Program: CBTRN03C

Archetype: batch_report · **Kind:** batch · **Source:** `data/expanded_source_v2/CBTRN03C.expanded.cbl` .
STB: `data/stb_unified_v2/CBTRN03C_unified_stb.json`

s01 — Program Narrative

Purpose

`CBTRN03C` is a batch COBOL report writer that produces the *Daily Transaction Report* (`'DALYREPT'`, `REPT-SHORT-NAME` at L176–L177; long-name literal `'Daily Transaction Report'` at L178–L179). The program reads a daily transaction extract (`TRANSACT-FILE`, L29), enriches each record with three reference-file lookups (card-account cross-reference, transaction type, transaction category) and a date-range window read from a parameter file (`DATE-PARMS-FILE`, L55), then writes a paginated report to `REPORT-FILE` (`TRANREPT`, L51).

`PROGRAM-ID. CBTRN03C` is declared at L23. The header comment block at L1–L21 records "*Function: Print the transaction detail report*". The program runs under the JCL job `TRANREPT` in step `STEP10R` (`jcl_facts.tier1_core.invocation`, `jcl/TRANREPT.jcl` L59).

Pipeline position. `STEP10R` is the **third and terminal step** of the `TRANREPT` job (`jcl_facts.tier2_enhanced.step_sequence` — order 1: `STEP05R.PRC001`/IDCAMS, order 2: `STEP05R`/SORT, order 3: `STEP10R`/CBTRN03C). `STEP05R` (SORT) is the named upstream step that feeds `TRANFILE` via `SORTOUT` (`jcl_facts.tier2_enhanced.pipeline.upstream_steps[0]`); `downstream_steps[]` is empty.

Step-ordering context. This program runs **after** the IDCAMS housekeeping sub-step (which copies and catalogs `AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS` to a backup) and **after** the SORT step (which date-range-filters and card-key-sorts the backup into `AWS.M2.CARDDEMO.TRANSACT.DALY(+1)`). CBTRN03C consumes that daily extract.

Prerequisites. `jcl_facts.tier2_enhanced.prerequisites[]` lists three reference KSDS lookups (`CARDXREF`, `TRANTYPE`, `TRANCATG`) and one parameter file (`DATEPARM`) — these must exist and be populated before this step runs. The JCL does **not** name any external prerequisite jobs.

Trigger / cadence. `jcl_facts.tier2_enhanced` carries no explicit schedule. The trigger is JCL submission; no scheduler-driven cadence, cron-equivalent, or run-frequency is declared in `jcl_facts`. The job name `TRANREPT` and the upstream `DALY` GDG naming convention are suggestive of a daily run, but that is not asserted by the STB and is not stated here as fact.

I/O Summary

The program declares six files in `ENVIRONMENT DIVISION` (L26–L57, `environment_division.file_control_enhanced`):

Logical file	DDNAME (assign)	Organization	Access	Record key	File-status field	Source line
`TRANSACT-FILE`	`TRANFILE`	Sequential	(sequential)	—	`TRANFILE-STATUS`	L29
`XREF-FILE`	`CARDXREF`	Indexed	RANDOM	`FD-XREF-`	`CARDXREF-STATUS`	L33

		Indexed	RANDOM	CARD-NUM		L39
TRANTYPE-FILE	TRANTYPE	Indexed	RANDOM	FD-TRAN-TYPE	TRANTYPE-STATUS	L39
TRANCATG-FILE	TRANCATG	Indexed	RANDOM	FD-TRAN-CAT-KEY	TRANCATG-STATUS	L45
REPORT-FILE	TRANREPT	Sequential	(sequential)	—	TRANREPT-STATUS	L51
DATE-PARMS-FILE	DATEPARM	Sequential	(sequential)	—	DATEPARM-STATUS	L55

I/O role per `io.fd_layouts` and `io.summary.by_file`:

- **Inputs (read):** `TRANSACT-FILE` (sequential primary driver), `XREF-FILE` (random lookup), `TRANTYPE-FILE` (random lookup), `TRANCATG-FILE` (random lookup), `DATE-PARMS-FILE` (sequential single-read).
- **Output (write):** `REPORT-FILE` — the `WRITE` site is `1111-WRITE-REPORT-REC` (`io.summary.by_file["FD-REPTFILE-REC"]`), the sole WRITE in the program (`statements.by_verb.WRITE = 1`).

Negative Assertions

Each item below comes from `notable_constructs.absence_facts[]` (STB-grounded deterministic absence):

- No embedded SQL (`no_sql`).
- No IMS DL/I calls (`no_ims_dli`).
- No SORT or MERGE operations *inside* this COBOL program (`no_sort_merge`). Note: the **JCL pipeline** contains a SORT step (`STEP05R`, `jcl/TRANREPT.jcl` L37–L55) that pre-filters and sorts the input; that SORT runs outside the program.
- No CALL to application subprograms (`no_app_calls`). The sole CALL target is the LE service routine `CEE3ABD` (L762), which is a language-environment abend, not an application module.
- No REWRITE or DELETE operations (`no_rewrite_delete`, `no_delete`) — the program is read-only against its input files.
- No STRING, UNSTRING, or INSPECT operations (`no_string_inspect`).
- No COMPUTE statements (`no_compute`) — arithmetic is via `ADD` and `SUBTRACT`.
- No ACCEPT statements (`no_accept`).
- No GO TO statements (`no_goto`).
- No `ON EXCEPTION` handling on the CALL statement (`no_on_exception`). The `CALL 'CEE3ABD' USING ABCODE, TIMING.` at L762 has no exception clause — the abend is itself the terminal action.

s02 — System-Level Context

JCL Context

From `jcl_facts.tier1_core.invocation` (`jcl/TRANREPT.jcl` L59):

- **Job:** `TRANREPT` (L1 of JCL).
- **Step:** `STEP10R` invoking `PGM=CBTRN03C` (L59 of JCL).
- **PARM:** none — `parm` is `null`. The program reads its date-range window from `DATEPARM` instead of from `PARM`.
- **MSGCLASS:** `0` (L2 of JCL).
- **NOTIFY:** `&SYSUID` (L2 of JCL).
- **JCLLIB:** `AWS.M2.CARDDEMO.PROC` (L19 of JCL).
- **CLASS:** `A` (`jcl_facts.tier3_operational.resources.class`).

DD Statements

From `jcl_facts.tier1_core.dd_mappings`:

DDNAME	DSN	DISP	Type	LRECL/RECFM	Notes
STEPLIB	AWS.M2.CARDDEMO.LOADLIB	SHR	Sequent	—	Program load library
SYSOUT	—	—	(SYSOUT)	—	DISPLAY message
SYSPRINT	—	—	(SYSOUT)	—	System print
TRANFILE	AWS.M2.CARDDEMO.TRANSACT.DALY(+1)	SHR	Sequent GDG	—	Pre-sorted daily extract (input)
CARDXREF	AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS	SHR	VSAM KSDS	—	Card→ac cross-reference (input lookup)
TRANTYPE	AWS.M2.CARDDEMO.TRANTYPE.VSAM.KSDS	SHR	VSAM KSDS	—	Transacti type catalog (input lookup)
TRANCATG	AWS.M2.CARDDEMO.TRANCATG.VSAM.KSDS	SHR	VSAM KSDS	—	Transacti category catalog (input lookup)
DATEPARM	AWS.M2.CARDDEMO.DATEPARM	SHR	Sequent	—	Reportin date-range paramete file
TRANREPT	AWS.M2.CARDDEMO.TRANREPT(+1)	NEW,CATLG,DELETE	Sequent GDG	LRECL=133, RECFM=FB, ASA carriage control	Output report (new generatic

`SPACE=(CYL,(1,1),RLSE)`, `UNIT=SYSDA` for `TRANREPT` (`jcl_facts.tier3_operational.storage.TRANREPT`).

Pipeline Step Sequence

From `jcl_facts.tier2_enhanced.step_sequence`:

1. `STEP05R.PRC001` invokes `IDCAMS` — copies `AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS` to `AWS.M2.CARDDEMO.TRANSACT.BKUP(+1)` (housekeeping per `jcl_facts.tier3_operational.housekeeping`).
2. `STEP05R` invokes `SORT` — reads `TRANSACT.BKUP(+1)`, filters by `INCLUDE COND=(TRAN-PROC-DT,GE,PARM-START-DATE,AND,TRAN-PROC-DT,LE,PARM-END-DATE)` with literal date window `'2022-01-01'` through `'2022-07-06'` (`jcl/TRANREPT.jcl` L40–L48), sorts ascending by `TRAN-CARD-NUM` at position 263 length 16 (`jcl_facts.tier1_core.sort_specifications.fields[0]`), and writes the result to `AWS.M2.CARDDEMO.TRANSACT.DALY(+1)` — the `TRANFILE` input to this step.
3. `STEP10R` invokes `CBTRN03C` — this program.

