PARITY REPORT · SAMPLE · BEERS LABS LLC

# Ironparse Parity Report — ACORD_AL3_NIGHTMARE.CPY

The artifact every pilot delivers, one per copybook. Every number reproduces from the input — re-run the engine inside your perimeter and the hashes match.

## 1 · ENGAGEMENT & VERDICT

| | |
|---|---|
| Copybook | ACORD_AL3_NIGHTMARE.CPY |
| Run environment | customer VPC — zero data egress |
| Verdict | **PASS** |

## 2 · PROVENANCE (REPRODUCE THIS)

| | |
|---|---|
| Input SHA-256 | a3297567bd4b766668235bd20706da1f2bae39e842c5bbf43f5224de2d01cf0c |
| Output SHA-256 | e3b252076d57ee9fe348d14c4dab30e0d4a896a5551dd043b8be75b2cb40249a |
| Determinism | same input → identical output hash, every run |

## 3 · STRUCTURE

| | |
|---|---|
| Records | 11 |
| Elementary fields | 80 |
| REDEFINES overlays | 2 |
| OCCURS (incl. ODO) | 8 |

## 4 · QUALITY GATES

| # | GATE | RESULT | WHAT IT PROVES | MEASURE |
|---|---|---|---|---|
| 01 | PARSER | **PASS** | Deterministic AST extraction yields a non-empty, field-bearing record set. No LLM involved. | 11 records · 80 elementary fields |
| 02 | SCHEMA_SANITY | **PASS** | Candidate schema parses as valid Zod — balanced delimiters, no markdown fences, no prose. | valid Zod module |
| 03 | FIELD_PARITY | **PASS** | len(COBOL elementary fields) === len(emitted schema leaves). Off by one and the build fails. | COBOL 80 ⇄ schema 80 |
| 04 | DARK_CORNER | **PASS** | Every REDEFINES overlay compiles to a union (discriminated where a record-type byte exists); every OCCURS to a dynamic array. | 2/2 overlays · 8/8 arrays |
| 05 | MOCK_STRUCTURE | **PASS** | A mock document generated from the schema re-validates against it — the schema is internally consistent. | 82 nodes round-tripped |

## 5 · FIELD-FOR-FIELD MAPPING · 80 FIELDS

| COBOL FIELD | PIC | STORAGE | SCHEMA PATH | TYPE |
|---|---|---|---|---|
| TRANSACTION-TYPE | X(4) | alphanumeric (4 chars) | acordAl3Record.transactionType | z.string().max(4) |
| MESSAGE-ID | X(8) | alphanumeric (8 chars) | acordAl3Record.messageId | z.string().max(8) |
| TIMESTAMP | X(26) | alphanumeric (26 chars) | acordAl3Record.timestamp | z.string().max(26) |
| STATE-CODE | X(2) | alphanumeric (2 chars) | acordAl3Record.stateCode | z.string().max(2) |
| CARRIER-CODE | X(4) | alphanumeric (4 chars) | acordAl3Record.carrierCode | z.string().max(4) |
| POLICY-NUMBER | X(20) | alphanumeric (20 chars) | acordAl3Record.policyNumber | z.string().max(20) |

