**API6 to SDP Data Processing Architecture**

**Technical Documentation**

**Table of Contents**

1. [Executive Summary](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#executive-summary)
2. [System Overview](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#system-overview)
3. [Architecture Components](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#architecture-components)
4. [Data Flow Process](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#data-flow-process)
5. [Configuration Management](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#configuration-management)
6. [Offset Management](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#offset-management)
7. [Error Handling and Recovery](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#error-handling-and-recovery)
8. [Reconciliation Process](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#reconciliation-process)
9. [Data Governance](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#data-governance)
10. [Operational Considerations](https://claude.ai/chat/462b5b0a-a574-47de-846f-000ff61cc3a9#operational-considerations)

**1. Executive Summary**

This document describes the end-to-end data processing architecture for streaming logs from the API6 Gateway through Kafka to the Hadoop cluster (SDP), with final distribution to tenant-specific Hive databases within the SIP (AIF) platform. The system implements a robust ETL pipeline with comprehensive error handling, offset management, and daily reconciliation processes to ensure data integrity and completeness.

**Key Objectives**

* Real-time log ingestion from LLM services via API6 Gateway
* Reliable streaming through Kafka messaging infrastructure
* Multi-tenant data segregation and distribution
* Comprehensive offset tracking and data reconciliation
* Automated error handling and recovery mechanisms

**2. System Overview**

**2.1 High-Level Architecture**

The architecture consists of several interconnected layers that work together to process and distribute log data:

**Layer 1: API Gateway Layer**

* API6 Gateway serves as the front-end interface for all LLM server requests
* Captures and logs all API transactions and interactions
* Generates structured log entries for downstream processing

**Layer 2: Message Streaming Layer**

* Kafka topic receives synchronized logs from API6
* Provides distributed, fault-tolerant message queuing
* Enables real-time data streaming to processing systems

**Layer 3: Data Processing Layer**

* Hadoop cluster (SDP) hosts all data processing components
* SIP (AIF) platform manages staging and tenant-specific databases
* Batch jobs orchestrate data movement and transformation

**Layer 4: Data Persistence Layer**

* Staging Hive database for initial data landing
* Tenant-specific Hive databases for final data distribution
* Metadata tables for offset management and error tracking

**2.2 Technology Stack**

**Infrastructure Components:**

* **API6**: API Gateway for LLM services
* **AIHPC Server**: LLM server infrastructure
* **Kafka**: Distributed streaming platform
* **Hadoop (SDP)**: Big data processing cluster
* **Hive**: Data warehouse infrastructure
* **SIP (AIF)**: Analytics and Intelligence Framework

**3. Architecture Components**

**3.1 API6 Gateway (API Gateway)**

**Purpose**: Serves as the primary entry point for all requests to the LLM servers, providing centralized access control, monitoring, and logging capabilities.

**Key Functions:**

* Request routing to AIHPC server infrastructure
* Authentication and authorization management
* API request and response logging
* Performance metrics collection
* Traffic management and rate limiting

**Log Generation:**

* Captures detailed information about each API transaction
* Records request metadata, timestamps, client information, and response details
* Generates structured log entries in real-time
* Synchronizes logs to Kafka topic for downstream processing

**3.2 Kafka Topic**

**Purpose**: Provides a distributed, fault-tolerant messaging system for streaming logs from API6 to the data processing infrastructure.

**Key Characteristics:**

* Persistent message storage with configurable retention
* High-throughput, low-latency message delivery
* Partition-based parallelism for scalability
* Offset-based message tracking for reliable consumption
* Built-in replication for fault tolerance

**Message Flow:**

* Receives log messages from API6 Gateway
* Maintains ordered message sequences within partitions
* Supports multiple consumer groups for parallel processing
* Preserves messages for reconciliation and recovery

**3.3 Hadoop Cluster (SDP)**

**Purpose**: Serves as the foundational infrastructure for distributed data storage and processing within the organization.

**Key Capabilities:**

* Distributed file system (HDFS) for large-scale data storage
* Resource management through YARN
* Support for various processing frameworks
* High availability and fault tolerance
* Scalable compute and storage resources

**SDP Environment:**

* Hosts all SIP (AIF) components
* Provides isolated environments for staging and production data
* Supports batch and streaming processing workloads
* Enables multi-tenant data segregation

**3.4 SIP (AIF) Platform**

**Purpose**: Analytics and Intelligence Framework that manages data organization, processing, and distribution across multiple tenants.

