**DIAL Step 1 - Comprehensive Testing Document**

**Executive Summary**

This document provides comprehensive testing specifications for the Java implementation of DIAL Step 1 operations. Step 1 consists of four main sub-processes that handle file processing, database statistics, exports, and synchronization operations.

**1. Input Requirements**

**1.1 Configuration Prerequisites**

**Environment Configuration**

* **DIAL.path file** must be properly configured with all environment variables
* **Oracle database connections** must be available for:
  + dial schema connection
  + dialrpt schema connection
  + als schema connection
  + alsrpt schema connection

**Required Directory Structure**

/als-ALS/app/

├── execloc/d.common/ # Contains DeclpRetr password files

├── execloc/d.dial/ # DIAL execution location

├── loadstage/

│ ├── AREADIR/ # Area processing directory

│ └── CONSOLDIR/ # Console output directory

├── rawfiles/

│ ├── DIAL.raw # Input raw file

│ ├── NEWDIAL.raw # New DIAL raw file

│ ├── OLDDIAL.raw # Backup raw file

│ ├── EXP\_DIR/ # Export files directory

│ └── XFILES\_DIR/ # Additional files

└── [Area Directories]/

└── DIALDIR/

├── TDA.raw # Transaction Detail Area raw file

└── TDI.raw # Transaction Detail Index raw file

**1.2 Input Files by Sub-Process**

**Dial1\_crRAW (DialCrRawService)**

**Primary Input Files:**

* **TDA.raw files** (per area)
  + Location: $ALSDIR/$place/DIALDIR/TDA.raw
  + Format: Fixed-width records, 660 characters per line
  + Content: Transaction detail data with TIN at positions 11-21
  + Expected: Recent files (modified within last 6 days)
* **TDI.raw files** (per area)
  + Location: $ALSDIR/$place/DIALDIR/TDI.raw
  + Format: Fixed-width records, 660 characters per line
  + Content: Transaction detail index data with TIN at positions 11-21
  + Expected: Recent files (modified within last 6 days)

**Configuration Input:**

* **Database paths** from config.getDb().getPath(): A11, A12, A13, A14, A15, A21, A22, A23, A24, A25, A26, A27, A35

**Dial1\_dothrcp (DialTableStatsService)**

**No specific input files required**

* Operates on existing database tables
* Uses stored procedures: @dialcopy and @dialcopy2

**Dial1\_exports (DialExportsService)**

**Parameter Files:**

* **EXP\_PAR** - Oracle export parameter file for first export
* **EXP2\_PAR** - Oracle export parameter file for second export
* Content: Export parameters defining tables, schemas, and options

**Dial1\_point2cp (DialSyncService)**

**No specific input files required**

* Executes stored procedure: @syn2cp

**1.3 Database Table Dependencies**

**Tables Read From:**

* **dial.dialaud** - Load audit information
  + Columns: loadarea, comments, loaddt
  + Used by: Load checking operations

**Tables Operated On:**

* **RAWDATA** (temporary table created/dropped)
  + Columns: tin VARCHAR2(11), dataline VARCHAR2(660)
  + Used by: COMBO.raw creation process
* **Statistics Tables:**
  + **TINSUMMARY2**, **DIALVCD2**, **DIALAUD2**, **DIALENT2**, **CONSOLEAD2**
  + Operation: Statistics gathering via DBMS\_STATS.GATHER\_TABLE\_STATS

**2. Output Specifications**

**2.1 Database Interactions**

**RAWDATA Table Operations (DialCrRawService)**

**Table Creation:**

CREATE TABLE rawdata (

tin VARCHAR2(11),

dataline VARCHAR2(660)

)

**Expected Data Volume:**

* Row count: Sum of all TDA.raw + TDI.raw lines across all areas
* Typical volume: 10,000 - 1,000,000+ records depending on data cycle
* Data content: Combined TDA/TDI records sorted by TIN

**Table Lifecycle:**

1. CREATE TABLE rawdata
2. INSERT data from TDA.raw and TDI.raw files
3. SELECT sorted data to create COMBO.raw
4. TRUNCATE TABLE rawdata
5. DROP TABLE rawdata

**Statistics Updates (DialTableStatsService)**

**Tables Updated:**

* **TINSUMMARY2**: Tax information summary statistics
* **DIALVCD2**: DIAL VCD (Verification Code) statistics
* **DIALAUD2**: DIAL audit statistics
* **DIALENT2**: DIAL entity statistics
* **CONSOLEAD2**: Console lead statistics

