**Dial4 Java Implementation - Test Document**

**1. Overview of Dial4 Functionality**

**Primary Purpose**

The Dial4 process is a multi-step ETL (Extract, Transform, Load) data loading system that populates critical database tables used for dial-related business operations. The process loads data from structured control files into various database schemas to support analytical and operational functions.

**Business Objective**

Dial4 serves as a foundational data loading process that:

* Maintains current dial entity information (DIALENT table)
* Updates dial modification records (DIALMOD table)
* Loads statistical scoring models (MODEL\_SCORE table)
* Populates summary/aggregation data (TINSUMMARY table)

The process ensures data consistency across multiple database schemas (dial, als) and provides comprehensive audit trails through detailed logging.

**High-Level Process Flow**

1. **Step 1**: Load DIALENT data (entity information)
2. **Step 2**: Load DIALMOD data (modification records)
3. **Step 3**: Load MODEL\_SCORE data (scoring models)
4. **Step 4**: Load TINSUMMARY data (summary information)
5. **Step 5**: Execute MODEL\_AVG calculation procedure

**2. Input Specifications**

**Data Sources**

**Control Files (.ctl files)**

The process expects four primary control files in the $CONSOLDIR directory:

1. **DIALENT.ctl**
   * **Purpose**: Contains dial entity master data
   * **Format**: Pipe-delimited (|) fields
   * **Expected Location**: $CONSOLDIR/DIALENT.ctl
2. **DIALMOD.ctl**
   * **Purpose**: Contains dial modification records
   * **Format**: Pipe-delimited (|) fields
   * **Expected Location**: $CONSOLDIR/DIALMOD.ctl
3. **MODELS.ctl**
   * **Purpose**: Contains model scoring data
   * **Format**: Pipe-delimited (|) fields
   * **Expected Location**: $CONSOLDIR/MODELS.ctl
4. **DIALSUM.ctl**
   * **Purpose**: Contains summary/aggregation data
   * **Format**: Pipe-delimited (|) fields
   * **Expected Location**: $CONSOLDIR/DIALSUM.ctl

**File Details and Expected Content Structure**

**DIALENT.ctl Structure**

Based on the ProC code analysis, DIALENT.ctl should contain the following fields:

* OUTSID, ENTTYPE, GL\_SOURCE, GL, PDT, ZPTD, DTASSIGN DATE, ASSACTDT DATE, RWMS
* REPEAT, LARGE, INIT\_PYR
* CURR\_PYR, FR041, FR120, FR1065, TDI\_XREF, AO\_TRNSF, TDACD, NATP2IND, ADTP, ADTP2, CITP, STTP
* CNTRY\_CD, NATP2, IA, DIS\_VTC
* PEN\_ENT\_CD, OIC\_ACC\_YR, RD\_BD\_IND, PDC\_ID\_CD, FR044, Q\_PYR\_IND, FD\_CNTRCT\_IND, PRIOR\_ASSGMNT\_NUM
* PRIOR\_ASSGMNT\_ACT\_DT DATE

**DIALMOD.ctl Structure**

* MODSID, MFTI, DFFER DATE, MODTYPE, CASECODE, BALDUF, STAT\_CYC, CSED\_REV
* CSED DATE, NOACT, ASED, ASED\_REV, OVERAGE, LIEN, BACK\_MTH, LIST\_CYC, CAF
* STAT, CC72, RECTYPE, SELECT\_CD, TDICYC, TDICYC\_O, TDI\_AG\_CYC, LAST\_AMT, DTASSD DATE
* MLT\_ASS, CTVP, ACCEL, NOICS, MOD\_PYR\_IND, NEWMOD, BODCD, BODOLCD, PREDIC\_CD, PREDIC\_UPDT\_CYC
* TDA\_COPYS\_SCORE, LATEST\_MOD\_CSEDDATE, SPECIAL\_PROJ\_CD, SPECIAL\_MOD\_TX\_IND, TOTAL\_DELQ\_YR
* PRIOR\_YR\_RET\_AGT\_AMT, PRIOR\_YR\_RET\_NET\_AMT, PDC\_MOD\_ID, TXPER\_TXPYR\_AMT, ADJ\_GRSS\_INCME\_AMT
* PASSPORT\_LEVY\_971\_IND, DM\_FATCATIND