**Components within SIP (AIF):**

**3.4.1 Staging Hive Database**

* **Role**: Initial landing zone for all log data from Kafka
* **Characteristics**:
  + Single, centralized database for all incoming logs
  + Temporary storage before tenant distribution
  + Contains raw, unprocessed log data
  + Optimized for high-volume writes
  + Minimal schema constraints for flexible ingestion

**3.4.2 Tenant Hive Databases**

* **Role**: Dedicated databases for each client/tenant
* **Characteristics**:
  + Isolated data storage per tenant
  + Enforces data segregation and security
  + Customizable schema per tenant requirements
  + Optimized for analytical queries
  + Long-term data retention

**3.4.3 Unprocessed\_Table**

* **Role**: Error quarantine for records with invalid or missing tenant mappings
* **Characteristics**:
  + Captures all records that fail tenant mapping validation
  + Preserves original log data for reprocessing
  + Includes error metadata and timestamps
  + Enables manual intervention and correction
  + Supports automated reprocessing after configuration updates

**3.5 Batch Job Process (ETL Pipeline)**

**Purpose**: Orchestrates the movement and transformation of data from staging to tenant-specific databases.

**Key Responsibilities:**

* Read log data from Staging Hive database
* Validate tenant mapping using configuration file
* Route data to appropriate tenant databases
* Update offset management metadata
* Handle errors and invalid records
* Monitor processing metrics and performance

**Processing Workflow:**

1. Initiate scheduled batch execution
2. Read configuration file for tenant mappings
3. Query staging database for new records
4. Validate each record against tenant mapping
5. Write valid records to tenant databases
6. Update offset management table
7. Move invalid records to Unprocessed\_Table
8. Generate processing logs and metrics

**3.6 Configuration File**

**Purpose**: Maintains the authoritative mapping between clients and their designated tenant Hive databases.

**Structure and Content:**

* Client identifier to tenant database mapping
* Database connection details
* Schema mapping specifications
* Data routing rules
* Version control and change history

**Management Considerations:**

* Centralized configuration management
* Version-controlled updates
* Validation before deployment
* Rollback capabilities
* Audit trail of all changes

**Update Process:**

* Configuration changes trigger validation checks
* Approved changes deployed to processing environment
* Batch jobs reload updated configuration
* Reprocessing initiated for previously failed records

**3.7 Offset Management Table**

**Purpose**: Tracks the processing state of Kafka messages and their distribution to tenant databases.

**Schema Components:**

* Kafka topic partition identifier
* Kafka message offset
* Tenant database identifier
* Processing timestamp
* Record count per offset
* Processing status

**Key Functions:**

* Maintains exactly-once processing semantics
* Enables recovery from processing failures
* Supports data lineage tracking
* Facilitates reconciliation processes
* Provides audit trail for compliance

**4. Data Flow Process**

**4.1 Stage 1: Log Generation and Collection**

**Step 1: API Request Processing**

* LLM server (AIHPC) receives incoming requests
* API6 Gateway intercepts and processes all API calls
* Gateway performs authentication, authorization, and routing
* Each transaction generates comprehensive log entries

**Step 2: Log Synchronization to Kafka**

* API6 Gateway serializes log data into message format
* Messages published to designated Kafka topic
* Kafka assigns partition and offset to each message
* Messages persisted to disk with configured retention
* Acknowledgment sent back to API6 upon successful write

**4.2 Stage 2: Data Streaming to Staging Database**

**Step 3: Kafka to Staging Database Stream**

* Streaming consumer connects to Kafka topic
* Consumer reads messages in order from assigned partitions
* Log data transformed into Hive-compatible format
* Records written to Staging Hive database in SIP (AIF)
* Consumer commits offsets upon successful write

**Streaming Characteristics:**

* Near real-time data ingestion (typically seconds of latency)
* Parallel processing across multiple Kafka partitions
* At-least-once delivery guarantee
* Automatic retry logic for transient failures
* Backpressure handling during peak loads

**4.3 Stage 3: Batch Processing and Distribution**

**Step 4: Configuration Loading**

* Batch job initiates at scheduled interval
* Configuration file loaded into memory
* Tenant mapping rules validated and parsed
* Database connection pools established
* Processing metrics initialized

**Step 5: Data Reading from Staging**

* Query staging database for unprocessed records
* Filter based on last processed offset/timestamp
* Read data in configurable batch sizes
* Apply data quality checks and validations
* Maintain processing state for recovery

