IMPERIUM-X1: TECHNICAL FACT SHEET

OMNIINGEST ABDM 2.0 (PHASE 0.1)

> ***Role & Intent:*** *Senior Health-Tech Architect | Independent Builder* ***Compliance Standard:*** *ABDM 2.0 | FHIR R5 | DPDP Act 2023*

1. SYSTEM ARCHITECTURE

A lightweight, compliance-first ingestion engine designed to bridge legacy hospital data (CSV/SQL) with India's National Health Stack (ABDM) without the overhead of heavy enterprise middleware.

- **Core Engine:** Python 3.12 (Type-Safe)
- **Interface:** Streamlit (Rapid Prototyping UI)
- **Data Standard:** HL7 FHIR R5 (Strict Schema Validation)
- **Security:** AES-256 Encryption (At-Rest) & TLS 1.3 (In-Transit Simulation)

2. CRITICAL COMPLIANCE FEATURES

A. FHIR R5 Native Adoption

Unlike many systems still on R4, OmniIngest is built on **R5** primitives. This is critical for future-proofing patient demographics handling.

Technical Evidence: *Handling the breaking change in R5 where `name` is strictly a list of complex types.*

```
# src/fhir_mapper.py (Snippet) def map_patient_r5(row): """ Strict R5
Compliance: 'name' attribute must be a list of HumanName objects. Legacy R4
string concatenation is rejected here. """ return { "resourceType":
"Patient", "id": generate_uuid(row['id']), "name": [ { "use": "official",
"family": row['last_name'], "given": [row['first_name']], # R5 Requirement:
explicit periods for initials if applicable } ], "gender":
normalize_gender(row['gender']), # Maps 'M' -> 'male' per ValueSet
"birthDate": format_date(row['dob']) # ISO-8601 YYYY-MM-DD }
```

B. The "Kill Switch" (DPDP Rule 8.3)

Compliance with the Digital Personal Data Protection (DPDP) Act 2023 requires more than database deletion. We implement **Cryptographic Shredding**.

Technical Evidence: *Simulating the destruction of encryption keys to render PII unrecoverable instantly.*

```
# src/security/crypto_shredder.py import secrets def
execute_kill_switch(session_id): """ DPDP Rule 8.3 Compliance: Instead of
slow disk overwrites, we destroy the ephemeral AES-256 key associated with
the session. The data blob remains but is mathematically garbage. """ # 1.
Locate Session Key target_key = key_store.get(session_id) # 2. Cryptographic
Shredding (Key Destruction) # Overwriting memory location with random bytes
before release ctypes.memset(id(target_key), 0, sys.getsizeof(target_key))
del key_store[session_id] # 3. Log Immutable Audit Trail
audit_logger.log_event( event_type="DATA_PRINCIPAL_REVOCATION",
severity="CRITICAL", message=f"Key destroyed for Session {session_id}. Data
unrecoverable." )
```

C. Immutable Audit Lineage

Every action in the system creates a hash-chained log entry, ensuring non-repudiation for forensic audits.

```
// logs/audit/audit_2026.json { "timestamp": "2026-01-15T10:00:01Z", "actor":
"admin_sys_01", "action": "INGEST_BATCH", "target_resource":
"Patient/sbx-5501", "compliance_check": "PASSED", "integrity_hash":
"e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855" }
```

3. ROADMAP (PHASE 0.2)

- **Vector Search**: Integrating vector embeddings for semantic patient matching.
- **Auto-consent**: Smart contract implementation for data access requests.
- **Edge deployment**: Docker containerization for on-premise hospital servers.