**MODELS.ctl Structure**

* MTIN, MFS, MTT, MCTRL, ROEMPID, NUM90SCR, NUM190SCR, NUM210SCR, NUM310SCR, NUM330SCR, NUM430SCR, NUM450SCR

**DIALSUM.ctl Structure**

* EMISTIN, EMISAO, AGGBALDUE, STAT\_FLAG, LFI\_FLAG, PYR\_FLAG, AGE\_FLAG, ENT\_SEL\_CD, LARGE\_FLAG
* REPEAT\_FLAG, ENT\_TDI\_XREF, EMISTT, EMISFS, TS\_FATCATIND, TS\_TOT\_DT DATE
* TS\_TOT\_FRP\_INCOME, AGI\_AMT, TPI\_AMT, AGI\_TPI\_TX\_YR, AGI\_TPI\_IND

**Data Constraints and Validations**

**File Size and Volume Expectations**

* Files should contain header records that can be validated
* Expected to process thousands to hundreds of thousands of records per file
* Maximum error threshold: 1,000,000 errors before process termination

**Data Type Constraints**

* **Date Fields**: Must be in YYYYMMDD format
* **Numeric Fields**: Must be valid numbers (integers, decimals as appropriate)
* **Character Fields**: Must respect database column length limits
* **Null Handling**: TRAILING NULLCOLS specification indicates null values are acceptable for trailing columns

**File Validation Requirements**

* Files must exist and be readable
* Files must have proper pipe-delimited format
* Files should not be empty (0 records would trigger error conditions)

**3. Output Specifications**

**Database Interactions**

**Table: dial.DIALENT**

* **Operation**: TRUNCATE and APPEND
* **Purpose**: Stores dial entity master information
* **Expected Columns**: All fields from DIALENT.ctl mapping
* **Row Count**: Should match successful records from DIALENT.ctl
* **Key Business Data**: Entity identifiers, dates, payment information, assignment details

**Table: dial.DIALMOD**

* **Operation**: TRUNCATE and APPEND
* **Purpose**: Stores dial modification records
* **Expected Columns**: All fields from DIALMOD.ctl mapping
* **Row Count**: Should match successful records from DIALMOD.ctl
* **Key Business Data**: Modification details, case codes, statistical cycles, amounts

**Table: als.MODEL\_SCORE**

* **Operation**: TRUNCATE and INSERT
* **Purpose**: Stores model scoring data for analytics
* **Expected Columns**: Model identifiers and various numeric scores
* **Row Count**: Should match successful records from MODELS.ctl
* **Key Business Data**: Model timing, scoring values, employment IDs

**Table: dial.TINSUMMARY**

* **Operation**: APPEND
* **Purpose**: Stores summary and aggregation information
* **Expected Columns**: All fields from DIALSUM.ctl mapping
* **Row Count**: Should match successful records from DIALSUM.ctl
* **Key Business Data**: Summary statistics, flags, amounts, tax year information

**Database Procedure: als.CMPT\_AVG**

* **Operation**: EXECUTE
* **Purpose**: Calculates model averages (likely updates MODEL\_AVG table)
* **Trigger**: Executed after successful completion of MODEL\_SCORE load
* **Expected Outcome**: Populates/updates average score calculations

**File Outputs**

**Log Files**

All log files are created in $CONSOLDIR directory:

1. **DIALENT.log**
   * Contains SQL\*Loader output for DIALENT processing
   * Records successful load counts and any errors
2. **DIALMOD.log**
   * Contains SQL\*Loader output for DIALMOD processing
   * Records successful load counts and any errors
3. **MODELS.log**
   * Contains SQL\*Loader output for MODELS processing
   * Records successful load counts and any errors
4. **DIALSUM.log**
   * Contains SQL\*Loader output for DIALSUM processing
   * Records successful load counts and any errors
5. **diallog** (Master Log File)
   * Contains comprehensive process execution log
   * Includes timestamps for each step
   * Records success/failure status for each component
   * Contains error messages and process completion status

**Error Handling & Logging**

**Error Detection Mechanisms**

* **File Existence**: Process checks for .bad files created by SQL\*Loader
* **Record Count Validation**: Compares successful vs. total records
* **SQL\*Loader Error Threshold**: Maximum 1,000,000 errors per load operation
* **Zero Record Detection**: Identifies when no records are successfully loaded

**Error Reporting**

* **Process Termination**: If critical errors occur, process exits with error status
* **Error Logging**: All errors written to individual component logs and master diallog
* **Timestamp Recording**: All error conditions include date/time stamps
* **Specific Error Messages**: Clear identification of which component failed

**Success Confirmation**

* **Completion Messages**: Each successful step logs completion confirmation
* **Record Counts**: Successful record counts logged for each table load
* **Process Status**: Overall process completion logged with timestamp