**Step 6: Tenant Mapping and Validation**

* Extract client identifier from each log record
* Lookup corresponding tenant database in configuration
* Validate mapping exists and is active
* Categorize records as valid or invalid
* Prepare data for respective destinations

**Step 7: Data Distribution**

**For Valid Mappings:**

1. Transform data according to tenant schema
2. Write records to designated tenant Hive database
3. Record successful write count and details
4. Update offset management table with:
   * Kafka offset processed
   * Target tenant database
   * Record count written
   * Processing timestamp
   * Status as 'COMPLETED'

**For Invalid Mappings:**

1. Preserve original log data without modification
2. Add error metadata (reason, timestamp, client ID)
3. Write records to Unprocessed\_Table
4. Record failure details in processing logs
5. Generate alerts for operational monitoring

**4.4 Stage 4: Offset Management Update**

**Step 8: Metadata Update**

* For each successfully processed batch:
  + Insert new record in offset management table
  + Include Kafka offset range processed
  + Link to tenant database destination
  + Record processing timestamp
  + Update processing statistics

**Offset Table Benefits:**

* Enables idempotent processing
* Supports exactly-once semantics
* Facilitates data lineage tracking
* Provides recovery point in case of failures
* Enables audit and compliance reporting

**4.5 Stage 5: Error Recovery**

**Step 9: Unprocessed Data Handling**

* Records in Unprocessed\_Table await configuration updates
* Operational team reviews and updates configuration file
* Once configuration corrected:
  1. Reprocessing job triggered (manual or automated)
  2. Records read from Unprocessed\_Table
  3. Tenant mapping re-validated
  4. Valid records moved to appropriate tenant databases
  5. Still-invalid records remain in Unprocessed\_Table
  6. Processing metadata updated accordingly

**5. Configuration Management**

**5.1 Configuration File Structure**

The configuration file serves as the central registry for client-to-tenant database mappings. It must be:

**Well-Structured:**

* Organized hierarchically for easy navigation
* Includes comprehensive metadata for each mapping
* Version-controlled with change tracking
* Documented with comments and descriptions

**Validated:**

* Schema validation before deployment
* Reference integrity checks
* Duplicate detection
* Completeness verification

**Maintained:**

* Regular reviews for accuracy
* Timely updates for new tenants
* Decommissioning of inactive mappings
* Archive of historical configurations

**5.2 Configuration Attributes**

**Client Information:**

* Client unique identifier
* Client name and description
* Contact information
* Service tier and SLA requirements

**Tenant Database Mapping:**

* Tenant Hive database name
* Database location within SDP
* Schema version
* Access credentials (encrypted)
* Connection parameters

**Data Routing Rules:**

* Filtering criteria
* Transformation specifications
* Validation rules
* Special processing requirements

**Operational Metadata:**

* Mapping effective date
* Last update timestamp
* Update by (user/system)
* Active/inactive status
* Comments and notes

**5.3 Configuration Update Process**

**Phase 1: Request and Approval**

1. Configuration change request submitted
2. Impact analysis performed
3. Approval obtained from stakeholders
4. Change scheduled during maintenance window

**Phase 2: Implementation**

1. Configuration file updated in source control
2. Automated validation tests executed
3. Deployment to staging environment
4. Integration testing performed
5. Production deployment approved

**Phase 3: Activation**

1. Updated configuration deployed to production
2. Batch jobs reload configuration
3. Monitoring enabled for processing changes
4. Reprocessing triggered for affected records

**Phase 4: Validation**

1. Verify data flowing to correct tenant databases
2. Confirm offset management updates
3. Check Unprocessed\_Table for reduction
4. Review processing logs and metrics
5. Generate configuration change report

**5.4 Configuration Governance**

**Access Control:**

* Role-based access to configuration files
* Audit logging of all configuration access
* Separation of duties for updates vs. approvals
* Secure storage with encryption

**Version Control:**

* All changes tracked in version control system
* Rollback capability to previous versions
* Branch management for testing changes
* Release tagging for deployments

**Documentation:**

* Comprehensive documentation of all mappings
* Change history with rationale
* Standard operating procedures
* Troubleshooting guides

**6. Offset Management**

**6.1 Purpose and Importance**

The offset management system provides critical functionality for:

**Data Integrity:**

* Prevents duplicate processing of messages
* Ensures exactly-once semantics
* Maintains data lineage and traceability
* Supports data quality validation

