and verifying scope includes the required scope for the requested operation.

6.3 IAM Security Controls

Control	Specification
Token lifetime	Maximum 900 seconds (15 minutes). Shorter lifetimes recommended for Tier 4/5 data.
Replay prevention	JTI uniqueness enforced. JTI cache maintained for token_lifetime + 30s.
Rate limiting	Maximum 60 token requests per minute per client_id. Excess returns HTTP 429.
IP restriction	Token endpoint accessible only from registered entity IP ranges.
Key rotation	IAM signing keys rotated every 90 days. Old keys retained in JWKS for 30 days post-rotation.
HSM requirement	All IAM private signing keys **MUST** reside in FIPS 140-2/3 Level 3 HSM.
Failure scenarios	See Table 6-2 below.

Table 6-2: IAM Failure and Error Scenarios

Scenario	Outcome
Invalid client credentials	Token request rejected: HTTP 401
Expired token submitted to API	API request rejected: AUTH_TOKEN_EXPIRED
Invalid JWT signature	Request rejected: AUTH_TOKEN_INVALID
Unauthorized scope requested	Token issued with intersection of requested and permitted scopes
Token replay detected (duplicate jti)	Request blocked: AUTH_TOKEN_REPLAY
IAM temporarily unavailable	Retry / failover to standby IAM instance

7 STANDARD API LAYER: ENDPOINT SPECIFICATIONS

The Standard API Layer defines the uniform interface contract through which all data exchange operations are performed within WASL. All endpoints are exposed at the API Gateway base URI: https://api.wasl.gov.pk/v1/. All requests **MUST** carry a valid Bearer access token, and **MUST** be made over mTLS connections.

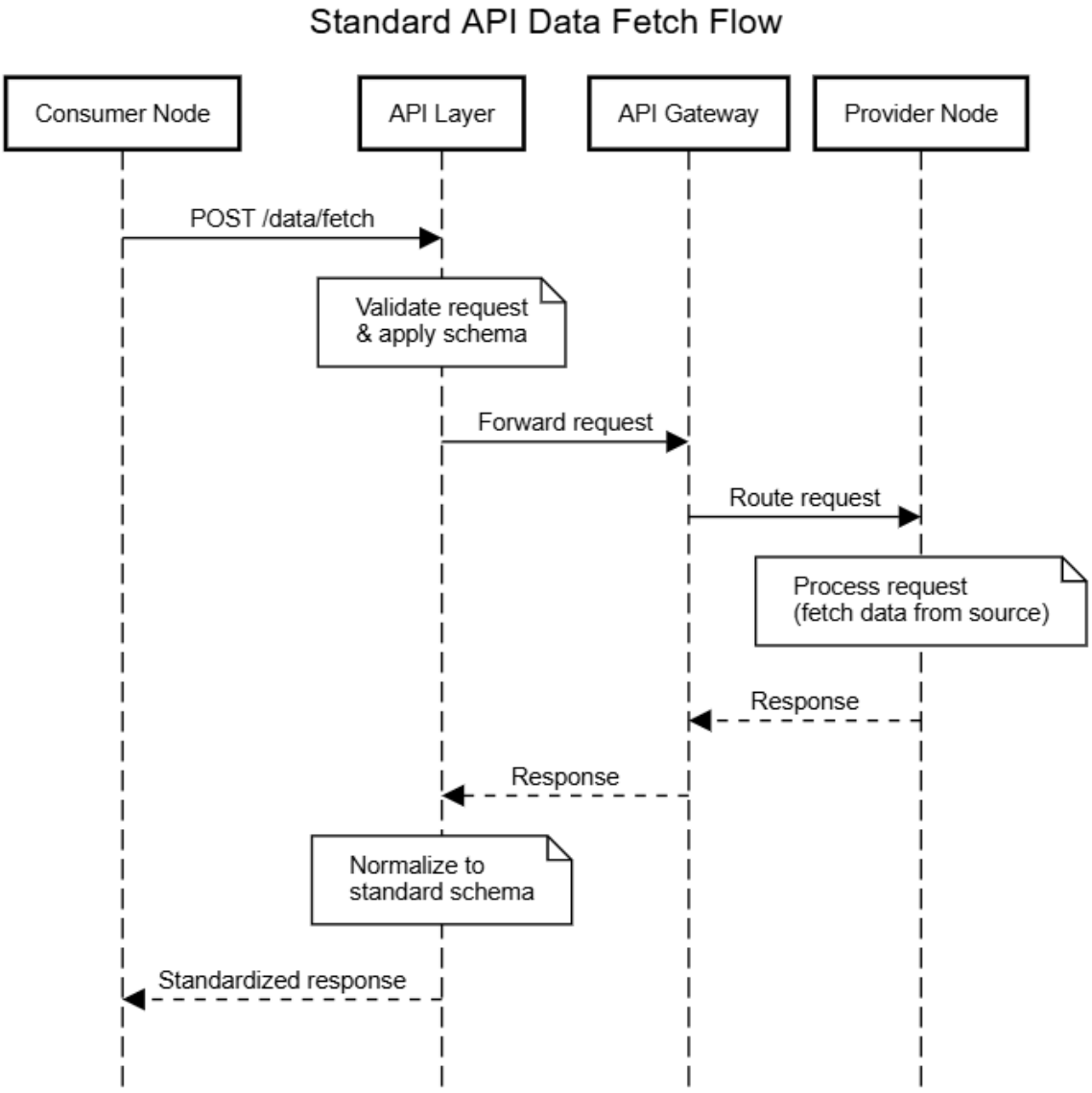

Figure 7-1: Standard API Interaction Model — consumer applications interact with provider systems through the standardised API interface mediated by the WASL platform and Client Nodes

This section specifies the Standard API Layer endpoint definitions and the API Gateway that enforces them at runtime. The Standard API Layer defines the uniform interface contract through which all data exchange operations are performed within WASL. Rather than allowing each provider to expose arbitrary APIs, WASL enforces a standardized interaction model ensuring that consumers integrate once and can interact with multiple providers, providers conform to a shared request and response structure, and platform-level controls (authentication, consent, audit, proof) are consistently enforced on every call.

The Standard API Layer is governed by five principles: Uniformity (consistent API structure

regardless of provider); Abstraction (consumers interact with logical datasets, not provider-specific implementations); Policy Enforcement (consent, authentication, and classification policies enforced uniformly on every call); Extensibility (supports asynchronous processing and event-driven exchange); and Interoperability (APIs defined using open standards including REST and OpenAPI v3.0). No API may bypass the WASL trust and governance framework. Standard APIs must always be used for core data exchange; custom APIs, where offered, remain subject to the same IAM, consent, and audit controls.

The API Gateway is the central runtime enforcement and routing layer for all WASL traffic. It is the single entry and exit point for all API requests, integrating with IAM, the Metadata Repository, the Consent Layer, and the Transaction Proof Layer to ensure every request is authenticated, authorized, consent-validated, schema-compliant, and auditable before being routed to any provider. It enforces rate limiting, IP allowlisting, WAF protections, and replay attack prevention. Routing is metadata-driven: the gateway resolves the correct provider's endpoint using the dataset identifier, provider availability, and load-balancing policies from the Metadata Repository.

The gateway also supports gRPC for high-throughput integrations, WebSocket connections for real-time event-streaming, and GraphQL for use cases requiring flexible, consumer-defined field selection over composite datasets published through the Custom API Marketplace. GraphQL endpoints are subject to the same IAM, consent, schema compliance, and audit controls as all other WASL API interactions; they do not provide a mechanism to bypass field-level access or consent policy enforcement.

7.1 Standard API Summary

API Endpoint	Method	URI	Required Scope
getData	POST	https://api.wasl.gov.pk/v1/data/fetch	data.fetch
getSchema	GET	https://api.wasl.gov.pk/v1/schema/{datasetId}	data.schema.read
getConsentInfo	GET	https://api.wasl.gov.pk/v1/consent/{consentId}	consent.read
getTransactionTPLStatus	GET	https://api.wasl.gov.pk/v1/tpl/{transactionId}	tpl.query
submitConsentDecision	POST	https://api.wasl.gov.pk/v1/consent/decision	consent.write
revokeConsent	DELETE	https://api.wasl.gov.pk/v1/consent/{consentId}	consent.write
listDatasets	GET	https://api.wasl.gov.pk/v1/metadata/datasets	data.schema.read
getProviderStatus	GET	https://api.wasl.gov.pk/v1/providers/{providerId}/status	data.schema.read

By constraining data exchange to these standardized APIs, WASL enables consumers to integrate once against the standard interface, providers to implement a single standardized response format, and the platform to enforce consent checks, signature verification, and proof recording on every getData call.

7.2 getData Endpoint

7.2.1 Request Specification

Property	Value
HTTP Method	POST
URI	https://api.wasl.gov.pk/v1/data/fetch
Content-Type	application/json

Authorization	Bearer <access_token>
Required Scope	data.fetch
mTLS	Required
Max Request Size	1 MB (encrypted payload)
Synchronous Timeout	30 seconds
Idempotency	Not idempotent — each call creates a new transaction record in TPL

7.2.2 HTTP Request Headers

Header	Required	Description
Authorization	REQUIRED	Bearer <JWT access token>
Content-Type	REQUIRED	application/json
X-WASL-Transaction-ID	REQUIRED	Consumer-generated UUID v4. Used for idempotency tracking and TPL recording.
X-WASL-Request-Timestamp	REQUIRED	ISO 8601 UTC timestamp. Max skew vs server time: 5 minutes.
X-WASL-Signature	REQUIRED	Base64url-encoded consumer digital signature over canonical request body.
X-WASL-API-Version	OPTIONAL	Requested API version. Defaults to current stable version if omitted.
Accept	OPTIONAL	application/json (default); application/jose+json for JWE-wrapped responses.

7.3 Custom API Marketplace and Developer Portal

While the Standard API Layer defines the mandatory baseline for data exchange, the Custom API Marketplace provides a controlled innovation layer enabling participating organizations to expose domain-specific APIs beyond the standard schema, such as composite eligibility workflows, credit-scoring services, or sector-specific processing, while remaining within the WASL trust, IAM, consent, and audit framework. All custom APIs must be registered in the Metadata Repository, conform to WASL authentication and authorization standards, enforce consent requirements where applicable, and produce auditable transactions. Custom APIs are categorized as Public (accessible to all authorized participants), Restricted (requiring approval), or Private (bilateral use only). Publication is subject to PDA governance and approval to prevent fragmentation.

The Developer Portal provides a self-service environment for: discovering available datasets and custom APIs; accessing schema definitions and documentation; registering applications and generating OAuth credentials; testing integrations in the certified sandbox environment; and managing API subscriptions and usage monitoring. The sandbox allows organizations to complete full integration testing, including consent flow validation, signature verification, and failure scenario handling, before production activation.

8 REQUEST AND RESPONSE ENVELOPE FORMATS

8.1 getData Request Envelope: JSON Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/request-envelope.json",
  "title": "WASL getData Request Envelope",
  "type": "object",
  "required": ["transactionId","datasetId","schemaVersion","subject",
               "purpose","requester","timestamp","signature"],
  "additionalProperties": false,
  "properties": {
    "transactionId": {"type": "string", "format": "uuid"},
    "datasetId": {
      "type": "string",
      "pattern": "^[a-z][a-z0-9]*\\.[a-z][a-z0-9._-]*$"
    },
    "schemaVersion": {"type": "string", "pattern":
"^\\d+\\.\\d+\\.\\d+$"},
    "subject": {
      "type": "object",
      "required": ["idType","idValue"],
      "properties": {
        "idType": { "type": "string",
                    "enum":
["CNIC","NTN","PASSPORT","FNIC","NICOP","ORG_REG"] },
        "idValue": {"type": "string", "minLength": 1, "maxLength":
100}
      }
    },
    "purpose": {"type": "string"},
    "consentId": { "type": "string",
      "description": "REQUIRED if datasetId is consent-governed." },
    "requester": {
      "type": "object",
      "required": ["orgId","clientId"],
      "properties": {
        "orgId": {"type": "string"},
        "clientId": { "type": "string" }
      }
    },
    "fields": { "type": "array", "items": { "type": "string" },
      "description": "Optional field-level filter. If omitted, all
consented fields returned." },
    "timestamp": { "type": "string", "format": "date-time" },
    "encryptedPayload": { "type": "string",
      "description": "JWE compact serialization if E2E encryption is
used." },
    "signature": { "type": "string",
```

```
      "description": "Base64url-encoded consumer digital signature
over canonical request." },
    "async": { "type": "boolean", "default": false },
    "callbackUrl": { "type": "string", "format": "uri" }


      "cryptoSuite": { "type": "string", "description": "Algorithm
suite             identifier selected by the Consumer for this
transaction, chosen from the  Cryptographic Parameter Registry.
Required when encryptedPayload is present.      SHALL match a suite
identifier currently listed as 'permitted' in the
    registry. Example: 'ECDH-ES+A256GCM-v1'.", "example": "ECDH-
ES+A256GCM-v1" }
  }
}
```

8.1.1 Example getData Request

```
POST /v1/data/fetch HTTP/1.1
Host: api.wasl.gov.pk
```

Authorization: Bearer eyJhbGciOiJFUzI1NiJ9...

```
Content-Type: application/json
```

X-WASL-Transaction-ID: 7f3d4a21-8b9c-4e1f-a2d3-5c6e7f8a9b0c

X-WASL-Request-Timestamp: 2026-04-18T10:00:00Z

X-WASL-Signature: MEUCIQDx3YjF...

```
{
  "transactionId": "7f3d4a21-8b9c-4e1f-a2d3-5c6e7f8a9b0c",
  "datasetId": "health.immunization",
  "schemaVersion": "1.0.0",
  "subject": { "idType": "CNIC", "idValue": "35201-1234567-3" },
  "purpose": "school_admission",
  "consentId": "cons-99821-abcd-ef01",
  "requester": { "orgId": "edu_dept_punjab", "clientId":
"school_portal_prod" },
  "timestamp": "2026-04-18T10:00:00Z",
  "signature": "MEUCIQDx3YjF..."
}
```

8.2 getData Response Envelope: JSON Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/response-envelope.json",
  "title": "WASL getData Response Envelope",
  "type": "object",
  "required": ["transactionId","status","providerId","timestamp",
```

```
                "providerSignature","proofRef"],
  "additionalProperties": false,
  "properties": {
    "transactionId": { "type": "string", "format": "uuid" },
    "status": {
      "type": "string",
      "enum": ["SUCCESS","PARTIAL_SUCCESS","CONSENT_REQUIRED",
               "PROVIDER_ERROR","SCHEMA_ERROR","AUTH_FAILED",

"NOT_FOUND","RATE_LIMITED","SERVICE_UNAVAILABLE","PENDING"]
    },
    "providerId": { "type": "string" },
    "data": { "type": "object",
      "description": "Schema-validated response payload. Null if
status != SUCCESS." },
    "encryptedData": { "type": "string",
      "description": "JWE compact serialization if E2E encryption was
requested." },
    "errorDetail": {
      "type": "object",
      "properties": {
        "code": { "type": "string" },
        "message": { "type": "string" },
        "detail": { "type": "string" }
      }
    },
    "timestamp": { "type": "string", "format": "date-time" },
    "providerSignature": { "type": "string" },
    "proofRef": { "type": "string" },
    "schemaVersion": { "type": "string" },
    "datasetId": { "type": "string" }


      "cryptoSuite": { "type": "string", "description": "Algorithm
suite
     identifier used by the Provider to encrypt the response. SHALL
match the
     suite declared in the corresponding request envelope. Required
when
     encryptedData is present." }
  }
}
```

8.2.1 Example getData Success Response

HTTP/1.1 200 OK

```
Content-Type: application/json
{
  "transactionId": "7f3d4a21-8b9c-4e1f-a2d3-5c6e7f8a9b0c",
  "status": "SUCCESS",
```

```
  "providerId": "prov_health_registry",
  "data": {
    "immunizationStatus": "COMPLETE",
    "lastDoseDate": "2025-01-10",
    "vaccineCount": 8
  },
  "timestamp": "2026-04-18T10:00:02Z",
  "providerSignature": "MEUCIQDy7KpX...",
  "proofRef": "tpl-889922-abc",
  "schemaVersion": "1.0.0",
  "datasetId": "health.immunization"
}
```


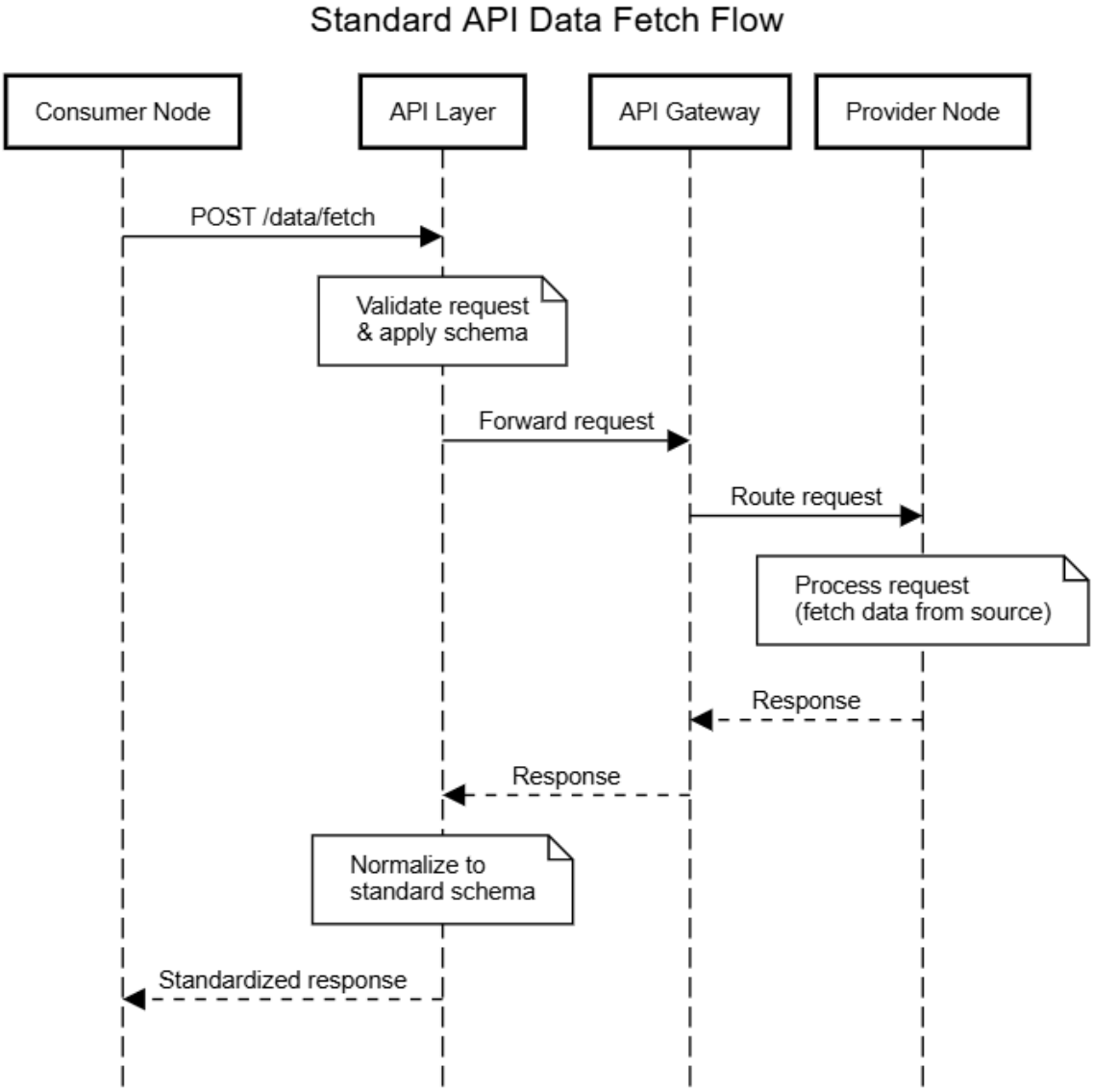

Figure 8-1: Standard API Data Fetch Flow: standardized data request through API Layer, Gateway, and Provider Node

8.3 getSchema Endpoint

8.3.1 Example Request and Response

```
GET /v1/schema/health.immunization?version=1.0.0 HTTP/1.1
Host: api.wasl.gov.pk
```

Authorization: Bearer eyJhbGci...