The date-range INCLUDE in SORT is *separate from* the date-range check inside the program at L305–L310: the SORT-step parameters `PARM-START-DATE` and `PARM-END-DATE` are JCL `SYMNAMES` literals (`jcl/TRANREPT.jcl` L43–L44), while `WS-START-DATE` and `WS-END-DATE` (L255, L257) are loaded by the program from `DATE-PARMS-FILE` at `0550-DATEPARM-READ` (L353). They are nominally aligned in production but are independently sourced.

s03 — Data Dictionary

File Section Records

From `data_division.record_layouts` / `io.fd_layouts`:

`FD-TRANFILE-REC` (L62) — record of `TRANSACT-FILE`:

Level	Field	PIC	Notes
01	`FD-TRANFILE-REC`	—	Implicit 350-byte record (5/304+5/26+5/20)
05	`FD-TRANS-DATA`	X(304)	Body (parsed via `TRAN-RECORD` 01-level in WORKING-STORAGE through `READ ... INTO`)
05	`FD-TRAN-PROC-TS`	X(26)	Processing timestamp — referenced at L305 for date-range gate
05	`FD-FILLER`	X(20)	Trailing filler

`FD-CARDXREF-REC` (L68) — record of `XREF-FILE`:

Level	Field	PIC	Notes
01	`FD-CARDXREF-REC`	—	50-byte record
05	`FD-XREF-CARD-NUM`	X(16)	RECORD KEY for the KSDS
05	`FD-XREF-DATA`	X(34)	Body (parsed via `CARD-XREF-RECORD` on `READ INTO`)

`FD-TRANTYPE-REC` (L73) — record of `TRANTYPE-FILE`:

Level	Field	PIC	Notes
01	`FD-TRANTYPE-REC`	—	60-byte record
05	`FD-TRAN-TYPE`	X(02)	RECORD KEY
05	`FD-TRAN-DATA`	X(58)	Body (parsed via `TRAN-TYPE-RECORD` on `READ INTO`)

`FD-TRAN-CAT-RECORD` (L78) — record of `TRANCATG-FILE`:

Level	Field	PIC	Notes

01	FD-TRAN-CAT-RECORD	—	60-byte record
05	FD-TRAN-CAT-KEY	—	RECORD KEY (concatenated)
10	FD-TRAN-TYPE-CD	X(02)	Component of key
10	FD-TRAN-CAT-CD	9(04)	Component of key
05	FD-TRAN-CAT-DATA	X(54)	Body (parsed via TRAN-CAT-RECORD on READ INTO)

FD-REPTFILE-REC (L85) — record of REPORT-FILE:

Level	Field	PIC	Notes
01	FD-REPTFILE-REC	X(133)	133-byte print line; MOVE'd from various report 01-levels at L427/440/452/etc. before each WRITE at L477

FD-DATEPARM-REC (L88) — record of DATE-PARMS-FILE:

Level	Field	PIC	Notes
01	FD-DATEPARM-REC	X(80)	80-byte record (parsed via WS-DATEPARM-RECORD on READ INTO at L353)

Working-Storage Key Areas

data_division.working_storage.fields lists 120 fields (27 of them at 01-level). Key areas:

Inlined transaction record from copybook CVTRA05Y (L93–L115):

Level	Field	PIC	Notes
01	TRAN-RECORD	—	350-byte WS landing zone for TRANSACT-FILE (READ ... INTO TRAN-RECORD at L381)
05	TRAN-ID	X(16)	Transaction id
05	TRAN-TYPE-CD	X(02)	Transaction type code
05	TRAN-CAT-CD	9(04)	Transaction category code
05	TRAN-SOURCE	X(10)	Source channel
05	TRAN-DESC	X(100)	Description
05	TRAN-AMT	S9(09)V99	Signed amount with 2 decimals — accumulator source at L332/419/429
05	TRAN-MERCHANT-ID	9(09)	Merchant id
05	TRAN-MERCHANT-NAME	X(50)	Merchant name
05	TRAN-MERCHANT-CITY	X(50)	Merchant city
05	TRAN-MERCHANT-ZIP	X(10)	Merchant ZIP
05	TRAN-CARD-NUM	X(16)	Card number — drives card-break detection at L313
05	TRAN-ORIG-TS	X(26)	Origination timestamp
05	TRAN-PROC-TS	X(26)	Processing timestamp — first 10 chars compared against date window at L305–L306

Inlined card-xref record from copybook CVACT03Y (L120–L132) — CARD-XREF-RECORD:

Level	Field	PIC
01	CARD-XREF-RECORD	—
05	XREF-CARD-NUM	X(16)
05	XREF-CUST-ID	9(09)
05	XREF-ACCT-ID	9(11)

Inlined transaction-type record from copybook `CVTRA03Y` (L137–L148) — `TRAN-TYPE-RECORD`:

Level	Field	PIC
01	`TRAN-TYPE-RECORD`	—
05	`TRAN-TYPE`	X(02)
05	`TRAN-TYPE-DESC`	X(50)

Inlined transaction-category record from copybook `CVTRA04Y` (L153–L166) — `TRAN-CAT-RECORD`:

Level	Field	PIC
01	`TRAN-CAT-RECORD`	—
05	`TRAN-CAT-KEY`	—
10	`TRAN-TYPE-CD`	X(02)
10	`TRAN-CAT-CD`	9(04)
05	`TRAN-CAT-TYPE-DESC`	X(50)

Inlined report layouts from copybook `CVTRA07Y` (L171–L245) — see *s13 Screen and Report Layouts* below for the detail.

File status field pairs (L116–L118, L133–L135, L149–L151, L167–L169, L246–L248, L250–L252) — one per file, two bytes each:

`TRANFILE-STATUS` / `CARDXREF-STATUS` / `TRANTYPE-STATUS` / `TRANCATG-STATUS` / `TRANREPT-STATUS` / `DATEPARM-STATUS`, each with `STAT1` + `STAT2` sub-fields.

Date-parameter landing zone `WS-DATEPARM-RECORD` (L254–L257):

Level	Field	PIC	Notes
01	`WS-DATEPARM-RECORD`	—	21-byte WS landing zone for `DATE-PARMS-FILE` (`READ ... INTO` at L353)
05	`WS-START-DATE`	X(10)	Start date — compared at L305
05	FILLER	X(01)	Separator
05	`WS-END-DATE`	X(10)	End date — compared at L306

Report control variables `WS-REPORT-VARS` (L259–L269):

Level	Field	PIC	VALUE	Notes
01	`WS-REPORT-VARS`	—	—	Group
05	`WS-FIRST-TIME`	X	`'Y'`	Latched-once header-trigger flag, flipped to `'N'` at L408
05	`WS-LINE-COUNTER`	9(09) COMP-3	0	Detail-line counter — pagination test at L414
05	`WS-PAGE-SIZE`	9(03) COMP-3	20	Lines per page (constant 20)
05	`WS-BLANK-LINE`	X(133)	SPACES	Blank line moved at L461
05	`WS-PAGE-TOTAL`	S9(09)V99	0	Per-page amount accumulator (L332/419/429–L430)
05	`WS-ACCOUNT-TOTAL`	S9(09)V99	0	Per-account (card) amount accumulator (L333/420/442)
05	`WS-GRAND-TOTAL`	S9(09)V99	0	Job-wide amount accumulator (L429)
05	`WS-CURR-CARD-NUM`	X(16)	SPACES	Card-break detection key (L313)

Application result band (L282–L284):

- `APPL-RESULT` PIC S9(9) COMP — every paragraph sets this to 0 (OK) / 16 (EOF) / 12 (other) via EVALUATE / IF on file status.
- 88-level `APPL-AOK` VALUE 0.
- 88-level `APPL-EOF` VALUE 16.

EOF flag `END-OF-FILE` PIC X(01) VALUE `'N'` (L286) — flipped to `'Y'` when `APPL-EOF` is true (L368, L396).

Abend operands `ABCODE` PIC S9(9) BINARY and `TIMING` PIC S9(9) BINARY (L287–L288) — passed to `CEE3ABD` at L762.

File-status display helpers (L271–L280): `IO-STATUS`, `IO-STAT1`, `IO-STAT2`, `TWO-BYTES-BINARY`, `TWO-BYTES-ALPHA` REDEFINES, `TWO-BYTES-LEFT`, `TWO-BYTES-RIGHT`, `IO-STATUS-04`, `IO-STATUS-0401`, `IO-STATUS-0403` — used by `9910-DISPLAY-IO-STATUS` (L765–L778) to format and emit the active file's status code.

Business Limits

- `WS-PAGE-SIZE` PIC 9(03) COMP-3 VALUE 20 (L263–L264) — fixed pagination at 20 detail lines per page.
- `WS-LINE-COUNTER` PIC 9(09) COMP-3 (L261–L262) — counter capacity 999,999,999 lines.
- `WS-PAGE-TOTAL`, `WS-ACCOUNT-TOTAL`, `WS-GRAND-TOTAL` all S9(09)V99 (L266–L268) — signed, ±999,999,999.99 each.
- `TRAN-AMT` S9(09)V99 (L103) — same precision as the accumulators receiving it.
- `ABCODE` value `999` (L761) — the abend code passed to `CEE3ABD`.

Parameter Record

The JCL invocation has no `PARM=` (`jcl_facts.tier1_core.invocation.parm = null`). The functional parameters are the date range read from `DATE-PARMS-FILE` at L353 into `WS-DATEPARM-RECORD` (L254–L257): `WS-START-DATE` X(10) and `WS-END-DATE` X(10) separated by a one-byte FILLER. The expected on-disk format is therefore a single 21+-byte sequential record holding `YYYY-MM-DD□YYYY-MM-DD`.

s04 — Hierarchical Structure Map

Paragraph Inventory

`procedure_division.paragraphs` lists 26 named paragraphs in the source. The PROCEDURE DIVISION header (L291) is followed by an *unnamed* in-line driver block (L292–L349) that precedes the first named paragraph at L352 *(source-derived — STB extraction gap; not in unified STB — `paragraphs[]` does not enumerate this anonymous driver)*.