**Operational Resilience:**

* Enables recovery from processing failures
* Provides restart capability at specific points
* Supports rollback scenarios
* Facilitates disaster recovery

**Compliance and Auditing:**

* Tracks complete processing history
* Provides audit trail for regulations
* Enables data governance reporting
* Supports investigation of data issues

**6.2 Offset Management Table Schema**

**Primary Attributes:**

**Kafka Source Information:**

* Kafka topic name
* Partition number
* Starting offset
* Ending offset
* Message count in range

**Target Information:**

* Tenant Hive database name
* Table name
* Record count written
* Write timestamp

**Processing Metadata:**

* Batch job identifier
* Processing start timestamp
* Processing end timestamp
* Processing duration
* Processing status (IN\_PROGRESS, COMPLETED, FAILED)

**Data Quality Metrics:**

* Records read
* Records written
* Records failed
* Data volume (bytes)
* Validation errors count

**6.3 Offset Update Process**

**During Normal Processing:**

1. **Pre-Processing:**
   * Query offset table for last processed offset per partition
   * Determine starting offset for current batch
   * Mark batch as IN\_PROGRESS in offset table
2. **Processing:**
   * Read and process messages from Kafka
   * Track offsets being processed
   * Maintain in-memory offset state
3. **Post-Processing:**
   * Upon successful write to tenant database
   * Insert new offset record with COMPLETED status
   * Include all relevant metadata
   * Commit transaction to ensure atomicity

**During Error Scenarios:**

1. **Partial Failure:**
   * Record offsets successfully processed
   * Mark failed offsets with FAILED status
   * Log error details and context
   * Preserve state for retry
2. **Complete Failure:**
   * Rollback offset updates
   * Maintain last known good state
   * Enable restart from last successful offset
   * Generate alerts for investigation

**6.4 Offset-Based Recovery**

**Scenario 1: Processing Interruption**

* Query offset table for last COMPLETED record per partition
* Resume processing from next offset
* Continue normal processing flow
* Ensure no data loss or duplication

**Scenario 2: Data Reprocessing**

* Identify offset range requiring reprocessing
* Mark existing offsets as REPROCESSING
* Execute processing for specified range
* Update offset records upon completion
* Maintain audit trail of reprocessing

**Scenario 3: Tenant Migration**

* Identify offsets processed to old tenant database
* Configure new tenant database mapping
* Optionally reprocess historical data
* Update offset records with new tenant reference
* Validate data consistency

**7. Error Handling and Recovery**

**7.1 Error Categories**

**Category 1: Configuration Errors**

* **Cause**: Missing or invalid tenant mapping in configuration file
* **Detection**: During tenant lookup in batch processing
* **Impact**: Records cannot be routed to tenant database
* **Resolution**: Update configuration file with correct mapping

**Category 2: Data Quality Errors**

* **Cause**: Malformed log data, missing required fields
* **Detection**: During data validation in batch job
* **Impact**: Records fail validation checks
* **Resolution**: Investigate data source, correct upstream issues

**Category 3: Infrastructure Errors**

* **Cause**: Database connectivity issues, resource constraints
* **Detection**: During database write operations
* **Impact**: Processing halts or slows significantly
* **Resolution**: Infrastructure team investigates and resolves

**Category 4: Processing Errors**

* **Cause**: Batch job failures, logic errors, resource exhaustion
* **Detection**: Through job monitoring and alerting
* **Impact**: Data processing stops, backlog accumulates
* **Resolution**: Investigate root cause, restart processing

**7.2 Unprocessed\_Table**

**Purpose:** The Unprocessed\_Table serves as a quarantine area for records that cannot be processed due to configuration or data issues, preventing data loss while allowing time for resolution.

**Table Structure:**

**Source Data:**

* Complete original log record from staging database
* Kafka offset reference
* Partition identifier
* Original timestamp

**Error Metadata:**

* Error type/category
* Error message and details
* Client identifier extracted (if available)
* Processing attempt timestamp
* Processing job identifier

**Status Tracking:**

* Processing status (PENDING, IN\_REVIEW, REPROCESSING, RESOLVED)
* Number of reprocessing attempts
* Last processing attempt timestamp
* Resolution notes
* Resolved by (user/system)