| COBOL FIELD | PIC | STORAGE | SCHEMA PATH | TYPE |
|---|---|---|---|---|
| NAME-TYPE | X(1) | alphanumeric (1 chars) | insuredNameData.nameType | z.string().max(1) |
| INSURED-LAST-NAME | X(30) | alphanumeric (30 chars) | insuredNameData.insuredLastName | z.string().max(30) |
| INSURED-FIRST-NAME | X(20) | alphanumeric (20 chars) | insuredNameData.insuredFirstName | z.string().max(20) |
| INSURED-MIDDLE-NAME | X(15) | alphanumeric (15 chars) | insuredNameData.insuredMiddleName | z.string().max(15) |
| INSURED-SUFFIX | X(10) | alphanumeric (10 chars) | insuredNameData.insuredSuffix | z.string().max(10) |
| ADDR-INDICATOR | X(1) | alphanumeric (1 chars) | insuredAddressData.addrIndicator | z.string().max(1) |
| ADDRESS-LINE-1 | X(35) | alphanumeric (35 chars) | insuredAddressData.addressLine1 | z.string().max(35) |
| ADDRESS-LINE-2 | X(35) | alphanumeric (35 chars) | insuredAddressData.addressLine2 | z.string().max(35) |
| CITY | X(25) | alphanumeric (25 chars) | insuredAddressData.city | z.string().max(25) |
| STATE | X(2) | alphanumeric (2 chars) | insuredAddressData.state | z.string().max(2) |
| ZIP-CODE | X(9) | alphanumeric (9 chars) | insuredAddressData.zipCode | z.string().max(9) |
| POLICY-EFF-DATE | X(8) | alphanumeric (8 chars) | policyInfo.policyEffDate | z.string().max(8) |
| POLICY-EXP-DATE | X(8) | alphanumeric (8 chars) | policyInfo.policyExpDate | z.string().max(8) |
| PREMIUM-AMOUNT | S9(9)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | policyInfo.premiumAmount | z.number() |
| PREMIUM-PAYMENT-MODE | X(1) | alphanumeric (1 chars) | policyInfo.premiumPaymentMode | z.string().max(1) |
| POLICY-STATUS | X(1) | alphanumeric (1 chars) | policyInfo.policyStatus | z.string().max(1) |
| VEHICLE-COUNT | 9(2) | display numeric (2 digits) | vehicleDetail.vehicleCount | z.string().regex(/^\d{1,2} |
| VIN | X(17) | alphanumeric (17 chars) | vehicleDetail.vehicleEntry[].vin | z.string().max(17) |
| YEAR | 9(4) | display numeric (4 digits) | vehicleDetail.vehicleEntry[].year | z.string().regex(/^\d{1,4} |
| MAKE | X(15) | alphanumeric (15 chars) | vehicleDetail.vehicleEntry[].make | z.string().max(15) |
| MODEL | X(20) | alphanumeric (20 chars) | vehicleDetail.vehicleEntry[].model | z.string().max(20) |

| COBOL FIELD | PIC | STORAGE | SCHEMA PATH | TYPE |
|---|---|---|---|---|
| VEHICLE-CLASS | X(3) | alphanumeric (3 chars) | vehicleDetail.vehicleEntry[].vehicleClass | z.string().max(3) |
| COVERAGE-AMOUNT | S9(7)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | vehicleDetail.vehicleEntry[].coverageAmount | z.number() |
| DEDUCTIBLE-AMOUNT | S9(5)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | vehicleDetail.vehicleEntry[].deductibleAmount | z.number() |
| DRIVER-COUNT | 9(1) | display numeric (1 digits) | vehicleDetail.vehicleEntry[].driverCount | z.string().regex(/^\d{1,1} |
| DRIVER-NAME | X(30) | alphanumeric (30 chars) | vehicleDetail.vehicleEntry[].driverList[].driverName | z.string().max(30) |
| DRIVER-LICENSE | X(20) | alphanumeric (20 chars) | vehicleDetail.vehicleEntry[].driverList[].driverLicense | z.string().max(20) |
| DRIVER-DOB | X(8) | alphanumeric (8 chars) | vehicleDetail.vehicleEntry[].driverList[].driverDob | z.string().max(8) |
| DRIVER-RELATION | X(1) | alphanumeric (1 chars) | vehicleDetail.vehicleEntry[].driverList[].driverRelation | z.string().max(1) |
| DH-RECORD-TYPE | X(1) | alphanumeric (1 chars) | driverHistory.dhRecordType | z.string().max(1) |
| DH-LAST-UPDATE | X(8) | alphanumeric (8 chars) | driverHistory.dhLastUpdate | z.string().max(8) |
| DH-ACCIDENT-COUNT | 9(2) | display numeric (2 digits) | driverHistory.dhAccidentCount | z.string().regex(/^\d{1,2} |
| DH-VIOLATION-COUNT | 9(2) | display numeric (2 digits) | driverHistory.dhViolationCount | z.string().regex(/^\d{1,2} |
| DH-MVR-SCORE | 9(3) | display numeric (3 digits) | driverHistory.dhMvrScore | z.string().regex(/^\d{1,3} |
| DH-CLAIM-DATE | X(8) | alphanumeric (8 chars) | driverHistory.dhClaimHistory[].dhClaimDate | z.string().max(8) |
| DH-CLAIM-AMOUNT | S9(6)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | driverHistory.dhClaimHistory[].dhClaimAmount | z.number() |
| DH-CLAIM-TYPE | X(2) | alphanumeric (2 chars) | driverHistory.dhClaimHistory[].dhClaimType | z.string().max(2) |
| COVERAGE-COUNT | 9(2) | display numeric (2 digits) | coverageOptions.coverageCount | z.string().regex(/^\d{1,2} |
| COVERAGE-TYPE | X(3) | alphanumeric (3 chars) | coverageOptions.coverageEntry[].coverageType | z.string().max(3) |