Line	Paragraph	Role
L292–L349	*(unnamed driver block)*	Top-of-PROCEDURE-DIVISION: open files, drive main loop, close files, GOBACK
L352–L379	`0550-DATEPARM-READ`	Read `DATE-PARMS-FILE` into `WS-DATEPARM-RECORD`; classify status via EVALUATE; on EOF set `END-OF-FILE='Y'`; on other error abend
L380–L405	`1000-TRANFILE-GET-NEXT`	READ `TRANSACT-FILE INTO TRAN-RECORD`; classify status via EVALUATE; on EOF set `END-OF-FILE='Y'`; on other error abend
		First-time header emission; pagination test (`MOD(WS-LINE-`

L406–L424	`1100-WRITE-TRANSACTION-REPORT`	`COUNTER, WS-PAGE-SIZE) = 0`); add amounts to page/account totals; emit detail
L425–L437	`1110-WRITE-PAGE-TOTALS`	MOVE `WS-PAGE-TOTAL` to `REPT-PAGE-TOTAL`; write page-total line; roll `WS-PAGE-TOTAL` into `WS-GRAND-TOTAL`; reset; write divider
L438–L449	`1120-WRITE-ACCOUNT-TOTALS`	MOVE `WS-ACCOUNT-TOTAL` to `REPT-ACCOUNT-TOTAL`; write account-total line; reset; write divider
L450–L455	`1110-WRITE-GRAND-TOTALS`	MOVE `WS-GRAND-TOTAL` to `REPT-GRAND-TOTAL`; write grand-total line
L456–L474	`1120-WRITE-HEADERS`	Emit `REPORT-NAME-HEADER`, blank line, `TRANSACTION-HEADER-1`, `TRANSACTION-HEADER-2` divider
L475–L492	`1111-WRITE-REPORT-REC`	WRITE `FD-REPTFILE-REC`; on TRANREPT-STATUS≠'00' abend
L493–L507	`1120-WRITE-DETAIL`	INITIALIZE `TRANSACTION-DETAIL-REPORT`; MOVE 8 detail fields; write detail; bump `WS-LINE-COUNTER`
L508–L525	`0000-TRANFILE-OPEN`	OPEN INPUT `TRANSACT-FILE`; abend on non-'00'
L526–L543	`0100-REPTFILE-OPEN`	OPEN OUTPUT `REPORT-FILE`; abend on non-'00'
L544–L561	`0200-CARDXREF-OPEN`	OPEN INPUT `XREF-FILE`; abend on non-'00'
L562–L579	`0300-TRANTYPE-OPEN`	OPEN INPUT `TRANTYPE-FILE`; abend on non-'00'
L580–L597	`0400-TRANCATG-OPEN`	OPEN INPUT `TRANCATG-FILE`; abend on non-'00'
L598–L615	`0500-DATEPARM-OPEN`	OPEN INPUT `DATE-PARMS-FILE`; abend on non-'00'
L616–L625	`1500-A-LOOKUP-XREF`	Random READ on `XREF-FILE` by `FD-XREF-CARD-NUM`; INVALID KEY → abend
L626–L635	`1500-B-LOOKUP-TRANTYPE`	Random READ on `TRANTYPE-FILE` by `FD-TRAN-TYPE`; INVALID KEY → abend
L636–L645	`1500-C-LOOKUP-TRANCATG`	Random READ on `TRANCATG-FILE` by `FD-TRAN-CAT-KEY`; INVALID KEY → abend
L646–L663	`9000-TRANFILE-CLOSE`	CLOSE `TRANSACT-FILE`; abend on non-'00'
L664–L682	`9100-REPTFILE-CLOSE`	CLOSE `REPORT-FILE`; abend on non-'00'
L683–L700	`9200-CARDXREF-CLOSE`	CLOSE `XREF-FILE`; abend on non-'00'
L701–L718	`9300-TRANTYPE-CLOSE`	CLOSE `TRANTYPE-FILE`; abend on non-'00'
L719–L736	`9400-TRANCATG-CLOSE`	CLOSE `TRANCATG-FILE`; abend on non-'00'
L737–L757	`9500-DATEPARM-CLOSE`	CLOSE `DATE-PARMS-FILE`; abend on non-'00'
L758–L764	`9999-ABEND-PROGRAM`	DISPLAY `'ABENDING PROGRAM'`; MOVE 0/999 to TIMING/ABCODE; CALL `'CEE3ABD' USING ABCODE, TIMING`
L765–end	`9910-DISPLAY-IO-STATUS`	Format `IO-STATUS` to a 4-digit display via `IO-STATUS-04` and DISPLAY

Paragraph count from `statements.by_verb`: 26 named paragraphs (the unnamed driver block at L291–L349 is invoked implicitly by program entry, not by PERFORM).

Structure Chart

```
(driver, L291–L349)
├── 0000-TRANFILE-OPEN          (L508)  → 9910-DISPLAY-IO-STATUS · 9999-ABEND-PROGRAM (on err
├── 0100-REPTFILE-OPEN          (L526)  → 9910 · 9999 (on error)
├── 0200-CARDXREF-OPEN          (L544)  → 9910 · 9999 (on error)
├── 0300-TRANTYPE-OPEN          (L562)  → 9910 · 9999 (on error)
├── 0400-TRANCATG-OPEN          (L580)  → 9910 · 9999 (on error)
├── 0500-DATEPARM-OPEN          (L598)  → 9910 · 9999 (on error)
├── 0550-DATEPARM-READ          (L352)  → 9910 · 9999 (on non-EOF error)
```

```
├── PERFORM UNTIL END-OF-FILE='Y'          (L302-L338, inline)
│   ├── 1000-TRANFILE-GET-NEXT  (L380)  → 9910 · 9999 (on non-EOF error)
│   ├── 1120-WRITE-ACCOUNT-TOTALS (L438) [card-break, when WS-FIRST-TIME='N']
│   │   └── 1111-WRITE-REPORT-REC (L475)  → 9910 · 9999 (on write error)
│   ├── 1500-A-LOOKUP-XREF      (L616)  → 9910 · 9999 (on INVALID KEY)
│   ├── 1500-B-LOOKUP-TRANTYPE  (L626)  → 9910 · 9999 (on INVALID KEY)
│   ├── 1500-C-LOOKUP-TRANCATG  (L636)  → 9910 · 9999 (on INVALID KEY)
│   ├── 1100-WRITE-TRANSACTION-REPORT (L406)
│   │   ├── 1120-WRITE-HEADERS  (L456) [first-time, or after page totals]
│   │   │   └── 1111-WRITE-REPORT-REC × 4
│   │   ├── 1110-WRITE-PAGE-TOTALS (L425) [when MOD=0]
│   │   │   └── 1111-WRITE-REPORT-REC × 2
│   │   └── 1120-WRITE-DETAIL   (L493)
│   │       └── 1111-WRITE-REPORT-REC
│   ├── 1110-WRITE-PAGE-TOTALS  (L425) [EOF-path branch, if reached]
│   └── 1110-WRITE-GRAND-TOTALS (L450) [EOF-path branch, if reached]
│       └── 1111-WRITE-REPORT-REC
├── 9000-TRANFILE-CLOSE         (L646)  → 9910 · 9999 (on error)
├── 9100-REPTFILE-CLOSE         (L664)  → 9910 · 9999 (on error)
├── 9200-CARDXREF-CLOSE         (L683)  → 9910 · 9999 (on error)
├── 9300-TRANTYPE-CLOSE         (L701)  → 9910 · 9999 (on error)
├── 9400-TRANCATG-CLOSE         (L719)  → 9910 · 9999 (on error)
├── 9500-DATEPARM-CLOSE         (L737)  → 9910 · 9999 (on error)
└── GOBACK                       (L349)
```

Source: `procedure_division.control_flow.perform_calls` (13 PERFORMs at driver scope; 72 PERFORMs program-wide per `statements.by_verb.PERFORM = 72`).

s05 — Control Flow Logic

Main Logic Walkthrough

Phase 1 — Open all files (L293–L298). The driver issues `PERFORM` to each of the six `*-OPEN` paragraphs in order: `0000-TRANFILE-OPEN`, `0100-REPTFILE-OPEN`, `0200-CARDXREF-OPEN`, `0300-TRANTYPE-OPEN`, `0400-TRANCATG-OPEN`, `0500-DATEPARM-OPEN`. Each follows the same shape: `MOVE 8 TO APPL-RESULT` (pessimistic default), `OPEN`, then a status check (`IF *-STATUS = '00'`) that either clears `APPL-RESULT` to 0 or sets it to 12, and the test `IF APPL-AOK CONTINUE ELSE ... PERFORM 9910-DISPLAY-IO-STATUS · PERFORM 9999-ABEND-PROGRAM`. Any failed OPEN aborts the program.

Phase 2 — Read the date parameter (L300). `PERFORM 0550-DATEPARM-READ` reads the single sequential record from `DATE-PARMS-FILE INTO WS-DATEPARM-RECORD` (L353). The EVALUATE on `DATEPARM-STATUS` (L354–L361) maps `'00'` → 0, `'10'` → 16, OTHER → 12 into `APPL-RESULT`. On AOK the program DISPLAYs "*Reporting from ... to ...*" (L364–L365). On `APPL-EOF` the program sets `END-OF-FILE='Y'` (L368) — i.e., an empty `DATEPARM` file is treated as a clean exit before the main loop. On any other status the program prints `'ERROR READING DATEPARM FILE'`, displays the status, and abends (L370–L373).