**Data Lifecycle:**

1. **Initial Write:**
   * Record moved from staging database
   * Error details captured
   * Status set to PENDING
   * Alert generated for operations team
2. **Investigation:**
   * Operations team reviews error
   * Root cause identified
   * Resolution plan determined
   * Status updated to IN\_REVIEW
3. **Resolution:**
   * Configuration file updated (if needed)
   * Data corrections applied (if needed)
   * Status updated to REPROCESSING
   * Reprocessing job triggered
4. **Reprocessing:**
   * Record read from Unprocessed\_Table
   * Tenant mapping re-validated
   * If valid: Move to tenant database, update offset table
   * If still invalid: Remain in Unprocessed\_Table, increment attempt count
   * Status updated to RESOLVED or remains PENDING
5. **Archival:**
   * Successfully resolved records archived
   * Configurable retention period
   * Historical data maintained for audit
   * Periodic cleanup of old resolved records

**7.3 Reprocessing Workflow**

**Trigger Conditions:**

* Configuration file updated with new/corrected mappings
* Manual reprocessing request from operations
* Scheduled periodic reprocessing job
* Infrastructure issues resolved

**Reprocessing Steps:**

1. **Preparation:**
   * Identify records eligible for reprocessing
   * Filter by status, error type, or time range
   * Load updated configuration
   * Validate infrastructure readiness
2. **Execution:**
   * Read records from Unprocessed\_Table in batches
   * Apply current tenant mapping logic
   * Validate data quality
   * Route to appropriate destination
3. **Disposition:**
   * **Success**: Write to tenant database, update offset table, mark as RESOLVED
   * **Still Invalid**: Keep in Unprocessed\_Table, update attempt count, log new error
   * **New Error**: Capture new error details, update status
4. **Reporting:**
   * Generate reprocessing summary
   * Report success/failure counts
   * Identify remaining issues
   * Provide recommendations for resolution

**7.4 Error Prevention Strategies**

**Configuration Validation:**

* Pre-deployment validation of configuration changes
* Automated testing of mapping rules
* Staging environment testing before production
* Gradual rollout of configuration updates

**Data Quality Checks:**

* Upstream data validation at API6 Gateway
* Schema enforcement in Kafka messages
* Early detection of malformed data
* Data profiling and anomaly detection

**Monitoring and Alerting:**

* Real-time monitoring of error rates
* Alerts for abnormal error patterns
* Dashboard visibility into Unprocessed\_Table growth
* Proactive notification before thresholds exceeded

**Capacity Planning:**

* Infrastructure sizing for peak loads
* Resource reservation for critical jobs
* Scalability testing under stress
* Auto-scaling configurations

**8. Reconciliation Process**

**8.1 Purpose and Objectives**

The end-of-day reconciliation process ensures data completeness and integrity by comparing:

* Messages available in Kafka topic
* Records processed and logged in offset management table
* Data written to tenant Hive databases

**Key Objectives:**

* Verify all Kafka messages have been processed
* Confirm offset management table accuracy
* Detect any data loss or duplication
* Validate processing completeness
* Generate compliance and audit reports

**8.2 Reconciliation Job Execution**

**Scheduling:**

* Executed at end of each business day
* Scheduled during low-activity periods
* Sufficient time allocated for completion
* Automated execution with manual override capability

**Pre-Execution Checks:**

* Verify all batch processing jobs completed
* Confirm Kafka topic stability
* Validate database connectivity
* Ensure sufficient resources available

**8.3 Reconciliation Algorithm**

**Phase 1: Data Collection**

**From Kafka:**

* Query Kafka for message count per partition
* Retrieve offset ranges for the day
* Calculate total messages available
* Capture Kafka metadata (timestamp ranges, partitions)

**From Offset Management Table:**

* Query for all offset records for the day
* Sum total messages processed per partition
* Calculate aggregate record counts
* Identify any gaps in offset sequences

**From Tenant Databases:**

* Query each tenant database for record counts
* Filter by timestamp for the current day
* Aggregate across all tenant databases
* Collect data volume statistics

**Phase 2: Comparison and Analysis**

**Kafka vs. Offset Table:**

* Compare message counts per partition
* Identify missing offset ranges
* Detect any unprocessed messages
* Calculate processing lag

**Offset Table vs. Tenant Databases:**

* Verify offset records match tenant database writes
* Compare record counts
* Check for any orphaned offset records
* Validate data distribution

**Completeness Check:**

* Total messages in Kafka = Total in offset table = Total in tenant DBs
* All offset ranges continuous with no gaps
* All partitions processed
* All tenant databases updated