HTTP/1.1 200 OK

```
Content-Type: application/json
```

Cache-Control: max-age=3600

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/health.immunization/1.0.0",
  "title": "Health Immunization Record",
  "version": "1.0.0",
  "datasetId": "health.immunization",
  "domain": "health",
  "classification": "restricted_personal",
  "consentRequired": true,
  "subjectType": "natural_person",
  "type": "object",
  "required": ["immunizationStatus"],
  "properties": {
    "immunizationStatus": { "type": "string", "enum":
["COMPLETE","PARTIAL","NONE"] },
    "lastDoseDate": { "type": ["string","null"], "format": "date" },
    "vaccineCount": { "type": "integer", "minimum": 0 },
    "nextScheduledDate": { "type": ["string","null"], "format": "date"
}
  }
}
```

8.4 getConsentInfo Endpoint

```
GET /v1/consent/cons-99821-abcd-ef01 HTTP/1.1
Host: api.wasl.gov.pk
```

Authorization: Bearer eyJhbGci...

HTTP/1.1 200 OK

```
{
  "consentId": "cons-99821-abcd-ef01",
  "subjectId": "CNIC:35201-1234567-3",
  "consumerId": "edu_dept_punjab",
  "providerId": "prov_health_registry",
```

```
  "datasetId": "health.immunization",
  "fields": ["immunizationStatus","lastDoseDate"],
  "purpose": "school_admission",
  "validFrom": "2026-04-18T10:00:00Z",
  "validTo": "2026-04-30T23:59:59Z",
  "status": "ACTIVE",
  "channel": "pak_id_app",
  "signature": "MEUCIQDzConsent..."
}
```

8.5 getTransactionTPLStatus Endpoint

```
GET /v1/tpl/7f3d4a21-8b9c-4e1f-a2d3-5c6e7f8a9b0c HTTP/1.1
Host: api.wasl.gov.pk
```

Authorization: Bearer eyJhbGci...

HTTP/1.1 200 OK

```
{
  "transactionId": "7f3d4a21-8b9c-4e1f-a2d3-5c6e7f8a9b0c",
  "status": "RECORDED",
  "requestHash": "sha256:a1b2c3d4e5f6...",
  "responseHash": "sha256:f6e5d4c3b2a1...",
  "consentHash": "sha256:c0nsent1234...",
  "consumerId": "edu_dept_punjab",
  "providerId": "prov_health_registry",
  "timestamp": "2026-04-18T10:00:03Z",
  "providerSignature": "MEUCIQDyProv...",
  "gatewaySignature": "MEUCIQDzGw...",
  "merkleProof": {
    "treeId": "tpl-batch-2026-04-18-001",
    "leaf": "sha256:leaf_hash_abc",
    "path": ["sha256:sibling1...", "sha256:sibling2..."],
    "root": "sha256:merkle_root_xyz"
  }
}
```

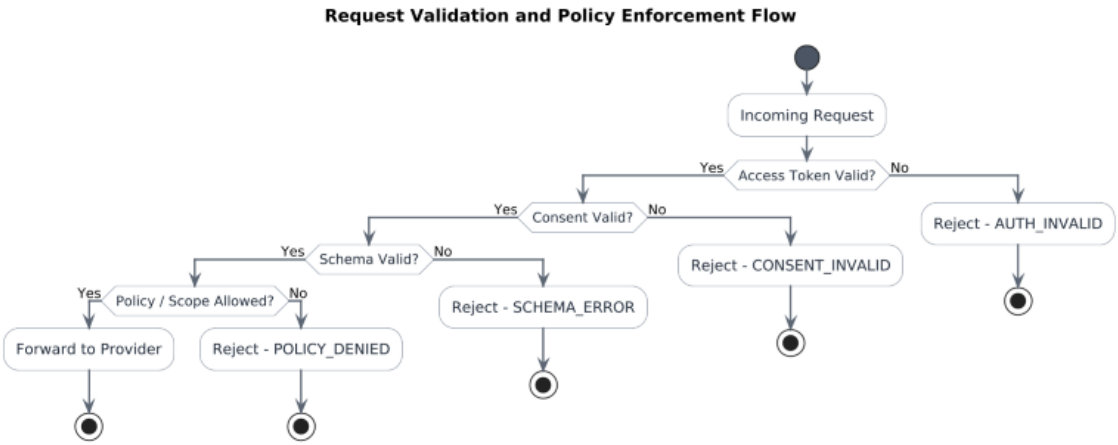

Figure 8-2: API Gateway Validation Pipeline, sequential validation steps before a request is allowed to reach the provider

8.6 Asynchronous Processing Pattern

For long-running or multi-step workflows, the getData API supports asynchronous processing. The consumer includes async: true in the request body. The API Gateway returns immediately with HTTP 202 Accepted and a polling URL.

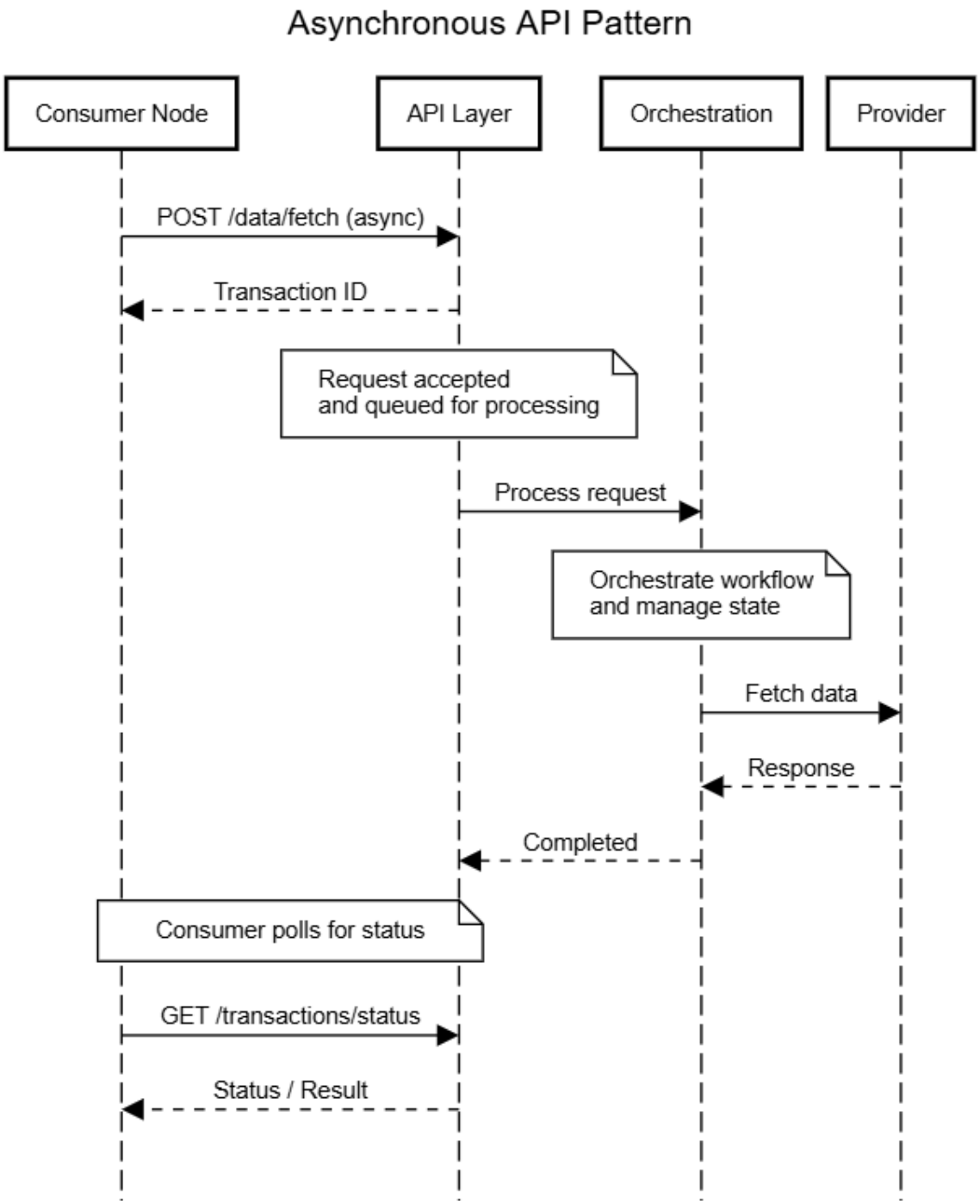

Figure 8-3: Asynchronous API Pattern, long-running data requests handled using transaction-based asynchronous processing

```
// Async request — additional fields in request body
{ "async": true, "callbackUrl": "https://api.bank.com/wasl/cb", ... }
// Immediate response (HTTP 202 Accepted)
{
  "transactionId": "7f3d4a21-...",
  "status": "PENDING",
  "pollUrl": "https://api.wasl.gov.pk/v1/async/7f3d4a21-.../status",
  "estimatedCompletionSeconds": 15
}
```

9 METADATA RECORD FORMAT

The Metadata Repository is the authoritative control-plane registry for the WASL ecosystem, maintaining machine-readable dataset definitions, provider bindings, consent policies, and classification rules. Unlike data, which remains federated with each owning organization, the Metadata Repository is deliberately centralized within the Central WASL Platform. This centralization is intentional: a single authoritative source for schemas, provider bindings, and consent policies is what enables the standardize-once, consume-everywhere model, any Client Node, in any province, connecting to any provider, resolves the same schema definitions and routing rules without bilateral coordination.
This section specifies the technical formats and schemas for metadata records. Sections 9.4 and 9.5 below cover ZKP capability annotations and the Provider Directory respectively.

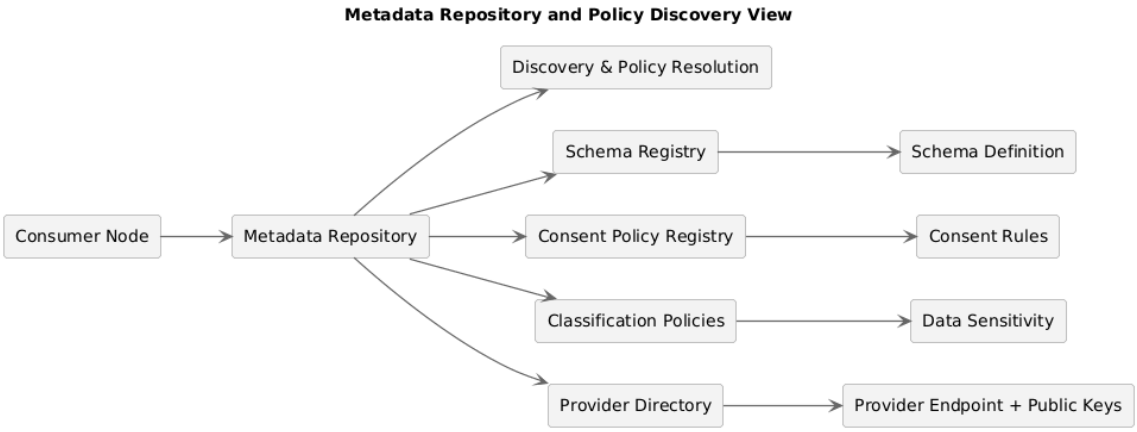

Figure 9-1: Metadata Repository in the WASL Control Plane, central control-plane registry exposing schemas, provider mappings, consent rules, and classification policies

9.1 Metadata Record JSON Schema

Every dataset or service registered in the WASL Metadata Repository **SHALL** conform to the following JSON schema. This is the authoritative format for all metadata records published via the getSchema and listDatasets APIs.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/metadata-record.json",
  "title": "WASL Metadata Record",
  "type": "object",
  "required": ["datasetId","datasetName","domain","providerId",
               "schemaVersion","subjectType","classification",
               "consentRequired","allowedPurposes","authMethod",
               "endpointRef","publicKeyRef","status"],
  "additionalProperties": false,
  "properties": {
    "datasetId": {
      "type": "string",
      "pattern": "^[a-z][a-z0-9]*\\.[a-z][a-z0-9._-]*$",
```

```
      "examples": ["health.immunization","land.record.ownership"]
    },
    "datasetName": { "type": "string", "maxLength": 200 },
    "description": { "type": "string", "maxLength": 2000 },
    "domain": {
      "type": "string",
      "enum": ["civil_registry","land","tax","health","education",
               "business","financial","social_protection","legal",
               "transport","custom"]
    },
    "providerId": { "type": "string" },
    "providerName": { "type": "string" },
    "schemaVersion": { "type": "string", "pattern":
"^\\d+\\.\\d+\\.\\d+$" },
    "schemaRef": { "type": "string", "format": "uri" },
    "subjectType": {
      "type": "string",
      "enum": ["natural_person","legal_person","non_personal"]
    },
    "classification": {
      "type": "string",
      "enum": ["public","internal","confidential",
               "restricted_personal","highly_sensitive"]
    },
    "consentRequired": { "type": "boolean" },
    "consentPolicyRef": { "type": "string" },
    "allowedPurposes": {
      "type": "array", "items": { "type": "string" }, "minItems": 1
    },
    "authMethod": {
      "type": "string", "enum": ["oauth2_mtls","oauth2_jwt","oidc"]
    },
    "endpointRef": { "type": "string" },
    "publicKeyRef": { "type": "string" },
    "zkpSupported": { "type": "boolean", "default": false },
    "zkpFields": { "type": "array", "items": { "type": "string" } },
    "eventSubscriptionSupported": { "type": "boolean", "default":
false },
    "status": {
      "type": "string",
      "enum": ["active","deprecated","retired","sandbox_only"]
    },
    "deprecationDate": { "type": "string", "format": "date" },
    "retirementDate": { "type": "string", "format": "date" },
    "backwardCompatibleVersions": {
      "type": "array", "items": { "type": "string" }
    },
    "publishedAt": { "type": "string", "format": "date-time" },
    "lastUpdated": { "type": "string", "format": "date-time" }
```

```
    }
}
```

9.2 Example Metadata Record

```
{
  "datasetId": "land.record.ownership",
  "datasetName": "Land Ownership Record",
  "description": "Authoritative land ownership records from provincial
land authorities.",
  "domain": "land",
  "providerId": "punjab_land_authority",
  "providerName": "Punjab Land Records Authority",
  "schemaVersion": "1.2.0",
  "schemaRef":
"https://schemas.wasl.gov.pk/v1/land.record.ownership/1.2.0",
  "subjectType": "natural_person",
  "classification": "restricted_personal",
  "consentRequired": true,
  "consentPolicyRef": "policy-land-personal-001",
  "allowedPurposes":
["subsidy_eligibility","credit_assessment","legal_verification"],
  "authMethod": "oauth2_mtls",
  "endpointRef": "/v1/data/fetch",
  "publicKeyRef": "pk-punjab-land-01",
  "zkpSupported": false,
  "eventSubscriptionSupported": true,
  "status": "active",
  "backwardCompatibleVersions": ["1.1.0","1.0.0"],
  "publishedAt": "2026-01-15T00:00:00Z",
  "lastUpdated": "2026-04-01T00:00:00Z"
}
```

9.3 Metadata Caching and Synchronisation

Client Nodes **SHALL** maintain a local cache of metadata records. The cache **SHALL** be refreshed on startup; on cache TTL expiry (Example default: 3600 seconds); on receipt of a cache invalidation notification; and on encountering an unknown datasetId. Cached metadata bundles **SHALL** be digitally signed by the WASL Metadata Repository. Client Nodes **SHALL** verify the signature before using cached metadata.

Scenario	Outcome
Dataset not registered	Request cannot proceed — SCHEMA_DATASET_NOT_FOUND
Schema version unsupported	Reject or invoke compatibility handling
Provider inactive	Routing blocked — PROVIDER_UNAVAILABLE
Metadata signature invalid	Cache update rejected
Consent policy missing	Treat as non-routable until resolved

Classification undefined	Reject publication or block use

9.4 Zero-Knowledge Proof (ZKP) Support in Data Schemas

Selected WASL schemas may include annotations indicating that certain fields are eligible for privacy-preserving proof-based responses. Instead of transmitting the full underlying value, a provider may return a cryptographic proof of a specific predicate, such as confirmation that the subject is above a required age threshold, that a compliance condition is satisfied, or that an eligibility criterion has been met. The metadata record for a ZKP-capable schema shall indicate: whether proof-based response is supported; which fields or predicates are eligible; what proof format is expected; and whether verifier-side validation rules apply. This is an advanced data-minimization capability and is not the default response mode.

9.5 Provider Directory and Service Binding

The Metadata Repository maintains a Provider Directory that binds each dataset to the organizations authorized to serve it. For each registered provider, the directory records: provider identifier and organization name; provider type; active datasets and endpoint references; public key reference or certificate chain; operational status; supported protocol types (REST, gRPC, GraphQL, event); and region or jurisdiction where relevant. This enables a Consumer Client Node to discover not only what dataset exists but which registered provider is authorized and technically available to serve it.

9.5a Province-Code Routing Key Pattern

Where a single national schema is served by multiple provincial or regional providers (for example, a national vehicle excise transfer schema served by four provincial Excise Departments), the Provider Directory **SHALL** support province-code routing keys so that a Consumer submits one request and the Orchestration Services resolve the correct provider at runtime. This avoids requiring a separate governance approval per province and enables a single schema publication to cover all providers. The routing key pattern works as follows: (a) the domain custodian (typically a federal ministry or PDA) publishes one national schema in the Metadata Repository with domain=custom and the metadata record includes a routingKeyField specifying which request field carries the routing discriminator (example: province_code); (b) each provincial Client Node registers as a provider against the same datasetId, declaring its supported routingKeyValues (example: ["KP","PB","SD","BL"]); (c) the Orchestration Services evaluate the routingKeyField at transaction time to dispatch to the appropriate provider; (d) if no routingKeyValue matches, the API Gateway returns PROVIDER_UNAVAILABLE. One PDA governance approval covers the national schema; each provincial provider's registration is an administrative act, not a new governance approval cycle.

The routingKeyField **SHALL** be declared in the metadata record as a top-level property: {"routingKeyField": "province_code", "routingKeyValues": ["KP","PB","SD","BL"]}. Provider registrations in the directory that include routingKeyValues are treated as regional providers for that dataset. A Consumer may target a specific province by populating the routing key field in the getData request envelope, or may request fan-out to all registered provincial providers (for aggregate queries) by omitting the field; in fan-out mode, Orchestration Services dispatch concurrently and aggregate results per the composite workflow pattern (Section 26.5).

9.5b: Provider Public Key Publication (JWKS)

Every registered Provider **SHALL** publish a JWK Set (JWKS) conforming to RFC 7517. The Metadata Repository **SHALL** expose each Provider's JWKS at a well-known path derived from the Provider's registered providerId:

```
GET /v1/providers/{providerId}/jwks.json
```

Each JWK in the set **SHALL** carry the following required fields: kty (key type, EC for ECDH/ECDSA keys); use (key use — enc for encryption keys used in E2E payload encryption, sig for signing keys); crv (curve, P-256 minimum); kid (key identifier, a stable UUID or SHA-256 thumbprint of the public key); x and y (public key coordinates, base64url-encoded); and nbf and exp (not-before and not-after timestamps for key validity). A key rotation schedule **SHALL** be specified: Provider encryption keys **SHALL** be rotated at minimum annually; the outgoing key **SHALL** remain published in the JWKS with use: enc and an exp timestamp for a minimum 7-day overlap period to allow in-flight transactions to complete. Client Nodes **SHALL** re-fetch the JWKS on cache TTL expiry (default 24 hours) and on receipt of a DECRYPTION_FAILED error indicating a possible key rotation event.

10 CONSENT POLICY FORMAT AND CONSENT ARTIFACT SPECIFICATION

This section specifies the technical representation, exchange, and validation of consent and other authorization artefacts within WASL. The determination of when consent is required, the lawful bases for processing personal data, organization eligibility for non-consent authorization, and applicable exemptions are governed by the WASL Consent Governance Guidelines (DNP-D.250_GDL_v0.1) issued separately by the Pakistan Digital Authority. Implementations **SHALL** enforce the consent policies published in the Consent Policy Registry. The Consent Layer manages the full lifecycle of citizen consent for personal data exchange. This section specifies the technical formats for consent policy records and consent artifacts. The sub-clauses below provide the operational context for how consent is collected, enforced at runtime, and revoked. The substantive policy framework governing which data categories require consent, which authorization bases are recognized, and which organization classes qualify for exemption or modified consent treatment is defined in Annex E (NORMATIVE): WASL Consent Policy Framework, which is normative and **SHALL** be read in conjunction with this section.

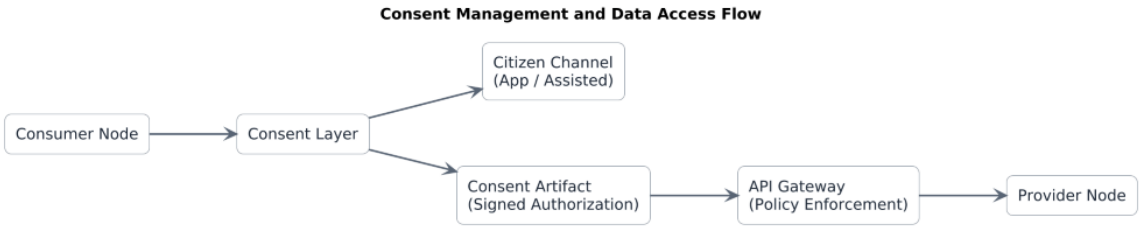

Figure 10-1: Consent Layer in WASL Architecture — consent is obtained, converted into a verifiable artifact, and enforced during data exchange

When a consumer requests data requiring citizen consent, the Consent Layer initiates an asynchronous consent collection flow. The citizen is notified through an approved channel, primarily the PAK-ID mobile application, and a later stage, possibly with USSD/IVR and in-

person assisted consent available for citizens without smartphone access, and presented with a plain-language description of what data is requested, by which organization, for what stated purpose, and for how long consent will be valid. Notifications shall be in Urdu and applicable regional languages. On citizen approval, a signed consent artifact is generated and the pending getData request proceeds. On rejection or timeout, the request is terminated with CONSENT_REQUIRED.