Phase 3 — Main processing loop (L302–L338). The loop header is `PERFORM UNTIL END-OF-FILE = 'Y'`. Its body is structured as:

```
IF END-OF-FILE = 'N'
   PERFORM 1000-TRANFILE-GET-NEXT
   IF TRAN-PROC-TS (1:10) >= WS-START-DATE
      AND TRAN-PROC-TS (1:10) <= WS-END-DATE
      CONTINUE
   ELSE
      NEXT SENTENCE
   END-IF
   IF END-OF-FILE = 'N'
```

```
      DISPLAY TRAN-RECORD
      IF WS-CURR-CARD-NUM NOT= TRAN-CARD-NUM
        IF WS-FIRST-TIME = 'N'
          PERFORM 1120-WRITE-ACCOUNT-TOTALS
        END-IF
        MOVE TRAN-CARD-NUM TO WS-CURR-CARD-NUM
        MOVE TRAN-CARD-NUM TO FD-XREF-CARD-NUM
        PERFORM 1500-A-LOOKUP-XREF
      END-IF
      MOVE TRAN-TYPE-CD OF TRAN-RECORD TO FD-TRAN-TYPE
      PERFORM 1500-B-LOOKUP-TRANTYPE
      MOVE TRAN-TYPE-CD OF TRAN-RECORD
        TO FD-TRAN-TYPE-CD OF FD-TRAN-CAT-KEY
      MOVE TRAN-CAT-CD OF TRAN-RECORD
        TO FD-TRAN-CAT-CD OF FD-TRAN-CAT-KEY
      PERFORM 1500-C-LOOKUP-TRANCATG
      PERFORM 1100-WRITE-TRANSACTION-REPORT
   ELSE
      DISPLAY 'TRAN-AMT ' TRAN-AMT
      DISPLAY 'WS-PAGE-TOTAL'  WS-PAGE-TOTAL
      ADD TRAN-AMT TO WS-PAGE-TOTAL
                      WS-ACCOUNT-TOTAL
      PERFORM 1110-WRITE-PAGE-TOTALS
      PERFORM 1110-WRITE-GRAND-TOTALS
   END-IF
END-IF
END-PERFORM.
```

Per-iteration the loop does (when not at EOF):

1. `1000-TRANFILE-GET-NEXT` (L304) reads the next transaction record into `TRAN-RECORD`. The EVALUATE on `TRANFILE-STATUS` (L383–L390) maps `'00'`→0, `'10'`→16, OTHER→12. On EOF it sets `END-OF-FILE='Y'` (L396).
2. The date-range gate (L305–L310) compares the first 10 characters of `TRAN-PROC-TS` against `WS-START-DATE` and `WS-END-DATE`. The TRUE branch is `CONTINUE` (fall through); the FALSE branch is `NEXT SENTENCE`.
3. **NEXT SENTENCE (L309) is `effect_class=exit_loop`** (`procedure_division.control_flow.next_sentence_jumps[0]`, `danger_level=high`, `target_line=338`). It jumps **past the next period in the source**, which is the period that terminates `END-PERFORM.` at L338 — i.e., it exits the entire `PERFORM UNTIL` block. The subsequent `IF END-OF-FILE = 'N'` at L311 and the inner detail-emission / EOF-totals path are *not* reached from this branch. Maintainers refactoring this code must **not** substitute `CONTINUE` for `NEXT SENTENCE` — they are not equivalent here (`CONTINUE` falls through to the next imperative statement in the current scope; `NEXT SENTENCE` exits the loop).
4. If the date-range gate passed (the `CONTINUE` arm), control reaches the inner `IF END-OF-FILE = 'N'` (L311). When the inner gate is true:
   - `DISPLAY TRAN-RECORD` (L312) is a per-record debug trace.
   - Card-break detection: `IF WS-CURR-CARD-NUM NOT= TRAN-CARD-NUM` (L313). When the card changes and `WS-FIRST-TIME='N'`, `PERFORM 1120-WRITE-ACCOUNT-TOTALS` (L315) emits the previous account's total. Then `WS-CURR-CARD-NUM` is updated and `1500-A-LOOKUP-XREF` is performed.
   - Type-code and category-code keys are built and `1500-B-LOOKUP-TRANTYPE` / `1500-C-LOOKUP-TRANCATG` are performed.
   - `1100-WRITE-TRANSACTION-REPORT` emits the detail line.
5. When the inner gate is false (i.e., `END-OF-FILE` flipped to `'Y'` during the same iteration *and the last record's date passed the range check*), the ELSE branch (L329–L336) DISPLAYs the trailing `TRAN-AMT` / `WS-PAGE-TOTAL`, adds `TRAN-AMT` into both running totals, and PERFORMs `1110-WRITE-PAGE-TOTALS` and `1110-WRITE-GRAND-TOTALS`. This is the program's *only* call site for `1110-WRITE-GRAND-`

`TOTALS` (`control_flow.perform_calls` shows no other site for that target).

Phase 4 — Close all files (L340–L345). Six sequential `PERFORM`s close each file in reverse-meaningful order: `9000-TRANFILE-CLOSE`, `9100-REPTFILE-CLOSE`, `9200-CARDXREF-CLOSE`, `9300-TRANTYPE-CLOSE`, `9400-TRANCATG-CLOSE`, `9500-DATEPARM-CLOSE`. Each follows the same `IF *-STATUS = '00' ... ELSE abend` shape as the OPEN paragraphs.

Phase 5 — Terminate (L347–L349). `DISPLAY 'END OF EXECUTION OF PROGRAM CBTRN03C'` then `GOBACK`. The `GOBACK` at L349 is the only `GOBACK` in the program (`statements.by_verb.GOBACK = 1`).

Boundary and edge-case behavior (inline)

- **Empty input.** If `TRANSACT-FILE` returns AT END on the first READ (zero records), `1000-TRANFILE-GET-NEXT` (L380) routes through `EVALUATE TRANFILE-STATUS WHEN '10'` (L386) → `MOVE 16 TO APPL-RESULT`, the `IF APPL-EOF` branch (L395) → `MOVE 'Y' TO END-OF-FILE`, and the paragraph EXITs. Back in the driver at L305, `TRAN-PROC-TS` was not freshly populated by a valid READ — the standard-COBOL behavior on AT END leaves the receiving record in an implementation-defined state. The date-range comparison `TRAN-PROC-TS (1:10) >= WS-START-DATE` at L305 fails (whether `TRAN-PROC-TS` holds SPACES, LOW-VALUES, or stale content, the gate is overwhelmingly likely to be FALSE), so control falls into the ELSE at L308 and `NEXT SENTENCE` at L309 executes. `NEXT SENTENCE` jumps **past** `END-PERFORM.` at L338 — the next period in the source. The subsequent `IF END-OF-FILE = 'N'` at L311 does **not** execute. The ELSE branch at L329 that writes page-total / grand-total does **not** execute. **No detail lines, no totals, and (in particular) no header lines are written** — the headers are emitted from `1100-WRITE-TRANSACTION-REPORT` inside the gated-by-`WS-FIRST-TIME` block (L407–L412), which is never reached on empty input. The program proceeds directly to the close-files sequence at L340 and the terminal `GOBACK`. The output `TRANREPT(+1)` is created and catalogued by JCL (DISP `NEW,CATLG,DELETE` at `jcl/TRANREPT.jcl` L76) but written with zero records.
- **Single record.** When exactly one transaction record sits in `TRANSACT-FILE` and its date passes the L305–L306 gate, the first iteration through the loop body proceeds: card-break detection at L313 fires (because `WS-CURR-CARD-NUM` is SPACES and `TRAN-CARD-NUM` is non-blank) but the inner `IF WS-FIRST-TIME='N'` at L314 is FALSE, so `1120-WRITE-ACCOUNT-TOTALS` is **skipped** on the first record. The three lookups run, then `1100-WRITE-TRANSACTION-REPORT` emits the headers (latched first-time path at L407–L412) and the detail. The second loop iteration calls `1000-TRANFILE-GET-NEXT` which now hits EOF (`END-OF-FILE='Y'`); the date-range gate then evaluates against an undefined `TRAN-PROC-TS`. As in the empty-input case, the gate is overwhelmingly likely to be FALSE, the ELSE branch takes `NEXT SENTENCE` at L309, and the loop exits without writing the page or grand totals — meaning a **single-record run emits header + detail but no totals line**. If, however, the date-range gate happens to evaluate TRUE on the stale single record's timestamp (e.g., the receiving area still holds the previously-valid `TRAN-PROC-TS`), the inner `IF END-OF-FILE = 'N'` at L311 is FALSE (because `1000-TRANFILE-GET-NEXT` just set it to `'Y'`), the ELSE at L329 runs, and page/grand totals *are* written. The branch taken on EOF is therefore data-dependent on whether `READ ... INTO` overwrites `TRAN-RECORD` on AT END — an implementation-defined COBOL behavior.
- **Termination signal.** The loop terminates when `END-OF-FILE = 'Y'`. That flag is set in exactly two places: `0550-DATEPARM-READ` at L368 (on AOK-EOF reading the date parameter, before the main loop starts) and `1000-TRANFILE-GET-NEXT` at L396 (on FILE STATUS `'10'` reading the transaction file). The termination check is the `PERFORM UNTIL END-OF-FILE = 'Y'` clause at L302; the flag is declared at L286 as `01 END-OF-FILE PIC X(01) VALUE 'N'.` and never written elsewhere.
- **Counter / array-index boundaries.** No fixed-size `OCCURS` arrays appear in the I/O path (`data_division.working_storage.fields[]` carries no `occurs` entries). Bound-overflow is

therefore not applicable in the array sense. The pagination counter `WS-LINE-COUNTER` PIC 9(09) COMP-3 has capacity 999,999,999; at `WS-PAGE-SIZE = 20` that allows roughly fifty million pages before counter wrap — practically unreachable for a daily report. The accumulators `WS-PAGE-TOTAL`, `WS-ACCOUNT-TOTAL`, `WS-GRAND-TOTAL` are each S9(09)V99 — a value with absolute magnitude exceeding 999,999,999.99 would lose digits on the `ADD` (no `SIZE ERROR` clause is coded on any `ADD` in this program).