**Expected Updates:**

* NUM\_ROWS: Accurate row counts
* LAST\_ANALYZED: Current timestamp
* AVG\_ROW\_LEN: Average row length statistics
* Index statistics updated via CASCADE => TRUE

**Synchronization Operations (DialSyncService)**

**Expected Changes:**

* Materialized view refreshes
* Sequence synchronization
* Schema statistics updates
* Data integrity validations

**2.2 File Outputs**

**COMBO.raw Files (DialCrRawService)**

**File Details:**

* **Location:** Current working directory (per area processing)
* **Naming Convention:** COMBO.raw
* **Format:** Fixed-width, 660 characters per line
* **Content:** Sorted combination of TDA.raw and TDI.raw data
* **Sorting:** Ordered by TIN (positions 11-21)
* **Expected Size:** Sum of input TDA.raw + TDI.raw file sizes

**File Validation Requirements:**

* Line count must equal sum of TDA.raw + TDI.raw line counts
* Each line must be exactly 660 characters
* Data must be sorted by TIN in ascending order
* Single list cycle validation (characters 159-164 must be consistent)

**Backup Files (DialCrRawService)**

**TDA/TDI Backup Files:**

* **Location:** $RAW\_DIR/
* **Naming:** TDA.$area.$date and TDI.$area.$date
* **Format:** Copy of original TDA.raw/TDI.raw files
* **Date Format:** MMDD (e.g., 1225 for December 25)

**Old File Backups:**

* **Location:** $RAW\_BKUP/
* **Naming:** x.$original\_filename
* **Permissions:** 666 (rw-rw-rw-)

**Export Files (DialExportsService)**

**Export File Outputs:**

* **Location:** $EXP\_DIR/
* **Files Created:**
  + dial.exp.$date (first export)
  + dial2.exp.$date (second export)
* **Compressed Files:**
  + dial.exp.$date.Z
  + dial2.exp.$date.Z
* **Content:** Oracle export dump files containing DIAL schema objects
* **Date Format:** MMDD

**2.3 Log Files and Monitoring Outputs**

**diallog File**

**Location:** $CONSOLDIR/diallog **Content Expectations:**

2024-12-25 10:30:00 === Step #1 - Dial\_crRAW - Creates COMBO.raw files ===

2024-12-25 10:30:01 Processing area: A11

2024-12-25 10:30:15 --- Creating Rawdata table for A11 ----

2024-12-25 10:30:30 --- Running sqlldr for A11 ----

2024-12-25 10:30:45 Data load successful - Error - in RAWDATA sqlload

2024-12-25 10:31:00 --- Create COMBO.raw from rawdata sorted by tin ----------

2024-12-25 10:31:15 Successful creation of COMBO.raw for A11

2024-12-25 10:31:20 === Step #1 - Creation of COMBO.raw files COMPLETED ===

**Statistics Output Files**

**dialcopy.out:** $CONSOLDIR/dialcopy.out **dialcopy2.out:** $CONSOLDIR/dialcopy2.out **Content:** Results of statistics gathering operations

**Synchronization Output**

**syn2cp.out:** $CONSOLDIR/syn2cp.out **Content:** Results of database synchronization operations

**Byte Check Output**

**byte\_check:** $CONSOLDIR/byte\_check **Content:** File validation and byte checking results

**3. Core Logic / Business Rules**

**3.1 COMBO.raw Creation Logic (DialCrRawService)**

**File Processing Sequence:**

1. **Area Validation:**
   * Check TDA.raw and TDI.raw exist and are current (< 6 days old)
   * Verify files are not empty (> 0 bytes)
2. **Data Extraction:**
   * Extract TIN from positions 11-21 (1-based indexing)
   * Extract full dataline (positions 1-660)
   * Combine TDA and TDI records
3. **Data Loading:**
   * Create temporary RAWDATA table
   * Load combined data with TIN indexing
   * Validate successful load completion
4. **COMBO.raw Generation:**
   * Sort data by TIN in ascending order
   * Apply index hint: /\*+ index\_asc(RAWDATA, SUBTIN\_IX) \*/
   * Filter: WHERE tin > '0'
   * Export to COMBO.raw file
5. **Validation Rules:**
   * Output line count = Input TDA lines + Input TDI lines
   * Single list cycle check (positions 159-164)
   * File integrity verification

**3.2 Statistics Processing Logic (DialTableStatsService)**

**Table Statistics Sequence:**

DBMS\_STATS.GATHER\_TABLE\_STATS(

ownname => 'DIAL',

tabname => [TABLE\_NAME],

cascade => TRUE,

estimate\_percent => DBMS\_STATS.AUTO\_SAMPLE\_SIZE

);