Citizens may revoke previously granted consent at any time through any supported consent channel. Upon revocation, the Consent Layer immediately updates the consent artifact status to REVOKED, propagates the revocation to the Consent Policy Registry, and ensures subsequent getData requests referencing the revoked consentId are rejected within 5 seconds. The revocation timestamp and reason are recorded in the consent log. Event subscriptions bound to the revoked consent are automatically terminated.

10.1 Consent Policy Record — JSON Schema

Consent policies define the rules governing whether, how, and under what conditions citizen consent is required for a given dataset, field, or purpose combination. Policies are published in the Consent Policy Registry and enforced at runtime by the Consent Layer, API Gateway, and Provider Fulfilment Layer.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/consent-policy.json",
  "title": "WASL Consent Policy Record",
  "type": "object",
  "required": ["policyId","datasetId","consentRequired",
               "authorizationBasis","fieldLevelPolicies",
               "validityModel","revocable","channels"],
  "properties": {
    "policyId": { "type": "string" },
    "datasetId": { "type": "string" },
    "version": { "type": "string" },
    "consentRequired": { "type": "boolean" },
    "authorizationBasis": {
      "type": "string",
      "enum": ["explicit_consent","implicit_consent",
               "statutory_authority","public_data","law_enforcement"]
    },
    "exemptOrgTypes": {
      "type": "array", "items": { "type": "string" }
    },
    "fieldLevelPolicies": {
      "type": "array",
      "items": {
        "type": "object",
        "required": ["fieldName","consentRequired"],
        "properties": {
          "fieldName": { "type": "string" },
          "consentRequired": { "type": "boolean" },
```

```
          "sensitivityLevel": {
            "type": "string", "enum":
["low","medium","high","critical"]
          },
          "maskingAllowed": { "type": "boolean" }
        }
      }
    },
    "validityModel": {
      "type": "object",
      "properties": {
        "maxValidityDays": { "type": "integer", "minimum": 1,
"maximum": 365 },
        "purposeBound": { "type": "boolean" },
        "renewalAllowed": { "type": "boolean" }
      }
    },
    "revocable": { "type": "boolean" },
    "channels": {
      "type": "array",
      "items": { "type": "string",
        "enum":
["pak_id_app","sms_ussd","assisted","call_center","offline_deferred"]
      }
    },
    "purposeRestrictions": { "type": "array", "items": { "type":
"string" } },
    "legislativeRef": { "type": "string" }
  }
}
```

10.2 Consent Artifact: JSON Schema

Every approved consent request **SHALL** produce a signed consent artifact. This is the authoritative evidence of citizen authorization and **SHALL** be attached to all getData requests for consent-governed datasets.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/consent-artifact.json",
  "title": "WASL Consent Artifact",
  "type": "object",
  "required": ["consentId","subjectId","consumerId","providerId",
               "datasetId","purpose","validFrom","validTo","status",
               "channel","signature"],
  "properties": {
    "consentId": { "type": "string", "format": "uuid" },
    "subjectId": { "type": "string",
      "description": "Format: <idType>:<idValue> e.g. CNIC:35201-
1234567-3" },
```

```
    "consumerId": { "type": "string" },
    "providerId": { "type": "string" },
    "datasetId": { "type": "string" },
    "fields": { "type": "array", "items": { "type": "string" },
      "description": "Field-level scope of consent. Empty = all
consented fields." },
    "purpose": { "type": "string" },
    "validFrom": { "type": "string", "format": "date-time" },
    "validTo": { "type": "string", "format": "date-time" },
    "status": {
      "type": "string",
      "enum": ["ACTIVE","EXPIRED","REVOKED","PENDING","REJECTED"]
    },
    "channel": { "type": "string" },
    "subjectSignature": { "type": "string",
      "description": "Digital signature from citizen via PAK-ID (where
applicable)." },
    "signature": { "type": "string",
      "description": "WASL platform digital signature over the consent
artifact." },
    "revocationTimestamp": { "type": "string", "format": "date-time"
},
    "revocationReason": { "type": "string" },
    "createdAt": { "type": "string", "format": "date-time" }
  }
}
```

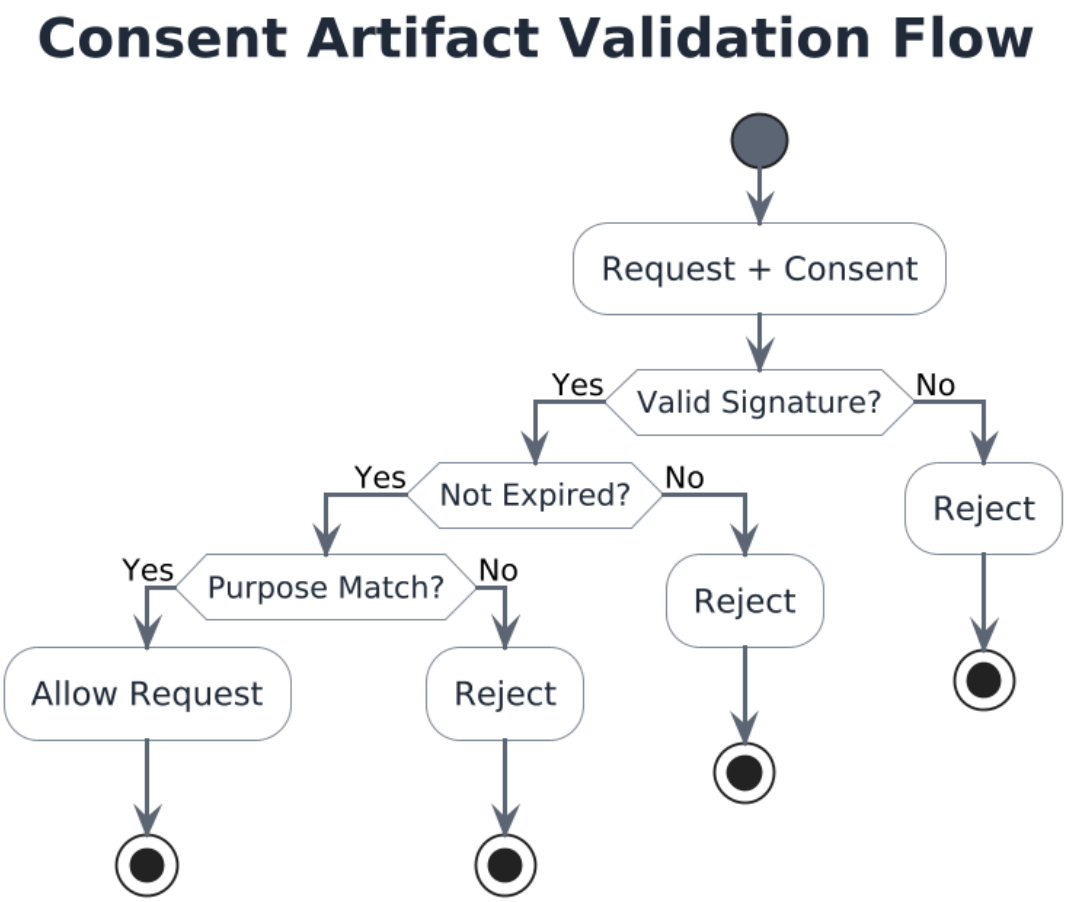

Figure 10-2: Consent Validation at Runtime, consent artifacts are validated during request processing at the API Gateway and Provider Node

10.3 Consent Validation Requirements at Runtime

Before a getData request proceeds, the following consent validations **SHALL** be performed in sequence:

Step	Check	Failure Code
1	Consent artifact present (if dataset is consent-governed)	CONSENT_REQUIRED
2	Consent artifact platform signature is valid	CONSENT_INVALID
3	consentId in artifact matches consentId in request envelope	CONSENT_MISMATCH
4	consumerId in artifact matches requesting entity org_id	CONSUMER_MISMATCH
5	providerId in artifact matches target provider	PROVIDER_MISMATCH
6	datasetId in artifact matches requested datasetId	DATASET_MISMATCH
7	purpose in artifact matches purpose in request envelope	PURPOSE_MISMATCH
8	Current timestamp is within [validFrom, validTo] window	CONSENT_EXPIRED

9	status field is ACTIVE (not REVOKED or EXPIRED)	CONSENT_REVOKED
10	Requested fields are within the consented field scope	FIELD_OUT_OF_SCOPE

11 DATA CLASSIFICATION TECHNICAL CONTROLS MAPPING

11.1 Classification Tiers

Tier	Label	Definition
1	Public	Data that may be shared without restriction. No consent required. Minimal access controls.
2	Internal	Data restricted to authorized institutions under defined use cases. No citizen consent required but organization-level access control applies.
3	Confidential	Sensitive institutional data requiring elevated access controls. Consent depends on policy; statutory authority may apply.
4	Restricted Personal	Personal data relating to natural persons. Citizen consent or statutory legal basis required. End-to-end encryption is mandatory.
5	Highly Sensitive	Data presenting heightened risk if disclosed. Enhanced controls, minimal retention, and elevated approval requirements.

11.2 Technical Controls Mapping

Control Domain	Public	Internal	Confidential	Restricted Personal	Highly Sensitive
Transport Encryption (TLS 1.3)	REQ	REQ	REQ	REQ	REQ
End-to-End Payload Encryption	—	REC	REQ	REQ	REQ
mTLS Authentication	REQ	REQ	REQ	REQ	REQ
Citizen Consent	—	—	DEPENDS	REQ	REQ (enhanced)
Purpose Binding	REC	REQ	REQ	REQ	REQ
Field-level Minimization	OPT	REC	REQ	REQ	REQ
Audit Logging	Standard	Standard	Restricted	Masked	Minimal & Masked
TPL Recording	REQ	REQ	REQ	REQ	REQ

Event Subscription Allowed	YES	Controlled	Controlled	Consent-bound	Exceptional only
Max Token Lifetime	900s	900s	600s	300s	120s
Approval Workflow	Self-service	Auto-approval	Manual review	Contractual	Exceptional approval

11.3 Field-level Classification Example: Civil Registry

Field	Classification	Consent Required	Maskable
full_name	Restricted Personal	Yes	Yes (partial)
date_of_birth	Restricted Personal	Yes	Yes (year only)
gender	Internal	No	No
nationality	Internal	No	No
cnic_number	Highly Sensitive	Yes	No
address	Confidential	Yes	Yes
marital_status	Restricted Personal	Yes	No
biometric_reference	Highly Sensitive	Exceptional only	No

12 ERROR CODES AND ERROR HANDLING

12.1 Error Response Format

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/error-response.json",
  "title": "WASL Error Response",
  "type": "object",
  "required": ["transactionId","errorCode","message","timestamp"],
  "properties": {
    "transactionId": { "type": "string" },
    "errorCode": { "type": "string" },
    "httpStatus": { "type": "integer" },
    "message": { "type": "string" },
    "detail": { "type": "string" },
    "retryAfter": { "type": "integer",
      "description": "Seconds until retry permitted (rate limit
errors)." },
    "timestamp": { "type": "string", "format": "date-time" }
  }
}
```

12.2 Authentication and Authorization Errors (4xx)

Error Code	HTTP	Description	Recovery Action

AUTH_TOKEN_MISSING	401	No access token in Authorization header.	Include valid Bearer token.
AUTH_TOKEN_INVALID	401	Token signature verification failed.	Obtain new token from IAM.
AUTH_TOKEN_EXPIRED	401	Token exp claim is in the past.	Obtain new token from IAM.
AUTH_TOKEN_REPLAY	401	Token jti has been seen before.	Obtain new token with fresh jti.
AUTH_MTLS_FAILED	401	mTLS certificate validation failed.	Verify certificate chain and OCSP status.
AUTH_CERT_REVOKED	401	Client certificate has been revoked.	Contact PDA/NADRA to reissue certificate.
AUTH_CERT_EXPIRED	401	Client certificate has expired.	Renew certificate via WASL PKI portal.
AUTHZ_SCOPE_INSUFFICIENT	403	Token scope does not permit requested operation.	Request token with required scope.
AUTHZ_PURPOSE_MISMATCH	403	Declared purpose not permitted for dataset.	Use a permitted purpose code.
AUTHZ_DATASET_NOT_SUBSCRIBED	403	Entity not subscribed to requested dataset.	Subscribe via Developer Portal.
AUTHZ_IP_BLOCKED	403	Request from unregistered IP address.	Register IP in entity profile.

12.3 Consent Errors (4xx)

Error Code	HTTP	Description	Recovery Action
CONSENT_REQUIRED	403	Dataset is consent-governed; no consent artifact provided.	Initiate consent flow and attach artifact.
CONSENT_INVALID	403	Consent artifact signature verification failed.	Obtain fresh consent artifact.
CONSENT_EXPIRED	403	Consent validTo timestamp is in the past.	Request citizen to renew consent.
CONSENT_REVOKED	403	Citizen has revoked this consent.	Request citizen to grant fresh consent.
CONSENT_PURPOSE_MISMATCH	403	Request purpose does not match consented purpose.	Use the consented purpose code.
CONSENT_FIELD_OUT_OF_SCOPE	403	Requested fields not within consent scope.	Request only consented fields.
CONSENT_CONSUMER_MISMATCH	403	Consent consumerId does not match requesting entity.	Use correct consent artifact.
CONSENT_PENDING	202	Consent request initiated; awaiting citizen decision.	Poll or subscribe for consent decision.
CONSENT_REJECTED	403	Citizen rejected the consent request.	Inform end-user; cannot proceed.

12.4 Schema and Payload Errors (4xx)

Error Code	HTTP	Description	Recovery Action
SCHEMA_DATASET_NOT_FOUND	404	datasetId not registered in Metadata Repository.	Verify dataset ID.
SCHEMA_VERSION_NOT_FOUND	404	Requested schema version does not exist.	Use current version from getSchema.
SCHEMA_VERSION_DEPRECATED	410	Requested schema version is deprecated.	Migrate to current version.
SCHEMA_VALIDATION_FAILED	422	Request payload does not conform to schema.	Validate request against schema.
REQUEST_MALFORMED	400	Request envelope is malformed JSON.	Fix JSON syntax.
REQUEST_SIGNATURE_INVALID	400	Consumer digital signature over request is invalid.	Re-sign request with entity private key.
REQUEST_TIMESTAMP_STALE	400	Request timestamp outside 5-minute skew window.	Sync clock and retry.
TRANSACTION_ID_DUPLICATE	409	transactionId has been used before.	Generate new UUID v4 transactionId.
SUBJECT_NOT_FOUND	404	No record found for the given subject identifier.	Verify subject identifier.

12.5 Provider and Platform Errors (5xx)

Error Code	HTTP	Description	Recovery Action
PROVIDER_UNAVAILABLE	503	Provider Client Node is unreachable.	Retry after retryAfter seconds.
PROVIDER_TIMEOUT	504	Provider did not respond within 30-second timeout.	Retry or use async pattern.
PROVIDER_SCHEMA_VIOLATION	502	Provider returned data not conforming to schema.	Report via PDA/NADRA support portal.
PROVIDER_SIGNATURE_INVALID	502	Provider response signature verification failed.	Do not use response; report to PDA/NADRA.
PROVIDER_INTERNAL_ERROR	502	Provider returned an internal error.	Retry; escalate if persistent.
PLATFORM_RATE_LIMITED	429	Platform rate limit exceeded.	Back off per retryAfter header.
TPL_RECORD_FAILED	500	Transaction proof could not be recorded.	Do not use response; report to PDA/NADRA.
IAM_UNAVAILABLE	503	IAM service temporarily unavailable.	Retry token request.

12.6 Error Handling Requirements

Error Category	Retry?	Max Retries	Backoff Strategy
AUTH_TOKEN_EXPIRED	Yes	1	Obtain new token, no delay
PROVIDER_UNAVAILABLE	Yes	3	Exponential: 2s, 4s, 8s

PROVIDER_TIMEOUT	Yes	2	Linear: 5s, 10s
PLATFORM_RATE_LIMITED	Yes	—	Honor retryAfter header
CONSENT_REQUIRED	No (user action needed)	—	Initiate consent flow
CONSENT_REVOKED	No (user action needed)	—	Inform user
REQUEST_SIGNATURE_INVALID	No (fix request)	—	Correct signature
PROVIDER_SIGNATURE_INVALID	No (security event)	—	Alert security team
TPL_RECORD_FAILED	No (security event)	—	Alert; do not use response

13 TRANSACTION PROOF LAYER: TECHNICAL SPECIFICATION

The Transaction Proof Layer (TPL) provides a verifiable, tamper-evident record of every data exchange transaction, recording only cryptographic proofs, not data content. This section specifies the technical formats for TPL proof records.

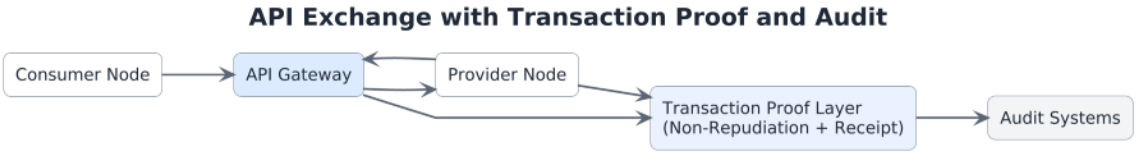

Figure 13-1: Transaction Proof Generation, request, response, and consent artifacts are hashed and combined into a signed transaction proof record

For each transaction, the TPL records: transaction identifier; consumer and provider identifiers; SHA-256 hash of the request payload (consumer-signed); SHA-256 hash of the response payload (provider-signed); reference to the consent artifact; WASL platform co-signature; transaction status; RFC 3161 trusted timestamp; and Merkle proof reference once the batch is anchored. No plaintext data, field values, or citizen identifiers are stored in the TPL.

The TPL is privacy-preserving by construction: dataset identifiers rather than full payloads are recorded; status and metadata rather than raw personal data are stored; and subject identifiers are hashed or masked per the data minimisation policy. TPL records provide mathematically unforgeable evidence that a specific transaction occurred, and what was committed to by each party, making them suitable for legal proceedings, regulatory audits, and dispute resolution.

13.1 TPL Proof Record JSON Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/tpl-proof-record.json",
  "title": "WASL TPL Proof Record",
  "type": "object",
  "required":
["transactionId","requestHash","responseHash","consentHash",

"consumerId","providerId","timestamp","cryptoSuite","tplStatus",
```

```
                "providerSignature","gatewaySignature","status"],
  "properties": {
    "transactionId": { "type": "string", "format": "uuid" },
    "requestHash": { "type": "string",
      "description": "SHA-256 of canonical request body." },
    "responseHash": { "type": "string",
      "description": "SHA-256 of response body." },
    "consentHash": { "type": "string",
      "description": "SHA-256 of consent artifact." },
    "consumerId": { "type": "string" },
    "providerId": { "type": "string" },
    "datasetId": { "type": "string" },
    "purpose": { "type": "string" },
    "timestamp": { "type": "string", "format": "date-time" },
    "tsaTimestamp": { "type": "string",
      "description": "Trusted timestamp from TSA (RFC 3161)." },
    "providerSignature": { "type": "string" },
    "gatewaySignature": { "type": "string" },
    "status": {
      "type": "string",
      "enum":
["INITIATED","VALIDATED","PROCESSING","COMPLETED","FAILED","RECORDED"]
    },
    "merkleProof": {
      "type": "object",
      "properties": {
        "treeId": { "type": "string" },
        "leaf": { "type": "string" },
        "path": { "type": "array", "items": { "type": "string" } },
        "root": { "type": "string" }
      }
    },
    "anchorRef": { "type": "string",
      "description": "External anchor reference CT-log tree ID and
leaf hash." },
    "cryptoSuite": { "type": "string",
      "description": "Cryptographic algorithm suite identifier from
the Cryptographic Parameter Registry. Required for forensic
traceability across algorithm migrations. Example: wasl-suite-v1-
ecdsa-p256-aesgcm256." },
    "tplStatus": { "type": "string",
      "enum":
["PENDING","ENDORSEMENT","COMMITTED","EXPIRED","ENDORSEMENT_FAILED"],
      "description": "TPL proof lifecycle state: PENDING \u2014
consumer proof recorded, provider pending; ENDORSEMENT \u2014 both
sides received, batch circulated; COMMITTED \u2014 quorum achieved,
immutably appended; EXPIRED \u2014 timeout; ENDORSEMENT_FAILED \u2014
quorum not achieved." },
    "endorserSignatures": { "type": "array",
```

```
      "description": "Endorser co-signatures over the Merkle batch
root. Populated when tplStatus=COMMITTED. Each entry: endorserId,
signature, timestamp.",
      "items": { "type": "object", "properties": { "endorserId":
{"type":"string"}, "signature": {"type":"string"}, "timestamp":
{"type":"string","format":"date-time"} } } }
  }
}
```