Data Operations

`statements.by_verb` summary (entire program):

Verb	Count
DISPLAY	27
PERFORM	72
IF	38
CONTINUE	15
MOVE	97
ADD	16
GOBACK	1
READ	5
EVALUATE	2
EXIT	24
WRITE	1
INITIALIZE	1
OPEN	6
CLOSE	6
SUBTRACT	2
CALL	1

INITIALIZE — one site (`notable_constructs.data_operations.initialize[0]`): `INITIALIZE TRANSACTION-DETAIL-REPORT` at L494 in `1120-WRITE-DETAIL`. Resets the detail-report 01-level before each MOVE-then-WRITE cycle.

Intrinsic functions — one site (`notable_constructs.intrinsic_functions[0]`): `FUNCTION MOD(WS-LINE-COUNTER, WS-PAGE-SIZE)` at L414 — pagination test.

ADD highlights:

- `ADD TRAN-AMT TO WS-PAGE-TOTAL WS-ACCOUNT-TOTAL` (L332–L333 in driver; L419–L420 in `1100-WRITE-TRANSACTION-REPORT`) — sums per-record amount into two running totals atomically.
- `ADD WS-PAGE-TOTAL TO WS-GRAND-TOTAL` (L429) — page-total roll-up at the end of each page.
- `ADD 1 TO WS-LINE-COUNTER` (six sites: L431, L434, L443, L446, L459, L463, L467, L471, L505) — line counter bump on each emitted line.
- `ADD 8 TO ZERO GIVING APPL-RESULT` (L647, L665) — pessimistic-default form used in `9000-TRANFILE-CLOSE` and `9100-REPTFILE-CLOSE`.

SUBTRACT highlights:

- `SUBTRACT APPL-RESULT FROM APPL-RESULT` (L650, L668) — idiomatic "zero APPL-RESULT" in the close paragraphs (matches the `ADD 8 TO ZERO GIVING APPL-RESULT` pessimistic-default pattern).

MOVE highlights (representative; full population is 97 sites — see `data_lineage.edges[]` for the structured ledger):

- `MOVE TRAN-CARD-NUM TO WS-CURR-CARD-NUM` (L317) — card-break key update.
- `MOVE TRAN-CARD-NUM TO FD-XREF-CARD-NUM` (L318) — xref lookup key.
- `MOVE TRAN-TYPE-CD OF TRAN-RECORD TO FD-TRAN-TYPE` (L321) — type lookup key.
- `MOVE TRAN-TYPE-CD OF TRAN-RECORD TO FD-TRAN-TYPE-CD OF FD-TRAN-CAT-KEY` (L323–L324) — composite category key (high half).
- `MOVE TRAN-CAT-CD OF TRAN-RECORD TO FD-TRAN-CAT-CD OF FD-TRAN-CAT-KEY` (L325–L326) — composite category key (low half).
- `MOVE WS-START-DATE TO REPT-START-DATE` and `MOVE WS-END-DATE TO REPT-END-DATE` (L409–L410) — emitted in the first-time header.

Loop Termination

The sole loop is `PERFORM UNTIL END-OF-FILE = 'Y'` at L302–L338 (`procedure_division.control_flow.loop_constructs[0]`, `kind = PERFORM_UNTIL`). The termination condition is the WS field `END-OF-FILE` flipping to `'Y'` — set by `0550-DATEPARM-READ` on `APPL-EOF` (L368) and by `1000-TRANFILE-GET-NEXT` on `APPL-EOF` (L396). `READ ... INTO` is used without explicit `AT END` clauses; status classification is performed by the trailing EVALUATE on `TRANFILE-STATUS` and `DATEPARM-STATUS` respectively, which translate the COBOL FILE STATUS `'10'` into the 88-level `APPL-EOF`.

Control Breaks

The single control-break field is `WS-CURR-CARD-NUM` (L269, declared `PIC X(16) VALUE SPACES`). The break is detected at L313: `IF WS-CURR-CARD-NUM NOT= TRAN-CARD-NUM`. On break, when `WS-FIRST-TIME = 'N'`, `1120-WRITE-ACCOUNT-TOTALS` (L438) is performed; that paragraph emits the prior account's total via `REPT-ACCOUNT-TOTAL`, resets `WS-ACCOUNT-TOTAL` to 0 (L442), and writes a `TRANSACTION-HEADER-2` divider (L444). After the totals emission, the driver updates `WS-CURR-CARD-NUM` (L317) and primes the xref lookup. There is no second-tier break (no transaction-type break, no date break) — only the per-card break.

NEXT SENTENCE control flow

`procedure_division.control_flow.next_sentence_jumps[]` lists one occurrence:

Line	Effect class	Danger	Target line	Notes
L309	`exit_loop`	high	L338	Jumps past the period that terminates `END-PERFORM` at L338, exiting the entire `PERFORM UNTIL END-OF-FILE = 'Y'` loop

The literal `NEXT SENTENCE` token at L309 sits in the ELSE branch of the date-range gate (L305–L310). When the date-range comparison is FALSE, control jumps past the next period in the source. Inside the `PERFORM UNTIL` body the only intervening period is the one on `END-PERFORM.` at L338, so `NEXT SENTENCE` exits the loop. This is **not** equivalent to `CONTINUE` (which would fall through to the inner `IF END-OF-FILE = 'N'` at L311 and continue the same iteration), and it is **not** equivalent to a no-op (which would resume the next iteration of the loop). Refactoring this branch as `CONTINUE` would alter the program's behavior on out-of-range records: out-of-range records currently *terminate the report*; with `CONTINUE` they would be silently skipped and the loop would advance to the next record.

In normal production this branch is operationally not reached because the JCL SORT step (`jcl/TRANREPT.jcl` L46–L48) pre-filters records to the same `[WS-START-DATE, WS-END-DATE]` window before the program ever sees them (see *s02 Pipeline Step Sequence*). The literal `exit_loop` semantic is still the load-bearing fact for any

maintainer modifying this code path or considering removal of the upstream SORT filter.

Operational Behavior

From `operational_behavior` (STB-grounded):

