**SIA ETL - Testing, Validation, and Deployment Framework**

**Document Overview**

This document provides comprehensive guidelines for Unit Testing, Pre-Load Validations, Break Testing, Error Reporting, GitHub repository structure, and ECP deployment for the modernized Java-based SIA ETL process. This framework emphasizes SIAExtract logging functionality integration and ensures consistency with legacy SIA processing.

**1. Unit Testing Strategy**

**1.1 Overview**

**Objective:** Validate individual components of the SIA ETL process with emphasis on SIAExtract logging functions and SIA-specific transformation logic.

**1.2 Testing Framework and Tools**

**Core Testing Stack:**

* **JUnit 5** - Test case development
* **Mockito** - Mocking dependencies and external services
* **PowerMock** - Testing static methods and legacy code integration
* **AssertJ** - Fluent, readable assertions
* **DBUnit** - Database testing and state management
* **Testcontainers** - Isolated database test environments
* **Maven Surefire** - Test execution and reporting
* **JaCoCo** - Code coverage measurement (target: minimum 85% coverage)

**1.3 Key Testing Scenarios**

**1.3.1 SIAExtract Functionality Testing**

**Purpose:** Validate correct data extraction from SIA source systems and SIAExtract logging function integration.

**Test Implementation:**

@Test

public void testSiaExtractExecution() {

// Arrange

SiaSourceConnection mockSource = mock(SiaSourceConnection.class);

when(mockSource.isAvailable()).thenReturn(true);

when(mockSource.extractData()).thenReturn(createTestSiaData());

// Act

SIAExtract extractor = new SIAExtract(mockSource);

SiaExtractResult result = extractor.execute();

// Assert

assertThat(result.isSuccessful()).isTrue();

assertThat(result.getRecordCount()).isEqualTo(expectedRecordCount);

assertThat(result.getExtractLog()).isNotNull();

}

@Test

public void testSiaExtractLoggingFunction() {

// Test SIAExtract logging captures all required audit information

SIAExtract extractor = new SIAExtract();

SIAExtractLogger logger = extractor.getLogger();

// Execute extraction

extractor.execute();

// Validate logging

List<LogEntry> logEntries = logger.getLogEntries();

assertThat(logEntries).isNotEmpty();

assertThat(logEntries).anyMatch(e -> e.getLogLevel() == LogLevel.INFO);

assertThat(logEntries).anyMatch(e -> e.getMessage().contains("Extraction started"));

assertThat(logEntries).anyMatch(e -> e.getMessage().contains("Extraction completed"));

}

@Test

public void testSiaExtractLogFormat() {

// Verify log entry format matches legacy SIAExtract format

SIAExtractLogger logger = new SIAExtractLogger();

LocalDateTime timestamp = LocalDateTime.now();

String message = "Test extraction message";

logger.log(LogLevel.INFO, message);

String logOutput = logger.getFormattedLog();

// Validate format: [TIMESTAMP] [LEVEL] MESSAGE

assertThat(logOutput).matches("\\[\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}\\] \\[INFO\\] .+");

}

@Test

public void testSiaExtractionFilters() {

// Test extraction filters and selection criteria

LocalDate filterDate = LocalDate.of(2025, 10, 1);

SiaExtractionCriteria criteria = new SiaExtractionCriteria();

criteria.setEffectiveDateAfter(filterDate);

SIAExtract extractor = new SIAExtract(criteria);

List<SiaRecord> extracted = extractor.extract();

// All records should meet filter criteria

assertThat(extracted).allMatch(r ->

!r.getEffectiveDate().isBefore(filterDate));

}

@Test

public void testIncrementalVsFullExtract() {

// Test incremental vs. full extract logic

SIAExtract extractor = new SIAExtract();

// Full extract

extractor.setExtractMode(ExtractMode.FULL);

SiaExtractResult fullResult = extractor.execute();

int fullCount = fullResult.getRecordCount();

// Incremental extract

extractor.setExtractMode(ExtractMode.INCREMENTAL);

extractor.setIncrementalDate(LocalDate.now().minusDays(1));

SiaExtractResult incrementalResult = extractor.execute();

int incrementalCount = incrementalResult.getRecordCount();

// Incremental should be subset of full

assertThat(incrementalCount).isLessThanOrEqualTo(fullCount);

}

@Test

public void testSiaSourceConnectionHandling() {

// Test source system connection handling and error cases

SiaSourceConnection mockSource = mock(SiaSourceConnection.class);

when(mockSource.isAvailable()).thenReturn(false);

SIAExtract extractor = new SIAExtract(mockSource);

// Should throw exception when source unavailable

assertThatThrownBy(() -> extractor.execute())

.isInstanceOf(SiaSourceUnavailableException.class)

.hasMessageContaining("SIA source system is unavailable");

}

**Test Cases:**

* Correct data extraction from SIA source systems
* SIAExtract logging function integration verified
* Extraction filters and selection criteria tested
* Incremental vs. full extract logic validated
* Source system connection handling tested
* Error cases explicitly covered

**1.3.2 SIAExtract Logging Validation**

**Purpose:** Comprehensive testing of SIAExtract logging functionality.

**Test Implementation:**

@Test

public void testSiaExtractLogEntryCapture() {

// Verify all required audit information captured

SIAExtract extractor = new SIAExtract();

SIAExtractLogger logger = extractor.getLogger();

extractor.execute();

List<LogEntry> entries = logger.getLogEntries();

// Verify required log entries exist

assertThat(entries).anyMatch(e -> e.getEventType() == EventType.EXTRACTION\_START);

assertThat(entries).anyMatch(e -> e.getEventType() == EventType.EXTRACTION\_COMPLETE);

assertThat(entries).anyMatch(e -> e.containsRecordCount());

assertThat(entries).anyMatch(e -> e.containsSourceSystemInfo());

}

@Test

public void testSiaExtractLogTimestamps() {

// Validate log timestamps are accurate

LocalDateTime beforeExtract = LocalDateTime.now();

SIAExtract extractor = new SIAExtract();

extractor.execute();

LocalDateTime afterExtract = LocalDateTime.now();

SIAExtractLogger logger = extractor.getLogger();

List<LogEntry> entries = logger.getLogEntries();

// All timestamps should be within extraction window

assertThat(entries).allMatch(e ->

!e.getTimestamp().isBefore(beforeExtract) &&

!e.getTimestamp().isAfter(afterExtract));

}

@Test

public void testSiaExtractLogUnderErrorConditions() {

// Test logging under normal and error conditions

SiaSourceConnection mockSource = mock(SiaSourceConnection.class);

when(mockSource.extractData()).thenThrow(new RuntimeException("Extraction error"));

SIAExtract extractor = new SIAExtract(mockSource);

SIAExtractLogger logger = extractor.getLogger();

try {

extractor.execute();

} catch (Exception e) {

// Expected

}

// Error should be logged

List<LogEntry> entries = logger.getLogEntries();

assertThat(entries).anyMatch(e ->

e.getLogLevel() == LogLevel.ERROR &&

e.getMessage().contains("Extraction error"));

}

@Test

public void testSiaExtractLogCorrelation() {

// Test log correlation with process execution

String correlationId = UUID.randomUUID().toString();

SIAExtract extractor = new SIAExtract();

extractor.setCorrelationId(correlationId);

extractor.execute();

SIAExtractLogger logger = extractor.getLogger();

List<LogEntry> entries = logger.getLogEntries();

// All log entries should have correlation ID

assertThat(entries).allMatch(e ->

e.getCorrelationId().equals(correlationId));

}

@Test

public void testSiaExtractLogRotation() {

// Test log rotation and retention behavior

SIAExtractLogger logger = new SIAExtractLogger();

logger.setMaxLogEntries(100);

// Generate 150 log entries

for (int i = 0; i < 150; i++) {

logger.log(LogLevel.INFO, "Test message " + i);

}

// Should only retain last 100 entries

List<LogEntry> entries = logger.getLogEntries();

assertThat(entries).hasSize(100);

assertThat(entries.get(0).getMessage()).contains("Test message 50");

}

@Test

public void testLegacySiaExtractLogCompatibility() {

// Ensure compatibility with legacy SIAExtract log format

SIAExtractLogger modernLogger = new SIAExtractLogger();

modernLogger.log(LogLevel.INFO, "Extraction started");

modernLogger.log(LogLevel.INFO, "Records extracted: 50000");

modernLogger.log(LogLevel.INFO, "Extraction completed");

String modernLog = modernLogger.getFormattedLog();

String expectedLegacyFormat = loadLegacyLogFormat();

// Log format should match legacy expectations

assertThat(modernLog).containsPattern(expectedLegacyFormat);

}

**Test Cases:**

* Log entry format and content validated
* Log timestamps accuracy verified
* Logging under error conditions tested
* Log correlation with process execution
* Log rotation functionality tested
* Legacy SIAExtract log format compatibility verified

**1.3.3 Record Count and Data Integrity Testing**

**Purpose:** Validate record counts and data preservation through transformation pipeline.

**Test Implementation:**

@Test

public void testSiaRecordCountVerification() {

// Arrange

int expectedRecordCount = 55000;

MockSiaSource mockSource = new MockSiaSource(expectedRecordCount);

// Act

SIAExtract extractor = new SIAExtract(mockSource);

SiaExtractResult result = extractor.execute();

// Assert

assertThat(result.getRecordCount()).isEqualTo(expectedRecordCount);

assertThat(result.getExtractedRecords()).hasSize(expectedRecordCount);

}

@Test

public void testSiaPassThroughMapping() {

// Validate pass-through mapping record counts

List<SiaRecord> sourceRecords = createSiaTestData(1000);

SiaMapper mapper = new SiaMapper();

List<SiaRecord> mappedRecords = mapper.mapPassThrough(sourceRecords);

assertThat(mappedRecords).hasSize(sourceRecords.size());

}

@Test

public void testSiaMinusQuery() {

// Execute minus queries to detect unexpected data changes

List<SiaRecord> sourceData = loadSiaSourceData();

List<SiaRecord> targetData = loadSiaTargetData();

SiaDataComparator comparator = new SiaDataComparator();

List<DataDifference> differences = comparator.performMinusQuery(sourceData, targetData);

// Only expected differences should exist

assertThat(differences).allMatch(d -> d.isExpectedDifference());

}

@Test

public void testSiaDataIntegrityThroughPipeline() {

// Test data preservation through transformation pipeline

List<SiaRecord> originalRecords = createSiaTestData(500);

// Extract -> Transform -> Load simulation

SiaTransformer transformer = new SiaTransformer();

List<SiaRecord> transformedRecords = transformer.transform(originalRecords);

// Critical fields should be preserved

for (int i = 0; i < originalRecords.size(); i++) {

SiaRecord original = originalRecords.get(i);