| COBOL FIELD | PIC | STORAGE | SCHEMA PATH | TYPE |
|---|---|---|---|---|
| COVERAGE-LIMIT | S9(7)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | coverageOptions.coverageEntry[].coverageLimit | z.number() |
| COVERAGE-PREMIUM | S9(6)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | coverageOptions.coverageEntry[].coveragePremium | z.number() |
| COVERAGE-STATUS | X(1) | alphanumeric (1 chars) | coverageOptions.coverageEntry[].coverageStatus | z.string().max(1) |
| DEDUCTIBLE-APPLIED | S9(5)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | coverageOptions.coverageEntry[].deductibleApplied | z.number() |
| SUPPLEMENT-COUNT | 9(2) | display numeric (2 digits) | supplementalExposures.supplementCount | z.string().regex(/^\d{1,2} |
| SUPPLEMENT-TYPE | X(4) | alphanumeric (4 chars) | supplementalExposures.supplementData[].supplementType | z.string().max(4) |
| SUPPLEMENT-CODE | X(6) | alphanumeric (6 chars) | supplementalExposures.supplementData[].supplementCode | z.string().max(6) |
| SUPPLEMENT-DESC | X(50) | alphanumeric (50 chars) | supplementalExposures.supplementData[].supplementDesc | z.string().max(50) |
| SUPPLEMENT-AMOUNT | S9(7)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | supplementalExposures.supplementData[].supplementAmount | z.number() |
| EXCLUSION-IND | X(1) | alphanumeric (1 chars) | supplementalExposures.supplementData[].exclusionInd | z.string().max(1) |
| TOTAL-CLAIMS | 9(3) | display numeric (3 digits) | claimHistoryRecord.totalClaims | z.string().regex(/^\d{1,3} |
| CLAIM-DATE | X(8) | alphanumeric (8 chars) | claimHistoryRecord.claimDetails[].claimDate | z.string().max(8) |
| CLAIM-TYPE | X(2) | alphanumeric (2 chars) | claimHistoryRecord.claimDetails[].claimType | z.string().max(2) |
| CLAIM-AMOUNT | S9(8)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | claimHistoryRecord.claimDetails[].claimAmount | z.number() |
| CLAIM-STATUS | X(1) | alphanumeric (1 chars) | claimHistoryRecord.claimDetails[].claimStatus | z.string().max(1) |
| CLAIM-DESCRIPTION | X(100) | alphanumeric (100 chars) | claimHistoryRecord.claimDetails[].claimDescription | z.string().max(100) |
| LOSS-DATE | X(8) | alphanumeric (8 chars) | claimHistoryRecord.claimDetails[].lossDate | z.string().max(8) |