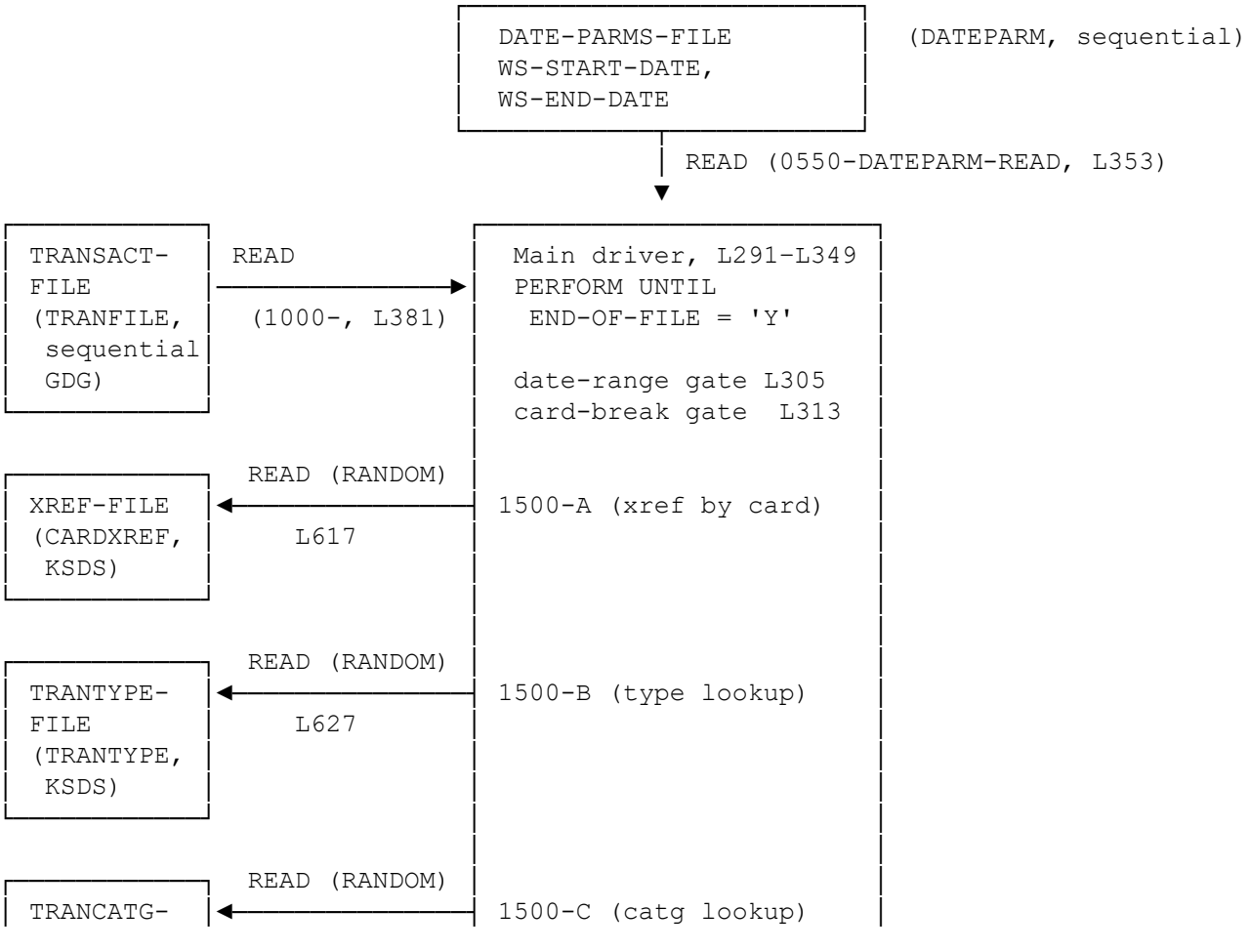
- **Keyed I/O ops** (`keyed_ops[]`, 3 entries): random READ on `TRANCATG-FILE` keyed by `FD-TRAN-CAT-KEY` at L637; random READ on `TRANTYPE-FILE` keyed by `FD-TRAN-TYPE` at L627; random READ on `XREF-FILE` keyed by `FD-XREF-CARD-NUM` at L617.
- **Pagination policy** (`effects[0]`): pagination on counter `WS-LINE-COUNTER` — MOD = 0 against `WS-PAGE-SIZE`; when the test succeeds, the program writes the running page totals (`1110-WRITE-PAGE-TOTALS`) and reprints headers (`1120-WRITE-HEADERS`). The test is at L414.
- **Error policy** (`effects[1..]`, 6 entries — one per file status field): on any I/O error (`io_error_CARDXREF-STATUS`, `io_error_DATEPARM-STATUS`, `io_error_TRANCATG-STATUS`, `io_error_TRANFILE-STATUS`, `io_error_TRANREPT-STATUS`, `io_error_TRANTYPE-STATUS`) the program DISPLAYs an error message, runs `9910-DISPLAY-IO-STATUS`, and PERFORMs `9999-ABEND-PROGRAM`. Every open, close, write, and lookup paragraph in the program uses this exact two-step response.

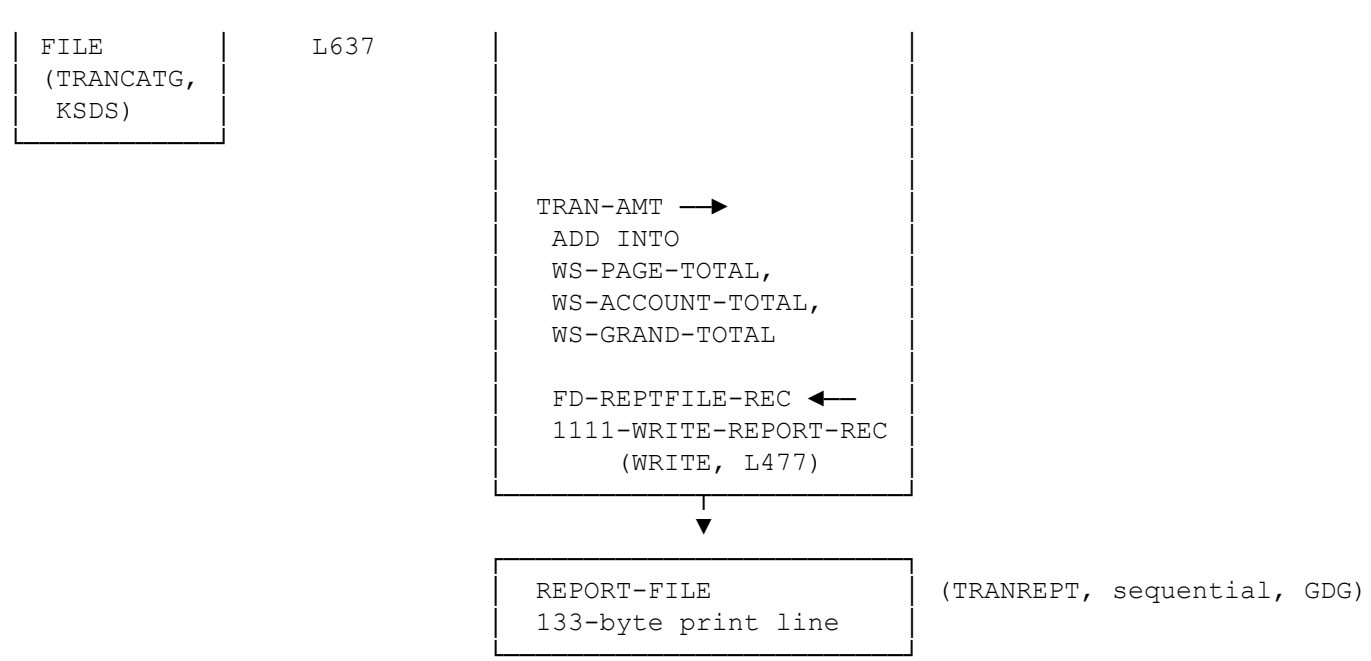
`metrics.cyclomatic_complexity` is 5; `metrics.perform_nesting_depth` is 2 (the driver loop's IF/IF nest plus the lookups it performs).

s06 — Integrated Diagrams

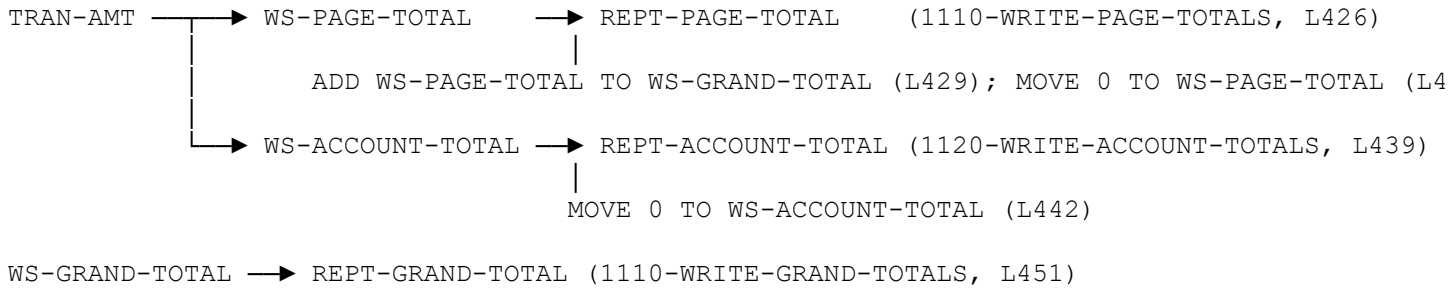
Data Flow Diagram

ASCII flow (per `io.summary` plus driver-block tracing L304–L328):





Accumulator Flow



s07 — External Dependencies

External Calls

`cross_corpus.call_sites[]` and `metrics.paragraph_external_calls[]` show one external CALL:

Target	Type	Paragraph	Line	USING	Category	Notes
CEE3ABD	static	9999-ABEND-PROGRAM	L762	ABCODE, TIMING	LE service routine	IBM Language Environment abend (`CALL 'CEE3ABD' USING ABCODE, TIMING.`) — no `ON EXCEPTION` clause (`notable_constructs.absence_facts.no_on_excepti`)

No application-program CALLs are present (`notable_constructs.absence_facts.no_app_calls`). The L762 call is the program's terminal action on any handled error.