SiaRecord transformed = transformedRecords.get(i);

assertThat(transformed.getRecordId()).isEqualTo(original.getRecordId());

assertThat(transformed.getCustomerId()).isEqualTo(original.getCustomerId());

assertThat(transformed.getTransactionDate()).isEqualTo(original.getTransactionDate());

}

}

@Test

public void testSiaChecksumValidation() {

// Test checksum and hash validation where applicable

SiaRecord record = createSiaRecord();

String expectedChecksum = calculateChecksum(record);

SiaChecksumValidator validator = new SiaChecksumValidator();

boolean isValid = validator.validate(record, expectedChecksum);

assertThat(isValid).isTrue();

}

**Test Cases:**

* Record counts between source and target verified
* Pass-through mappings maintain exact counts
* Minus queries detect only expected differences
* Data preservation through pipeline validated
* Checksum validation where applicable

**1.3.4 SIA-Specific Business Logic Testing**

**Purpose:** Test SIA-specific calculations, transformations, and business rules.

**Test Implementation:**

@Test

public void testSiaCalculationLogic() {

// Test SIA-specific calculation and derivation logic

SiaRecord record = createSiaRecord();

record.setBaseAmount(new BigDecimal("1000.00"));

record.setTaxRate(new BigDecimal("0.08"));

SiaBusinessRules rules = new SiaBusinessRules();

BigDecimal totalAmount = rules.calculateTotalAmount(record);

// Total = Base + (Base \* Tax Rate)

assertThat(totalAmount).isEqualByComparingTo("1080.00");

}

@Test

public void testSiaTransformationRules() {

// Test complex transformations unique to SIA

SiaRecord rawRecord = createRawSiaRecord();

rawRecord.setStatus("A");

rawRecord.setEffectiveDate(LocalDate.of(2025, 1, 1));

SiaTransformer transformer = new SiaTransformer();

SiaRecord transformed = transformer.transform(rawRecord);

// Validate transformation logic

assertThat(transformed.getIsActive()).isTrue();

assertThat(transformed.getEffectiveDate()).isEqualTo(LocalDate.of(2025, 1, 1));

}

@Test

public void testSiaDataEnrichment() {

// Test supplemental data integration

SiaRecord baseRecord = createSiaRecord();

baseRecord.setCustomerId("CUST-12345");

SiaEnrichmentService enrichmentService = new SiaEnrichmentService();

SiaRecord enrichedRecord = enrichmentService.enrich(baseRecord);

// Enriched fields should be populated

assertThat(enrichedRecord.getCustomerName()).isNotNull();

assertThat(enrichedRecord.getCustomerSegment()).isNotNull();

}

@Test

public void testSiaDerivedFields() {

// Test derived field calculations

SiaRecord record = createSiaRecord();

record.setTransactionDate(LocalDate.of(2025, 10, 15));

SiaFieldDerivation derivation = new SiaFieldDerivation();

derivation.deriveFields(record);

// Derived fields should be calculated

assertThat(record.getTransactionYear()).isEqualTo(2025);

assertThat(record.getTransactionMonth()).isEqualTo(10);

assertThat(record.getTransactionQuarter()).isEqualTo(4);

}

@Test

public void testSiaLookupIntegration() {

// Test lookup and reference data integration

String accountCode = "ACC-001";

SiaLookupService lookupService = new SiaLookupService();

SiaAccountInfo accountInfo = lookupService.getAccountInfo(accountCode);

assertThat(accountInfo).isNotNull();

assertThat(accountInfo.getAccountCode()).isEqualTo(accountCode);

assertThat(accountInfo.getAccountType()).isNotNull();

}

@Test

public void testSiaAggregationLogic() {

// Test aggregation and summarization logic

List<SiaTransaction> transactions = createSiaTransactions(100);

SiaAggregator aggregator = new SiaAggregator();

SiaSummary summary = aggregator.aggregate(transactions);

// Validate aggregation

assertThat(summary.getTotalCount()).isEqualTo(100);

BigDecimal expectedTotal = transactions.stream()

.map(SiaTransaction::getAmount)

.reduce(BigDecimal.ZERO, BigDecimal::add);

assertThat(summary.getTotalAmount()).isEqualByComparingTo(expectedTotal);

}

**Test Cases:**

* SIA-specific calculation logic validated
* Transformation rules tested
* Data enrichment processes verified
* Derived field calculations tested
* Lookup and reference data integration validated
* Aggregation and summarization logic tested

**1.3.5 SQL Query and Database Interaction Testing**

**Purpose:** Validate SQL queries, database operations, and performance.

**Test Implementation:**

@Test

public void testSiaQueryExecution() {

// Test SQL query execution

SiaQueryExecutor executor = new SiaQueryExecutor();

String query = loadSiaExtractQuery();

List<SiaRecord> results = executor.executeQuery(query);

assertThat(results).isNotEmpty();

}

@Test

public void testSiaFilterQueries() {

// Test filter conditions and where clauses

LocalDate filterDate = LocalDate.of(2025, 10, 1);

SiaQueryBuilder queryBuilder = new SiaQueryBuilder();

queryBuilder.addDateFilter("effective\_date", filterDate, FilterOperator.GREATER\_THAN\_OR\_EQUAL);

String query = queryBuilder.build();

SiaQueryExecutor executor = new SiaQueryExecutor();

List<SiaRecord> results = executor.executeQuery(query);

// All results should meet filter criteria

assertThat(results).allMatch(r -> !r.getEffectiveDate().isBefore(filterDate));

}

@Test

public void testSiaJoinLogic() {

// Test join logic and relationship handling

List<SiaRecord> siaRecords = loadSiaRecords();

List<SiaCustomer> customers = loadCustomers();

SiaJoinProcessor joinProcessor = new SiaJoinProcessor();

List<SiaRecordWithCustomer> joined = joinProcessor.joinRecordsWithCustomers(siaRecords, customers);

assertThat(joined).hasSize(siaRecords.size());

assertThat(joined).allMatch(j -> j.getCustomer() != null);

}

@Test

public void testSiaTransactionManagement() {

// Test transaction management and rollback

SiaTransactionManager txManager = new SiaTransactionManager();

txManager.beginTransaction();

try {

SiaRecord record = createSiaRecord();

txManager.insert(record);

// Simulate error

if (record.getAmount().compareTo(BigDecimal.ZERO) < 0) {

throw new IllegalArgumentException("Negative amount not allowed");

}

txManager.commit();

} catch (Exception e) {

txManager.rollback();

}

// Verify transaction state

assertThat(txManager.getTransactionState()).isEqualTo(TransactionState.COMPLETED);

}

@Test

public void testSiaBatchProcessing() {

// Test batch processing operations

List<SiaRecord> records = createSiaRecords(10000);

int batchSize = 1000;

SiaBatchProcessor batchProcessor = new SiaBatchProcessor(batchSize);

BatchProcessingResult result = batchProcessor.processBatch(records);

assertThat(result.getTotalProcessed()).isEqualTo(10000);

assertThat(result.getBatchCount()).isEqualTo(10);

}

@Test

public void testSiaDatabaseConstraintHandling() {

// Test database constraint handling

SiaRecord record = createSiaRecord();

record.setRecordId("DUPLICATE-ID");

SiaLoader loader = new SiaLoader();

// First insert should succeed

assertThatCode(() -> loader.insert(record)).doesNotThrowAnyException();

// Duplicate insert should throw constraint violation

assertThatThrownBy(() -> loader.insert(record))

.isInstanceOf(ConstraintViolationException.class);

}

@Test

public void testSiaQueryPerformance() {

// Verify query performance meets requirements

int recordCount = 100000;

List<SiaRecord> testData = generateSiaTestData(recordCount);

long startTime = System.currentTimeMillis();

SiaQueryExecutor executor = new SiaQueryExecutor();

executor.executeComplexAggregation(testData);

long duration = System.currentTimeMillis() - startTime;

// Should complete within 45 seconds for 100k records

assertThat(duration).isLessThan(45000);

}

**Test Cases:**

* SQL query execution tested
* Filter conditions validated
* Join logic verified
* Transaction management tested
* Batch processing operations validated
* Database constraint handling tested
* Query performance verified

**1.3.6 Error Handling and Exception Testing**

**Purpose:** Comprehensive testing of error handling and recovery scenarios.

**Test Implementation:**

@Test

public void testSiaNullPointerHandling() {

// Test handling of null pointer exceptions

SiaRecord record = new SiaRecord();

record.setCustomerId(null); // Intentional null

SiaValidator validator = new SiaValidator();

ValidationResult result = validator.validate(record);

assertThat(result.isValid()).isFalse();

assertThat(result.getErrors()).anyMatch(e ->

e.getFieldName().equals("customerId") &&

e.getErrorType() == ErrorType.NULL\_VALUE);

}

@Test

public void testSiaDatabaseExceptionHandling() {

// Validate database exception handling

SiaLoader loader = new SiaLoader();

DatabaseConnection mockConnection = mock(DatabaseConnection.class);

when(mockConnection.execute(any())).thenThrow(new SQLException("Connection lost"));

loader.setConnection(mockConnection);

assertThatThrownBy(() -> loader.load(createSiaRecords(10)))

.isInstanceOf(SiaLoadException.class)

.hasCauseInstanceOf(SQLException.class);

}

@Test

public void testSiaFileIOExceptionHandling() {

// Test file I/O error conditions

String nonExistentFile = "/path/to/nonexistent/sia\_file.dat";

SiaFileReader reader = new SiaFileReader();

assertThatThrownBy(() -> reader.readFile(nonExistentFile))

.isInstanceOf(FileNotFoundException.class);

}

@Test

public void testSiaTimeoutHandling() {

// Verify timeout and retry logic

SiaSourceConnection mockSource = mock(SiaSourceConnection.class);

when(mockSource.extractData())

.thenAnswer(invocation -> {

Thread.sleep(120000); // 2 minutes - exceeds timeout

return null;

});

SIAExtract extractor = new SIAExtract(mockSource);

extractor.setTimeout(60000); // 1 minute timeout

assertThatThrownBy(() -> extractor.execute())

.isInstanceOf(SiaExtractionTimeoutException.class);

}

@Test

public void testSiaGracefulDegradation() {

// Test graceful degradation scenarios

SiaEnrichmentService enrichmentService = mock(SiaEnrichmentService.class);

when(enrichmentService.enrich(any())).thenThrow(new RuntimeException("Enrichment service down"));

SiaTransformer transformer = new SiaTransformer(enrichmentService);

transformer.setGracefulDegradation(true);

SiaRecord record = createSiaRecord();

// Should not throw exception, should process with reduced functionality

assertThatCode(() -> transformer.transform(record)).doesNotThrowAnyException();