13.2 Transaction Lifecycle States

Transaction Status Lifecycle

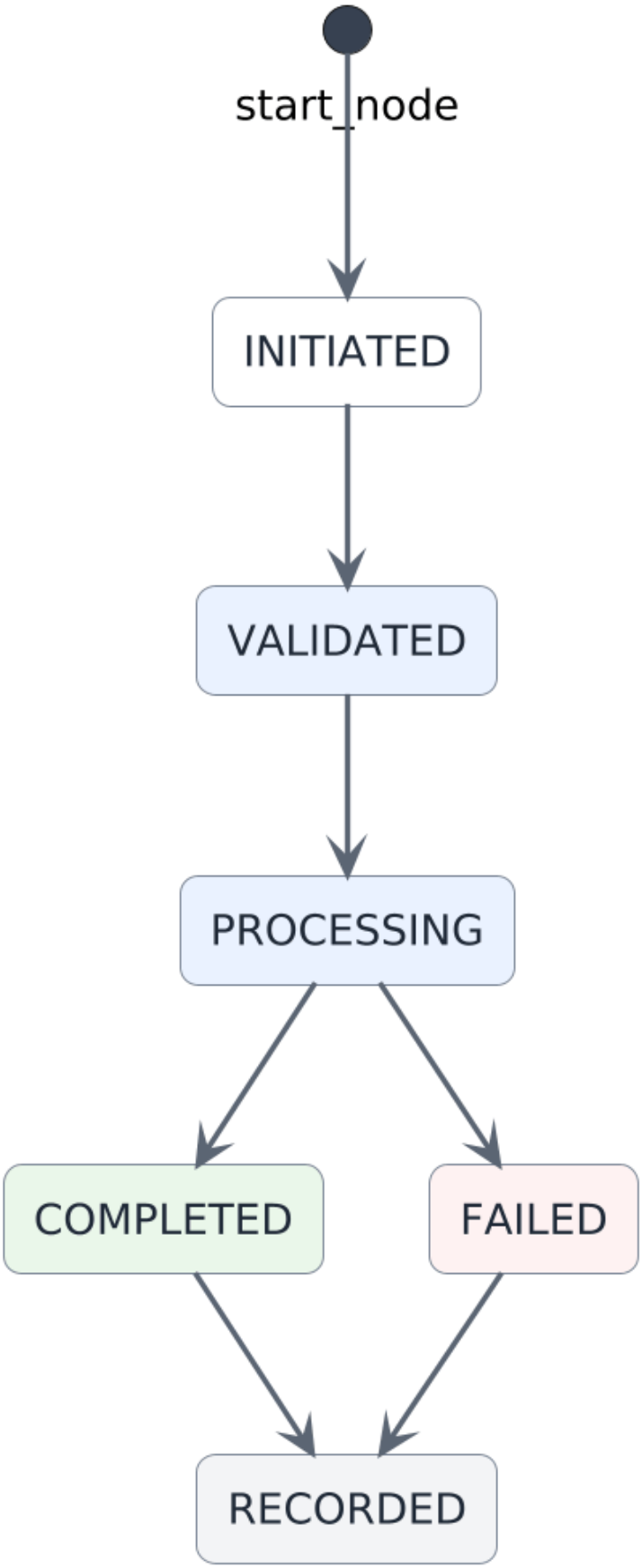

Figure 13-2: Transaction Lifecycle: a WASL transaction progresses from INITIATED through RECORDED in the Transaction Proof Layer

State	Description	Triggered by
INITIATED	Consumer has submitted getData request; Transaction ID assigned.	Consumer Client Node
VALIDATED	API Gateway has verified authentication, consent, and schema.	API Gateway
PROCESSING	Request forwarded to Provider Client Node.	Secure Connectivity Layer
COMPLETED	Provider has returned signed response.	Provider Client Node
FAILED	Error occurred at any stage.	Any component
RECORDED	Cryptographic proof written to TPL.	Transaction Proof Layer

13.3 Signature Computation

13.3.1 Consumer Request Signature

The consumer Client Node **SHALL**: (1) canonicalize the request body as compact JSON with keys sorted alphabetically; (2) compute SHA-256 of the canonical JSON bytes; (3) sign the SHA-256 digest using ECDSA-P256, Ed25519, or RSA-PSS-4096 with the entity's private key (protected per the entity's assigned Client Category, see Section 14.2a); (4) encode the signature as base64url; and (5) include in X-WASL-Signature header and the signature field of the request envelope.

13.3.2 Provider Response Signature

The provider Fulfilment Layer **SHALL**: (1) concatenate requestHash || responseBodyHash (SHA-256 of response data) || timestamp (ISO 8601 UTC); (2) sign the concatenated bytes using the provider's private key (protected per the provider's assigned Client Category, see Section 14.2a); and (3) include as providerSignature in the response envelope and in the TPL proof record.

13.4 External Anchoring (CT-Log)

The TPL **SHALL** periodically anchor Merkle roots of proof records to an externally verifiable, publicly auditable log. This provides an independent, tamper-evident external anchor for all TPL proof records. Proof records are batched into Merkle trees at configurable intervals (default: every 1,000 records or 1 hour, whichever occurs first). The Merkle root hash is submitted to the designated CT-log service. The anchor reference (tree ID and leaf hash) is stored in the anchorRef field of each proof record in the batch. The external anchoring mechanism is technology-neutral at this specification level, consistent with RA Section 11.4.3. The CT-log approach is the authoritative baseline. Alternative anchoring constructions, including Trillian-style federated append-only logs with multi-party witness signatories or purpose-built multi-signature Merkle-tree constructions, may be evaluated and confirmed in downstream operational documentation provided they satisfy the append-only, cryptographic-consistency, and independently-auditable requirements. No specific distributed ledger or blockchain technology is mandated at this level.

13.5 Federated Endorsement Model

The TPL is operated as a federated, multi-party, append-only ledger. PDA is one endorser among several. A TPL batch is considered Committed only when a policy-defined quorum of endorsers has countersigned it. This section specifies the technical integration requirements for Client Nodes and the platform; the governance parameters (endorser roster, quorum threshold, legal accountability) are defined in the companion TPL Federation Regulation to be issued as a REG under the DNP-D series prior to WASL production launch.

13.5.1 Endorsement Flow

The TPL endorsement flow proceeds through the following stages for each batch: (1) Batch assembly: the TPL aggregates proof records into a Merkle tree on the configured interval trigger; (2) Root signing request: the TPL broadcasts the Merkle root hash and batch metadata to all registered endorser nodes; (3) Endorser signing: each endorser node independently verifies the batch integrity and countersigns the root hash using its HSM-protected endorser key (FIPS 140-2/3 Level 3 mandatory for endorsers per RA Section 14.2); (4) Quorum collection: the TPL collects endorser signatures until the policy-defined quorum threshold is achieved; (5) Commitment: the batch is marked COMMITTED and the endorserSignatures array in each proof record is populated; (6) CT-log anchoring: the committed batch Merkle root is submitted to the external CT-log per Section 13.4.

13.5.2 Lifecycle State Transitions

The tplStatus field (defined in Section 13.1) tracks the proof lifecycle independently of the transaction operational status. State transitions: PENDING (consumer-side proof recorded; provider-side not yet received) to ENDORSEMENT (both sides received; Merkle batch assembled and circulated to endorsers) to COMMITTED (quorum achieved; batch immutably appended and CT-log anchored). Failure paths: PENDING records unresolved beyond a configurable timeout (default: 10 minutes) transition to EXPIRED and are themselves recorded as evidence of a non-completion event. If a quorum cannot be achieved within the configured endorsement window (default: 30 minutes), the batch transitions to ENDORSEMENT_FAILED; this event is flagged for operational investigation and does not silently disappear. Neither EXPIRED nor ENDORSEMENT_FAILED records are deleted; both remain in the TPL as permanent evidence.

13.5.3 Endorser Node Technical Requirements

Each TPL endorser node **SHALL**: maintain a persistent connection to the WASL TPL service over mTLS (per Section 5.2); hold its endorser signing key in a FIPS 140-2/3 Level 3 HSM (mandatory; no software keystore exception); respond to signing requests within the operational latency commitment specified in the TPL Federation Regulation; maintain its own local copy of all committed batch records for independent auditability; implement key rotation procedures per the TPL Federation Regulation; and expose a public verification endpoint at which any party may retrieve endorser public keys to independently verify committed batch signatures. Endorser availability SLA is defined in the TPL Federation Regulation; the technical minimum is 99.5% measured monthly. An endorser that fails to respond within the signing window for three consecutive batches is marked inactive and the remaining endorsers continue toward quorum if the threshold remains achievable.

14 SECURITY ARCHITECTURE: TECHNICAL SPECIFICATION

This section specifies the cryptographic algorithm requirements, PKI architecture, end-to-end encryption protocol, and threat mitigation controls for WASL. The security architecture implements a Zero Trust model combined with a three-party cryptographic trust framework: the consumer independently verifies the provider's response signature using the locally cached provider public key; the provider independently verifies the consumer's request signature; and both parties verify WASL-issued consent artifacts using the platform's public key. Neither party needs to extend blind trust to the central platform for data integrity.

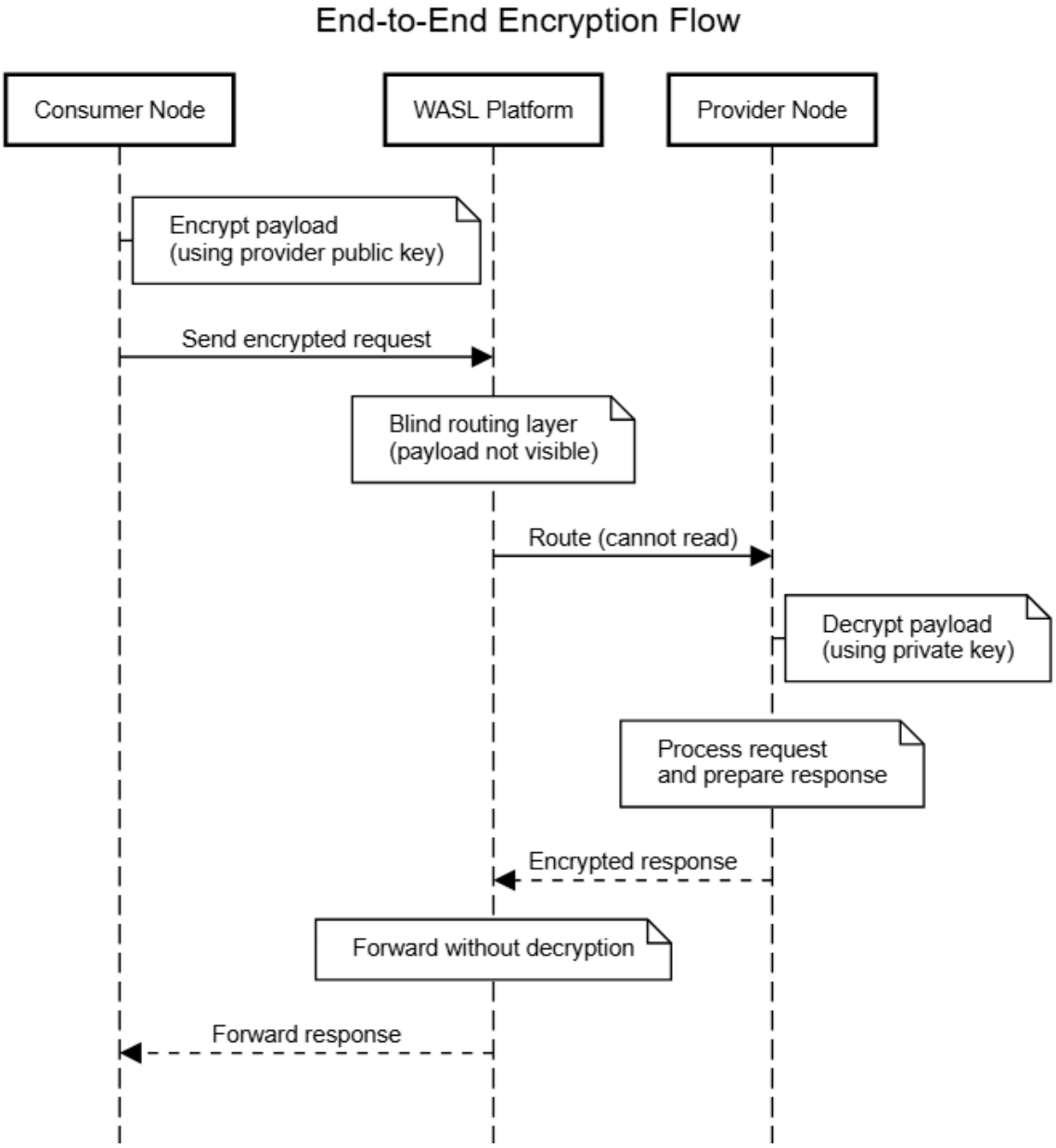

Figure 14-1: End-to-End Encryption and Data Blindness, WASL routes encrypted payloads without accessing underlying data

14.1 Cryptographic Algorithm Requirements

Function	Required Algorithm	Key Size	Standard	Min PSS CP Grade

Digital Signatures (Entity)	ECDSA with P-256	256-bit	FIPS 186-5	Level B (CAT-C/D only, see Section14.2a(iv))
Digital Signatures (CA)	ECDSA with P-384 or RSA-PSS	384-bit / 4096-bit	FIPS 186-5	Level B / Level C
JWT Signing (IAM)	ES256 (ECDSA P-256) or RS256	256-bit / 2048-bit min	RFC 7518	Level B (ES256) /Level A (RS256)
Symmetric Encryption	AES-256-GCM	256-bit	FIPS 197	Level C (satisfies all categories)
Session Key Encapsulation	ECDH-ES (P-256) + AES-256-GCM	256-bit	RFC 7518	Level B (ES256) /Level C (AES-256-GCM)
Hash Function	SHA-256 minimum; SHA-384 preferred	256/384-bit	FIPS 180-4	Level A (SHA-256)/Level B (SHA-384)
TLS Cipher Suite	TLS_AES_256_GCM_SHA384	256-bit	RFC 8446	Level C (AES-256) /Level B (SHA-384)
Key Protection (Entity)	Key Protection: per Client Category (Section 14.2a): CAT-A (TPL Endorsers): FIPS 140-2/3 Level 3 HSM mandatory. CAT-B (high-risk Tier 4/5 providers): HSM strongly recommended. CAT-C/D: software keystore acceptable with minimum controls per Section 14.2a. IAM platform signing keys: FIPS 140-2/3 Level 3 HSM mandatory regardless of entity category.	N/A	FIPS 140-2/3	N/A, Key Protection is a CE Security Level control, not a CP Grade control.

14.2 PKI Architecture

Component	Role	Key Storage
Root CA	Trust anchor for Government Sector will be DCS while for all other entities, it will be ECAC. Offline, air-gapped. Issues Intermediate CA certs only.	FIPS 140-2/3 Level 4 HSM
Intermediate CA (Entity Signing)	Issues entity signing certificates.	FIPS 140-2/3 Level 3 HSM
Intermediate CA (VPN Device)	Issues VPN device certificates for IPSec.	FIPS 140-2/3 Level 3 HSM
Registration Authority, Government (RA-GOV	Validates identity of government Participating Entities before certificate issuance under the DCS root CA. Operated by the Department of Communication Security (DCS).	Software-based with MFA
Registration Authority, Non-Government (RA-PRIV)	Validates identity of regulated private-sector and other non-government Participating Entities before certificate issuance under the ECAC root CA. Operated under arrangement between PDA and the Electronic Certification Accreditation Council (ECAC).	Software-based with MFA

Validation Authority (VA)	Provides OCSP responses and CRL.	HSM-protected signing key
Timestamping Authority (TSA)	Issues RFC 3161 timestamps for TPL proof records.	FIPS 140-2/3 Level 3 HSM

14.2a Key Management: Client Category Framework

Per RA Section 14.2, private key protection requirements are risk-differentiated by entity type and the data tiers served. WASL defines four Client Categories. An entity's Client Category is determined at onboarding (see Section 18), recorded in the Provider Directory, and must be declared in the PKI Certificate Signing Request. Category upgrades trigger automatically when dataset subscriptions change; downgrades require approval.

CAT-A TPL Federated Endorsers

Applies to: designated TPL endorser institutions. Key protection: FIPS 140-2/3 Level 3 Hardware Security Module, mandatory. No software keystore exception is permitted for TPL endorser signing keys. This requirement applies only to the endorser role; the same institution's participation as a Data Provider or Consumer is governed by its applicable data-role category below.

CAT-B High-Risk Data Providers and Consumers

Applies to: entities serving or consuming Tier 4 (Restricted Personal) or Tier 5 (Highly Sensitive) data as a primary Provider; includes NADRA (civil registry), FBR (tax), provincial health departments, provincial land record authorities, and SECP-regulated financial data custodians. Key protection: FIPS 140-2/3 Level 3 HSM, strongly recommended. A CAT-B entity that cannot deploy an HSM **SHALL** submit a written derogation request documenting compensating controls; Approval is required before production activation. Approved compensating controls must satisfy all requirements in Section 14.2a(ii) plus additional controls agreed with PDA.

CAT-C Standard Data Providers and Consumers

Applies to: regulated financial institutions, federal ministries, and provincial agencies serving Tier 2 (Internal) or Tier 3 (Confidential) data, or consuming Tier 4 data without holding the authoritative source. Key protection: software keystore acceptable, subject to minimum controls in Section 14.2a(ii). HSM adoption is strongly recommended where Tier 4 data is routinely served; HSM for CAT-C entities upon evidence of elevated operational risk maybe required.

CAT-D Low-Risk and Public Data Consumers

Applies to: research institutions, public-interest analytics consumers, and government agencies consuming only Tier 1 (Public) or Tier 2 (Internal) data; developer sandbox participants. Key protection: software keystore acceptable. HSM is not required but is never prohibited.

Minimum Controls for Software Keystores (CAT-C and CAT-D)

Where a software keystore is used, the following minimum controls **SHALL** apply: (a) Encryption at rest, private key material **SHALL** be encrypted using AES-256-GCM; the encryption key **SHALL** be stored separately from the private key, preferably in a dedicated

secrets management service or platform KMS; (b) Access control, key access **SHALL** be restricted to the WASL Client Node process identity; no human operator **SHALL** have direct programmatic access to the private key in production; (c) Audit logging, all key access events (use, rotation, deletion) **SHALL** be recorded and correlated with WASL transaction identifiers; (d) Key rotation, keys **SHALL** be rotated at least annually or immediately upon suspected compromise; rotation events **SHALL** be reported through the entity-notification channel; (e) Infrastructure isolation, the Client Node runtime **SHALL** be isolated from general-purpose workloads (container-level isolation with seccomp/AppArmor profiles as minimum); (f) Secrets management platform, use of a FIPS-validated software KMS or dedicated secrets manager is strongly recommended for CAT-C entities.

IAM Platform Key Management

The IAM platform's own signing keys (used to sign access tokens and consent artefacts) are subject to a separate mandatory HSM requirement per Section 6.3 (FIPS 140-2/3 Level 3 minimum). This requirement applies to the platform operator regardless of the Client Category system, which governs only Participating Entity key management.

14.2a(iv): PSS Security Level and CP Grade Requirements by Client Category

In accordance with Pakistan Security Standards, each WASL Client Category is assigned a minimum PSS Cryptographic Equipment (CE) Security Level and Cryptographic Primitive (CP) Security Grade as follows. These are floor requirements; entities may implement at a higher level.

Client Category	Entity Examples	Min PSS CE Level	Min PSS CP Grade	Rationale
CAT-A — TPL Endorsers	Designated statutory endorser institutions	CE Level 4	CP Grade C	Highest assurance required; underpins non-repudiation of national transaction record
CAT-B — High-Risk Providers/Consumers	NADRA, FBR, provincial health depts, land record authorities, SECP-regulated financial custodians	CE Level 3	CP Grade A	Tier 4/5 personal data; elevated physical security, identity-based auth, and stronger primitives required
CAT-C — Standard Providers/Consumers	Federal ministries, regulated financial institutions, provincial agencies (Tier 2–3 data)	CE Level 2	CP Grade B	Standard regulated context; tamper-evident physical controls and Grade A primitives sufficient
CAT-D — Low-Risk/Public Consumers	Research institutions, analytics consumers, Tier 1–2 only, developer sandbox	CE Level 1	CP Grade A	Minimum baseline; basic security requirements apply

Where a single entity holds multiple roles (e.g., a CAT-A TPL Endorser that is also a CAT-B Data Provider), the highest applicable category's cryptographic requirements apply to all its WASL signing keys.