Copybook References

`data_division.copybooks` lists 5 copybook references, all `inlined`:

Copybook	Inlined at line	Purpose
CVTRA05Y	L93	Transaction record layout (`TRAN-RECORD`)
CVACT03Y	L120	Card cross-reference layout (`CARD-XREF-RECORD`)

`CVTRA03Y`	L137	Transaction type layout (`TRAN-TYPE-RECORD`)
`CVTRA04Y`	L153	Transaction category layout (`TRAN-CAT-RECORD`)
`CVTRA07Y`	L171	Report layouts (`REPORT-NAME-HEADER`, `TRANSACTION-DETAIL-REPORT`, `TRANSACTION-HEADER-1`, `TRANSACTION-HEADER-2`, `REPORT-PAGE-TOTALS`, `REPORT-ACCOUNT-TOTALS`, `REPORT-GRAND-TOTALS`)

All five are stamped "*Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19*" in the inlined source (L113, L130, L146, L164, L243).

s08 — Error and Exception Handling

File Status Checks

From `error_handling.patterns[]`:

- **IF file-status check** (`if_file_status`, count 13, lines: 477, 510, 528, 546, 564, 582, 600, 648, 666, 685, 703, 721). The shape is uniform: `IF *-STATUS = '00' MOVE 0 TO APPL-RESULT ELSE MOVE 12 TO APPL-RESULT END-IF` followed by `IF APPL-AOK CONTINUE ELSE DISPLAY '<msg>' · MOVE *-STATUS TO IO-STATUS · PERFORM 9910-DISPLAY-IO-STATUS · PERFORM 9999-ABEND-PROGRAM END-IF`.
- **EVALUATE file-status check** (`evaluate_file_status`, count 2, lines: 354, 383). Used in `0550-DATEPARM-READ` and `1000-TRANFILE-GET-NEXT`. Three-way classification: `WHEN '00' MOVE 0`, `WHEN '10' MOVE 16`, `WHEN OTHER MOVE 12` — the only paragraphs that distinguish EOF (`'10'`) from a hard error.
- **INVALID KEY clause** (`invalid_key`, count 3, lines: 617, 627, 637). On the three random-read lookups (`XREF-FILE`, `TRANTYPE-FILE`, `TRANCATG-FILE`) the `INVALID KEY` path DISPLAYs `'INVALID <thing> : '` followed by the key value, MOVEs `23` to `IO-STATUS`, runs `9910-DISPLAY-IO-STATUS`, and PERFORMs `9999-ABEND-PROGRAM`.
- **ABEND on error** (`abend`, count 1, line 759). Mechanism: `CALL 'CEE3ABD'` with `ABCODE = 999` (L761). The abend paragraph is `9999-ABEND-PROGRAM`.

Abend Codes

`9999-ABEND-PROGRAM` (L758–L764):

```
DISPLAY 'ABENDING PROGRAM'        (L759)
MOVE 0 TO TIMING                  (L760)
MOVE 999 TO ABCODE                (L761)
CALL 'CEE3ABD' USING ABCODE, TIMING.  (L762)
```

The user abend code passed to LE is the literal `999` (L761). `TIMING` is set to `0` (no delay).

I/O Status Display

`9910-DISPLAY-IO-STATUS` (L765–L778) is a shared utility. The body classifies `IO-STATUS`:

- If `IO-STATUS` is **not** numeric (or `IO-STAT1 = '9'`, the VSAM extended-return marker), the paragraph builds a 4-digit display by stuffing `IO-STAT1` into byte 1 of `IO-STATUS-04` and converting `IO-STAT2` from binary-via-`TWO-BYTES-RIGHT` into the 3-digit `IO-STATUS-0403` (L768–L771), then DISPLAYs `'FILE STATUS IS: NNNN'` followed by `IO-STATUS-04` (L772).
- Else the paragraph zero-pads `IO-STATUS-04` (`MOVE '0000' TO IO-STATUS-04`, L774) and overlays

the two-character `IO-STATUS` in positions 3–4 (L775), then DISPLAYs the same line (L776).

The redefinition `TWO-BYTES-ALPHA REDEFINES TWO-BYTES-BINARY` (L275) is what enables the VSAM extended-status numeric conversion at L770–L771.

I/O Event Chronology

From `io.summary.by_file`:

File	OPEN paragraph (line)	READ/WRITE paragraph(s) (line)	CLOSE paragraph (line)
`TRANSACT-FILE`	`0000-TRANFILE-OPEN` (L508)	READ in `1000-TRANFILE-GET-NEXT` (L381)	`9000-TRANFILE-CLOSE` (L646)
`REPORT-FILE`	`0100-REPTFILE-OPEN` (L526)	WRITE on `FD-REPTFILE-REC` in `1111-WRITE-REPORT-REC` (L477)	`9100-REPTFILE-CLOSE` (L664)
`XREF-FILE`	`0200-CARDXREF-OPEN` (L544)	READ in `1500-A-LOOKUP-XREF` (L617)	`9200-CARDXREF-CLOSE` (L683)
`TRANTYPE-FILE`	`0300-TRANTYPE-OPEN` (L562)	READ in `1500-B-LOOKUP-TRANTYPE` (L627)	`9300-TRANTYPE-CLOSE` (L701)
`TRANCATG-FILE`	`0400-TRANCATG-OPEN` (L580)	READ in `1500-C-LOOKUP-TRANCATG` (L637)	`9400-TRANCATG-CLOSE` (L719)
`DATE-PARMS-FILE`	`0500-DATEPARM-OPEN` (L598)	READ in `0550-DATEPARM-READ` (L353)	`9500-DATEPARM-CLOSE` (L737)

File Closure Verification

All six files declared in `ENVIRONMENT DIVISION` have matching `CLOSE` paragraphs (one per file), each invoked by the driver at L340–L345. There are no files opened-but-never-closed, and no `CLOSE` without a preceding `OPEN` (per `io.summary.by_paragraph` — every CLOSE site is paired with a corresponding OPEN site for the same file).

Literal Error Message Text

`notable_constructs.display_statements.statements[]` carries 19 error-class DISPLAY literals (`purpose = "error"`):

Line	Paragraph	Literal text
L370	`0550-DATEPARM-READ`	`'ERROR READING DATEPARM FILE'`
L398	`1000-TRANFILE-GET-NEXT`	`'ERROR READING TRANSACTION FILE'`
L486	`1111-WRITE-REPORT-REC`	`'ERROR WRITING REPTFILE'`
L519	`0000-TRANFILE-OPEN`	`'ERROR OPENING TRANFILE'`
L537	`0100-REPTFILE-OPEN`	`'ERROR OPENING REPTFILE'`
L555	`0200-CARDXREF-OPEN`	`'ERROR OPENING CROSS REF FILE'`
L573	`0300-TRANTYPE-OPEN`	`'ERROR OPENING TRANSACTION TYPE FILE'`
L591	`0400-TRANCATG-OPEN`	`'ERROR OPENING TRANSACTION CATG FILE'`
L609	`0500-DATEPARM-OPEN`	`'ERROR OPENING DATE PARM FILE'`
L619	`1500-A-LOOKUP-XREF`	`'INVALID CARD NUMBER : '` (concatenated with `FD-XREF-CARD-NUM`)
L629	`1500-B-LOOKUP-TRANTYPE`	`'INVALID TRANSACTION TYPE : '` (concatenated with `FD-TRAN-TYPE`)

L639	`1500-C-LOOKUP-TRANCATG`	`'INVALID TRAN CATG KEY : '` (concatenated with `FD-TRAN-CAT-KEY`)
L657	`9000-TRANFILE-CLOSE`	`'ERROR CLOSING POSTED TRANSACTION FILE'`
L675	`9100-REPTFILE-CLOSE`	`'ERROR CLOSING REPORT FILE'`
L694	`9200-CARDXREF-CLOSE`	`'ERROR CLOSING CROSS REF FILE'`
L712	`9300-TRANTYPE-CLOSE`	`'ERROR CLOSING TRANSACTION TYPE FILE'`
L730	`9400-TRANCATG-CLOSE`	`'ERROR CLOSING TRANSACTION CATG FILE'`
L748	`9500-DATEPARM-CLOSE`	`'ERROR CLOSING DATE PARM FILE'`
L759	`9999-ABEND-PROGRAM`	`'ABENDING PROGRAM'`

Additional non-error DISPLAYs include `'START OF EXECUTION OF PROGRAM CBTRN03C'` (L292), `'END OF EXECUTION OF PROGRAM CBTRN03C'` (L347), `'Reporting from ' WS-START-DATE ' to ' WS-END-DATE` (L364–L365), `DISPLAY TRAN-RECORD` (L312, per-record trace), `DISPLAY 'TRAN-AMT ' TRAN-AMT` and `DISPLAY 'WS-PAGE-TOTAL' WS-PAGE-TOTAL` (L330–L331, totals-path trace), and the file-status `'FILE STATUS IS: NNNN'` lines at L772 and L776.

s10 — PLM Metadata

- **Program ID:** `CBTRN03C`
- **Archetype:** `batch_report`
- **Program kind:** `batch`
- **Author:** `AWS` (`identification_division.author`, L24)
- **Source file:** `data/expanded_source_v2/CBTRN03C.expanded.cbl` (781 lines)
- **STB:** `data/stb_unified_v2/CBTRN03C_unified_stb.json`
- **JCL:** `data/input/jcl/TRANREPT.jcl`
- **Copybooks:** `CVACT03Y`, `CVTRA03Y`, `CVTRA04Y`, `CVTRA05Y`, `CVTRA07Y`
- **Header-comment function statement:** "*Print the transaction detail report*" (L5)
- **JCL parser version:** `2.0.1` (`jcl_facts.harvest_metadata.parser_version`)

s11 — Business Rules and Validation

Validation Rules

The program performs no field-level input validation in the conventional sense (no per-field `IF ... NOT NUMERIC` checks on `TRAN-RECORD` fields, no range checks on `TRAN-AMT`, no enum checks on `TRAN-TYPE-CD` or `TRAN-CAT-CD`). The validation it does perform is **referential**:

Rule	Field(s)	Mechanism	Source line	Failure handling
Card-number must resolve to a cross-reference row	`TRAN-CARD-NUM` → `FD-XREF-CARD-NUM`	Random `READ XREF-FILE` with `INVALID KEY` clause	L617–L623	DISPLAY `'INVALID CARD NUMBER : '`; abend via `9999-ABEND-PROGRAM`
Transaction-type code must resolve to a type catalog row	`TRAN-TYPE-CD` → `FD-TRAN-TYPE`	Random `READ TRANTYPE-FILE` with `INVALID KEY` clause	L627–L633	DISPLAY `'INVALID TRANSACTION TYPE : '`; abend
Transaction-type-and-category code must resolve to a category catalog row	`TRAN-TYPE-CD`, `TRAN-CAT-CD` → `FD-TRAN-CAT-KEY`	Random `READ TRANCATG-FILE` with `INVALID KEY` clause	L637–L643	DISPLAY `'INVALID TRAN CATG KEY : '`; abend
Transaction processing date must lie in `[WS-START-DATE, WS-`	`TRAN-PROC-TS`	`IF ... >= WS-START-DATE AND`	L305–L306	ELSE branch: `NEXT SENTENCE` exits the main loop (see *NEXT SENTENCE control flow* in s05).