}

**Test Cases:**

* Null pointer exception handling tested
* Database exception handling validated
* File I/O error conditions tested
* Timeout and retry logic verified
* Graceful degradation scenarios tested

**1.4 Test Coverage Requirements**

**Coverage Targets:**

* Minimum 85% line coverage for all SIA modules
* 100% coverage for critical business logic components
* All public methods require test coverage
* Edge cases and boundary conditions explicitly tested
* Legacy integration points (SIAExtract) fully tested
* Error handling paths covered

**1.5 Mock Data Strategy**

**Test Data Organization:**

src/test/resources/test-data/sia/

├── sia-extracts/

│ ├── sia\_extract\_small\_001.dat (100 records)

│ ├── sia\_extract\_medium\_001.dat (10,000 records)

│ └── sia\_extract\_large\_001.dat (100,000 records)

├── sia-logs/

│ ├── sia\_extract\_log\_001.log

│ └── sia\_extract\_log\_002.log

├── edge-cases/

│ ├── empty\_extract.dat

│ ├── single\_record.dat

│ ├── null\_values.dat

│ └── special\_characters.dat

└── reference-data/

├── sia\_accounts.dat

└── sia\_customers.dat

**Data Generation Strategy:**

* Create representative SIA test datasets
* Include SIAExtract log samples
* Maintain test data version control
* Automated refresh from sanitized production samples
* Mock SIAExtract logging output for various scenarios

**1.6 Legacy Parity Testing**

**SIAExtract Parity Validation:**

@Test

public void testSiaExtractLegacyParity() {

// Ensure parity with legacy SIAExtract

List<SiaRecord> testData = loadSiaTestDataset();

// Simulate legacy SIAExtract

LegacySIAExtractSimulator legacyExtractor = new LegacySIAExtractSimulator();

SiaExtractResult legacyResult = legacyExtractor.execute(testData);

// Modern SIAExtract

SIAExtract modernExtractor = new SIAExtract();

SiaExtractResult modernResult = modernExtractor.execute(testData);

// Outputs should match

assertThat(modernResult.getRecordCount()).isEqualTo(legacyResult.getRecordCount());

assertThat(modernResult.getExtractedRecords()).isEqualTo(legacyResult.getExtractedRecords());

}

@Test

public void testSiaExtractLoggingLegacyParity() {

// Verify logging output matches legacy format

SIAExtractLogger modernLogger = new SIAExtractLogger();

modernLogger.log(LogLevel.INFO, "Extraction started");

String modernLogOutput = modernLogger.getFormattedLog();

String expectedLegacyFormat = loadLegacySiaExtractLogFormat();

assertThat(modernLogOutput).matchesPattern(expectedLegacyFormat);

}

**2. Pre-Load Validation Checklist**

**2.1 Overview**

**Purpose:** Automated validation of SIA extracts before transformation and loading, with particular attention to SIAExtract logging requirements and SIA-specific data quality rules.

**Legacy Reference:**

* SIAExtract validation and logging functions
* SIA-specific validation rules from legacy environment

**2.2 File and Metadata Validation**

**✓ Required File Verification**

**Validation Rules:**

* Verify presence of all expected SIA extract files
* Validate SIAExtract output files are available
* Check for required metadata or control files
* Confirm log files from SIAExtract process exist
* Alert if any critical files missing beyond grace period (15 minutes)

**Expected File Patterns:**

* Extract: SIA\_YYYYMMDD\_HHMMss.dat
* Control: SIA\_YYYYMMDD\_HHMMss.ctl
* Log: SIA\_EXTRACT\_YYYYMMDD\_HHMMss.log
* Metadata: SIA\_YYYYMMDD\_HHMMss.meta

**Error Actions:**

* Halt processing if SIA extract file missing
* Send alert to SIA ETL support team
* Log missing file details with expected paths
* Cross-reference with SIAExtract log to determine if extraction occurred

**✓ SIAExtract Log File Validation**

**Validation Rules:**

* Verify SIAExtract log file exists and is accessible
* Validate log entries for current processing run
* Check for error or warning messages in logs
* Verify completion status logged by SIAExtract
* Cross-reference log timestamps with file timestamps
* Ensure all required audit information present in logs

**Expected Log Content:**

[2025-10-07 00:15:00] [INFO] SIA Extraction started

[2025-10-07 00:15:01] [INFO] Source system: SIA-PROD-DB

[2025-10-07 00:15:01] [INFO] Extraction criteria: effective\_date >= 2025-10-06

[2025-10-07 00:45:23] [INFO] Records extracted: 55000

[2025-10-07 00:45:23] [INFO] Extraction completed successfully

[2025-10-07 00:45:23] [INFO] Output file: SIA\_20251007\_001500.dat

**Error Actions:**

* Alert if SIAExtract log missing
* Flag if log contains error messages
* Cross-validate log record count with actual file count
* Escalate if extraction completion not logged

**✓ File Accessibility and Permissions**

**Validation Rules:**

* Verify read permissions on all source files
* Validate write permissions on staging areas
* Check file ownership and access controls
* Confirm no file locks preventing access

**Error Actions:**

* Alert on permission issues
* Log access control violations
* Escalate to infrastructure team if needed

**✓ File Naming Convention Validation**

**Validation Rules:**

* Validate files follow naming standard
* Verify sequence numbers (if multi-part files)
* Check file extensions match expected types (.dat, .log, .ctl, .meta)
* Validate timestamp in filename is reasonable

**Error Actions:**

* Reject files with non-conforming names
* Log naming violations
* Alert on unexpected file patterns

**✓ File Date and Timestamp Validation**

**Validation Rules:**

* Verify file creation dates align with processing schedule
* Validate file modification timestamps are recent
* Check for stale or outdated files
* Ensure files represent current processing period
* Compare file timestamps with SIAExtract log entries

**Expected Time Windows:**

* SIA extract expected: 12:00 AM - 1:00 AM daily
* Grace period: 15 minutes
* Maximum file age: 48 hours

**Error Actions:**

* Alert on late-arriving files
* Flag stale files for investigation
* Require approval for old files

**✓ File Integrity and Completeness**

**Validation Rules:**

* Calculate and verify file checksums (MD5 or SHA-256)
* Compare with checksum values in control files
* Validate file sizes within expected ranges
* Check for file truncation (compare with control totals)
* Verify complete file transfer from source

**Historical Baselines:**

* Average SIA file size: 120-180 MB
* Alert if deviation > 25% from baseline

**Error Actions:**

* Reject corrupted files
* Alert on checksum mismatches
* Log integrity check failures

**2.3 File Structure and Schema Validation**

**✓ Header and Footer Record Validation**

**Validation Rules:**

* Verify header record structure and required metadata
* Validate header contains expected version and format information
* Check footer/trailer record exists and well-formed
* Validate record counts in footer match actual record count
* Verify control totals and checksums in footer

**Header Format:**

HDR|SIA|2025-10-07|VERSION\_1.9|RECORD\_COUNT|EXTRACT\_TIMESTAMP

**Footer Format:**

TRL|RECORD\_COUNT|TOTAL\_AMOUNT|CHECKSUM|END\_TIMESTAMP

**Error Actions:**

* Reject file if header/footer missing
* Alert on count discrepancies
* Log header/footer validation failures

**✓ Schema Conformance**

**Validation Rules:**

* Validate number of columns matches expected schema
* Verify column order aligns with specification
* Check delimiter consistency (pipe-delimited)
* Validate field widths for fixed-width files (if applicable)
* Ensure all required fields present in schema

**SIA Expected Schema:**

RECORD\_ID|ACCOUNT\_ID|CUSTOMER\_ID|TRANSACTION\_DATE|AMOUNT|TAX\_AMOUNT|TOTAL\_AMOUNT|STATUS|EFFECTIVE\_DATE|EXPIRY\_DATE|CREATE\_TIMESTAMP

**Error Actions:**

* Reject file if schema doesn't match
* Log schema mismatch details
* Generate schema comparison report

**✓ Data Type Pre-Validation**

**Validation Rules:**

* Sample records to verify data types match expectations
* Check numeric fields contain only valid numeric characters
* Validate date fields follow expected format (YYYY-MM-DD)
* Verify text fields within expected length limits
* Check for unexpected special characters or encoding issues

**Error Actions:**

* Log data type mismatches
* Sample and report invalid records
* Halt if error rate exceeds threshold

**2.4 Data Quality and Business Rule Validation**

**✓ Mandatory Field Validation**

**Validation Rules:**

* Verify all required SIA business fields are populated
* Check for null values in mandatory columns
* Validate primary key fields are never null
* Ensure critical business identifiers present

**Mandatory Fields for SIA:**

* RECORD\_ID, ACCOUNT\_ID, CUSTOMER\_ID, TRANSACTION\_DATE, AMOUNT, STATUS, CREATE\_TIMESTAMP

**Error Actions:**

* Reject records with null mandatory fields
* Log null violations with record identifiers
* Generate data quality exception report
* Quarantine invalid records

**✓ Data Type and Format Validation**

**Validation Rules:**

* Validate numeric fields contain valid numbers (no alpha characters)
* Verify date fields use standard format (YYYY-MM-DD)
* Check timestamp fields include time zone information
* Validate decimal precision for currency fields
* Verify status fields contain only valid values (A=Active, I=Inactive, D=Deleted)
* Check string fields for invalid characters

**Field-Level Validations:**

| **Field Name** | **Data Type** | **Format** | **Validation Rule** |
| --- | --- | --- | --- |
| RECORD\_ID | String | SIA-[0-9]+ | Max 20 chars, pattern match |
| ACCOUNT\_ID | String | ACC-[0-9]+ | Max 15 chars, pattern match |
| CUSTOMER\_ID | Integer | Numeric | Positive integer, max 12 digits |
| AMOUNT | Decimal | 999999.99 | Precision(10,2), non-negative |
| TAX\_AMOUNT | Decimal | 99999.99 | Precision(8,2), non-negative |
| TRANSACTION\_DATE | Date | YYYY-MM-DD | Valid date, not future |
| EFFECTIVE\_DATE | Date | YYYY-MM-DD | Valid date |
| STATUS | Char | A/I/D | One of: A, I, D |

**Error Actions:**

* Log data type mismatches
* Quarantine invalid records
* Continue processing valid records if error rate < 5%
* Halt if error rate >= 5%

**✓ Referential Integrity Pre-Checks**

**Validation Rules:**

* Validate foreign key values exist in reference tables
* Check ACCOUNT\_ID exists in account master
* Verify CUSTOMER\_ID exists in customer master
* Test dimensional conformity with data warehouse

**Error Actions:**

* Flag orphaned records
* Alert on referential integrity violations
* Generate orphaned records report
* Route to data stewardship team