14.3 End-to-End Encryption Protocol

For Tier 4 and Tier 5 data, the following end-to-end encryption protocol **SHALL** be used between Consumer and Provider Client Nodes:

Step	Operation	Technical Detail
1	Consumer obtains provider public key	Retrieved from local cache or Metadata Repository (provider JWK)
2	Consumer generates ephemeral key pair	ECDH ephemeral key pair (P-256)
3	Compute shared secret	ECDH key agreement: consumer ephemeral private key + provider public key
4	Derive symmetric session key	HKDF-SHA256 over shared secret; context: transactionId + datasetId
5	Encrypt payload	AES-256-GCM with derived key; 96-bit random IV in JWE header
6	Construct JWE	JOSE JWE compact serialisation (RFC 7516); alg: ECDH-ES, enc: A256GCM
7	Transmit via WASL	Encrypted JWE routed through Secure Connectivity Layer; WASL cannot decrypt
8	Provider decrypts	Provider Client Node derives session key and decrypts using its private key
8a	Provider verifies HKDF context binding	Before decryption, Provider **SHALL** verify that the transactionId and datasetId values in the JWE protected header match those in the outer request envelope. If they do not match, the Provider **SHALL** return DECRYPTION_FAILED and record a TPL anomaly event.
9	Provider encrypts response	The response payload **SHALL** be encrypted using the same session key derived in Step 4, with a fresh 96-bit random IV. The Consumer decrypts using the session key it derived at Step 4. No additional key exchange is required for the response.

The Consumer **SHALL** include transactionId and datasetId as additional members in the JWE protected header so that the Provider can verify the HKDF derivation context before and independently of decryption. These values bind the session key to the specific transaction and prevent cross-transaction key reuse. The session key derived in Step 4 is valid for the duration of the transaction (request + response pair) only. A new ephemeral key pair **SHALL** be generated for each new transaction

14.3a Cryptographic Agility and Post-Quantum Cryptography Roadmap

Per RA Section 6 (Core Design Principle: Cryptographic Agility and Post-Quantum Readiness) and RA Section 14, WASL implements cryptographic agility from v1. Algorithm suites are negotiated per session using parameters published in the Cryptographic Parameter Registry within the Metadata Repository. This isolates cryptographic migration

from protocol migration; PQC algorithms can be introduced, preferred, and eventually mandated without redesigning the protocol. Each TPL proof record carries the cryptoSuite identifier (Section 13.1) for forensic traceability across algorithm migrations. Where Pakistan Security Standards (PSS) specify requirements, PSS prevails per RA Section 14.1.1.

Cryptographic Parameter Registry

The Cryptographic Parameter Registry is a sub-component of the Metadata Repository. Client Nodes **SHALL** retrieve and cache the current registry on startup and on TTL expiry. The registry is digitally signed; Client Nodes **SHALL** verify the signature before use. The registry specifies: (a) permitted signature algorithms with minimum key sizes; (b) permitted key encapsulation algorithms; (c) permitted symmetric encryption algorithms; (d) deprecated algorithms (verifiable for historical records, not permitted for new issuance); (e) forbidden algorithms; and (f) effective dates for each rule. Per-session negotiation: Consumer and Provider Client Nodes **SHALL** select the strongest mutually supported suite from the registry. The chosen suite identifier **SHALL** be included in the request envelope and recorded in the TPL cryptoSuite field.

Stage 1 Algorithm Suite Additions (v1 Launch)

The Stage 1 permitted suite (table in Section 14.1) is extended as follows: (a) Ed25519 (RFC 8032) is added as a permitted signature algorithm alongside ECDSA P-256; (b) ChaCha20-Poly1305 is added as a permitted symmetric cipher alongside AES-256-GCM; (c) RSA keys below 3072 bits are deprecated for new issuance (RSA-2048 existing certificates remain valid until their scheduled expiry); (d) SHA-3-256 is added as a permitted hash function. Forbidden for new issuance: RSA PKCS#1 v1.5 signatures; MD5; SHA-1; RSA keys below 2048 bits; ECDSA on curves below P-256; pre-TLS 1.3 transport.

Stage 2 — Hybrid PQC Mode (Planned)

Stage 2 introduces hybrid classical + post-quantum algorithm suites. The following NIST-standardized PQC algorithms are targeted for integration into the Cryptographic Parameter Registry at Stage 2 activation (timeline confirmed by PDA in alignment with Pakistan Security Standards): NIST FIPS 204 (ML-DSA), for entity and TPL endorser digital signatures; NIST FIPS 203 (ML-KEM), for session key encapsulation; NIST FIPS 205 (SLH-DSA), as a stateless hash-based signature alternative for high-assurance contexts. In hybrid mode, both classical and PQC components are applied; a transaction is secure only if both remain unbroken. Stage 3 targets full migration to pure PQC aligned with PSS guidance. Client Node implementations **SHALL** implement the cryptographic agility interfaces from v1 so that Stage 2 adoption does not require Client Node redesign.

14.3b: Payload Protection by Data Classification Tier

Data Classification Tier	E2E Payload Encryption	Basis
Tier 1 - Public	Not required. Payload transmitted as plaintext JSON over TLS 1.3.	No confidentiality obligation; TLS transport security is sufficient
Tier 2 - Internal	Not required by default. Payload transmitted as plaintext JSON over TLS 1.3. Dataset policy record MAY designate E2E encryption as required; if so, Section 14.3 protocol applies	Organisation-level access control; no citizen PII
Tier 3 - Confidential	Required where the dataset policy record designates the encryptionRequired flag as true.	Dataset-dependent; may involve sensitive organisational data

	Plain payload over TLS 1.3 permissible only where the dataset policy record explicitly designates encryptionRequired: false Where E2E encryption is required then Section 14.3 protocol applies	
Tier 4 – Restricted Personal	Mandatory. Section 14.3 protocol **SHALL** be used for all transactions	Contains personal data; data blindness guarantee required
Tier 5 – Highly Sensitive	Mandatory. Section 14.3 protocol **SHALL** be used for all transactions	Highest sensitivity; cryptographic data blindness non-negotiable

14.4 Threat Mitigation Controls

Threat	Control Implemented
Man-in-the-Middle	mTLS at application layer + IPSec VPN at network layer
Token Replay Attack	JTI uniqueness enforcement + short token lifetimes (max 15 minutes)
Request Replay	Timestamp skew validation (5 min) + transactionId uniqueness enforcement
Signature Forgery	Private keys protected per Client Category (Section 14.2a) + ECDSA/Ed25519/RSA-PSS on all requests/responses
Data Interception (WASL)	End-to-end payload encryption for Tier 4/5 — WASL sees only ciphertext
Unauthorized Access	OAuth 2.1 + mTLS + RBAC + dataset subscription model + IP allowlisting
Consent Bypass	Consent validation independently at API Gateway AND Provider Client Node
DDoS / API Abuse	Rate limiting per entity + WAF + anomaly detection + circuit breakers
CA Compromise	Offline Root CA + HSM + multi-person access + Certificate Transparency logs
Insider Threat (PDA)	Data blindness: WASL cryptographically cannot read exchanged data
Audit Tampering	Append-only logs + Merkle tree integrity + external CT-log anchoring + federated endorsement quorum

14.5: PSS Certification Obligations

WASL Client Node implementations that incorporate cryptographic hardware or software modules claiming security functionality are subject to PSS evaluation under PS: 5544-2021 (PSS-GB-CRYPTOSec), administered by NTISB and evaluated through PNAC-accredited CEVaL laboratories.

The following obligation applies:

Prior to being deployed/used any software or hardware security/cryptographic component either dealing with PII or being used in Critical Information infrastructure will undergo PSS evaluation through NTISB.

15 WASL CLIENT NODE: TECHNICAL SPECIFICATION

The WASL Client Node is the primary execution and trust enforcement component deployed within each participating organization's own infrastructure. All cryptographic operations, signing, verification, encryption, decryption, occur within the Client Node, never at the central platform. This section specifies the deployment requirements, local API specification, Fulfilment Layer verification sequence, event management, and gRPC transport binding for conformant Client Node implementations.

The Client Node comprises: (a) Secure Connectivity - firewall and IPSec VPN providing encrypted, mutually authenticated connectivity with mTLS at the application layer; (b) Local Storage: cached dataset schemas, provider public keys, and event subscription records; (c) Signature Verification and Encryption Services: the cryptographic core implementing the three-party trust model per Section 14; (d) Local APIs: integration surface abstracting all WASL protocol complexity from the entity's internal applications; (e) Fulfilment Layer (provider-side), specified in Section15.3; and (f) Event Management, specified in Section 15.4.

The Consumer Client Node's Local APIs are callable by an AI Agent operating within the consuming organization's own infrastructure. An AI-driven workflows such as automated loan underwriting, benefits eligibility determination, or fraud detection, accesses WASL data through the same authenticated, consent-gated, TPL-recorded path as any human-initiated request. From WASL's perspective an AI Agent is indistinguishable from any other authorized consumer application: it must authenticate via OAuth 2.1, all requests are subject to consent requirements, and all transactions are recorded in the TPL with full non-repudiation. No special AI-specific access privileges exist.

15.1 Deployment Requirements

Requirement	Specification
Deployment Model	Containerized microservices. Official OCI-compliant images published to WASL Container Registry.
Orchestration	Kubernetes (minimum version 1.27).
Network	Dedicated interface for WASL VPN tunnel. Firewall-enforced ingress/egress.
HSM Integration	PKCS#11 interface **SHALL** be implemented for any entity using a hardware HSM. For CAT-C/D entities using software keystores, PKCS#11 is not required but a standards-compliant key management interface **SHOULD** be used. Software keystore in sandbox is acceptable for all categories. See Section 14.2a for category-specific requirements.
High Availability	Minimum 2 replicas per service component in production
Key Storage	Key protection is determined by the entity's assigned Client Category (Section 14.2a). CAT-A (TPL Endorsers): FIPS 140-2/3 Level 3 HSM — mandatory for endorser signing keys. CAT-B (high-risk Tier 4/5 providers): FIPS 140-2/3 Level 3 HSM, strongly recommended; derogation requires written approval with compensating controls. CAT-C (standard providers/consumers, Tier 2–4): software keystore with minimum controls per Section 14.2a(ii) acceptable. CAT-D (low-risk/public data consumers, Tier 1–2 only): software keystore — acceptable. The entity's Client Category is recorded in its registered

	profile and declared in the PKI Certificate Signing Request at onboarding (Section 18.2).

15.2 Local API Specification

Local API	Method	URI	Description
fetchData	POST	/local/v1/data/fetch	Initiates end-to-end getData flow. Returns schema-validated, signature-verified response.
getLocalSchema	GET	/local/v1/schema/{datasetId}	Returns cached schema. Refreshes from central repository if stale.
initiateConsent	POST	/local/v1/consent/initiate	Triggers consent collection flow.
getConsentStatus	GET	/local/v1/consent/{consentId}/status	Returns local cache of consent artifact status.
subscribeEvent	POST	/local/v1/events/subscribe	Registers event subscription for a subject/dataset.
getNodeStatus	GET	/local/v1/health	Returns Client Node health and connectivity status.

15.3 Fulfilment Layer: Verification Sequence

The Fulfilment Layer executes the following 12-step sequence for every incoming getData request. No step may be skipped:

Step	Action	Failure Outcome
1	Receive request from Secure Connectivity Layer	SCL_ROUTING_ERROR
2	Verify consumer digital signature on request	REQUEST_SIGNATURE_INVALID
3	Verify WASL consent artifact platform signature	CONSENT_INVALID
4	Validate consent: status, expiry, scope, purpose	CONSENT_EXPIRED / CONSENT_REVOKED
5	Decrypt request payload (if E2E encrypted)	DECRYPTION_FAILED
6	Validate request against registered schema	SCHEMA_VALIDATION_FAILED
7	Call internal backend system API	PROVIDER_INTERNAL_ERROR
8	Validate response against published schema	PROVIDER_SCHEMA_VIOLATION
9	Sign response with provider private key (HSM)	SIGNING_FAILED
10	Encrypt response (if E2E encryption was requested)	ENCRYPTION_FAILED
11	Submit TPL proof record to central TPL	TPL_RECORD_FAILED

12	Return signed, encrypted response to consumer via SCL	—

15.4 Event Management Technical Specification

15.4.1 Event Subscription Request

```
POST /local/v1/events/subscribe HTTP/1.1
Content-Type: application/json
{
  "subscriptionId": "sub-edu-health-001",
  "consumerId": "edu_dept_punjab",
  "providerId": "prov_health_registry",
  "datasetId": "health.immunization",
  "subjectId": "CNIC:35201-1234567-3",
  "fields": ["immunizationStatus","lastDoseDate"],
  "consentId": "cons-99821-abcd-ef01",
  "callbackUrl": "https://api.edu.punjab.gov.pk/wasl/events",
  "expiresAt": "2026-04-30T23:59:59Z"
}
```

15.4.2 Event Notification Format

```
{
  "eventId": "evt-20260418-001234",
  "subscriptionId": "sub-edu-health-001",
  "eventType": "DATA_CHANGED",
  "datasetId": "health.immunization",
  "subjectId": "CNIC:35201-1234567-3",
  "changedFields": ["immunizationStatus"],
  "timestamp": "2026-04-18T14:30:00Z",
  "providerSignature": "MEUCIQDzEvent...",
  "consentRef": "cons-99821-abcd-ef01"
}
```

15.4.3: Key Rotation Handling for Active Event Subscriptions

When a Provider rotates its public encryption key while active event subscriptions exist, the following procedure **SHALL** apply:

The Provider **SHALL** publish the new key to its JWKS with a new kid value before retiring the old key. During the overlap period (minimum 7 days per Section 9.5b), the Provider **SHALL** continue to accept subscription-management requests encrypted under either the old or new key, disambiguating by kid. For outbound event notifications sent after the rotation effective date, the Provider **SHALL** encrypt using the new key and **SHALL** include the new kid in the notification envelope. Consumer Client Nodes that receive an event notification with an unrecognised kid **SHALL** trigger a JWKS re-fetch for that Provider before attempting decryption. If decryption fails after a re-fetch, the Consumer Client Node **SHALL** return a SUBSCRIPTION_DECRYPTION_FAILED error and the subscription **SHALL** enter a

KEY_MISMATCH suspended state pending Consumer re-acknowledgement. The KEY_MISMATCH state **SHALL** be surfaced in the getConsentInfo and subscription-status endpoints so that the Consumer's operations team is notified. Subscriptions in KEY_MISMATCH state do not require the citizen to re-consent; they require only a Consumer-side technical re-acknowledgement through the subscribeEvent Local API.

15.5 gRPC Transport Binding

For high-throughput, low-latency integrations, WASL supports gRPC as an alternative transport. The gRPC service definition (proto3) for the WASL getData service is as follows:

```
syntax = "proto3";
package wasl.v1;
import "google/protobuf/timestamp.proto";
service WASLDataExchange {
rpc GetData (GetDataRequest) returns (GetDataResponse);
rpc StreamEvents (EventSubscription) returns (stream DataEvent);
}
message GetDataRequest {
string transaction_id = 1;
string dataset_id = 2;
string schema_version = 3;
SubjectIdentifier subject = 4;
string purpose = 5;
string consent_id = 6;
RequesterInfo requester = 7;
google.protobuf.Timestamp timestamp = 8;
bytes signature = 9;
bytes encrypted_payload = 10;
}
message SubjectIdentifier {
string id_type = 1;
string id_value = 2;
}
message RequesterInfo {
string org_id = 1;
string client_id = 2;
}
```

16 END-TO-END TRANSACTION FLOW

This section describes the complete lifecycle of a data exchange transaction in WASL, from the consumer application's initial call to receipt of verified, schema-validated data.

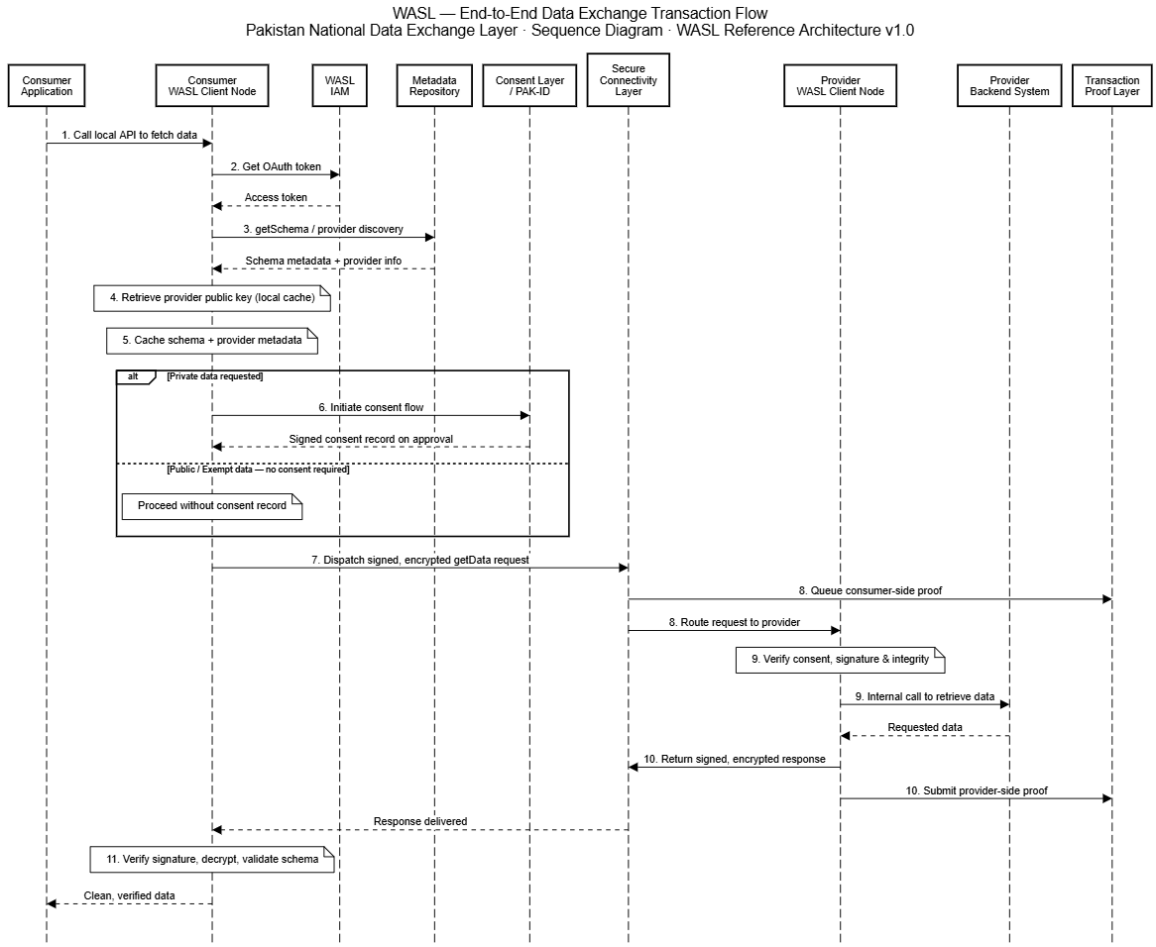

Figure 16-1: WASL End-to-End Data Exchange Transaction Flow, complete lifecycle from consumer request initiation through authentication, metadata discovery, consent handling, secure routing, provider verification, data retrieval, and response delivery

This section describes the complete lifecycle of a WASL data exchange transaction from the consumer application's initial call through to receipt of verified, schema-validated data. Section 16.1 provides the normative summary table; Section16.2 specifies the Secure Connectivity Layer transport envelopes.