**4. Core Logic and Business Rules**

**Sequential Processing Requirements**

The Dial4 process must execute steps in the following order:

1. DIALENT load must complete successfully before DIALMOD
2. DIALMOD load must complete successfully before MODELS
3. MODELS load must complete successfully before DIALSUM
4. DIALSUM load must complete successfully before CMPT\_AVG execution

**Error Handling Logic**

* **Fail-Fast Approach**: Any step failure terminates the entire process
* **File Validation**: Each step validates input file existence before processing
* **Load Validation**: Each step validates successful record loading before proceeding
* **Rollback Consideration**: Failed processes may leave data in inconsistent state

**Data Transformation Rules**

* **TRUNCATE Operations**: DIALENT, DIALMOD, and MODEL\_SCORE tables are completely refreshed
* **APPEND Operations**: TINSUMMARY table accumulates data (historical preservation)
* **Date Formatting**: All date fields must be properly formatted for database insertion
* **Null Handling**: Trailing null columns are acceptable and handled appropriately

**Business Rule Validations**

* **Data Integrity**: Foreign key relationships between tables must be maintained
* **Referential Consistency**: Entity IDs in DIALMOD should correspond to DIALENT entities
* **Temporal Consistency**: Date fields should represent logical business dates
* **Scoring Validity**: Model scores should fall within expected numerical ranges

**5. Test Case Scenarios & Data**

**5.1 Positive/Happy Path Scenarios**

**Test Case 1: Complete Successful Processing**

**Objective**: Verify end-to-end successful execution with valid data

**Input Conditions**:

* All four .ctl files present and properly formatted
* Each file contains representative valid data (100-1000 records)
* All required directories exist and are writable
* Database connections are available
* All target tables are accessible

**Expected Outputs**:

* dial.DIALENT: Records loaded matching DIALENT.ctl count
* dial.DIALMOD: Records loaded matching DIALMOD.ctl count
* als.MODEL\_SCORE: Records loaded matching MODELS.ctl count
* dial.TINSUMMARY: Records appended matching DIALSUM.ctl count
* als.CMPT\_AVG procedure executes successfully
* All log files created with success messages
* Master diallog shows all steps completed
* No .bad files generated

**Success Criteria**:

* Process completes with exit code 0
* All expected database records present
* All log files indicate successful processing
* Record counts match between files and database tables

**Test Case 2: Large Volume Processing**

**Objective**: Verify performance with realistic data volumes

**Input Conditions**:

* DIALENT.ctl: 50,000+ records
* DIALMOD.ctl: 100,000+ records
* MODELS.ctl: 10,000+ records
* DIALSUM.ctl: 25,000+ records

**Expected Outputs**:

* Process completes within acceptable time limits
* Memory usage remains stable
* All records processed successfully
* Log files reflect accurate counts

**5.2 Negative/Error Scenarios**

**Test Case 3: Missing Input Files**

**Objective**: Verify proper error handling when input files are missing

**Input Conditions**:

* One or more .ctl files missing from $CONSOLDIR
* Test each file missing individually
* Test multiple files missing simultaneously

**Expected Outputs**:

* Process terminates with appropriate error message
* Error logged to diallog with timestamp
* No database modifications occur
* Clear indication of which file(s) are missing

**Test Case 4: File Format Errors**

**Objective**: Verify handling of malformed input files

**Input Conditions**:

* Files with incorrect delimiters
* Files with wrong number of columns
* Files with invalid date formats
* Files with invalid numeric data
* Empty files (0 bytes)

**Expected Outputs**:

* SQL\*Loader generates .bad files
* Error counts logged appropriately
* Process fails when error threshold exceeded
* Specific format errors identified in logs

**Test Case 5: Database Connection Failures**

**Objective**: Verify error handling for database connectivity issues

**Input Conditions**:

* Database unavailable during processing
* Invalid database credentials
* Target tables locked or inaccessible
* Insufficient database permissions

**Expected Outputs**:

* Clear database error messages in logs
* Process terminates gracefully
* No partial data commits
* Appropriate error codes returned

**Test Case 6: Disk Space/Permission Issues**

**Objective**: Verify handling of file system problems

**Input Conditions**:

* Insufficient disk space for log files
* No write permissions to $CONSOLDIR
* Log directory does not exist

**Expected Outputs**:

* Process handles file system errors gracefully
* Appropriate error messages generated
* Process terminates without corruption