**✓ Business Logic Validation**

**Validation Rules:**

* Apply SIA-specific business rules and constraints
* Validate value ranges and boundary conditions
* Check logical consistency (e.g., EFFECTIVE\_DATE <= EXPIRY\_DATE)
* Verify TOTAL\_AMOUNT = AMOUNT + TAX\_AMOUNT
* Test cross-field validation rules

**SIA Business Rules:**

1. TRANSACTION\_DATE cannot be in the future
2. AMOUNT must be non-negative
3. EXPIRY\_DATE must be after or equal to EFFECTIVE\_DATE
4. TOTAL\_AMOUNT must equal AMOUNT + TAX\_AMOUNT (within $0.01 tolerance)
5. Active records (STATUS='A') must have valid EFFECTIVE\_DATE

**Error Actions:**

* Log business rule violations
* Quarantine violating records
* Generate business rule violation report

**✓ Duplicate Detection**

**Validation Rules:**

* Scan for duplicate primary keys (RECORD\_ID)
* Identify duplicate business keys (ACCOUNT\_ID + TRANSACTION\_DATE)
* Check for records violating unique constraints in target table

**Error Actions:**

* Log duplicates with full record details
* Reject duplicate records
* Generate duplicate records report

**2.5 Volume and Statistical Validation**

**✓ Count Reconciliation**

**Validation Rules:**

* Compare record counts with control file totals
* Validate counts against historical averages
* Flag significant deviations (>15% variance) for review
* Cross-check counts with SIAExtract log entries
* Validate counts by record type or category

**Historical Baselines:**

* SIA Daily: Average 50,000-60,000 records
* Alert if < 25,000 or > 120,000

**Error Actions:**

* Send warning alert for volume anomalies
* Require manual approval if outside thresholds
* Log count reconciliation with variance

**✓ Statistical Anomaly Detection**

**Validation Rules:**

* Calculate summary statistics (count, min, max, avg, std dev)
* Compare against established baselines
* Flag outliers requiring investigation
* Validate data completeness percentages

**Statistical Checks:**

* Average AMOUNT should be $50-$500
* Average TAX\_AMOUNT should be 5-10% of AMOUNT
* FLAG transactions with AMOUNT > $10,000 for review

**Error Actions:**

* Log statistical anomalies
* Generate anomaly report
* Continue processing but flag records

**2.6 SIAExtract Log Content Validation**

**✓ Log Completeness Verification**

**Validation Rules:**

* Verify all required audit information captured in SIAExtract log
* Validate record counts in logs match file counts
* Check for extraction errors or exceptions logged
* Verify source system connection details logged
* Ensure data lineage information captured

**Required Log Elements:**

* Extraction start timestamp
* Extraction end timestamp
* Source system identifier
* Extraction criteria/filters applied
* Record count extracted
* Completion status
* Output file name
* Any errors or warnings

**Error Actions:**

* Alert if log incomplete
* Flag missing required elements
* Cross-validate log with actual file

**✓ Audit Trail Completeness**

**Validation Rules:**

* Verify start and end times logged
* Validate user/service account information captured
* Check source system version information
* Verify extraction parameters and filters logged
* Ensure all SIAExtract logging functions from legacy replicated

**Error Actions:**

* Flag incomplete audit trail
* Log missing audit elements
* Escalate if critical information missing

**2.7 Final Pre-Load Readiness**

**✓ Data Conversion Compatibility**

**Validation Rules:**

* Final verification all source data types can convert to target types
* Validate no data truncation will occur during load
* Check for potential numeric overflow conditions
* Ensure character encoding compatibility (UTF-8)
* Verify date/time formats compatible with target database

**Error Actions:**

* Alert on conversion compatibility issues
* Prevent load if data would be truncated

**✓ Target Environment Readiness**

**Validation Rules:**

* Verify target database connectivity and availability
* Confirm sufficient storage space for load
* Validate write permissions on target tables
* Check for table locks or blocking conditions
* Verify staging area has adequate space

**Error Actions:**

* Alert if target unavailable
* Halt if insufficient storage
* Log environment readiness results

**✓ Dependency Validation**

**Validation Rules:**

* Verify all prerequisite processes completed successfully
* Check dependent reference data is loaded and current
* Validate dimension tables updated before fact load
* Ensure no conflicting processes running

**Error Actions:**

* Halt if dependencies not met
* Log dependency check failures

**2.8 Automated Response Actions**

**Critical Validation Failures:**

* Immediately halt processing
* Send urgent alert to SIA ETL support team
* Notify SIA business data owners
* Create P1 incident ticket
* Log detailed failure information
* Enable automatic retry after corrective action

**Data Quality Issues:**

* Quarantine records failing validation
* Log all validation failures
* Continue processing valid records if error rate < 5%
* Generate data quality exception report
* Route exceptions to data stewardship team

**Warning Conditions:**

* Log warnings to monitoring system
* Continue processing with flagged records
* Include warnings in post-process summary
* Route to appropriate team for follow-up

**3. Break Testing Methodology**

**3.1 Overview**

**Objective:** Comprehensively test SIA ETL resilience through intentional failure scenarios, stress testing, and edge case validation, with emphasis on SIAExtract integration and recovery capabilities.

**3.2 Negative Testing - File and Data Failures**

**3.2.1 Missing and Delayed File Testing**

**Test Scenarios:**

1. **SIA Extract File Missing**
   * Execute SIA ETL with primary extract file missing
   * Expected: Error detected within 3 minutes
   * Expected: SIAExtract log checked for extraction status
   * Expected: Processing halts, no partial commits
   * Expected: Alert sent to SIA support team
2. **SIAExtract Log File Missing**
   * Extract file present, but SIAExtract log missing
   * Expected: Validation warning flagged
   * Expected: Processing continues with reduced audit trail
   * Expected: Alert sent for missing log
3. **Control File Missing**
   * SIA data file present, control file missing
   * Expected: Count validation skipped or uses default
   * Expected: Warning logged
   * Expected: Processing continues or halts based on configuration
4. **Late-Arriving Files**
   * Files arrive 30 minutes after expected window
   * Expected: Late arrival detected from SIAExtract log
   * Expected: Automatic pickup after arrival
   * Expected: SLA reporting reflects late start

**Validation Criteria:**

* Error messages identify missing files accurately
* SIAExtract log consulted for extraction status
* No database modifications occur before validation
* Alerts contain file names, paths, and expected times

**3.2.2 Corrupted and Malformed File Testing**

**Test Scenarios:**

1. **Truncated Files**
   * SIA file cut off mid-record
   * Expected: Trailer count mismatch detected
   * Expected: Processing halts before load
   * Expected: Corruption location logged
2. **Malformed Records**
   * Records with missing delimiters
   * Records with extra columns
   * Expected: Schema validation catches issues
   * Expected: Line numbers logged
3. **Character Encoding Issues**
   * Mixed UTF-8 and ASCII encoding
   * Special characters causing parsing errors
   * Expected: Encoding errors detected
   * Expected: Character positions logged
4. **SIAExtract Log Corruption**
   * SIAExtract log file corrupted or incomplete
   * Expected: Log parsing handles gracefully
   * Expected: Warning flagged but processing continues
   * Expected: Alert sent to review log issues

**Validation Criteria:**

* Corruption location identified (file, line, column)
* Detailed error messages for troubleshooting
* No partial data loaded
* Graceful error handling

**3.3 Chaos Engineering - Infrastructure Failures**

**3.3.1 Database Failure Scenarios**

**Test Scenarios:**

1. **Source Database Unavailable During SIAExtract**
   * Simulate SIA source database down
   * Expected: SIAExtract logs connection error
   * Expected: Retry logic engages (3 attempts: 30s, 60s, 120s)
   * Expected: Critical alert after retries exhausted
2. **Connection Loss Mid-Transformation**
   * Database connection dropped during SIA processing
   * Expected: Transaction rolled back
   * Expected: No partial data committed
   * Expected: Error logged with transaction state
3. **Tablespace Full During Load**
   * Simulate target tablespace exhausted
   * Expected: Load fails with specific error
   * Expected: Rollback occurs
   * Expected: Alert includes storage details
4. **Deadlock Scenarios**
   * Simulate concurrent access causing deadlock
   * Expected: Deadlock detected
   * Expected: Transaction retried
   * Expected: Successful completion after retry

**Validation Criteria:**

* Retry logic functions as designed
* Database remains consistent
* All errors logged with full context
* Alerts sent to appropriate teams

**3.3.2 Network and Connectivity Chaos**

**Test Scenarios:**

1. **Network Interruption During File Transfer**
   * Simulate network drop during SIA file download
   * Expected: Transfer failure detected
   * Expected: Retry mechanism engages
   * Expected: Successful transfer after retry
2. **SIA Source System Unreachable**
   * SIAExtract unable to reach source
   * Expected: Timeout after configured period (120 seconds)
   * Expected: SIAExtract logs connection failure
   * Expected: Alert with source system details
3. **DNS Resolution Failures**
   * Simulate DNS unable to resolve hostnames
   * Expected: DNS error caught and logged
   * Expected: Fallback to IP address if configured

**Validation Criteria:**

* Network errors properly handled
* Retry attempts logged
* Alerts provide actionable information

**3.4 Boundary Value and Edge Case Testing**

**3.4.1 Volume Extremes**

**Test Scenarios:**

1. **Zero Records**
   * SIA file with zero transaction records
   * SIAExtract log shows 0 records extracted
   * Expected: Processing completes successfully
   * Expected: Appropriate logging
2. **Single Record**
   * SIA file with exactly 1 transaction
   * Expected: Record processed correctly
   * Expected: All validations execute
3. **Maximum Volume**
   * 10x normal volume: 600,000 SIA transactions
   * Expected: Processing completes within extended SLA (4 hours)
   * Expected: Memory usage within limits (< 12GB)
   * Expected: SIAExtract log shows high volume handled
4. **Large Individual Records**
   * Records with maximum field lengths
   * Expected: Records processed without truncation
   * Expected: No buffer overflow errors

**Validation Criteria:**

* All volume scenarios handled gracefully
* Memory consumption monitored and acceptable
* Processing time scales linearly
* No memory leaks

**3.4.2 SIA-Specific Edge Cases**

**Test Scenarios:**

1. **Missing SIAExtract Log Entries**
   * SIAExtract log missing expected entries
   * Expected: Warning flagged
   * Expected: Processing continues with reduced audit
2. **SIAExtract Log Shows Errors**
   * Log indicates extraction errors
   * Expected: Pre-load validation flags errors
   * Expected: Investigation triggered
   * Expected: Processing halts or continues based on severity
3. **Conflicting Record Counts**
   * SIAExtract log count vs. actual file count mismatch
   * Expected: Count reconciliation fails
   * Expected: Alert sent
   * Expected: Manual review required