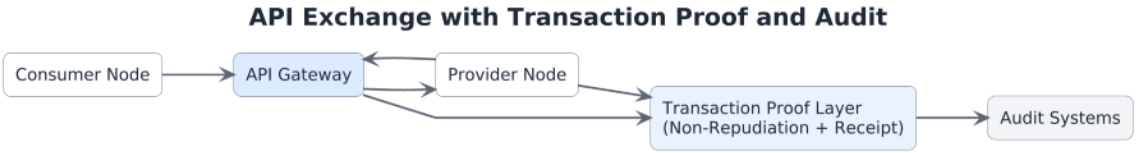

Figure 16-2: WASL E2E Encryption and Transaction Signing Flow: Key Derivation, Payload Encryption, Signature Computation, and TPL Recording across Consumer Client Node, WASL Platform, and Provider Client Node (Sections 14.3, 13.3, 15.3)

16.1 Transaction Steps

Step	Action	Key Technical Operation
1	Consumer application calls local API on its Client Node	Provides subject identifier and dataset ID only; Client Node handles all protocol complexity

2	Client Node requests OAuth 2.1 access token from IAM	private_key_jwt client assertion; mTLS connection to IAM Token Endpoint
3	Schema and provider discovery	Client Node checks local cache; calls getSchema if stale; identifies target provider
4	Retrieve provider public key	From local cache or Metadata Repository; required for payload encryption and response signature verification
5	Cache metadata locally	Reduces central platform load for subsequent requests
6	Initiate consent (if data is Tier 4/5)	Consent Layer contacted; citizen notified via PAK-ID; signed consent artifact returned on approval
7	Assemble and dispatch signed, encrypted request	Consumer generates ephemeral ECDH P-256 key pair; derives session key K1 via HKDF-SHA256 bound to transactionId and datasetId; encrypts payload using AES-256-GCM with K1 and a 96-bit random IV; packages as JWE compact serialisation (RFC 7516; alg: ECDH-ES, enc: A256GCM); signs SHA-256 hash of canonical request body with consumer private key; attaches consent artefact; dispatches via Secure Connectivity Layer.
8	Route and record (consumer-side)	Secure Connectivity Layer routes to provider; consumer-side proof elements queued for TPL recording
9	Provider verification and backend call	Fulfilment Layer verifies consumer digital signature; verifies consent artefact platform signature; validates consent status, expiry, scope, and purpose; verifies HKDF context binding (transactionId and datasetId in JWE header must match outer envelope); derives K1 via HKDF-SHA256 from ECDH shared secret; decrypts payload using AES-256-GCM with K1; validates decrypted request against published schema; calls internal backend API.
10	Provider signs and encrypts response	Response validated against published schema; SHA-256 response hash computed; provider signs concatenation of requestHash, responseHash, and timestamp with provider private key (HSM-protected); response encrypted using same session key K1 with a fresh 96-bit random IV (AES-256-GCM); provider-side TPL proof record submitted to central TPL simultaneously with response dispatch.
11	Consumer verification	Consumer Client Node verifies provider signature; decrypts

		payload; validates against schema; returns clean data to application

16.2 Secure Connectivity Layer: Transport Envelopes

The Secure Connectivity Layer transports the following data for each transaction. Business payloads are encrypted, and the platform cannot read their content.

16.2.1 Consumer Request Transport Envelope

```
{
  "transactionId": "txn-2026-00001",
  "consumerId": "edu_dept_punjab",
  "providerId": "prov_health_registry",
  "requestDigest": "sha256:abc123...",
  "consentRef": "cons-99821-abcd-ef01",
  "timestamp": "2026-04-18T10:00:00Z",
  "payloadFormat": "encrypted-json",
  "routing": { "protocol": "https", "targetNode": "prov-health-node-
01" }
}
```

16.2.2 Provider Response Transport Envelope

```
{
  "transactionId": "txn-2026-00001",
  "providerId": "prov_health_registry",
  "consumerId": "edu_dept_punjab",
  "requestDigest": "sha256:abc123...",
  "responseDigest": "sha256:def456...",
  "timestamp": "2026-04-18T10:00:02Z",
  "deliveryStatus": "DELIVERED",
  "payloadFormat": "encrypted-json"
}
```

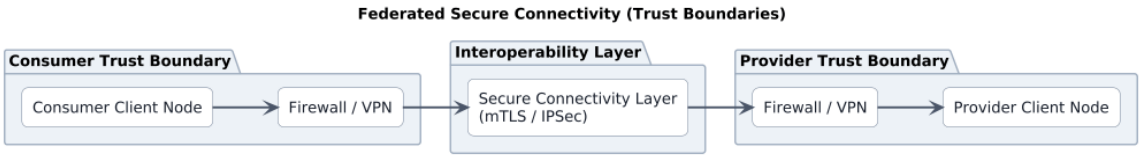

Figure 16-2: Layered Transport Security Model, application-level signing and encryption combined with IPSec VPN secure transport channels

17 CONFORMANCE TEST REQUIREMENTS

All WASL Client Node implementations, Central Platform implementations, and Fulfilment Layer implementations **SHALL** pass the conformance tests defined in this section in the certified WASL sandbox environment before production onboarding is permitted.

17.1 Test Category Summary

Category	Code	Applicability
Authentication and Token	CT-AUTH	Consumer Node, Provider Node, Platform
Standard API	CT-API	Consumer Client Node
Request Envelope	CT-REQ	Consumer Client Node
Response Verification	CT-RESP	Consumer Node, Provider Node
Consent Handling	CT-CONS	Consumer Node, Provider Node, Platform
Schema Validation	CT-SCHEMA	Consumer Node, Provider Node
Digital Signature	CT-SIG	Consumer Node, Provider Node
End-to-End Encryption	CT-ENC	Consumer Node, Provider Node
TPL Recording	CT-TPL	Provider Node, Platform
Error Handling	CT-ERR	Consumer Node, Provider Node
Event Management	CT-EVENT	Consumer Node, Provider Node
Fulfilment Layer	CT-FULFIL	Provider Client Node
Delegated Channel	CT-DELEGATED	Sponsoring Entity Client Node, Platform

17.2 Authentication and Token Tests (CT-AUTH)

Test ID	Test Name	Expected Outcome
CT-AUTH-01	Valid mTLS-bound token acceptance	Request proceeds normally.
CT-AUTH-02	Expired token rejection	HTTP 401 AUTH_TOKEN_EXPIRED.
CT-AUTH-03	Invalid JWT signature rejection	HTTP 401 AUTH_TOKEN_INVALID.
CT-AUTH-04	JTI replay prevention	Second use of same token within expiry returns HTTP 401 AUTH_TOKEN_REPLAY.
CT-AUTH-05	mTLS certificate mismatch	Cert not matching cnf.x5t#S256 returns HTTP 401 AUTH_MTLS_FAILED.
CT-AUTH-06	Revoked certificate rejection	HTTP 401 AUTH_CERT_REVOKED.
CT-AUTH-07	Insufficient scope	HTTP 403 AUTHZ_SCOPE_INSUFFICIENT.
CT-AUTH-08	Unpermitted purpose	HTTP 403 AUTHZ_PURPOSE_MISMATCH.
CT-AUTH-09	Unregistered IP address	HTTP 403 AUTHZ_IP_BLOCKED.
CT-AUTH-10	Clock skew validation	Token with iat > now + 30s returns HTTP 401 AUTH_TOKEN_INVALID.

17.3 Consent Tests (CT-CONS)

Test ID	Test Name	Expected Outcome

CT-CONS-01	Consent-governed dataset without artifact	HTTP 403 CONSENT_REQUIRED.
CT-CONS-02	Valid consent accepted	Request proceeds; data returned.
CT-CONS-03	Expired consent rejected	HTTP 403 CONSENT_EXPIRED.
CT-CONS-04	Revoked consent rejected	HTTP 403 CONSENT_REVOKED.
CT-CONS-05	Purpose mismatch rejection	HTTP 403 CONSENT_PURPOSE_MISMATCH.
CT-CONS-06	Field scope enforcement	HTTP 403 CONSENT_FIELD_OUT_OF_SCOPE.
CT-CONS-07	Tampered consent artifact rejected	HTTP 403 CONSENT_INVALID.
CT-CONS-08	Consent revocation propagation	getData rejected within 5 seconds of revocation.

17.4 Signature Tests (CT-SIG)

Test ID	Test Name	Expected Outcome
CT-SIG-01	Valid consumer request signature	Request proceeds.
CT-SIG-02	Tampered request body post-signature	HTTP 400 REQUEST_SIGNATURE_INVALID.
CT-SIG-03	Wrong signing key used	HTTP 400 REQUEST_SIGNATURE_INVALID.
CT-SIG-04	Valid provider response signature verification	Consumer Client Node accepts response.
CT-SIG-05	Tampered provider response	Consumer Client Node rejects — PROVIDER_SIGNATURE_INVALID.
CT-SIG-06	TPL proof signature verification	Caller verifies TPL proof using platform public key.

17.5 Schema Tests (CT-SCHEMA)

Test ID	Test Name	Expected Outcome
CT-SCHEMA-01	Valid request schema	Request proceeds.
CT-SCHEMA-02	Missing required field (transactionId)	HTTP 400 SCHEMA_VALIDATION_FAILED.
CT-SCHEMA-03	Invalid datasetId format	HTTP 400 SCHEMA_VALIDATION_FAILED.
CT-SCHEMA-04	Unknown datasetId	HTTP 404 SCHEMA_DATASET_NOT_FOUND.
CT-SCHEMA-05	Deprecated schema version	HTTP 410 SCHEMA_VERSION_DEPRECATED.
CT-SCHEMA-06	Provider response schema violation	Consumer receives HTTP 502 PROVIDER_SCHEMA_VIOLATION.

17.6 Encryption Tests (CT-ENC)

Test ID	Test Name	Expected Outcome
CT-ENC-01	JWE-encrypted request processing	Provider successfully decrypts and processes encrypted request.
CT-ENC-02	JWE-encrypted response processing	Consumer successfully decrypts provider response.

CT-ENC-03	Platform data blindness verification	Platform audit logs contain no plaintext payload data for E2E encrypted transactions.
CT-ENC-04	Ephemeral key uniqueness	Each request uses unique ephemeral key pair.
CT-ENC-05	Wrong provider key rejection	Provider returns DECRYPTION_FAILED for payload encrypted with wrong key.

17.7 Fulfilment Layer Tests (CT-FULFIL)

Test ID	Test Name	Expected Outcome
CT-FULFIL-01	Complete 12-step verification sequence	All steps performed in correct order; no step skipped.
CT-FULFIL-02	Invalid consumer signature rejected at provider	REQUEST_SIGNATURE_INVALID without calling backend.
CT-FULFIL-03	Schema validation before signing	Non-conforming backend data not returned; PROVIDER_SCHEMA_VIOLATION raised.
CT-FULFIL-04	TPL proof submitted on success	TPL record created with correct hashes and provider signature.
CT-FULFIL-05	TPL proof submitted on failure	Failed transactions recorded with status=FAILED.
CT-FULFIL-06	Backend timeout handling	Provider returns PROVIDER_TIMEOUT if backend exceeds 25 seconds.

17.8 Conformance Certification Process

To obtain WASL conformance certification, an implementation **SHALL**: (1) execute all applicable tests in the WASL certified sandbox using the provided test harness; (2) submit test logs and results within 30 days of test completion; (3) resolve all CRITICAL and HIGH severity non-conformances before production onboarding approval; (4) complete a final certification interview; and (5) maintain conformance through re-certification upon each major specification revision.

17.9 Delegated Channel Tests (CT-DELEGATED)

Test ID	Test Name	Expected Outcome
CT-DELEGATED-01	API key/secret exchange for OAuth token	Sponsoring entity-issued API key and secret exchanged via the IAM token endpoint (https://iam.wasl.gov.pk/oauth2/token); scoped delegated_channel token issued with correct delegated_by and delegation_scope claims; subsequent API calls routed through the API Gateway under delegation_scope enforcement.
CT-DELEGATED-02	Scope enforcement — dataset outside delegation_scope rejected	Request for a dataset not listed in delegation_scope returns HTTP 403 AUTHZ_SCOPE_EXCEEDED. Sponsoring entity's own subscription does not override the restriction.

CT-DELEGATED-03	Scope enforcement — dataset within scope proceeds	Request for a dataset within delegation_scope is routed normally through the API Gateway and Secure Connectivity Layer; response returned to the Delegated Channel application as originating caller.
CT-DELEGATED-04	Missing delegated_by claim on delegated_channel token rejected	Token presenting token_type=delegated_channel without delegated_by returns HTTP 401 AUTH_TOKEN_INVALID at API Gateway.
CT-DELEGATED-05	TPL record captures sponsoring entity identity	TPL proof record for delegated transaction includes both the sponsoring entity org_id (from delegated_by) and the delegated application client_id as distinct fields.
CT-DELEGATED-06	Consent artefact references sponsoring entity, not delegated application	For personal data requests, consent artefact cites the sponsoring entity as the data accessor; citizen notification identifies the entity, not the third-party application.
CT-DELEGATED-07	Province-code routing key resolves correct provider	getData request carrying province_code routing key is dispatched to the matching provincial Client Node; incorrect or absent routing key returns PROVIDER_UNAVAILABLE.

18 ENTITY ONBOARDING — TECHNICAL REQUIREMENTS

Figure 18-1: Entity Onboarding Lifecycle, the sequential stages required for an organization to become an active participant in the WASL ecosystem

18.1 Onboarding Stages

Stage	Name	Pre-Requisite	Action Required
1	Registration and Approval	None	Yes — multi-level review
2	Data Sharing Agreement /MoU Execution	Stage 1	Yes — legal review and signing
3	PKI Certificate Issuance	Stage 2	Yes — RA validates identity
4	VPN Connectivity Establishment	Stage 3	Yes — NOC validates tunnel
5	Client Node Deployment	Stage 4	Technical assistance available
6	Schema Subscription and API Registration	Stage 5	Provider schemas reviewed by PDA/NADRA
7	Sandbox Certification	Stage 6	Yes — PDA runs conformance tests
8	Production Activation	Stage 7	Yes — PDA approves activation

18.2 PKI Certificate Types

Certificate Type	Purpose	Validity	Key Algorithm
VPN Device Certificate	IPSec tunnel authentication	24 months	ECDSA P-256 or RSA-4096
Entity Signing Certificate	Transaction request/response signing	24 months	ECDSA P-256 or RSA-4096

18.3 Registration Payload

```
{
  "organizationName": "Punjab Education Department",
  "organizationType": "government",
  "federalProvincial": "provincial",
  "province": "Punjab",
  "roles": ["consumer"],
  "authorizedRepresentative": {
    "name": "Ali Khan",
    "designation": "Director IT",
    "email": "ali.khan@edupunjab.gov.pk",
    "phone": "+92-42-XXXXXXXX"
  },
  "technicalContact": { "name": "IT Admin", "email":
"it@edupunjab.gov.pk" },
  "intendedDatasets": ["health.immunization","civil.registry.basic"],
  "intendedPurposes": ["school_admission"]
}
```

18.4 Onboarding Roles

Organizations may participate as: Data Consumer (calls getData to retrieve information from registered providers; subscribes to relevant schemas and generates OAuth credentials); Data Provider (exposes data through the Fulfilment Layer conforming to published schemas; responsible for schema validation of all responses before signing); or both Consumer and Provider simultaneously, with capabilities onboarded separately for each role.

18.5 Sandbox Certification

Before production activation, all implementations **SHALL** complete certification in the WASL certified sandbox environment using the provided test harness. Sandbox certification covers: API integration against real schema definitions; consent flow simulation; signature and encryption verification; and all conformance tests defined in Section 17. Only after passing all CRITICAL and HIGH severity test categories and receiving the concerned technical team clearance may an entity proceed to Stage 8 (Production Activation). Re-certification is required upon each major specification revision per Section 17.8.

18.6 Stage Narrative

Stage 1: Registration and Approval: Organization registers via the WASL self-service portal providing authorized representative details, legal documentation, role declaration, and

technical contact. PDA/NADRA conducts multi-level review with status notifications at each step.

Stage 2: Data Sharing Agreement Execution: Approved organizations execute a formal tripartite DSA/MoU (PDA and NADRA) with terms of use, data handling obligations, SLA commitments, and liability provisions.

Stage 3: PKI Certificate Issuance: The WASL Registration Authority validates entity identity and issues VPN Device Certificates (24-month validity) and Entity Signing Certificates (24-month validity) per Section18.2. Issued only after successful legal onboarding.

Stage 4: VPN Connectivity Establishment: Organization configures its firewall for site-to-site IPSec VPN to WASL's Secure Connectivity Layer. An NOC validates tunnel establishment, certificate validity, routing, and failover.

Stage 5: Client Node Deployment: Organization deploys official WASL Client Node OCI container images within Kubernetes infrastructure per Section15.1. The Client Node must reside within the organization's own secure infrastructure boundary.

Stage 6: Schema Subscription and API Registration: Consumers subscribe to schemas via the Developer Portal and generate OAuth credentials. Providers register their domain, configure the Fulfilment Layer, and complete schema review before production publication.

Stage 7: Sandbox Certification: Per Section 18.5 above.

Stage 8: Production Activation: PDA/NADRA approves activation upon successful sandbox certification. The organization is provisioned with production certificates, listed as active in the Provider Directory, and authorized to participate in live WASL transactions.

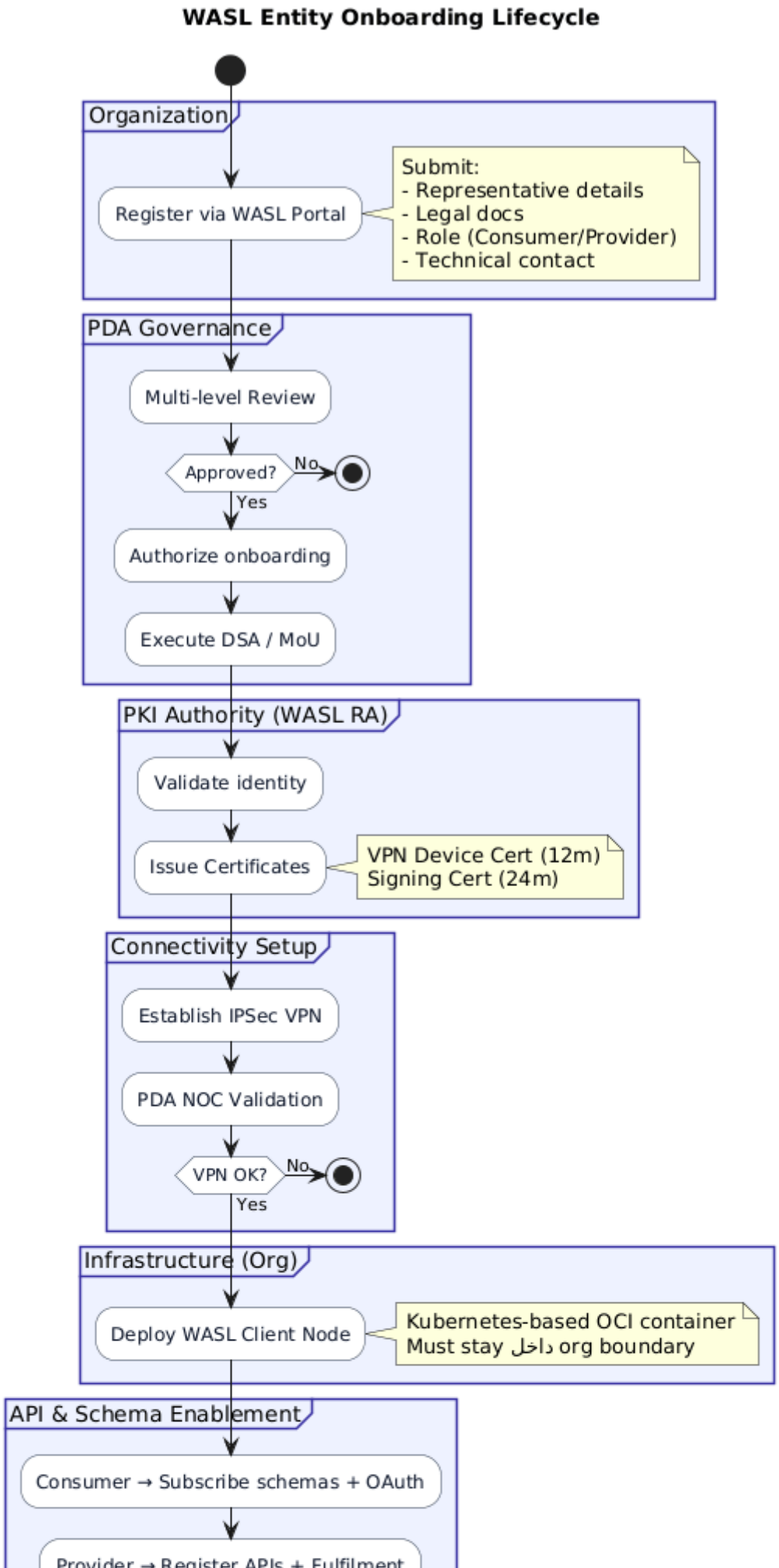
WASL Entity Onboarding Lifecycle
Organization
Register via WASL Portal
Submit:
- Representative details
- Legal docs
- Role (Consumer/Provider)
- Technical contact
PDA Governance
Multi-level Review
Approved?
No
Yes
Authorize onboarding
Execute DSA / MoU
PKI Authority (WASL RA)
Validate identity
Issue Certificates
VPN Device Cert (12m)
Signing Cert (24m)
Connectivity Setup
Establish IPSec VPN
PDA NOC Validation
VPN OK?
No
Yes
Infrastructure (Org)
Deploy WASL Client Node
Kubernetes-based OCI container
Must stay داخل org boundary
API & Schema Enablement
Consumer → Subscribe schemas + OAuth
Provider → Register APIs + Fulfilment

Figure 18-2 WASL - Entity Onboarding Lifecycle, End-to-end process from organizational registration and legal onboarding through PKI provisioning, secure connectivity setup, client node deployment, schema/API enablement, sandbox certification, and final production activation under PDA governance.