**5.3 Edge Cases**

**Test Case 7: Boundary Data Values**

**Objective**: Test processing of edge case data values

**Input Conditions**:

* Maximum length character fields
* Minimum/maximum numeric values
* Edge case dates (leap years, century boundaries)
* Records with all trailing nulls
* Records with mixed null/non-null trailing fields

**Expected Outputs**:

* All valid boundary values processed correctly
* Appropriate handling of null values
* No truncation or overflow errors

**Test Case 8: Zero Record Files**

**Objective**: Verify handling of files with header only or no data

**Input Conditions**:

* Files containing only headers
* Files with headers but no data records
* Files with only whitespace

**Expected Outputs**:

* Process detects zero record condition
* Appropriate warning or error messages
* Decision logic for whether to continue or terminate

**Test Case 9: Duplicate Data Processing**

**Objective**: Verify handling of duplicate records

**Input Conditions**:

* Files containing duplicate key values
* Files with identical records
* Multiple executions with same data

**Expected Outputs**:

* Database constraints handle duplicates appropriately
* Process behavior is predictable and documented
* Truncate operations properly clear previous data

**5.4 Performance/Volume Testing**

**Test Case 10: Stress Testing**

**Objective**: Validate performance under maximum expected load

**Input Conditions**:

* Maximum anticipated file sizes
* Concurrent execution attempts
* Resource constrained environments

**Expected Outputs**:

* Process completes within SLA timeframes
* Resource usage remains within acceptable limits
* Error handling remains robust under stress

**Performance Benchmarks**:

* Memory usage monitoring
* CPU utilization tracking
* Database connection pooling efficiency
* I/O throughput measurements

**5.5 Data Validation Test Cases**

**Test Case 11: Data Integrity Verification**

**Objective**: Ensure data accuracy through the transformation process

**Input Conditions**:

* Known test data with calculated expected outcomes
* Reference data for cross-validation
* Data with specific business rule implications

**Expected Outputs**:

* Loaded data matches expected transformations
* Business rule validations pass
* Referential integrity maintained across tables

**Validation Methods**:

* Source-to-target data reconciliation
* Business rule compliance checking
* Cross-table relationship validation

**5.6 Recovery and Restart Testing**

**Test Case 12: Process Interruption Recovery**

**Objective**: Verify system behavior when process is interrupted

**Input Conditions**:

* Process termination at various stages
* System crashes during execution
* Network interruptions during database operations

**Expected Outputs**:

* System state remains consistent
* Clear indication of where process stopped
* Ability to restart process cleanly

**Test Case 13: Rollback Scenarios**

**Objective**: Verify data consistency when errors occur mid-process

**Input Conditions**:

* Errors occurring after partial data loads
* Database constraint violations
* System resource exhaustion

**Expected Outputs**:

* Database remains in consistent state
* Clear rollback procedures documented
* No orphaned or inconsistent data

**6. Test Data Requirements**

**6.1 Minimal Test Data Sets**

Create small representative datasets for each .ctl file containing:

* Valid records covering all field types
* Records with various null/non-null combinations
* Records representing different business scenarios

**6.2 Comprehensive Test Data Sets**

Develop larger datasets that include:

* Statistical distribution of typical production data
* Edge cases and boundary conditions
* Invalid data for error testing
* Performance testing volumes

**6.3 Test Data Management**

* Establish procedures for test data creation and maintenance
* Document expected outcomes for each test dataset
* Create automated data validation scripts
* Maintain version control for test data sets

**7. Environment and Configuration Testing**

**7.1 Environment Variables**

**Critical Variables to Test**:

* $CONSOLDIR path configuration
* $DIAL path configuration
* Database connection parameters
* File permission settings

**7.2 Configuration Validation**

* Verify all paths are accessible
* Confirm database connectivity
* Validate file permissions
* Test directory creation capabilities

**8. Monitoring and Alerting**

**8.1 Process Monitoring Requirements**

* Real-time process status monitoring
* Progress tracking for long-running operations
* Resource utilization monitoring
* Error detection and alerting

**8.2 Log Analysis Requirements**

* Automated log parsing for error detection
* Performance metrics extraction
* Trend analysis capabilities
* Historical process comparison

**9. Documentation and Maintenance**

**9.1 Process Documentation**

* Complete process flow documentation
* Error code reference guide
* Troubleshooting procedures
* Performance tuning guidelines

**9.2 Test Maintenance**

* Regular test case review and updates
* Test data refresh procedures
* Automated test execution capabilities
* Results tracking and analysis

This comprehensive test document provides the foundation for thorough testing of the Dial4 Java implementation, ensuring that all aspects of the legacy ProC functionality are properly validated and maintained in the new system.