**Validation Criteria:**

* SIAExtract integration edge cases handled
* Clear error messages
* Audit trail maintained

**3.4.3 Data Value Boundaries**

**Test Scenarios:**

1. **Numeric Boundaries**
   * AMOUNT: $0.00 (minimum), $999,999.99 (maximum)
   * TAX\_AMOUNT: $0.00, $99,999.99
   * Expected: All boundary values processed correctly
2. **Date Boundaries**
   * EFFECTIVE\_DATE: 1900-01-01, 2099-12-31
   * Leap year dates: 2024-02-29
   * Year-end: 2025-12-31
   * Expected: All dates validated and processed
3. **String Boundaries**
   * Empty strings vs. nulls
   * Maximum length strings (255 characters)
   * Special characters
   * Expected: Correct handling

**Validation Criteria:**

* No overflow or underflow errors
* Boundary values processed correctly
* Special characters handled

**3.5 Stress and Performance Testing**

**3.5.1 High Volume Processing**

**Test Scenarios:**

1. **Sustained High Volume**
   * 5x normal volume for 5 consecutive days
   * 300,000 SIA transactions/day
   * Expected: Consistent performance
   * Expected: No degradation over time
2. **Peak Month-End Volume**
   * Simulate month-end with 10x volume
   * Expected: Completion within extended SLA (4 hours)
   * Expected: Resource usage monitored
3. **SIAExtract Performance at High Volume**
   * Test SIAExtract with maximum data extraction
   * Expected: Extraction completes within timeout (30 minutes)
   * Expected: Log file remains manageable size

**Performance Benchmarks:**

| **Scenario** | **Normal Volume** | **Max Volume** | **SLA Target** | **Benchmark** |
| --- | --- | --- | --- | --- |
| SIA Extract | 55,000 records | 600,000 records | 30 minutes | 20 minutes |
| SIA Transform | 55,000 records | 600,000 records | 1.5 hours | 1 hour |
| SIA Load | 55,000 records | 600,000 records | 1.5 hours | 1 hour |
| End-to-End | Combined | Combined | 3 hours | 2 hours |

**Metrics to Track:**

* Records processed per minute (target: 50,000/min minimum)
* Memory consumption (average, peak)
* CPU utilization
* Database connection pool usage
* I/O throughput
* SIAExtract execution time

**Validation Criteria:**

* SLAs met even at high volumes
* Resource utilization within limits
* No memory leaks
* Linear performance scaling

**3.6 Recovery and Resilience Testing**

**3.6.1 Restart and Resume Capability**

**Test Scenarios:**

1. **Mid-Process Failure Recovery**
   * Simulate failure at 50% completion
   * Expected: Checkpoint created before failure
   * Expected: Resume from checkpoint on restart
   * Expected: No duplicate processing
2. **Transaction Rollback**
   * Force rollback during SIA load
   * Expected: Database returns to pre-load state
   * Expected: No orphaned records
   * Expected: Audit trail of rollback
3. **Idempotency Testing**
   * Run same SIA ETL twice with same input
   * SIAExtract log shows duplicate run
   * Expected: Identical results
   * Expected: No duplicate records created
4. **SIAExtract Log Recovery**
   * Test recovery when SIAExtract log lost
   * Expected: Processing continues with warning
   * Expected: Reduced audit trail noted

**Validation Criteria:**

* Checkpoints function correctly
* Rollback complete and consistent
* Restart successful
* Idempotency maintained
* SIAExtract logging preserved where possible

**3.7 Break Testing Execution**

**Testing Schedule:**

* Full suite: Quarterly
* Critical scenarios: Monthly
* Automated chaos: Weekly (dev/test)
* Performance regression: With each release
* SIAExtract integration: With each release

**Documentation:**

* All scenarios documented with expected outcomes
* Results logged with pass/fail status
* Failures tracked with remediation actions
* Performance metrics trended
* SIAExtract-specific tests documented

**4. Error Reporting and Notification**

**4.1 Overview**

**Purpose:** Comprehensive error identification, logging, communication, and tracking for SIA ETL processes, with particular emphasis on integrating SIAExtract logging functionality.

**4.2 Error Detection and Classification**

**SIA Error Categories:**

1. **SIAExtract and Source System Errors**
   * SIA source system connectivity failures
   * SIAExtract process execution failures
   * Source query timeout or performance issues
   * Authentication and authorization failures
   * Source data extraction incomplete or failed
   * SIAExtract logging function errors
2. **File System and I/O Errors**
   * SIA extract file not found or inaccessible
   * File permission or access control errors
   * File corruption or integrity check failures
   * Insufficient disk space for staging
   * File transfer or network errors
   * Control file missing or invalid
   * SIAExtract log file missing or corrupted
3. **Data Validation and Quality Errors**
   * Schema validation failures
   * Data type mismatches or conversion errors
   * Mandatory field null violations
   * Referential integrity constraint violations
   * Duplicate key violations
   * Business rule validation failures
   * Data format or encoding errors
4. **Transformation and Processing Errors**
   * SIA-specific transformation logic failures
   * Calculation or derivation errors
   * Lookup or reference data failures
   * Aggregation or summarization errors
   * Data enrichment process failures
   * Unexpected null pointer or runtime exceptions
5. **Database and Load Errors**
   * Target database connection failures
   * SQL execution errors during load
   * Constraint violations during insert/update
   * Transaction deadlock or timeout
   * Tablespace or storage exhaustion
   * Database performance degradation
6. **System and Infrastructure Errors**
   * Application out-of-memory errors
   * CPU or resource exhaustion
   * Thread pool exhaustion
   * Network connectivity failures
   * Container or orchestration platform errors
   * Unexpected system exceptions or crashes

**4.3 Error Logging Framework**

**Logging Technology Stack:**

* **SLF4J** (Simple Logging Facade for Java) API
* **Logback** as logging implementation
* **Structured JSON logging** for machine parsing
* **MDC** (Mapped Diagnostic Context) for correlation tracking
* **Asynchronous appenders** for performance
* **Centralized log aggregation** (Splunk or ELK Stack)

**SIAExtract Logging Integration:**

*Legacy Logging Preservation:*

* Maintain SIAExtract log format for backward compatibility
* Capture all audit information from legacy logging function
* Cross-reference modern logs with SIAExtract log entries
* Preserve extraction timestamps and source system metadata

*Enhanced Logging for Modernization:*

* Add structured logging on top of legacy SIAExtract logs
* Include correlation IDs linking extract logs to load logs
* Enrich with additional context (container ID, orchestration metadata)
* Maintain audit trail from source extraction through final load

**Detailed Log Entry Structure:**

{

"timestamp": "2025-10-07T00:47:15.234Z",

"processName": "SIA\_ETL",

"processId": "SIA-20251007-001",

"correlationId": "s1i2a3-4b5c6d-7e8f9g",

"severity": "ERROR",

"errorCode": "SIA-LOAD-005",

"errorCategory": "DATABASE\_ERROR",

"errorType": "CONSTRAINT\_VIOLATION",

"errorMessage": "Unique constraint violation on SIA\_PRIMARY\_KEY",

"recordIdentifier": "RECORD\_ID=SIA-98765, Record #3421",

"sourceFile": "SIA\_20251007\_001500.dat",

"targetTable": "SIA\_FACT\_TABLE",

"affectedRecordCount": 1,

"totalRecordCount": 55000,

"stackTrace": "java.sql.SQLException: ORA-00001...",

"siaExtractLog": {

"logReference": "SIA\_EXTRACT\_20251007\_001500.log",

"extractTimestamp": "2025-10-07T00:15:00Z",

"sourceSystem": "SIA-PROD-DB",

"extractedRecordCount": 55000

},

"additionalContext": {

"processingPhase": "LOAD",

"batchNumber": 18,

"retryAttempt": 1,

"databaseConnection": "PROD-DB-01",

"userContext": "svc\_sia\_etl"

},

"hostInfo": {

"hostname": "ecp-sia-pod-15",

"ipAddress": "10.20.30.55",

"container": "sia-etl-container-v1.9.2"

}

}

**Log Storage and Management:**

*Log File Organization:*

/app/logs/

├── sia-etl/

│ ├── sia\_etl\_20251007\_004500.log

│ ├── sia\_etl\_20251007\_004500\_error.log

│ └── archived/

│ └── sia\_etl\_20251006.log.gz

├── sia-extract/

│ ├── sia\_extract\_20251007\_001500.log (from SIAExtract)

│ └── archived/

└── audit/

└── sia\_audit\_20251007.log

*Log Retention Policy:*

* Online logs: 90 days in searchable format
* Archived logs: 1 year in compressed format
* Compliance logs: 7 years (per regulatory requirements)
* Critical incident logs: Permanent retention
* SIAExtract logs: 1 year online, 7 years archived

*Centralized Log Aggregation:*

* Real-time streaming to centralized logging platform
* Indexed and searchable within 30 seconds
* Dashboard visualization of log data
* Alert triggers based on log patterns
* Correlation across distributed system components
* SIAExtract logs integrated into centralized platform

**4.4 Error Severity Classification**

**CRITICAL - Immediate Emergency Response**

**Criteria:**

* Complete SIA ETL failure preventing any data processing
* Data corruption or integrity compromise detected
* Security breach or unauthorized access attempt
* Source system extraction complete failure (SIAExtract unable to execute)
* SIAExtract process unable to execute
* Critical database failure preventing all operations

**Automated Response:**

* **0-5 min:** PagerDuty alert to on-call SIA engineer (24/7/365)
* **0-5 min:** Automated phone call escalation if not acknowledged
* **0-5 min:** Email to SIA ETL Leadership and SIA Business Owners
* **0-5 min:** Slack critical alert posted to #sia-critical channel
* **0-5 min:** ServiceNow P1 incident auto-created with full context
* **0-10 min:** Status page updated for downstream consumers
* **0-15 min:** Executive notification if unresolved
* **Continuous:** Automated diagnostics captured and attached to incident
* **Special:** SIAExtract logs retrieved and attached to incident

**Response SLAs:**

* Acknowledgment: 5 minutes maximum
* Initial assessment: 15 minutes
* Resolution target: 2 hours maximum
* Post-incident review: Within 24 hours

**ERROR - Urgent Attention Required**

**Criteria:**

* Partial SIA process failure affecting >10% of records
* Repeated database connection failures after retries
* Data validation failures exceeding threshold (>5% error rate)
* SLA breach occurring or imminent
* SIAExtract completed with errors (check log)
* Transformation failures affecting critical data elements
* File corruption detected but partial processing possible