**Phase 3: Discrepancy Resolution**

**Missing Records:**

* Identify specific offset ranges not processed
* Check if records in Unprocessed\_Table
* Verify no processing job failures
* Trigger reprocessing if needed

**Count Mismatches:**

* Investigate root cause of discrepancies
* Check for duplicate processing
* Verify data quality issues
* Review batch job logs

**Offset Gaps:**

* Identify missing offset ranges
* Determine if processing skipped offsets
* Verify no Kafka message expiration
* Plan recovery processing

**8.4 Reconciliation Reporting**

**Summary Report:**

* Total messages in Kafka for the day
* Total messages processed
* Total messages in tenant databases
* Total messages in Unprocessed\_Table
* Reconciliation status (PASS/FAIL)

**Detailed Analysis:**

* Per-partition reconciliation results
* Per-tenant database statistics
* Processing lag metrics
* Error and exception summary

**Discrepancy Details:**

* List of unprocessed offset ranges
* Count mismatches by partition
* Missing data incidents
* Recommended actions

**Compliance Information:**

* Data completeness percentage
* Processing SLA adherence
* Data retention compliance
* Audit trail confirmation

**8.5 Reconciliation Actions**

**Upon Successful Reconciliation:**

* Archive reconciliation report
* Update operational dashboards
* Notify stakeholders of successful completion
* Trigger any dependent processes

**Upon Failed Reconciliation:**

* Generate high-priority alerts
* Notify on-call operations team
* Create incident ticket
* Initiate investigation process
* Hold dependent processes until resolution

**Follow-Up Actions:**

* Schedule reprocessing for missing data
* Update offset management table as needed
* Communicate with stakeholders about any data gaps
* Implement corrective measures to prevent recurrence

**9. Data Governance**

**9.1 Data Quality Management**

**Data Quality Dimensions:**

**Completeness:**

* All log entries captured by API6
* No message loss in Kafka
* All records processed to tenant databases
* Verified through reconciliation

**Accuracy:**

* Correct tenant routing
* Accurate offset tracking
* Proper data transformations
* Validation against business rules

**Consistency:**

* Uniform data formats across tenants
* Consistent processing logic
* Standardized error handling
* Synchronized metadata

**Timeliness:**

* Near real-time streaming from Kafka
* Batch processing within SLA windows
* End-of-day reconciliation completion
* Timely error resolution

**Validity:**

* Schema validation at ingestion
* Data type enforcement
* Referential integrity checks
* Business rule validation

**9.2 Data Lineage and Traceability**

**Source Tracking:**

* Original API request captured by API6
* Kafka partition and offset recorded
* Processing timestamps maintained
* Tenant database destination logged

**Transformation Tracking:**

* Processing job identifiers
* Configuration version used
* Data transformations applied
* Enrichment or filtering logged

**Audit Trail:**

* Complete processing history in offset table
* Configuration change history
* Reprocessing events documented
* Error and resolution tracking

**9.3 Data Security and Privacy**

**Access Controls:**

* Role-based access to databases
* Separate credentials per tenant
* Encrypted connections
* API Gateway authentication

**Data Segregation:**

* Dedicated tenant databases
* No cross-tenant data access
* Isolated processing contexts
* Network segmentation

**Encryption:**

* Data in transit encryption (TLS/SSL)
* Data at rest encryption in Hive
* Credential encryption in configuration
* Secure key management

**Compliance:**

* Data retention policies enforced
* Privacy regulations adherence
* Regular security audits
* Incident response procedures

**9.4 Data Retention and Archival**

**Staging Database:**

* Short-term retention (e.g., 7-30 days)
* Automatic purging after successful processing
* Exception: Failed records retained longer
* Configurable retention policy

**Tenant Databases:**

* Long-term retention based on tenant requirements
* Archival to cold storage after defined period
* Compliance-driven retention schedules
* Automated lifecycle management

**Offset Management Table:**

* Retain for audit and compliance period
* Historical data for trend analysis
* Archival of old records
* Purge after retention period

**Unprocessed\_Table:**

* Retain until resolved
* Archive resolved records
* Periodic cleanup of old entries
* Long-term retention for investigation

**Kafka Messages:**

* Configurable retention (e.g., 7-30 days)
* Sufficient for reconciliation needs
* Balance storage costs vs. recovery requirements
* Compaction for key-based topics