| COBOL FIELD | PIC | STORAGE | SCHEMA PATH | TYPE |
|---|---|---|---|---|
| REPORT-DATE | X(8) | alphanumeric (8 chars) | claimHistoryRecord.claimDetails[].reportDate | z.string().max(8) |
| BILL-ACCOUNT-NUMBER | X(15) | alphanumeric (15 chars) | billingAccount.billAccountNumber | z.string().max(15) |
| BILL-PAYMENT-TYPE | X(1) | alphanumeric (1 chars) | billingAccount.billPaymentType | z.string().max(1) |
| BILL-CREDIT-CARD | X(16) | alphanumeric (16 chars) | billingAccount.billCreditCard | z.string().max(16) |
| BILL-EXPIRY | X(4) | alphanumeric (4 chars) | billingAccount.billExpiry | z.string().max(4) |
| BILL-AUTH-CODE | X(10) | alphanumeric (10 chars) | billingAccount.billAuthCode | z.string().max(10) |
| LAST-PAYMENT-DATE | X(8) | alphanumeric (8 chars) | billingAccount.lastPaymentDate | z.string().max(8) |
| LAST-PAYMENT-AMOUNT | S9(7)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | billingAccount.lastPaymentAmount | z.number() |
| BALANCE-DUE | S9(8)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | billingAccount.balanceDue | z.number() |
| PAY-DATE | X(8) | alphanumeric (8 chars) | billingAccount.paymentHistory[].payDate | z.string().max(8) |
| PAY-AMOUNT | S9(6)V99 COMP-3 | packed decimal (COMP-3, signed, scaled) | billingAccount.paymentHistory[].payAmount | z.number() |
| PAY-RESULT | X(2) | alphanumeric (2 chars) | billingAccount.paymentHistory[].payResult | z.string().max(2) |
| NOTE-COUNT | 9(2) | display numeric (2 digits) | notesAndRemarks.noteCount | z.string().regex(/^\d{1,2} |
| NOTE-DATE | X(8) | alphanumeric (8 chars) | notesAndRemarks.noteEntry[].noteDate | z.string().max(8) |
| NOTE-AUTHOR | X(20) | alphanumeric (20 chars) | notesAndRemarks.noteEntry[].noteAuthor | z.string().max(20) |
| NOTE-TYPE | X(1) | alphanumeric (1 chars) | notesAndRemarks.noteEntry[].noteType | z.string().max(1) |
| NOTE-TEXT | X(200) | alphanumeric (200 chars) | notesAndRemarks.noteEntry[].noteText | z.string().max(200) |
| NOTE-STATUS | X(1) | alphanumeric (1 chars) | notesAndRemarks.noteEntry[].noteStatus | z.string().max(1) |

```
import { z } from "zod";

export const RecordSchema = z.object({
  acordAl3Record: z.object({
    transactionType: z.string().max(4),
    messageId: z.string().max(8),
    timestamp: z.string().max(26),
    stateCode: z.string().max(2),
    carrierCode: z.string().max(4),
    policyNumber: z.string().max(20),
  }),
  // REDEFINES overlay: INSURED-NAME-DATA / INSURED-ADDRESS-DATA
  insuredNameData: z.union([
    z.object({
      nameType: z.string().max(1),
      insuredLastName: z.string().max(30),
      insuredFirstName: z.string().max(20),
      insuredMiddleName: z.string().max(15),
      insuredSuffix: z.string().max(10),
    }),
    z.object({
      addrIndicator: z.string().max(1),
      addressLine1: z.string().max(35),
      addressLine2: z.string().max(35),
      city: z.string().max(25),
      state: z.string().max(2),
      zipCode: z.string().max(9),
    })
  ]),
  policyInfo: z.object({
    policyEffDate: z.string().max(8),
    policyExpDate: z.string().max(8),
    premiumAmount: z.number(),
    premiumPaymentMode: z.string().max(1),
    policyStatus: z.string().max(1),
  }),
  // REDEFINES overlay: VEHICLE-DETAIL / DRIVER-HISTORY
  vehicleDetail: z.union([
    z.object({
      vehicleCount: z.string().regex(/^\d{1,2}$/),