**Automated Response:**

* Email alert to SIA ETL Support team
* Slack notification posted to #sia-errors channel
* ServiceNow P2 incident auto-created
* Dashboard alert prominently displayed
* Automated diagnostics gathered
* Incident assigned to on-call engineer
* SIAExtract logs reviewed automatically

**Response SLAs:**

* Acknowledgment: 30 minutes
* Initial assessment: 1 hour
* Resolution target: 4 hours
* Follow-up review: Within 48 hours

**WARNING - Monitor and Investigate**

**Criteria:**

* Minor data quality issues (<5% of records)
* Performance degradation within acceptable SLA limits
* Recoverable errors after automatic retry
* Unusual data patterns detected but not blocking
* SIAExtract warnings or informational messages
* Non-critical validation failures
* Transient infrastructure issues that self-resolved

**Automated Response:**

* Warning logged to dedicated warning log file
* Included in hourly warning digest email
* Dashboard notification (yellow flag)
* Detailed context captured for investigation
* No immediate action required
* Aggregated in daily summary report

**Response SLAs:**

* Review during business hours
* Investigation within 24 hours if pattern detected
* No immediate resolution required
* Monthly trend analysis

**INFO - Informational and Audit Trail**

**Criteria:**

* Successful SIA ETL process completion
* Processing statistics and performance metrics
* Configuration changes or parameter updates
* Checkpoint and milestone events
* SIAExtract successful completion messages
* Normal operational logging
* Audit trail entries

**Automated Response:**

* Logged to main process log file
* Included in daily operational summary
* Available for audit and compliance reporting
* No alerts generated
* Dashboard shows green status

**4.5 Notification Channels**

**Multi-Channel Alert Distribution:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Severity | PagerDuty | Email | Slack | ServiceNow | Phone | Status Page | Dashboard |
| CRITICAL | Yes (immediate) | Yes (urgent) | #sia-critical | P1 Auto | Yes (escalation) | Red Alert | Red Alert |
| ERROR | No | Yes | #sia-errors | P2 Auto | No | Orange Notice | Orange Alert |
| WARNING | No | Digest | Optional | Manual | No | No | Yellow Flag |
| INFO | No | Summary | No | No | No | No | Green Status |

**Email Distribution Lists:**

*Primary Distribution:*

* **SIA\_ETL\_CRITICAL:** Senior engineers, tech leads, SIA business owners, on-call engineers
* **SIA\_ETL\_SUPPORT:** Full SIA ETL support team, data engineers
* **SIA\_BUSINESS\_USERS:** Business analysts, data consumers, reporting teams
* **SIA\_OPERATIONS:** Operations team, infrastructure support

**Slack Integration:**

*Dedicated Channels:*

* #sia-critical: Critical alerts only, monitored 24/7
* #sia-errors: Error-level alerts, monitored during business hours
* #sia-operations: General SIA ETL operations and status
* #sia-extract-issues: SIAExtract-specific issues and monitoring

*Alert Format Example:*

🔴 CRITICAL: SIA ETL Database Failure

Process: SIA\_ETL

Process ID: SIA-20251007-001

Error: Database connection failed after 3 retries

Impact: 55,000 records not loaded, SLA breach imminent

SIAExtract: Completed successfully (see log: SIA\_EXTRACT\_20251007\_001500.log)

File: SIA\_20251007\_001500.dat

Time: 2025-10-07 00:47:15 EST

Incident: INC0012347

Action: On-call engineer paged

[View Dashboard] [View Logs] [View SIAExtract Log]

**ServiceNow Integration:**

*Automated Incident Creation:*

* Incident auto-populated with full error context
* Log excerpts and stack traces attached
* SIAExtract log automatically retrieved and attached
* Correlation ID for easy log searching
* Assignment to appropriate support group
* SLA clock starts automatically
* Related configuration items linked

*Incident Tracking:*

* All SIA errors tracked through resolution
* Status updates reflected in monitoring dashboard
* Resolution documented with root cause
* Knowledge base updated for recurring issues
* SIAExtract-specific issues tagged appropriately

**4.6 Reporting and Analytics**

**4.6.1 Real-Time Monitoring Dashboard**

**Key Metrics Displayed:**

* Current SIA ETL process status (Running/Failed/Completed/Queued)
* Real-time error count by severity and category
* SLA compliance meter with countdown
* Record processing rate and throughput
* Data quality scorecard (% records passing validation)
* SIAExtract integration status and health
* Historical performance trend charts (last 7 days)
* Resource utilization (CPU, memory, I/O)
* SIAExtract execution time and record counts

**Dashboard Sections:**

1. **Process Status:** Current state of SIA ETL
2. **Error Summary:** Count by severity with drill-down
3. **Performance Metrics:** Processing time, throughput, SLA compliance
4. **Data Quality:** Validation pass/fail rates, error categories
5. **SIAExtract Health:** Extraction status, log availability, record counts
6. **Historical Trends:** 7-day and 30-day trend charts

**Visualization:**

* Green/Yellow/Red status indicators
* Time-series charts for trends
* Error distribution pie charts
* Processing volume bar charts
* SLA compliance gauge
* SIAExtract performance metrics

**4.6.2 Daily SIA Error Summary Report**

**Automated Report (8:00 AM Daily):**

**Content:**

* Executive summary of SIA ETL health (past 24 hours)
* All errors encountered with severity classification
* Error categorization and frequency analysis
* Top 10 error types by occurrence
* Data quality metrics and trends
* Record processing statistics (total, successful, failed)
* SLA performance (met/missed)
* SIAExtract performance and any issues
* Comparison to previous day and weekly average
* Outstanding incidents and resolution status

**Distribution:** SIA ETL team, SIA business owners, data quality team

**4.6.3 Weekly SIA Analysis Report**

**Automated Report (Monday 9:00 AM):**

**Content:**

* Aggregated error patterns and trends over week
* Data quality trend analysis
* Performance metrics (avg processing time, throughput)
* SLA compliance statistics
* Recurring issue identification with frequency
* Capacity and resource utilization trends
* SIAExtract logging analysis
* Top 5 issues requiring attention
* Recommendations for process improvements
* Comparison to previous weeks (4-week trend)

**Distribution:** SIA ETL team, management, SIA business stakeholders

**4.6.4 Monthly SIA Executive Summary**

**Automated Report (1st of Month):**

**Content:**

* High-level SIA ETL health scorecard
* Major incidents, root causes, and resolutions
* Month-over-month SLA compliance trends
* Data quality improvements or degradations
* Capacity planning insights
* Process optimization recommendations
* Cost analysis (resource consumption)
* Roadmap items and modernization progress
* SIAExtract integration status

**Distribution:** Executive team, SIA business owners, IT leadership

**4.6.5 Ad-Hoc and Custom Reports**

**Available On-Demand:**

* Error reports by date range
* Specific error code deep-dive analysis
* Data quality exception reports
* Performance analysis reports
* SIAExtract audit reports
* Compliance and audit trail reports

**4.7 Error Resolution Workflow**

**Standard Resolution Process:**

**1. Error Detection and Alert Generation**

* Automated monitoring detects error condition
* Error logged with full context and diagnostics
* SIAExtract logs reviewed automatically if extraction-related
* Severity assessed and classified
* Appropriate alerts generated based on severity
* Incident ticket created in ServiceNow

**2. Incident Triage and Assignment**

* On-call engineer receives alert
* Initial assessment of impact and urgency
* Review SIAExtract logs for root cause clues
* Incident assigned to appropriate resolver group
* Priority and SLA established
* Stakeholders notified of incident

**3. Investigation and Diagnosis**

* Review error logs and stack traces
* Analyze SIAExtract logs for source issues
* Review recent code or configuration changes
* Check system resource availability
* Identify root cause (extraction, transformation, or load phase)

**4. Resolution and Remediation**

* Apply fix (data correction, code fix, configuration change, infrastructure repair)
* If SIAExtract issue, coordinate with source system team
* Test fix in lower environment if time permits
* Apply fix to production
* Validate resolution
* Monitor for recurrence

**5. Verification and Reprocessing**

* Verify SIA ETL completes successfully
* Validate data quality and completeness
* Confirm record counts and reconciliation
* Cross-check with SIAExtract log record counts
* Check SLA compliance
* Verify downstream consumers not impacted

**6. Documentation and Closure**

* Document root cause in incident ticket
* Record resolution steps taken
* Include SIAExtract log analysis if relevant
* Update runbook if new procedure
* Close incident with complete documentation
* Update knowledge base for future reference

**7. Post-Incident Activities**

* Conduct post-mortem for Critical incidents
* Identify preventive measures
* Create remediation tasks or backlog items
* Update monitoring or alerting if gaps identified
* Improve SIAExtract logging if needed
* Share lessons learned with team

**Resolution SLA Targets:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Severity | Acknowledgment | Initial Assessment | Resolution Target | Post-Incident Review |
| CRITICAL | 5 minutes | 15 minutes | 2 hours | Within 24 hours |
| ERROR | 30 minutes | 1 hour | 4 hours | Within 48 hours |
| WARNING | Business hours | 4 hours | 24 hours | Monthly rollup |
| INFO | N/A | N/A | N/A | N/A |

**4.8 Root Cause Analysis (RCA)**

**RCA Mandatory For:**

* All Critical severity incidents
* Recurring Error incidents (3+ occurrences in 30 days)
* SLA breaches
* Data integrity or corruption incidents
* Any incident impacting business operations
* SIAExtract failures or integration issues

**RCA Documentation Template:**

1. **Incident Summary:** What happened, when, impact
2. **Timeline:** Detailed chronology of events
3. **Root Cause:** Underlying cause, not just symptoms
4. **Contributing Factors:** Environmental or process factors
5. **Resolution:** Immediate fix applied
6. **SIAExtract Analysis:** If applicable, analysis of extraction logs and process
7. **Preventive Measures:** Long-term solutions to prevent recurrence
8. **Action Items:** Specific tasks with owners and due dates
9. **Lessons Learned:** Key takeaways for team and organization