**Tables Processed:**

* TINSUMMARY2, DIALVCD2, DIALAUD2, DIALENT2, CONSOLEAD2

**3.3 Export Processing Logic (DialExportsService)**

**Export Sequence:**

1. **Cleanup:** Remove old export files matching dial.exp.\*.Z and dial2.exp.\*.Z
2. **First Export:** Execute export using EXP\_PAR parameters
3. **Second Export:** Execute export using EXP2\_PAR parameters
4. **Compression:** Compress export files to .Z format
5. **Validation:** Verify "Export terminated successfully" messages

**3.4 Synchronization Logic (DialSyncService)**

**Multi-Connection Sync:**

1. **dialrpt connection:** Execute @syn2cp, commit
2. **als connection:** Execute @syn2cp, commit
3. **alsrpt connection:** Execute @syn2cp, commit

**4. Test Case Specifications**

**4.1 Happy Path Test Cases**

**Test Case 1.1: Complete Step 1 Execution**

**Objective:** Verify successful end-to-end Step 1 processing

**Prerequisites:**

* Valid TDA.raw and TDI.raw files in all area directories
* Database connectivity to all required schemas
* Proper directory permissions and disk space

**Test Steps:**

1. Execute dialStep1Orchestrator.executeStep1Operations()
2. Verify all 4 sub-services complete successfully
3. Validate all output files are created
4. Check database statistics are updated

**Expected Results:**

* All services return success status
* COMBO.raw files created for each area
* Export files created and compressed
* Statistics updated for all specified tables
* Synchronization completed for all connections
* No error messages in diallog

**Test Case 1.2: Individual Service Execution**

**Objective:** Verify each service can execute independently

**Test Steps:**

1. Execute each service individually:
   * dialStep1Orchestrator.executeStep1Operation("crraw")
   * dialStep1Orchestrator.executeStep1Operation("stats")
   * dialStep1Orchestrator.executeStep1Operation("exports")
   * dialStep1Orchestrator.executeStep1Operation("sync")

**Expected Results:**

* Each service completes successfully when run individually
* Appropriate output files/database changes occur
* Proper logging occurs for each operation

**4.2 Data Validation Test Cases**

**Test Case 2.1: COMBO.raw Content Validation**

**Objective:** Verify COMBO.raw contains correctly sorted and combined data

**Test Data Setup:**

* Create TDA.raw with known TIN values: 111111111, 333333333, 555555555
* Create TDI.raw with known TIN values: 222222222, 444444444, 666666666

**Test Steps:**

1. Execute DialCrRawService.processComboRawFiles()
2. Read generated COMBO.raw file
3. Validate sorting and content

**Expected Results:**

* COMBO.raw contains 6 lines
* TINs are sorted: 111111111, 222222222, 333333333, 444444444, 555555555, 666666666
* Each line is exactly 660 characters
* Original dataline content is preserved

**Test Case 2.2: Statistics Update Validation**

**Objective:** Verify database statistics are properly updated

**Test Steps:**

1. Record current statistics for target tables
2. Execute DialTableStatsService.gatherTableStatistics()
3. Query updated statistics

**Expected Results:**

* LAST\_ANALYZED timestamp updated to current time
* NUM\_ROWS reflects current table counts
* All target tables show updated statistics

**4.3 Edge Cases and Error Handling**

**Test Case 3.1: Missing Input Files**

**Objective:** Verify graceful handling of missing TDA/TDI files

**Test Setup:**

* Remove TDA.raw from one area directory
* Remove TDI.raw from another area directory

**Test Steps:**

1. Execute DialCrRawService.processComboRawFiles()
2. Check error handling and logging

**Expected Results:**

* Service continues processing other areas
* Error logged: "ERROR TDA/TDI.raw files are not current for [area]"
* Processing continues for areas with valid files
* Overall process does not fail catastrophically

**Test Case 3.2: Empty Input Files**

**Objective:** Verify handling of zero-byte input files

**Test Setup:**

* Create empty TDA.raw file (0 bytes)
* Create empty TDI.raw file (0 bytes)

**Test Steps:**

1. Execute DialCrRawService.processComboRawFiles()
2. Verify carriage return addition logic

**Expected Results:**

* Empty files detected and carriage returns added
* Logging indicates carriage return addition
* Processing continues appropriately