**10. Operational Considerations**

**10.1 Monitoring and Alerting**

**Key Performance Indicators (KPIs):**

**Throughput Metrics:**

* Messages per second through Kafka
* Records per hour processed to tenant databases
* Batch job execution time
* End-to-end latency

**Quality Metrics:**

* Processing success rate
* Error rate by category
* Unprocessed\_Table growth rate
* Reconciliation pass rate

**Availability Metrics:**

* API6 Gateway uptime
* Kafka cluster availability
* Hadoop/SDP cluster health
* Database availability

**Resource Metrics:**

* CPU and memory utilization
* Disk space usage
* Network bandwidth
* Queue depths

**Alert Conditions:**

* Processing lag exceeds threshold
* Error rate spikes
* Unprocessed\_Table growth exceeds limit
* Reconciliation failures
* Infrastructure component failures
* SLA violations

**10.2 Capacity Planning**

**Growth Considerations:**

* Projected increase in API traffic
* New tenant onboarding
* Data volume growth trends
* Retention policy changes

**Scalability Planning:**

* Kafka partition strategy
* Hadoop cluster expansion
* Database sizing and partitioning
* Batch job parallelization

**Resource Allocation:**

* Compute resources for batch jobs
* Storage for staging and tenant databases
* Network bandwidth for data transfer
* Buffer capacity for peak loads

**10.3 Performance Optimization**

**Kafka Optimization:**

* Optimal partition count
* Message batching configuration
* Compression settings
* Consumer group tuning

**Database Optimization:**

* Hive table partitioning strategy
* Indexing for frequent queries
* Statistics maintenance
* Query optimization

**Batch Processing Optimization:**

* Parallel processing tuning
* Batch size optimization
* Resource allocation tuning
* Caching strategies

**10.4 Disaster Recovery**

**Backup Strategy:**

* Configuration file backups
* Offset management table backups
* Tenant database backups
* Kafka message retention

**Recovery Procedures:**

* Kafka replay from offsets
* Database restoration from backups
* Configuration rollback
* Processing resumption from last known good state

**Business Continuity:**

* Failover to secondary Kafka cluster
* Database replication
* Multi-region deployment considerations
* RTO/RPO definitions

**10.5 Maintenance Windows**

**Scheduled Maintenance:**

* Configuration updates
* System upgrades
* Database maintenance
* Performance tuning

**Maintenance Procedures:**

* Pre-maintenance backups
* Processing pause/resume
* Validation after maintenance
* Rollback procedures if needed

**10.6 Troubleshooting Guide**

**Common Issues and Resolutions:**

**Issue: Increasing lag in processing**

* Check batch job performance
* Review resource utilization
* Verify Kafka consumer group health
* Scale processing resources if needed

**Issue: Growing Unprocessed\_Table**

* Review error patterns
* Check configuration file accuracy
* Investigate data quality issues
* Prioritize configuration updates

**Issue: Reconciliation failures**

* Review processing logs
* Check for infrastructure issues
* Verify offset table consistency
* Manually investigate discrepancies

**Issue: Slow query performance**

* Review query execution plans
* Check table statistics
* Verify partitioning effectiveness
* Consider index additions

**10.7 Change Management**

**Change Control Process:**

1. Change request documentation
2. Impact assessment
3. Stakeholder approval
4. Testing in non-production environment
5. Scheduled deployment
6. Post-deployment validation
7. Documentation update

**Communication:**

* Advance notice of maintenance
* Status updates during deployment
* Post-deployment summary
* Lessons learned documentation

**Conclusion**

This architecture provides a robust, scalable, and resilient solution for processing and distributing log data from API6 Gateway through Kafka to multi-tenant Hive databases. The comprehensive offset management, error handling, and reconciliation processes ensure data integrity and completeness while supporting operational excellence.

**Key Success Factors:**

* Reliable message streaming through Kafka
* Accurate tenant mapping configuration
* Comprehensive offset tracking
* Effective error handling and recovery
* Daily reconciliation for data validation
* Strong data governance practices
* Proactive monitoring and alerting
* Clear operational procedures

**Future Enhancements:**

* Real-time stream processing
* Advanced analytics on processing metrics
* Machine learning for anomaly detection
* Self-healing automation
* Enhanced multi-region support
* Improved data quality frameworks

**Document Control**

* **Version**: 1.0
* **Last Updated**: October 2025
* **Document Owner**: Data Engineering Team
* **Review Cycle**: Quarterly
* **Classification**: Internal Use