**5. GitHub Repository Structure**

**5.1 SIA Repository Organization**

etl-modernization/

├── sia-etl/

│ ├── sia-core/

│ │ ├── src/

│ │ │ ├── main/

│ │ │ │ ├── java/com/company/etl/sia/

│ │ │ │ │ ├── SiaEtlApplication.java

│ │ │ │ │ ├── extraction/

│ │ │ │ │ │ ├── SIAExtract.java

│ │ │ │ │ │ ├── SiaExtractor.java

│ │ │ │ │ │ └── SiaSourceConnection.java

│ │ │ │ │ ├── logging/

│ │ │ │ │ │ ├── SIAExtractLogger.java

│ │ │ │ │ │ ├── LogEntry.java

│ │ │ │ │ │ └── LogFormatter.java

│ │ │ │ │ ├── transformation/

│ │ │ │ │ │ ├── SiaTransformer.java

│ │ │ │ │ │ ├── SiaEnrichmentService.java

│ │ │ │ │ │ └── SiaAggregator.java

│ │ │ │ │ ├── validation/

│ │ │ │ │ │ ├── SiaValidator.java

│ │ │ │ │ │ ├── PreLoadValidation.java

│ │ │ │ │ │ ├── DataQualityChecker.java

│ │ │ │ │ │ └── ExtractLogValidator.java

│ │ │ │ │ ├── loading/

│ │ │ │ │ │ ├── SiaLoader.java

│ │ │ │ │ │ ├── DatabaseWriter.java

│ │ │ │ │ │ └── BatchProcessor.java

│ │ │ │ │ ├── error/

│ │ │ │ │ │ ├── SiaErrorHandler.java

│ │ │ │ │ │ └── SiaErrorLogger.java

│ │ │ │ │ └── util/

│ │ │ │ │ ├── SiaDateUtils.java

│ │ │ │ │ ├── SiaFileUtils.java

│ │ │ │ │ └── ChecksumCalculator.java

│ │ │ │ └── resources/

│ │ │ │ ├── config/

│ │ │ │ │ ├── application.properties

│ │ │ │ │ └── application-prod.properties

│ │ │ │ ├── sql/

│ │ │ │ │ ├── sia-extract.sql

│ │ │ │ │ └── sia-load.sql

│ │ │ │ └── log4j2.xml

│ │ │ └── test/

│ │ │ ├── java/com/company/etl/sia/

│ │ │ │ ├── extraction/

│ │ │ │ │ ├── SIAExtractTest.java

│ │ │ │ │ └── SiaExtractorTest.java

│ │ │ │ ├── logging/

│ │ │ │ │ └── SIAExtractLoggerTest.java

│ │ │ │ ├── transformation/

│ │ │ │ │ └── SiaTransformerTest.java

│ │ │ │ ├── validation/

│ │ │ │ │ └── SiaValidatorTest.java

│ │ │ │ └── integration/

│ │ │ │ ├── SiaIntegrationTest.java

│ │ │ │ └── SiaExtractParityTest.java

│ │ │ └── resources/

│ │ │ └── test-data/

│ │ │ ├── sia\_test\_001.dat

│ │ │ ├── sia\_extract\_log\_sample.log

│ │ │ └── sia\_edge\_cases.dat

│ │ ├── pom.xml

│ │ └── README.md

│ │

│ └── sia-common/

│ ├── src/

│ │ ├── main/

│ │ │ ├── java/com/company/etl/sia/common/

│ │ │ │ ├── model/

│ │ │ │ │ ├── SiaRecord.java

│ │ │ │ │ ├── SiaExtractResult.java

│ │ │ │ │ └── SiaSummary.java

│ │ │ │ ├── config/

│ │ │ │ │ └── SiaConfiguration.java

│ │ │ │ └── constants/

│ │ │ │ └── SiaConstants.java

│ │ │ └── resources/

│ │ └── test/

│ └── pom.xml

│

├── database/

│ ├── migrations/

│ │ ├── V1\_\_sia\_schema.sql

│ │ ├── V2\_\_sia\_tables.sql

│ │ └── V3\_\_sia\_audit\_tables.sql

│ └── stored-procedures/

│ └── sia/

│ ├── sp\_sia\_load.sql

│ └── sp\_sia\_validate.sql

│

├── docker/

│ ├── Dockerfile.sia

│ └── docker-compose-sia.yml

│

└── docs/

└── sia/

├── architecture.md

├── sia-extract-specification.md

├── data-dictionary.md

└── runbooks/

├── sia-etl-runbook.md

├── sia-extract-troubleshooting.md

└── sia-logging-guide.md

**5.2 Branching Strategy**

**Git Flow Model for SIA:**

**Permanent Branches:**

1. **main** - Production-ready code
   * Protected branch: requires PR and 2 approvals
   * Auto-deploys to production
   * Tagged with version numbers
2. **develop** - Integration branch
   * Protected branch: requires PR and 1 approval
   * Auto-deploys to development environment
   * Base for all feature branches
3. **test** - QA testing branch
   * Auto-deploys to test/UAT environment

**Temporary Branches:**

1. **feature/SIA-**\* - Feature development
   * Created from: develop
   * Naming: feature/SIA-450-enhance-extract-logging
   * Merged back to: develop
2. **bugfix/SIA-**\* - Bug fixes
   * Created from: develop or test
   * Naming: bugfix/SIA-502-fix-null-pointer-error
   * Merged back to: develop
3. **hotfix/SIA-**\* - Production hotfixes
   * Created from: main
   * Naming: hotfix/SIA-999-critical-extraction-failure
   * Merged to: main AND develop

**5.3 Commit Message Standards**

**Format:**

[SIA-###] Short summary (50 chars max)

Detailed explanation:

- What changed

- Why changed

- Impact on SIAExtract integration

Fixes: SIA-###

**Examples:**

* [SIA-450] Enhance SIAExtract logging with correlation IDs
* [SIA-502] Fix null pointer in SIA transformation logic
* [SIA-625] Improve SIAExtract log parsing for error detection
* [SIA-701] Add comprehensive unit tests for SIAExtract integration

**5.4 Code Review Requirements**

**All Pull Requests Must Have:**

* Clear description of SIA-specific changes
* Link to JIRA ticket
* Evidence of testing (unit test results)
* Verification of SIAExtract integration if applicable
* No merge conflicts
* Passing CI/CD pipeline (all tests pass, coverage >85%)

**SIA Code Owners:**

# CODEOWNERS for SIA ETL

/sia-etl/sia-core/ @sia-etl-team @senior-engineer-sia

/sia-etl/sia-core/src/main/java/com/company/etl/sia/extraction/ @sia-extract-specialists

/sia-etl/sia-core/src/main/java/com/company/etl/sia/logging/ @sia-extract-specialists

/sia-etl/sia-common/ @sia-architects @tech-leads

**5.5 Release Management**

**Semantic Versioning:** MAJOR.MINOR.PATCH

**Example:** sia-v1.9.2

**Release Process:**

1. Create release branch: release/sia-v2.0.0
2. Final testing in test environment
3. Update CHANGELOG.md
4. Verify SIAExtract integration compatibility
5. Merge to main
6. Tag release: git tag sia-v2.0.0
7. Auto-deploy to production
8. Merge back to develop

**Release Cadence:**

* Major releases: Quarterly
* Minor releases: Monthly
* Patch releases: As needed (weekly if required)
* Hotfixes: Immediate (within hours for critical issues)

**6. ECP (Enterprise Container Platform) Execution Environment**

**6.1 Overview**

**ECP Technology Stack for SIA:**

* **Container Runtime:** Docker 24.x
* **Orchestration:** Kubernetes 1.28.x
* **Container Registry:** harbor.company.com/etl/sia
* **Service Mesh:** Istio 1.19.x

**ECP Environments:**

| **Environment** | **Purpose** | **Update Frequency** | **Data Type** |
| --- | --- | --- | --- |
| **dev** | Development | Every commit | Synthetic |
| **test** | QA Testing | Daily builds | Sanitized production |
| **uat** | User acceptance | Weekly releases | Production-like |
| **staging** | Pre-production | Release candidates | Production mirror |
| **prod** | Production | Scheduled releases | Live production |

**6.2 SIA Containerization**

**SIA ETL Dockerfile:**

FROM openjdk:17-jre-slim

LABEL app.name="sia-etl"

LABEL app.version="1.9.2"

LABEL component="sia"

LABEL sia-extract.integrated="true"

# Create non-root user

RUN groupadd -r sia && useradd -r -g sia sia

# Set up directories

RUN mkdir -p /app /app/config /app/logs /app/logs/sia-extract /app/data

RUN chown -R sia:sia /app

WORKDIR /app

# Copy application

COPY --chown=sia:sia target/sia-etl-1.9.2.jar /app/app.jar

COPY --chown=sia:sia src/main/resources/config /app/config

COPY --chown=sia:sia scripts/start-sia.sh /app/start.sh

RUN chmod +x /app/start.sh

# Environment configuration

ENV JAVA\_OPTS="-Xms2g -Xmx5g -XX:+UseG1GC"

ENV LOG\_LEVEL="INFO"

ENV SIA\_EXTRACT\_TIMEOUT="1800"

ENV SIA\_EXTRACT\_LOGGING\_ENABLED="true"

ENV DB\_CONNECTION\_POOL\_SIZE="10"

# Switch to non-root user

USER sia

# Health check

HEALTHCHECK --interval=30s --timeout=10s --start-period=40s --retries=3 \

CMD curl -f http://localhost:8080/health || exit 1

EXPOSE 8080

ENTRYPOINT ["/app/start.sh"]

CMD ["run"]

**Image Build and Push:**

# Build SIA ETL image

docker build -f docker/Dockerfile.sia -t harbor.company.com/etl/sia:v1.9.2 .

# Tag as latest

docker tag harbor.company.com/etl/sia:v1.9.2 harbor.company.com/etl/sia:latest

# Push to registry

docker push harbor.company.com/etl/sia:v1.9.2

docker push harbor.company.com/etl/sia:latest

**Image Tagging Strategy:**

* **Semantic version:** v1.9.2 (for releases)
* **Git commit SHA:** sha-a1b2c3d (for traceability)
* **Environment tag:** dev-latest, test-stable, prod-v1.9.2
* **Latest tag:** latest (points to most recent build in develop)

**6.3 Kubernetes Deployment**

**SIA ETL CronJob:**

apiVersion: batch/v1

kind: CronJob

metadata:

name: sia-etl-job

namespace: etl-prod

labels:

app: sia-etl

component: sia

sia-extract: enabled

spec:

schedule: "0 0 \* \* \*" # Midnight daily

timeZone: "America/New\_York"

concurrencyPolicy: Forbid

successfulJobsHistoryLimit: 5

failedJobsHistoryLimit: 3

jobTemplate:

metadata:

labels:

app: sia-etl

version: v1.9.2

spec:

backoffLimit: 2

activeDeadlineSeconds: 10800 # 3-hour timeout

template:

metadata:

labels:

app: sia-etl

version: v1.9.2

annotations:

prometheus.io/scrape: "true"

prometheus.io/port: "8080"

prometheus.io/path: "/metrics"

spec:

restartPolicy: OnFailure

serviceAccountName: sia-etl-sa

securityContext:

runAsNonRoot: true

runAsUser: 1000

fsGroup: 1000

containers:

- name: sia-etl

image: harbor.company.com/etl/sia:v1.9.2

imagePullPolicy: IfNotPresent

resources:

requests:

memory: "4Gi"

cpu: "2000m"

limits:

memory: "10Gi"

cpu: "5000m"

env:

- name: ENVIRONMENT

value: "production"

- name: SIA\_PROCESS\_DATE

value: "T-1"

- name: LOG\_LEVEL

valueFrom:

configMapKeyRef:

name: sia-config

key: log.level

- name: DB\_HOST

valueFrom:

configMapKeyRef:

name: sia-config

key: db.host

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: sia-db-credentials

key: password

- name: SIA\_SOURCE\_TOKEN

valueFrom:

secretKeyRef:

name: sia-source-credentials

key: token

- name: SIA\_EXTRACT\_ENABLED

value: "true"

- name: SIA\_EXTRACT\_LOGGING\_PATH

value: "/app/logs/sia-extract"

volumeMounts:

- name: sia-data

mountPath: /app/data

- name: sia-logs

mountPath: /app/logs

- name: sia-config

mountPath: /app/config

readOnly: true

volumes:

- name: sia-data

persistentVolumeClaim:

claimName: sia-data-pvc

- name: sia-logs

persistentVolumeClaim:

claimName: sia-logs-pvc

- name: sia-config

configMap:

name: sia-config

nodeSelector:

workload-type: etl

environment: production

tolerations:

- key: "etl-workload"

operator: "Equal"

value: "true"

effect: "NoSchedule"

**6.4 Configuration Management**

**SIA ConfigMap:**

apiVersion: v1

kind: ConfigMap

metadata:

name: sia-config

namespace: etl-prod

data:

# Logging

log.level: "INFO"

log.rotation: "daily"

log.retention.days: "30"

# Database

db.host: "prod-db.company.com"

db.port: "5432"

db.name: "sia\_warehouse"

db.connection.pool.size: "10"

db.connection.timeout: "30000"

# SIA Configuration

sia.source.system: "SIA-PROD-DB"

sia.source.path: "/data/sia"

sia.batch.size: "7500"

sia.max.retries: "3"

sia.timeout.seconds: "3600"

sia.sla.hours: "3"

# SIAExtract Configuration

sia.extract.enabled: "true"

sia.extract.timeout: "1800"

sia.extract.logging.enabled: "true"

sia.extract.logging.path: "/app/logs/sia-extract"

sia.extract.logging.format: "legacy"

sia.extract.retry.attempts: "3"

# Validation

validation.enabled: "true"

validation.fail.threshold: "0.05"

validation.mandatory.fields: "RECORD\_ID,ACCOUNT\_ID,CUSTOMER\_ID,TRANSACTION\_DATE,AMOUNT,STATUS"

validation.extract.log.required: "true"

# Monitoring

monitoring.enabled: "true"

monitoring.prometheus.port: "8080"

monitoring.metrics.enabled: "true"

# Error Handling

error.notification.enabled: "true"

error.slack.webhook.url: "https://hooks.slack.com/services/XXX"

error.email.recipients: "sia-etl-support@company.com"

**SIA Secrets:**

apiVersion: v1

kind: Secret

metadata:

name: sia-db-credentials

namespace: etl-prod

type: Opaque

stringData:

username: "sia\_etl\_service"

password: "ENCRYPTED\_PASSWORD"

jdbc.url: "jdbc:postgresql://prod-db.company.com:5432/sia\_warehouse?ssl=true"

---

apiVersion: v1

kind: Secret

metadata:

name: sia-source-credentials

namespace: etl-prod

type: Opaque

stringData:

username: "sia\_source\_user"

password: "ENCRYPTED\_SOURCE\_PASSWORD"

token: "ENCRYPTED\_API\_TOKEN"

**6.5 Persistent Storage**

**SIA Data PVC:**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: sia-data-pvc

namespace: etl-prod

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 250Gi

storageClassName: ecp-fast-ssd

**SIA Logs PVC:**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: sia-logs-pvc

namespace: etl-prod

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 100Gi

storageClassName: ecp-standard-hdd

**6.6 Monitoring and Observability**

**SIA Prometheus Metrics:**

Exposed at http://sia-etl:8080/metrics

**Key Metrics:**

* sia\_job\_duration\_seconds - Job execution time
* sia\_extract\_duration\_seconds - SIAExtract execution time
* sia\_records\_processed\_total - Total records processed
* sia\_records\_failed\_total - Failed record count
* sia\_validation\_errors\_total - Validation error count
* sia\_extract\_status - SIAExtract execution status (1=success, 0=failure)
* sia\_extract\_record\_count - Records extracted by SIAExtract
* sia\_extract\_log\_available - SIAExtract log file availability (1=available, 0=missing)
* sia\_sla\_compliance - SLA compliance (1=met, 0=missed)
* sia\_database\_connection\_pool\_active - Active DB connections
* sia\_memory\_usage\_bytes - Memory consumption
* jvm\_\* - Standard JVM metrics

**Grafana Dashboards:**

* SIA ETL Dashboard
* SIAExtract Performance Dashboard
* SIA Health Overview Dashboard

**Alerting Rules:**

groups:

- name: sia\_etl\_alerts

rules:

- alert: SiaETLFailure

expr: sia\_job\_status == 0

for: 5m

labels:

severity: critical

component: sia

annotations:

summary: "SIA ETL job failed"

description: "SIA ETL has failed. Check SIAExtract logs and processing logs."

- alert: SiaExtractFailure

expr: sia\_extract\_status == 0

for: 3m

labels:

severity: critical

component: sia-extract

annotations:

summary: "SIAExtract failed"

description: "SIAExtract process failed. Review SIAExtract logs immediately."

- alert: SiaExtractLogMissing

expr: sia\_extract\_log\_available == 0

labels:

severity: warning

component: sia-extract

annotations:

summary: "SIAExtract log file missing"

description: "SIAExtract log file not found. Audit trail incomplete."

- alert: SiaSLABreach

expr: sia\_job\_duration\_seconds > 10800

labels:

severity: error

component: sia

annotations:

summary: "SIA ETL SLA breach"

description: "SIA ETL exceeded 3-hour SLA."

- alert: SiaHighErrorRate

expr: rate(sia\_records\_failed\_total[5m]) > 0.05

labels:

severity: warning

component: sia

annotations:

summary: "SIA ETL high error rate detected"

description: "SIA error rate exceeds 5%."

**6.7 Deployment Procedures**

**Production Deployment Process:**

1. **Pre-Deployment:**
   * Change request approved
   * Release notes prepared (including SIAExtract changes)
   * Backup current production state
   * Verify SIAExtract integration compatibility
   * Schedule maintenance window (if needed)
2. **Deployment:**
3. # Update image version in CronJob manifest
4. kubectl set image cronjob/sia-etl-job \
5. sia-etl=harbor.company.com/etl/sia:v2.0.0 \
6. -n etl-prod
7. # Verify update
8. kubectl get cronjob sia-etl-job -n etl-prod -o yaml | grep image
9. **Validation:**
   * Verify CronJob updated: kubectl get cronjob -n etl-prod
   * Check next scheduled run
   * Monitor first execution
   * Validate data quality and counts
   * Verify SIAExtract log generation
10. **Post-Deployment:**
    * 24-hour monitoring period
    * Daily health checks
    * Review SIAExtract logs for issues
    * Rollback plan ready

**Manual Job Execution:**

# Trigger SIA ETL manually

kubectl create job sia-manual-20251007 \

--from=cronjob/sia-etl-job \

-n etl-prod

# Monitor job execution

kubectl logs -f job/sia-manual-20251007 -n etl-prod

# Check job status

kubectl get job sia-manual-20251007 -n etl-prod

# View SIAExtract logs (if accessible from pod)

kubectl exec -it $(kubectl get pod -l job-name=sia-manual-20251007 -n etl-prod -o name | head -1) -n etl-prod -- cat /app/logs/sia-extract/SIA\_EXTRACT\_20251007.log

**6.8 Operational Procedures**

**Troubleshooting:**

# View SIA ETL logs

kubectl logs -f $(kubectl get pod -l app=sia-etl -n etl-prod -o name | head -1) -n etl-prod

# View recent events

kubectl get events -n etl-prod --sort-by='.lastTimestamp' | grep sia

# Describe CronJob

kubectl describe cronjob sia-etl-job -n etl-prod

# View job history

kubectl get jobs -n etl-prod | grep sia

# Access pod for debugging

kubectl exec -it $(kubectl get pod -l app=sia-etl -n etl-prod -o name | head -1) -n etl-prod -- /bin/bash

# Check SIAExtract logs inside pod

kubectl exec -it $(kubectl get pod -l app=sia-etl -n etl-prod -o name | head -1) -n etl-prod -- ls -la /app/logs/sia-extract/

# Retrieve SIAExtract log file

kubectl cp etl-prod/$(kubectl get pod -l app=sia-etl -n etl-prod -o name | head -1 | cut -d'/' -f2):/app/logs/sia-extract/SIA\_EXTRACT\_20251007.log ./sia\_extract\_log\_local.log

**Backup and Disaster Recovery:**

**Backup Strategy:**

* Persistent volume snapshots: Daily
* Configuration backups: With each change
* Container images: Retained for 90 days
* Database backups: Managed separately (hourly)
* SIAExtract logs: Backed up with standard logs (90 days online, 1 year archived)

**Recovery Procedure:**

1. Identify failure scope (extraction, transformation, or load)
2. Review SIAExtract logs to determine extraction status
3. Restore PVC from latest snapshot
4. Redeploy pods from last known good image
5. Validate configuration (especially SIAExtract settings)
6. Re-run failed jobs
7. Verify data integrity
8. Cross-check record counts with SIAExtract logs

**Recovery Objectives:**

* **RTO (Recovery Time Objective):** 2 hours
* **RPO (Recovery Point Objective):** 24 hours

**Document Summary**

This comprehensive document provides detailed guidance for the SIA ETL modernization effort, covering:

* **Unit Testing:** Complete testing framework with emphasis on SIAExtract logging functions, legacy parity testing, and comprehensive error handling scenarios
* **Pre-Load Validations:** Detailed validation checklist with special attention to SIAExtract log validation and integration
* **Break Testing:** Systematic approach including SIAExtract-specific scenarios and recovery testing
* **Error Reporting:** Multi-tier severity classification with SIAExtract log integration, automated alerting, and comprehensive analytics
* **GitHub Repository:** Detailed structure with dedicated SIAExtract logging modules and clear organization
* **ECP Deployment:** Complete containerization strategy with SIAExtract integration, Kubernetes orchestration, and operational procedures

This framework ensures the modernized SIA ETL process maintains full parity with legacy SIAExtract functionality while incorporating cloud-native best practices, comprehensive logging, and enterprise-grade operational support.