**Test Case 3.3: Database Connectivity Issues**

**Objective:** Verify handling of database connection failures

**Test Setup:**

* Temporarily disable database connection
* Or provide invalid database credentials

**Test Steps:**

1. Execute any database-dependent service
2. Verify error handling

**Expected Results:**

* Appropriate database connection error messages
* Service fails gracefully without hanging
* Error details logged for troubleshooting
* No partial database updates left in inconsistent state

**Test Case 3.4: Disk Space Issues**

**Objective:** Verify handling of insufficient disk space

**Test Setup:**

* Fill disk space to near capacity
* Attempt file operations

**Test Steps:**

1. Execute services that create large files
2. Monitor disk space and error handling

**Expected Results:**

* Disk space errors detected and logged
* Services fail gracefully
* No corrupted output files left behind
* Clear error messages indicating disk space issue

**4.4 Performance Test Cases**

**Test Case 4.1: Large Data Volume Processing**

**Objective:** Verify performance with production-sized data

**Test Setup:**

* Create TDA.raw files with 100,000+ records
* Create TDI.raw files with 100,000+ records
* Multiple areas with large datasets

**Test Steps:**

1. Execute complete Step 1 process
2. Monitor execution time and resource usage
3. Verify memory usage remains reasonable

**Expected Results:**

* Processing completes within acceptable timeframe (< 30 minutes)
* Memory usage remains stable (no memory leaks)
* Output files are correctly generated
* Database operations complete successfully

**Test Case 4.2: Concurrent Processing**

**Objective:** Verify behavior under concurrent access

**Test Setup:**

* Attempt to run multiple Step 1 processes simultaneously
* Or run Step 1 while other database operations are active

**Test Steps:**

1. Start Step 1 process
2. Attempt to start second Step 1 process
3. Monitor for deadlocks or conflicts

**Expected Results:**

* Appropriate locking or serialization occurs
* No data corruption from concurrent access
* Clear error messages if concurrent execution not supported

**4.5 Data Integrity Test Cases**

**Test Case 5.1: TIN Extraction Validation**

**Objective:** Verify correct TIN extraction from positions 11-21

**Test Data:**

0123456789012345678901234567890... (positions)

1111111111 (TIN at positions 11-21)

**Test Steps:**

1. Create test TDA.raw with known TIN patterns
2. Execute COMBO.raw creation
3. Verify extracted TINs match expected values

**Expected Results:**

* TINs extracted correctly from positions 11-21
* Leading zeros preserved if present
* Invalid TINs handled appropriately

**Test Case 5.2: List Cycle Validation**

**Objective:** Verify single list cycle validation (positions 159-164)

**Test Data:**

* Mix records with different values at positions 159-164
* Some records with consistent list cycle values

**Test Steps:**

1. Execute COMBO.raw creation with mixed list cycles
2. Check validation error detection

**Expected Results:**

* Error logged: "ERROR: there is more than one list\_cyc in COMBO.raw"
* Processing may continue but error is flagged

**4.6 Configuration Test Cases**

**Test Case 6.1: Configuration Validation**

**Objective:** Verify proper handling of configuration values

**Test Steps:**

1. Test with missing configuration values
2. Test with invalid directory paths
3. Test with inaccessible directories

**Expected Results:**

* Clear error messages for missing configuration
* Graceful handling of invalid paths
* Directory creation attempted where appropriate

**Test Case 6.2: Environment Variable Dependencies**

**Objective:** Verify proper use of ApplicationConfig

**Test Steps:**

1. Modify configuration values
2. Verify services use updated configuration
3. Test configuration reload scenarios

**Expected Results:**

* Services use current configuration values
* Configuration changes reflected in processing
* No hardcoded paths bypass configuration

**5. Test Data Requirements**

**5.1 Minimal Test Dataset**

**Small Dataset (for unit testing):**

* **TDA.raw:** 10 records, various TIN values
* **TDI.raw:** 10 records, various TIN values
* **Total processing time:** < 1 minute
* **Areas:** 2-3 test areas (A11, A12, A13)

**Medium Dataset (for integration testing):**

* **TDA.raw:** 1,000 records per area
* **TDI.raw:** 1,000 records per area
* **Total processing time:** 5-10 minutes
* **Areas:** 5-7 areas

**Large Dataset (for performance testing):**

* **TDA.raw:** 100,000+ records per area
* **TDI.raw:** 100,000+ records per area
* **Total processing time:** 15-30 minutes
* **Areas:** All production areas (13+ areas)