      vehicleEntry: z.array(z.object({
        vin: z.string().max(17),
        year: z.string().regex(/^\d{1,4}$/),
        make: z.string().max(15),
        model: z.string().max(20),
        vehicleClass: z.string().max(3),
        coverageAmount: z.number(),
        deductibleAmount: z.number(),
        driverCount: z.string().regex(/^\d{1,1}$/),
        driverList: z.array(z.object({
          driverName: z.string().max(30),
          driverLicense: z.string().max(20),
          driverDob: z.string().max(8),
          driverRelation: z.string().max(1),
        })).max(5) /* OCCURS 0..5 DEPENDING ON DRIVER-COUNT */,
      })).max(10) /* OCCURS 0..10 DEPENDING ON VEHICLE-COUNT */,
    }),
    z.object({
      dhRecordType: z.string().max(1),
      dhLastUpdate: z.string().max(8),
      dhAccidentCount: z.string().regex(/^\d{1,2}$/),
      dhViolationCount: z.string().regex(/^\d{1,2}$/),
      dhMvrScore: z.string().regex(/^\d{1,3}$/),
      dhClaimHistory: z.array(z.object({
        dhClaimDate: z.string().max(8),
        dhClaimAmount: z.number(),
        dhClaimType: z.string().max(2),
      })).max(5) /* OCCURS 5 */,
    })
  ]),
  coverageOptions: z.object({
    coverageCount: z.string().regex(/^\d{1,2}$/),
```

```
    coverageEntry: z.array(z.object({
      coverageType: z.string().max(3),
      coverageLimit: z.number(),
      coveragePremium: z.number(),
      coverageStatus: z.string().max(1),
      deductibleApplied: z.number(),
    })).max(8) /* OCCURS 0..8 DEPENDING ON COVERAGE-COUNT */,
  }),
  supplementalExposures: z.object({
    supplementCount: z.string().regex(/^\d{1,2}$/),
    supplementData: z.array(z.object({
      supplementType: z.string().max(4),
      supplementCode: z.string().max(6),
      supplementDesc: z.string().max(50),
      supplementAmount: z.number(),
      exclusionInd: z.string().max(1),
    })).max(20) /* OCCURS 0..20 DEPENDING ON SUPPLEMENT-COUNT */,
  }),
  claimHistoryRecord: z.object({
    totalClaims: z.string().regex(/^\d{1,3}$/),
    claimDetails: z.array(z.object({
      claimDate: z.string().max(8),
      claimType: z.string().max(2),
      claimAmount: z.number(),
      claimStatus: z.string().max(1),
      claimDescription: z.string().max(100),
      lossDate: z.string().max(8),
      reportDate: z.string().max(8),
    })).max(15) /* OCCURS 0..15 DEPENDING ON TOTAL-CLAIMS */,
  }),
  billingAccount: z.object({
    billAccountNumber: z.string().max(15),
    billPaymentType: z.string().max(1),
    billCreditCard: z.string().max(16),
    billExpiry: z.string().max(4),
    billAuthCode: z.string().max(10),
    lastPaymentDate: z.string().max(8),
    lastPaymentAmount: z.number(),
    balanceDue: z.number(),
    paymentHistory: z.array(z.object({
      payDate: z.string().max(8),
      payAmount: z.number(),
      payResult: z.string().max(2),
    })).max(12) /* OCCURS 12 */,
  }),
  notesAndRemarks: z.object({
    noteCount: z.string().regex(/^\d{1,2}$/),
    noteEntry: z.array(z.object({
      noteDate: z.string().max(8),
      noteAuthor: z.string().max(20),
      noteType: z.string().max(1),
      noteText: z.string().max(200),
      noteStatus: z.string().max(1),
    })).max(10) /* OCCURS 0..10 DEPENDING ON NOTE-COUNT */,
  }),
});

export type Record = z.infer<typeof RecordSchema>;
```