19 MONITORING, LOGGING, AND AUDIT: TECHNICAL REQUIREMENTS

19.1 Structured Log Record Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.wasl.gov.pk/v1/log-record.json",
  "title": "WASL Structured Log Record",
  "type": "object",
  "required":
["logId","timestamp","component","eventType","transactionId"],
  "properties": {
    "logId": { "type": "string" },
    "timestamp": { "type": "string", "format": "date-time" },
    "component": { "type": "string",
      "enum":
["API_GATEWAY","IAM","CONSENT_LAYER","TPL","ORCHESTRATION",

"SCL","AUDIT","CLIENT_NODE_CONSUMER","CLIENT_NODE_PROVIDER"] },
    "eventType": { "type": "string" },
    "transactionId": { "type": "string" },
    "actor": {
      "type": "object",
      "properties": {
        "type": { "type": "string",
          "enum": ["consumer","provider","platform","admin","citizen"]
},
        "id": { "type": "string" }
      }
    },
    "datasetId": { "type": "string" },
    "purpose": { "type": "string" },
    "status": { "type": "string" },
    "errorCode": { "type": ["string","null"] },
    "durationMs": { "type": "integer" },
    "ipAddress": { "type": "string",
      "description": "Masked per data minimisation policy." }
  }
}
```

19.2 Mandatory Log Events

Event Type	Component	Required Fields
TOKEN_ISSUED	IAM	transactionId, actor.id, scope, exp

TOKEN_REJECTED	IAM	transactionId, errorCode, reason
REQUEST_RECEIVED	API_GATEWAY	transactionId, actor.id, datasetId, purpose
CONSENT_VALIDATED	API_GATEWAY	transactionId, consentId, status
REQUEST_ROUTED	SCL	transactionId, providerId, routingLatencyMs
RESPONSE_RECEIVED	API_GATEWAY	transactionId, providerId, status, durationMs
PROOF_RECORDED	TPL	transactionId, requestHash, responseHash, status
CONSENT_GRANTED	CONSENT_LAYER	transactionId, consentId, subjectId (masked), channel
CONSENT_REVOKED	CONSENT_LAYER	transactionId, consentId, reason
ERROR_OCCURRED	Any	transactionId, component, errorCode, detail

19.3 Log Retention Policy

Log Category	Retention Period	Encryption at Rest	Access Control
Operational logs	90 days	AES-256	Platform operations team
Security / audit logs	3 years	AES-256	Security & Compliance
TPL proof records	10 years	AES-256	Read-only for authorised parties
Consent logs	7 years post-consent-expiry	AES-256	PDA Legal & Compliance
Incident logs	7 years	AES-256	Security Operations

20 INFRASTRUCTURE AND DEPLOYMENT SPECIFICATION

20.1 Central Platform SLAs

Component	Availability SLA	Deployment Model
API Gateway	99.9%	Active-Active, multi-AZ; HAProxy/Nginx load balancing
IAM Service	99.9%	Active-Active, multi-AZ; distributed session state
Metadata Repository	99.5%	Active-Passive with hot standby; database replication
Consent Layer	99.5%	Active-Passive; asynchronous replication
Transaction Proof Layer	99.9% read / 99.5% write	Active-Passive write, Active-Active read; eventual consistency acceptable for writes
Secure Connectivity Layer	99.9%	Active-Active; redundant VPN termination
Audit and Logging	99.5%	Append-only, distributed; 7-day minimum local cache

20.2 Disaster Recovery

Recovery Time Objective (RTO)	4 hours for critical services (IAM, API Gateway, SCL)
Recovery Point Objective (RPO)	1 hour for Metadata Repository and Consent Layer; 0 for TPL (synchronous replication)
Geographic Redundancy	Primary: Islamabad; DR site: Lahore/Karachi
Failover Trigger	Automated for network-level failures; manual approval for data-centre failover
DR Test Frequency	Full failover test minimum annually; tabletop exercise quarterly

21 COMPLIANCE WITH INTERNATIONAL STANDARDS

Category	Standard	Application in WASL DNP-D.300 TS
API Standards	OpenAPI Specification v3.0	All WASL Standard APIs defined per OAS v3.0
API Standards	JSON Schema Draft 2020-12	All schemas, metadata records, consent artifacts, error responses
Authentication	OAuth 2.1	Token issuance, client credentials grant, scope enforcement
Authentication	OpenID Connect Core 1.0	Citizen-facing authentication via PAK-ID
Authentication	IETF RFC 8705	mTLS client authentication and certificate-bound access tokens
Token Format	IETF RFC 7519 (JWT)	Access token structure and all claim definitions
Key Infrastructure	IETF RFC 7517 (JWK)	JWKS endpoint and public key distribution
Encryption	JOSE RFC 7516 (JWE)	End-to-end payload encryption format
Transport Security	IETF RFC 8446 (TLS 1.3)	Mandatory transport encryption; cipher suite requirements
Network Security	IKEv2 (RFC 7296)	IPSec VPN tunnel establishment; DH group requirements
Digital Signatures	FIPS 186-5	ECDSA and RSA-PSS algorithm requirements
Crypto Modules	FIPS 140-2/3 Level 3	HSM requirements for entity and CA private keys
Information Security	ISO/IEC 27001:2022	Security management framework
Privacy	ISO/IEC 27018	PII protection in cloud environments
Privacy Framework	ISO/IEC 29100	Privacy framework for consent and data handling
Zero Trust	NIST SP 800-207	Zero trust architecture principles
Continuity	ISO 22301	Business continuity and disaster recovery
Timestamps	RFC 3161	Trusted timestamping for TPL proof records

22 ORCHESTRATION SERVICES: TECHNICAL SPECIFICATION

Orchestration Services coordinate multi-step, multi-system workflows that cannot be completed through a single API request. They manage state across interactions with IAM, the Metadata Repository, the Consent Layer, the API Gateway, and multiple Provider Nodes to execute complex transactions such as multi-agency eligibility determinations and asynchronous processing flows.

22 Section 22

22.1 Workflow Execution Model

Orchestration Services **SHALL** execute workflows as state machines where each transaction progresses through defined steps and transitions. Workflow state **MUST** be persisted including current workflow step; intermediate results (as hashes, not plaintext data); timestamps; transaction identifiers; and error states. A typical orchestrated multi-provider workflow includes validate request; retrieve metadata; initiate consent for all required datasets; fetch data from each required provider; aggregate and validate results; generate signed composite response; and record proof in TPL for all sub-transactions.

22.2 Asynchronous Workflow Support

Orchestration Services **SHALL** support asynchronous execution of workflows involving multiple systems, external citizen consent, or long processing times. A transaction ID **SHALL** be returned immediately to the consumer; processing continues in the background; and status is retrievable via the getTransactionTPLStatus endpoint (Section 8.5). This addresses multi-step orchestration flows and is distinct from the single-call async pattern specified in Section 8.6.

22.3 Retry and Failure Handling

Orchestration Services **SHALL** implement automatic retries with exponential backoff for transient failures; circuit breakers for consistently failing providers; and partial result handling where one provider fails but others succeed. All failures **SHALL** be recorded in the TPL with status FAILED. If consent expires mid-workflow, the workflow **MUST** abort and return CONSENT_EXPIRED. If the orchestration service restarts, it **MUST** resume from the last persisted workflow state.

22.4 Governance Constraints

All data aggregations performed by Orchestration Services **MUST**: respect the consent scope for each contributing dataset; apply the classification controls of the highest-tier data in the aggregate; and be fully auditable with traceability to each source transaction. Orchestration **MUST NOT** bypass WASL IAM, consent, or schema validation requirements at any step.

23 SECURE CONNECTIVITY LAYER: TECHNICAL SPECIFICATION

The Secure Connectivity Layer (SCL) is the trusted transport and routing fabric connecting WASL Client Nodes with the Central WASL Platform. It provides the secure communications

backbone for synchronous request-response traffic, asynchronous event notifications, consent signaling, and routing metadata. The SCL does not perform consent decisioning, schema validation, access control, or business payload interpretation. Where end-to-end encryption is applied, the SCL transports ciphertext and routing metadata without visibility into the underlying business payload.

23.1 Network Architecture Requirements

All participating entities **SHALL** connect to WASL through mutually authenticated transport channels: PKI-authenticated IPSec VPN tunnels (IKEv2, AES-256-GCM, DH Group 19 or 20 per Section 5.5); dedicated VPN termination infrastructure at the WASL platform; firewall-enforced ingress and egress controls at both platform and participant boundaries; private or logically isolated routing for WASL traffic; and mTLS at the application layer above the network tunnel. WASL traffic **SHALL NOT** be routed over unsecured public internet paths.

23.2 Message Routing and Delivery

For synchronous request-response flows, the SCL **SHALL**: receive a validated request from the API Gateway; resolve the target provider endpoint from the Metadata Repository provider binding; forward the message to the destination Client Node; and return the provider response to the originating path. For asynchronous operations, the SCL **SHALL** support: at-least-once delivery semantics; retry with exponential backoff; delivery acknowledgment tracking; and dead-letter handling for undeliverable messages.

23.3 SCL Security Controls

The SCL **SHALL** enforce: mutually authenticated node connectivity using WASL PKI-issued certificates; encrypted network transport (TLS 1.3 at application layer, IPSec at network layer per Section 5); routing only to registered and active Client Nodes per the Provider Directory (Section 9.5); certificate validation and OCSP revocation checking on all connections; and message integrity verification at the transport boundary. These controls complement but do not replace application-layer signing and encryption performed by Client Nodes per Section 14.

23.4 Transport Telemetry

The SCL **SHALL** generate structured telemetry for: delivery success and failure per node; round-trip latency; node availability; queue depth and backlog; and reconnect and retry events. This telemetry feeds into the monitoring infrastructure (Section 19) and is used for SLA reporting (Section 20.1) and capacity planning.

24 AI INTEGRATION SERVICES: TECHNICAL SPECIFICATION

This section specifies the technical integration requirements for AI systems interacting with WASL, implementing RA Section 12. Two integration patterns are defined: AI in WASL (platform-side AI capabilities operating on metadata and audit streams, never on citizen data content) and AI as a Consumer of WASL (external AI systems invoking WASL APIs as authorized Consumers). All AI interactions are subject to identical authentication, consent, and audit obligations as human-originated interactions, with additional provenance metadata requirements per RA Section 12.3. Governance of AI model lifecycle, evaluation, and safety is defined under the DNP-A series; this section specifies only the WASL integration interface.

24.1 AI Identity Binding: Technical Requirements

When a WASL Client Node is operated by or on behalf of an AI-driven application, the access token issued by IAM **SHALL** carry the following additional claims (per RA Sections 7.1.1 and 12.2.1): ai_model_id (REQUIRED), the registered model identifier from the DNP-A model registry, format: <registry_code>:<model_identifier>:<version_string>; ai_invoking_principal (REQUIRED), the human operator, service account, or orchestrating system that triggered the AI action; ai_model_version (REQUIRED), the model version string for forensic traceability. These claims **SHALL** be bound to the token at issuance. Resource servers and the API Gateway **SHALL** validate their presence for any token carrying ai_model_id. The TPL proof record **SHALL** capture these values for every AI-originated transaction so that any AI-driven decision can be traced to the specific model, version, and invoking context. AI agents carry no additional data access privileges; access control decisions are made identically on org_id, scope, purpose, and consent artefact as for human-operated requests.

24.2 AI Model Registry Integration

AI-driven consumer applications **SHALL** register the model(s) they use in the AI model registry under DNP-A prior to production WASL participation. IAM **SHALL** validate ai_model_id values at token issuance against the registered model list. Tokens referencing unregistered models **SHALL** be rejected (HTTP 400 AI_MODEL_NOT_REGISTERED). Model deregistration (e.g., following a safety incident) **SHALL** propagate to IAM within 5 minutes; tokens bound to a deregistered model **SHALL** be revoked within 15 minutes. These events are mandatory log events (Section 19.2).

24.3 Model Context Protocol (MCP) Integration Surface

WASL exposes an MCP server surface through the Developer Portal (RA Section 12.4), enabling MCP-compatible AI clients to discover WASL schemas, consent requirements, available datasets, and integration patterns through a standardized protocol. The MCP surface does not bypass any WASL security control: MCP clients still obtain IAM tokens, attach consent artefacts, and transit the API Gateway under full enforcement. MCP is a discovery and invocation convenience layer, not a privileged access channel. The MCP server exposes: dataset catalogue discovery (maps to listDatasets API); schema retrieval (maps to getSchema); consent policy inspection (maps to getConsentInfo metadata); and sandbox integration patterns. All MCP tool calls resulting in a WASL API invocation are subject to full TPL recording. The MCP server surface is published at: https://api.wasl.gov.pk/mcp/v1.

24.4 AI Governance Guardrails (Technical Implementation)

Technical implementation of RA Section 12.3 guardrails: (a) No autonomous consent granting, the Consent Layer **SHALL** reject consent initiation by an AI agent token unless a pre-existing citizen-signed delegated-consent artefact (v2 capability) is present; (b) Parity of audit obligation, the API Gateway **SHALL** reject tokens carrying ai_model_id without ai_invoking_principal and ai_model_version; TPL records for AI transactions **SHALL** carry ai_originated=true; (c) No privileged data access — scope and dataset subscription rules apply identically to AI and human operators; no AI-specific endpoint or scope bypass exists; (d) Purpose binding, every AI request is evaluated against declared purpose and consent artefact purpose identically to human requests; (e) Model accountability, bound ai_model_id

is required in TPL records enabling post-incident filter by model; (f) Revocability, citizen consent enabling AI access is revocable with immediate effect under Sections 10 and 15.4.

24.5 Delegated-Consent Scaffolding (v1 Structural; v2 Operational)

Per RA Section 12.2.3 and Annex I, WASL v1 includes structural scaffolding for citizen-delegated AI agents (present in schema, not operationally active). The consent artefact schema (Section 10.2) includes reserved fields for delegatedAgentId, delegationScope, delegationPurpose, delegationExpiry, and revocationAddress. Operationalisation requires policy resolution under DNP-A (liability chain, model accreditation, safeguards against agent manipulation) and formal PDA Standards Board decision planned for v2. Until then, requests presenting a delegated-consent artefact type **SHALL** be rejected (HTTP 501 DELEGATED_CONSENT_NOT_YET_ACTIVE).

25 BILLING AND MONETIZATION: TECHNICAL REQUIREMENTS

Per RA Section 7.1.10, the Central WASL Platform includes a Billing and Monetization component supporting usage-based billing for data exchange. The detailed technical specification is deferred to a dedicated supplement under the DNP-D series. This section establishes the architectural commitments the billing implementation **SHALL** satisfy.

25.1 Billing Architecture Principles

The billing system **SHALL**: (a) meter API usage at the transaction level, correlated by transactionId and consumerId from the TPL; (b) support differentiated billing policies by exchange type (government-to-government: free or nominal cost as default per DPI principles; government-to-regulated-private-sector; commercial consumer access); (c) generate automated purchase orders and invoices compatible with Pakistan's government e-procurement systems; (d) integrate with designated digital payment infrastructure; (e) be data-blind, billing metadata derived from TPL records (entity IDs, dataset IDs, transaction counts, data classification tiers), not payload content; (f) produce billing records independently auditable and reconcilable against TPL transaction records; (g) support Data Provider-specific charging arrangements approved through the Billing Policy process.

25.2 Usage Metering

The metering subsystem **SHALL** consume TPL proof records as its authoritative source for billable events. Metering counters are maintained per (consumerId, providerId, datasetId, billing period) tuple. Billable events: successful getData calls (status SUCCESS or PARTIAL_SUCCESS); completed asynchronous transactions; event subscription activations/deliveries where applicable under the billing policy. Failed transactions (CONSENT_REQUIRED, AUTH errors, etc.) are not billable. Usage reports available to each Participating Entity through the Developer Portal for the rolling 12-month period. The detailed metering schema, billing policy format, and payment integration specification are published in the billing supplement.

26 DELEGATED CHANNEL INTEGRATION PATTERN

This section specifies the technical requirements for the Delegated Channel integration pattern, the recommended pattern for integrating third-party applications, including Super

Apps, composite service platforms, and delegated citizen-facing applications, with WASL. This is an additive capability. It does not modify or override any existing integration pattern, entity onboarding model, or standard API specification in this document. All Participating Entity interactions defined in the preceding sections remain unchanged. The Delegated Channel pattern operates entirely within the existing WASL architecture as defined in DNP-D.002 RA; no RA modification is required.

26.1 Pattern Overview and Rationale

A Delegated Channel application is a third-party platform (such as a government Super App, a private sector composite services portal, or an authorized integration middleware) that acts on behalf of a WASL Participating Entity with explicitly scoped permissions. The sponsoring entity, the Participating Entity that takes responsibility for the delegated application, defines what the application may discover and call, generates the credentials, and remains legally and operationally accountable for all transactions the application performs on its behalf.

The Delegated Channel pattern is the preferred integration approach for use cases that are: state-mutating (requiring an instruction payload to be routed to a provider's Fulfilment Layer, such as an ownership transfer or a record update); multi-party (involving multiple consent artefacts, multiple agencies, or parallel verification steps); or legally constitutive (where the TPL must anchor the entire transaction lifecycle as a single verifiable record). For such use cases, routing via the API Gateway is mandated because it preserves centralized policy enforcement, automatic TPL capture, and Consent Layer integration without requiring additional bespoke mechanisms at each entity. The API Gateway is the single entry and exit point for all synchronous WASL API requests; requests from Delegated Channel applications are not exempt from this path.

A Delegated Channel application does not require full WASL Participating Entity onboarding (the 8-stage process in Section 18). It does not hold a WASL PKI certificate, does not deploy a Client Node, and is not listed as an independent entity in the Provider Directory. Its authority is entirely derived from and bounded by the sponsoring entity's permissions. However, Delegated Channel applications **SHALL** be registered as first-class actors in the WASL ecosystem, recorded against the sponsoring entity's profile in the Metadata Repository.

26.2 Credential Model: API Key/Secret Issuance and Scoping

The sponsoring entity generates an API key and secret for the Delegated Channel application through the WASL Developer Portal. The API key/secret is a scoped credential; it cannot exceed the entity's own WASL permissions. The following rules govern issuance and use:

(a) Issuance: the sponsoring entity's authorized representative generates the API key via the Developer Portal. The key is bound to a declared delegation scope, specifying the dataset IDs and API endpoints the delegated application is permitted to invoke. The scope is a strict subset of the entity's own dataset_subscriptions and granted scopes.

(b) Token exchange: the Delegated Channel application presents the API key and secret to the WASL IAM token endpoint (https://iam.wasl.gov.pk/oauth2/token) with grant_type=api_key_delegation. Following successful token issuance, all subsequent API calls by the Delegated Channel application are routed through the API Gateway under standard enforcement.

(c) Scope enforcement: the API Gateway enforces delegation_scope on every request. A request for a dataset or endpoint not listed in delegation_scope is rejected with HTTP 403 AUTHZ_SCOPE_EXCEEDED regardless of the sponsoring entity's own broader subscriptions.

(d) Revocation: the sponsoring entity may revoke an API key at any time via the Developer Portal. Revocation propagates to IAM within 60 seconds. Active tokens bound to a revoked key are invalidated within 60 seconds.

(e) Accountability: all transactions by the Delegated Channel application are recorded in the TPL and audit log under the sponsoring entity's identity, with the delegated application's client_id captured as a secondary identifier. The sponsoring entity is the accountable party for all delegated transactions.

26.3 Token Validation for Delegated Channel Tokens

In addition to all standard token validation requirements defined in Section 6.2.4, resource servers and the API Gateway **SHALL** apply the following additional checks to tokens presenting token_type=delegated_channel:

(a) delegated_by **MUST** be present and **MUST** resolve to an active Participating Entity in the Provider Directory;

(b) delegation_scope **MUST** be present and non-empty;

(c) the requested operation's required scope **MUST** appear in delegation_scope (not merely in the sponsoring entity's subscription list);

(d) the sponsoring entity identified in delegated_by **MUST** itself hold an active subscription to each dataset in delegation_scope, a scope granted to the delegated application but not held by the sponsor is invalid and the token **SHALL** be rejected with HTTP 401 AUTH_DELEGATION_INVALID;

(e) mTLS certificate binding applies to the connection between the Delegated Channel application and the WASL API Gateway; the certificate is the Delegated Channel application's own TLS certificate (not a WASL PKI entity certificate); the cnf.x5t#S256 claim reflects this certificate. The API Gateway enforces this binding on every delegated_channel request alongside all standard token validation checks. Token lifetime for delegated_channel tokens is capped at 900 seconds (standard maximum), with shorter lifetimes recommended for high-sensitivity scope sets. The mTLS requirement specified in Section 5.2 applies to all Delegated Channel connections; the API Gateway **SHALL** verify the certificate thumbprint binding per RFC 8705 on every delegated_channel token request.

26.4 Six-Step Interaction Flow

The Delegated Channel integration pattern operates in two phases: Setup (performed once per application registration) and Transact (performed at runtime for each user workflow).

Setup Phase (Steps 1–2)

Step 1: Entity publishes and approves delegation: the sponsoring entity publishes its APIs and metadata on the WASL platform via the standard metadata publication process (Section 9). The entity then uses the Developer Portal to register the Delegated Channel application, define the delegation scope, and generate an API key/secret pair. The generated credentials are provided to the Delegated Channel application operator through a secure out-of-band channel.

Step 2: Schema and API discovery: the Delegated Channel application uses the API key/secret to call the WASL metadata discovery APIs (listDatasets, getSchema) via the API Gateway. During the initial design-time setup phase, the API key/secret is presented directly to the API Gateway for discovery calls; the API Gateway validates the key, resolves the sponsoring entity, and enforces the declared delegation_scope before returning results. Only datasets and APIs within the declared scope are returned. The application uses the discovered schemas to design and configure its workflow experience. This discovery step **MUST** route via the API Gateway so that policy-governed scope enforcement applies to discovery as well as to data access. At transaction runtime, a full delegated_channel OAuth token is obtained from the IAM token endpoint before any getData or custom API calls are made (Step 3).

Transaction Phase (Steps 3–6)

Step 3: Token exchange: when a user initiates a transaction, the Delegated Channel application presents its API key and secret to the WASL IAM token endpoint and receives a delegated_channel OAuth token scoped to the declared delegation scope. This applies equally to all parties interacting through the Delegated Channel application, including temporally separated parties such as an initiating party and a responding party in a multi-party workflow; each interaction requires a fresh token carrying the same delegated_by and delegation_scope claims bound to the sponsoring entity. The token is short-lived (maximum 900 seconds).

Step 4: Request submission: the Delegated Channel application constructs a getData or custom API request using the delegated_channel token and submits it to the API Gateway at https://api.wasl.gov.pk/v1/. The API Gateway validates the delegated_channel token per Section 26.3 before routing the request onward.

Step 5: Routing and fulfilment: the API Gateway routes the validated request to the appropriate Client Node via the Secure Connectivity Layer. The primary pattern for Delegated Channel use cases is the sponsoring entity's own published APIs: the entity's Fulfilment Layer receives the request, queries its own backend systems, and returns the response entirely within its own infrastructure boundary.

Where the sponsoring entity is additionally authorized to access another Participating Entity's data through WASL, the sponsoring entity's Client Node may sign and dispatch a getData request to that provider per the standard flow in Section 16; this is a secondary pattern and requires the target dataset to be within the delegated application's declared delegation_scope. All standard requirements apply to this cross-entity getData call without exception: a valid consent artefact is required for any personal data request (Section 10); the target provider's Client Node submits a TPL proof record on every response (Section 13); and the API Gateway enforces token validation, consent validation, and schema compliance on the request before routing it onward. The secondary pattern does not reduce or modify any of these obligations.

Step 6: Response delivery: the response is routed back through the Secure Connectivity Layer and API Gateway to the Delegated Channel application (not to the sponsoring entity's internal systems) because the delegated application is the originating caller.

26.5 Composite Workflow Pattern for State-Mutating Transactions

For state-mutating, multi-party workflows (such as vehicle ownership transfer, property registration, or benefit enrolment) the sponsoring entity **SHALL** publish the workflow as a

single composite endpoint via the Custom API Marketplace (Section 7.3) rather than exposing the individual sub-steps as separate API calls. Internal orchestration is contained within the sponsoring entity's Fulfilment Layer; the Delegated Channel application sees a single endpoint and a single transaction lifecycle, not the internal sequence of WASL round-trips. This keeps the API Gateway and Secure Connectivity Layer stateless with respect to the workflow and prevents partial-failure complexity from surfacing at the platform layer.

The composite workflow endpoint **SHALL** implement an explicit transaction state machine. Each state transition is an independent TPL anchor point. The following generic state machine applies to all composite workflow use cases; domain-specific states, transition conditions, and parallel fan-out configurations **SHALL** be defined in the relevant sectoral Profile (PRF) published under the DNP-D series.

Generic states:

INITIATED: workflow started; all required consent requests dispatched and pending.

VERIFIED: prerequisite checks completed (identity verification, eligibility conditions, or other domain-specific pre-conditions as defined in the applicable PRF); parallel fan-out via Orchestration Services is permitted for independent checks where the dependency graph allows.

PROCESSING: domain-specific fulfilment steps underway; this state accommodates intermediate steps such as payment confirmation, encumbrance checks, regulatory approvals, or other sequential dependencies defined in the applicable PRF.

COMMITTED: all fulfilment steps successfully executed within the sponsoring entity's Fulfilment Layer; final TPL proof record anchored.

FAILED: terminal failure state; permanently recorded; reason code and last-completed state captured in TPL.

Transition rules applicable to all use cases:

Transitions from INITIATED to VERIFIED MAY run concurrently with other non-dependent checks via Orchestration Services parallel fan-out, where the dependency graph permits and as specified in the applicable PRF. Transitions that mutate authoritative records or confirm irreversible external actions (equivalent to a write or payment confirmation) **SHALL** always be sequential and **SHALL** not proceed until all preceding states are COMMITTED or VERIFIED as applicable. A FAILED record is never deleted; it serves as permanent evidence of the attempted and incomplete transaction. Partial rollback obligations, where applicable, are defined in the sectoral Profile for the relevant domain.

Domain custodians publishing composite workflow endpoints via the Custom API Marketplace **SHALL** document the complete state machine for their use case in the applicable PRF, including: all domain-specific states inserted between VERIFIED and COMMITTED; the dependency graph governing parallel versus sequential transitions; timeout and retry parameters per state; and the rollback or compensation obligations on FAILED.

Not all state-mutating workflows require a composite endpoint. Where the sponsoring entity's Fulfilment Layer natively manages workflow state across multiple discrete API calls, and the Delegated Channel application interacts with those APIs sequentially as independent transactions, it is not necessary to publish a single composite endpoint. In this pattern, each API call is an independent WASL transaction with its own TPL record; the workflow state is held entirely within the sponsoring entity's own systems and is not visible to WASL as a

unified lifecycle. This pattern is appropriate where the entity's own backend is the authoritative state machine and the Delegated Channel application is a thin caller rather than a workflow orchestrator. The composite endpoint pattern defined above is mandated only where the workflow involves multiple participating entities, multiple consent artefacts collected across parties, or where a single verifiable TPL chain of custody across the full lifecycle is a legal or regulatory requirement.

26.6 Consent Handling for Delegated Channel Transactions

Consent requirements for Delegated Channel transactions are identical to those for standard Participating Entity transactions (Section 10). The following additional rules apply:

(a) Consent notification identifies the sponsoring entity as the data accessor. The Delegated Channel application's name is not presented to the citizen in the consent request. This is consistent with the principle that the sponsoring entity is the accountable party for all delegated transactions: the citizen's legal relationship is with the sponsoring entity, not with the third-party application acting on its behalf. The consent artefact and TPL record fully capture the delegation chain for regulatory and audit purposes, even though the citizen-facing notification presents only the sponsoring entity.

(b) For multi-party workflows requiring multiple consent artefacts (for example, a transaction requiring consent from an initiating party and a responding party), each consent artefact is collected independently through the Consent Layer. The composite workflow endpoint **SHALL** not proceed to VERIFIED state until all required consent artefacts are active and valid.

(c) Consent artefacts reference the sponsoring entity’s org_id. TPL records capture the full set of consent artefact hashes for the transaction.

(d) Consent revocation applies immediately to all active requests referencing the revoked artefact, consistent with Section 10.

(e) Where a workflow requires consent from multiple parties whose interactions are temporally separated, for example, a transaction initiated by one party and completed by a second party arriving independently at a later time, each consent artefact is collected at the point of that party's interaction with the Delegated Channel application. The workflow MAY proceed through intermediate states on the basis of the first party's consent; steps requiring the second party's consent **SHALL** be gated on collection and validation of that party's consent artefact before proceeding. Each consent artefact is independently recorded in the TPL with its own hash and timestamp, providing a complete audit trail of when each party authorised their respective step.

26.7 TPL Recording for Delegated Channel Transactions

In addition to the standard TPL proof record fields (Section 13.1), TPL records for delegated_channel transactions **SHALL** include: delegated_by (the sponsoring entity org_id, from the token claim); delegated_client_id (the Delegated Channel application’s client_id); delegation_scope_used (the specific dataset or endpoint called, as a subset confirmation of delegation_scope); and workflow_state (for composite workflow transactions, the state at which the TPL record was anchored, using the state machine values defined in Section 26.5).

For composite multi-step workflows, the TPL records a proof entry at each state transition, triggered via the Orchestration Services and Fulfilment Layer, creating a complete, independently verifiable chain of custody from INITIATED to COMMITTED or FAILED. The

API Gateway and Secure Connectivity Layer remain stateless with respect to the workflow; all state is held within the sponsoring entity's Fulfilment Layer.

26.8 Known Considerations and Mitigations

Consideration	Mitigation
Cumulative latency across multiple sequential WASL round-trips in multi-step workflows	Parallel fan-out via Orchestration Services: prerequisite checks (such as identity verification, eligibility conditions, and domain-specific pre-conditions) dispatched concurrently where the dependency graph permits (Section 26.5). Only terminal steps requiring sequential dependency, as defined in the applicable sectoral Profile, execute sequentially. Latency is bounded by the longest parallel branch, not the sum of all steps.
API Gateway and Secure Connectivity Layer used as stateful transaction coordinator beyond their primary routing design	Composite workflows are published as a single endpoint via the Custom API Marketplace (Section 7.3). All internal orchestration state is held within the sponsoring entity's Fulfilment Layer. The API Gateway and Secure Connectivity Layer route individual requests; neither holds workflow state. The Orchestration Services (Section 22) coordinate multi-step dispatch but remain stateless between transactions.
Partial failure leaves TPL records in intermediate states requiring reconciliation	The explicit state machine defined in Section 26.5 makes FAILED a first-class terminal state. Each state transition is a distinct TPL anchor. Reconciliation is deterministic: the highest committed state in the TPL chain is the authoritative transaction state. No silent abandonment is possible.
Governance overhead if multiple provincial APIs for the same domain are treated as separate publications	Province-code routing key pattern (Section 9.5a): one national schema publication with a routingKeyField covers all provincial providers. Each provincial Client Node registers against the same datasetId with its routingKeyValues. One PDA governance approval; n provider registrations. Approval is not replicated per province.

Annex A (informative) — Purpose Code Registry

The following example purpose codes can be registered in the WASL Purpose Code Registry. Additional codes are added through the metadata governance process without requiring a revision to this specification.

Purpose Code	Description	Applicable Datasets (examples)
credit_assessment	Assessment of creditworthiness for lending	income.tax, civil.registry.basic, land.record.ownership
kyc_verification	Know Your Customer identity verification	civil.registry.basic, civil.registry.biographic
school_admission	Verification for school enrollment	health.immunization, civil.registry.basic
subsidy_eligibility	Eligibility determination for government subsidies	income.tax, land.record.ownership, civil.registry.basic
benefits_administration	Social protection benefit administration	civil.registry.basic, income.tax, health.disability
insurance_underwriting	Risk assessment for insurance products	health.records, civil.registry.basic
legal_verification	Legal proceedings and due diligence	civil.registry.basic, business.registration
fraud_detection	Real-time fraud detection and prevention	civil.registry.basic, income.tax
employment_verification	Employment background checks	civil.registry.basic, tax.registration
regulatory_compliance	Regulatory reporting and compliance	business.registration, income.tax
public_service_delivery	Delivery of government services to citizens	civil.registry.basic, health.immunization
statistical_research	Anonymized aggregate statistical research (subject to PDA data ethics review)	Any — aggregated and anonymized only
law_enforcement	Authorised law enforcement access (statutory basis; subject to judicial oversight)	Any — with judicial authorisation

Annex B (informative) — Openapi 3.0 Skeleton

The abbreviated OpenAPI 3.0 skeleton for the WASL Standard API is shown below. The complete machine-readable specification is published at https://standards.dnp.gov.pk/DNP-D.300/openapi.yaml.