**5.2 Test Data Characteristics**

**Valid TIN Patterns:**

* 9-digit numbers: 123456789
* 11-character strings: 12-3456789
* Edge cases: 000000000, 999999999

**Invalid Data Scenarios:**

* Records shorter than 660 characters
* Records longer than 660 characters
* Non-numeric TINs in positions 11-21
* Missing line terminators

**6. Monitoring and Validation Points**

**6.1 Key Metrics to Monitor**

**File Operations:**

* File creation timestamps
* File sizes (before/after compression)
* File permissions
* Directory access patterns

**Database Operations:**

* Query execution times
* Row counts processed
* Transaction commit/rollback status
* Connection pool usage

**System Resources:**

* Memory usage patterns
* Disk space consumption
* CPU utilization during processing
* Network activity (for database connections)

**6.2 Success Criteria**

**Functional Success:**

* ✅ All required output files created
* ✅ Database operations complete successfully
* ✅ No error messages in logs
* ✅ Data integrity validations pass

**Performance Success:**

* ✅ Processing completes within expected timeframe
* ✅ Memory usage remains stable
* ✅ No resource leaks detected

**Data Quality Success:**

* ✅ Output file content matches input data
* ✅ Sorting and filtering applied correctly
* ✅ Statistics reflect current data state
* ✅ Synchronization operations complete

**7. Test Environment Setup**

**7.1 Database Setup Requirements**

-- Ensure test tables exist

CREATE TABLE TINSUMMARY2 (...);

CREATE TABLE DIALVCD2 (...);

CREATE TABLE DIALAUD2 (...);

CREATE TABLE DIALENT2 (...);

CREATE TABLE CONSOLEAD2 (...);

-- Create test data in dial.dialaud

INSERT INTO dial.dialaud (loadarea, comments, loaddt) VALUES

('A11', 'COMBO Tape.', SYSDATE),

('A12', 'COMBO Tape.', SYSDATE);

-- Ensure stored procedures exist or create mocks

CREATE OR REPLACE PROCEDURE dialcopy AS BEGIN NULL; END;

CREATE OR REPLACE PROCEDURE dialcopy2 AS BEGIN NULL; END;

CREATE OR REPLACE PROCEDURE syn2cp AS BEGIN NULL; END;

**7.2 File System Setup**

# Create directory structure

mkdir -p /als-ALS/app/execloc/d.common

mkdir -p /als-ALS/app/loadstage/CONSOLDIR

mkdir -p /als-ALS/app/rawfiles/EXP\_DIR

mkdir -p /als-ALS/app/A11/DIALDIR

mkdir -p /als-ALS/app/A12/DIALDIR

# Create test TDA/TDI files

echo "Sample TDA content with TIN 12345678901..." > /als-ALS/app/A11/DIALDIR/TDA.raw

echo "Sample TDI content with TIN 98765432109..." > /als-ALS/app/A11/DIALDIR/TDI.raw

# Set appropriate permissions

chmod 666 /als-ALS/app/\*/DIALDIR/\*.raw

**7.3 Configuration Setup**

# application-test.properties

app.alsdir=/als-ALS/app

app.consoldir=/als-ALS/app/loadstage/CONSOLDIR

app.raw.dir=/als-ALS/app/rawfiles

app.raw.bkup=/als-ALS/app/rawfiles/backup

app.exp.dir=/als-ALS/app/rawfiles/EXP\_DIR

db.path=A11,A12,A13

**8. Automated Testing Strategy**

**8.1 Unit Test Framework**

@SpringBootTest

@TestPropertySource(locations = "classpath:application-test.properties")

class DialStep1ServiceTests {

@Test

void testComboRawCreation() {

// Arrange: Create test TDA/TDI files

// Act: Execute DialCrRawService

// Assert: Verify COMBO.raw content and sorting

}

@Test

void testStatisticsGathering() {

// Arrange: Ensure test tables exist

// Act: Execute DialTableStatsService

// Assert: Verify statistics updated

}

}

**8.2 Integration Test Framework**

@SpringBootTest

@Sql("/test-data.sql")

class DialStep1IntegrationTests {

@Test

void testCompleteStep1Process() {

// Execute full orchestrator and verify end-to-end functionality

}

}

**8.3 Performance Test Framework**

@Test

@Timeout(value = 30, unit = TimeUnit.MINUTES)

void testLargeDataVolumeProcessing() {

// Test with production-sized datasets

}

This comprehensive testing document provides the foundation for thorough validation of your DIAL Step 1 Java implementation, ensuring both functional correctness and production readiness.