```
openapi: '3.0.3'
```

info:

title: WASL Standard API

version: '1.0.0'

description: Pakistan National Data Exchange Layer Standard API

contact:

email: standards@pda.gov.pk

servers:

- url: https://api.wasl.gov.pk/v1

security:

- BearerAuth: []

mTLS: []

```
paths:
  /data/fetch:
    post:
      summary: getData — Fetch data for a subject
      operationId: getData
      security:
        - BearerAuth: [data.fetch]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/GetDataRequest'
      responses:
        '200': { description: Success }
        '400': { $ref: '#/components/responses/BadRequest' }
        '401': { $ref: '#/components/responses/Unauthorized' }
        '403': { $ref: '#/components/responses/Forbidden' }
        '404': { $ref: '#/components/responses/NotFound' }
        '429': { $ref: '#/components/responses/RateLimited' }
        '500': { $ref: '#/components/responses/InternalError' }
components:
  securitySchemes:
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
    mTLS:
      type: mutualTLS
```

Annex C (normative) — Alignment With Dnp-D.002 Ra

This technical specification is subordinate to DNP-D.002 RA (WASL Reference Architecture). The following table maps each RA component and principle to the specific sections of this specification providing the concrete technical implementation requirements.

RA Element (DNP-D.002 RA)	TS Section (DNP-D.300 TS)	Notes
Identity and Access Management	Section 6: IAM Technical Specification	OAuth 2.1, OIDC, JWT structure, JWKS

OAuth 2.1 / OIDC Protocol Binding	Section 5.3, Section 5.4: Protocol Bindings	Token endpoint, scopes, PKCE
mTLS Binding	Section 5.2: mTLS Binding	RFC 8705, certificate-bound tokens
IPSec VPN Binding	Section 5.5: IPSec VPN Binding	IKEv2, AES-256-GCM, DH Group 19/20
TLS 1.3 Requirement	Section 5.1: Transport Security	Cipher suites, certificate requirements
Standard API Layer	Section 7: Standard API Endpoint Specifications	Endpoint table, HTTP headers
Request Envelope	Section 8.1: getData Request Envelope JSON Schema	Complete JSON Schema (Draft 2020-12)
Response Envelope	Section 8.2: getData Response Envelope JSON Schema	Complete JSON Schema (Draft 2020-12)
Metadata Repository	Section 9: Metadata Record Format	Metadata record JSON Schema
Consent Policy Registry	Section 10.1: Consent Policy Format	Consent policy JSON Schema
Consent Artifact Model	Section 10.2: Consent Artifact JSON Schema	Consent artifact schema and 10-step validation
Data Classification	Section 11: Data Classification Technical Controls	5-tier classification + controls matrix
Error Handling	Section 12: Error Codes and Error Handling	Complete error code registry (40+ codes)
Transaction Proof Layer	Section 13: TPL Technical Specification	Proof record schema, signature computation
External Anchoring (CT-Log) / Federated Endorsement	Section13.4: External Anchoring (CT-Log); Section13.5: Federated Endorsement Model	Merkle root anchoring process
Three-Party Trust / Security	Section 14: Security Architecture	Crypto algorithms, PKI hierarchy, E2E encryption
Client Node Autonomy	Section 15: Client Node Technical Specification	Deployment reqs, Fulfilment Layer 12-step sequence
Event-Driven Architecture	Section 15.4: Event Management	Subscription format, notification schema
End-to-End Data Exchange Flow	Section 16: End-to-End Transaction Flow	11-step transaction sequence; transport envelopes
Conformance	Section 17: Conformance Test Requirements	CT-AUTH through CT-FULFIL test categories
Entity Onboarding	Section 18: Entity Onboarding Technical Requirements	8-stage onboarding; PKI certificate types
Audit and Logging	Section 19: Monitoring, Logging, and Audit	Log schema, mandatory events, retention policy
Key Management (RA Section 14.2)	Section 14.2a: Client Category Key Management Framework	4-category risk-differentiated key protection replacing blanket HSM mandate per RA §14.2
Cryptographic Agility and PQC (RA Section 6, Section 14)	Section 14.3a: Cryptographic Agility and PQC Roadmap	Crypto Parameter Registry, Ed25519/ChaCha20, ML-KEM/ML-DSA/SLH-DSA staged roadmap
AI Integration Services; MCP (RA Section 12)	Section 24: AI Integration Services Technical Specification	AI identity binding, model registry, MCP surface, governance guardrails, delegated-consent scaffolding

Billing and Monetization (RA Section 7.1.10)	Section 25: Billing and Monetization Technical Requirements	Billing architecture, usage metering; detailed billing TS deferred to separate DNP-D supplement
Custom API Marketplace; API Gateway; Consent Layer; TPL; Orchestration Services (RA Sections 7.1.3, 7.1.4, 7.1.5, 7.1.6, 7.1.7)	Section 26: Delegated Channel Integration Pattern	Delegated Channel credential model, six-step interaction flow, composite workflow state machine, consent handling, and TPL recording for third-party delegated applications (Super Apps, composite service platforms). No RA modification required; operates entirely within existing RA component model.

Annex D — Revision History And Forward Compatibility

This Technical Specification is a live document. Publication of a version does not signify that its contents are final or immutable. As WASL implementation progresses, as technology evolves, and as operational experience accumulates, this document will be updated to reflect new requirements, clarifications, and improvements. Revisions are issued in accordance with the PDA Standards Board publication process defined in DNP-X.001 FWK, under which working drafts, pilot drafts, and published versions each carry a defined maturity designation.

However, each published version carries full normative force for the period during which it is current. All Participating Entities, platform operators, and implementers **SHALL** comply with the requirements of the version in effect at the time of their onboarding and **SHALL** transition to a superseding version within the transition window specified in that version's release notice. Continued operation under a superseded version beyond its stated transition deadline constitutes non-compliance with WASL participation obligations.

Implementers and Participating Entities are advised to subscribe to PDA update notifications at standards.dnp.gov.pk and to verify they are working from the current published version before commencing or extending any WASL integration. Each substantive revision is recorded in the revision history table below, with a description of the changes made and the approval authority.

D.1 Version History

Version	Date	Description	Approved by
1.0	18 April 2026	Initial publication: technical specification	

D.2 Expected Revision Cadence

Revision Type	Triggers	Backward Compatible?	Migration Period
Minor (e.g., 1.1)	New optional fields, clarifications, additional error/purpose codes	Yes	None required
Patch (e.g., 1.0.1)	Corrections, editorial changes, security advisories	Yes	None required

Major (e.g., 2.0)	Breaking changes to envelope formats, API signatures, or authentication requirements	No	Minimum 12 months dual-version support

D.3 Security Advisory Process

Critical security vulnerabilities identified in this specification shall be addressed via an expedited advisory process outside the normal revision cycle. Security Advisories (SA) on the standards portal shall be published, and all registered participating entities directly within 24 hours of a CRITICAL severity finding will be notified.

Bibliography

The following non-normative references support the interpretation and implementation of this Technical Specification. Entries are prefixed "b-" per DNP-X.001 Annex A.6.

[b-1] Pakistan Digital Authority. DNP-D.002 RA — WASL Reference Architecture.

[b-2] Pakistan Digital Authority. DNP-X.001 FWK — Nomenclature, Series Structure, and Issuance Framework (03/2026).

[b-3] UNDP / Office of the UN Secretary-General's Envoy on Technology. The Universal DPI Safeguards Framework, 2023.

[b-4] G20. Framework for Systems of Digital Public Infrastructure, 2023.

[b-5] Republic of Estonia. X-Road Technical Specifications (reference implementation).

[b-6] European Commission. European Interoperability Framework (EIF).

[b-7] IETF. RFC 8446 — The Transport Layer Security (TLS) Protocol Version 1.3, 2018.

[b-8] IETF. RFC 8705 — OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens.

Machine-Readable Metadata (JSON-LD)

The following JSON-LD block is the structured metadata record for this publication, as required by DNP-X.001 clause 13. It accompanies the publication package as metadata.jsonld.


```
{
  "@context": "https://schema.org/",
  "@type": "TechArticle",
  "identifier": "DNP-D.300 TS (04/2026)",
  "urn": "urn:dnp:D:300:TS:0.1:en",
  "publishingAuthority": { "code": "PDA", "name": "Pakistan Digital
Authority" },
  "title": "WASL — Pakistan National Data Exchange Layer: Technical
Specification",
  "titleUrdu": "وصل — پاکستان قومی ڈیٹا ایکسچینج لیئر: تکنیکی تفصیلات",
  "series": "D", "type": "TS", "status": "WD",
```

```
  "maturityLevel": "PILOT", "classification": "PUBLIC",
  "normativeReferences": [
    "DNP-X.001 FWK","DNP-D.002 RA","IETF RFC 8446","IETF RFC 8705",
    "OAuth 2.1","OpenID Connect Core 1.0","ISO/IEC 27001","FIPS 140-2/3"
  ],
  "dateApproved": null, "dateEffective": null, "dateReview": "2029-04-
26",
  "jurisdiction": "PK", "gazetteReference": "N/A",
  "persistentUrl": "https://standards.dnp.gov.pk/DNP-D.300", "doi":
"pending",
  "parentDocument": "DNP-D.002 RA"
}
```