START-DATE, WS-END-DATE]	(1:10)	... <= WS-END-DATE	Note: in production the input is already pre-filtered by the JCL SORT step.

Note the asymmetry: lookup failures abend; an out-of-range date short-circuits the run. The card-break detection at L313 (`IF WS-CURR-CARD-NUM NOT= TRAN-CARD-NUM`) is not a validation — it is a control-break trigger.

s12 — Key Algorithms

Page-break detection

`1100-WRITE-TRANSACTION-REPORT` (L406–L424) uses an arithmetic-MOD pagination test:

```
IF FUNCTION MOD(WS-LINE-COUNTER, WS-PAGE-SIZE) = 0
   PERFORM 1110-WRITE-PAGE-TOTALS
   PERFORM 1120-WRITE-HEADERS
END-IF
```

with `WS-PAGE-SIZE = 20` (L263–L264). After every 20 detail lines emitted, the program writes the running page-total and reprints the headers. Note that `WS-LINE-COUNTER` is incremented by 1 in every write paragraph (`1110-WRITE-PAGE-TOTALS` at L431/L434, `1120-WRITE-ACCOUNT-TOTALS` at L443/L446, `1120-WRITE-HEADERS` at L459/L463/L467/L471, `1120-WRITE-DETAIL` at L505), so the MOD test counts *all* emitted lines, not just detail lines — header reprints and totals lines contribute to the running counter and thus to the MOD denominator.

Accumulator hierarchy

Three running totals are maintained:

1. `WS-PAGE-TOTAL` — sum of `TRAN-AMT` over the current page; emitted by `1110-WRITE-PAGE-TOTALS` (L426); reset to 0 at L430 immediately after emission; rolled into `WS-GRAND-TOTAL` at L429 *before* reset.
2. `WS-ACCOUNT-TOTAL` — sum of `TRAN-AMT` over the current card-number (account); emitted by `1120-WRITE-ACCOUNT-TOTALS` (L439); reset to 0 at L442; **not** rolled into a higher level (the grand total comes from page totals).
3. `WS-GRAND-TOTAL` — sum of all `WS-PAGE-TOTAL` values rolled in at L429; emitted by `1110-WRITE-GRAND-TOTALS` (L451) on the EOF path.

The two `ADD TRAN-AMT TO WS-PAGE-TOTAL WS-ACCOUNT-TOTAL` sites (L332–L333 in the driver EOF-path, L419–L420 in `1100-WRITE-TRANSACTION-REPORT`) sum into both totals atomically per record.

Cyclomatic complexity

`metrics.cyclomatic_complexity = 5` — modest. The dominant control-flow knots are the main loop's nested IF/IF (date-range gate + EOF gate + card-break gate, all at L302–L336) and the file-status branches in each I/O paragraph.

s13 — Screen and Report Layouts

Report Lines

The report is composed of 7 record layouts from copybook `CVTRA07Y` (L171–L245). All are 133-byte print lines with ASA carriage control as imposed by the JCL DCB on `TRANREPT` (`DCB=(LRECL=133,RECFM=FB,BLKSIZE=0)` and `carriage_control: ASA` per `jcl_facts.tier1_core.dd_mappings.TRANREPT`).

`REPORT-NAME-HEADER` (L175–L184) — report banner line (page header):

Field	PIC	VALUE	Width
`REPT-SHORT-NAME`	X(38)	`'DALYREPT'`	38
`REPT-LONG-NAME`	X(41)	`'Daily Transaction Report'`	41
`REPT-DATE-HEADER`	X(12)	`'Date Range: '`	12
`REPT-START-DATE`	X(10)	SPACES (filled at L409 by MOVE `WS-START-DATE`)	10
FILLER	X(04)	`' to '`	4
`REPT-END-DATE`	X(10)	SPACES (filled at L410 by MOVE `WS-END-DATE`)	10

Total: 125 bytes; remaining 8 bytes blank.

`TRANSACTION-DETAIL-REPORT` (L186–L202) — the detail row:

Field	PIC	Source
`TRAN-REPORT-TRANS-ID`	X(16)	MOVE'd from `TRAN-ID` (L495)
FILLER	X(01)	space separator
`TRAN-REPORT-ACCOUNT-ID`	X(11)	MOVE'd from `XREF-ACCT-ID` (L496)
FILLER	X(01)	space separator
`TRAN-REPORT-TYPE-CD`	X(02)	MOVE'd from `TRAN-TYPE-CD OF TRAN-RECORD` (L497)
FILLER	X(01)	`'-'` separator
`TRAN-REPORT-TYPE-DESC`	X(15)	MOVE'd from `TRAN-TYPE-DESC` (L498)
FILLER	X(01)	space
`TRAN-REPORT-CAT-CD`	9(04)	MOVE'd from `TRAN-CAT-CD OF TRAN-RECORD` (L499)
FILLER	X(01)	`'-'` separator
`TRAN-REPORT-CAT-DESC`	X(29)	MOVE'd from `TRAN-CAT-TYPE-DESC` (L500)
FILLER	X(01)	space
`TRAN-REPORT-SOURCE`	X(10)	MOVE'd from `TRAN-SOURCE` (L501)
FILLER	X(04)	spaces
`TRAN-REPORT-AMT`	-ZZZ,ZZZ,ZZZ.ZZ	MOVE'd from `TRAN-AMT` (L502)
FILLER	X(02)	spaces

Detail width: 16+1+11+1+2+1+15+1+4+1+29+1+10+4+14+2 = 113 bytes; remaining 20 bytes blank.

`TRANSACTION-HEADER-1` (L204–L217) — column-heading row:

Position	Literal	Width
1	`'Transaction ID'`	X(17)
18	`'Account ID'`	X(12)
30	`'Transaction Type'`	X(19)
49	`'Tran Category'`	X(35)
	`'Tran Source'`	

	Tran Source	
84		X(14)
98	SPACE	X(01)
99	' Amount'	X(16)

TRANSACTION-HEADER-2 (L219) — `PIC X(133) VALUE ALL '-'` — full-width dashed divider.

REPORT-PAGE-TOTALS (L221–L225):

Field	PIC	VALUE
FILLER	X(11)	'Page Total'
FILLER	X(86)	ALL '·'
REPT-PAGE-TOTAL	+ZZZ,ZZZ,ZZZ.ZZ	(filled from WS-PAGE-TOTAL at L426)

REPORT-ACCOUNT-TOTALS (L227–L231):

Field	PIC	VALUE
FILLER	X(13)	'Account Total'
FILLER	X(84)	ALL '·'
REPT-ACCOUNT-TOTAL	+ZZZ,ZZZ,ZZZ.ZZ	(filled from WS-ACCOUNT-TOTAL at L439)

REPORT-GRAND-TOTALS (L233–L237):

Field	PIC	VALUE
FILLER	X(11)	'Grand Total'
FILLER	X(86)	ALL '·'
REPT-GRAND-TOTAL	+ZZZ,ZZZ,ZZZ.ZZ	(filled from WS-GRAND-TOTAL at L451)

Lines Per Page

`WS-PAGE-SIZE` PIC 9(03) COMP-3 VALUE 20 (L263–L264). Pagination test at L414: `IF FUNCTION MOD(WS-LINE-COUNTER, WS-PAGE-SIZE) = 0`. Note: because every line-write paragraph bumps `WS-LINE-COUNTER`, the test counts header reprints and totals lines as well as detail lines.

Page Counter

The program does **not** maintain an explicit page-number counter — `[[ABSENT:WS-PAGE-NUMBER|no page-number field declared; pagination is tracked only via the WS-LINE-COUNTER MOD WS-PAGE-SIZE test at L414]]`. Page numbers are not printed on the report banner.

Line Counter

`WS-LINE-COUNTER` PIC 9(09) COMP-3 VALUE 0 (L261–L262). Bumped by 1 in every WRITE-emitting paragraph (`1110-WRITE-PAGE-TOTALS` L431/L434, `1120-WRITE-ACCOUNT-TOTALS` L443/L446, `1120-WRITE-HEADERS` L459/L463/L467/L471, `1120-WRITE-DETAIL` L505). The counter is never reset within the program — it grows monotonically from 0 across the run and is consumed only by the MOD pagination test at L414.

Accumulator Fields

Field	PIC	Resets at	Emitted at

WS-PAGE-TOTAL	S9(09)V99 VALUE 0	L430 (MOVE 0 TO WS-PAGE-TOTAL, after roll into grand)	L426
WS-ACCOUNT-TOTAL	S9(09)V99 VALUE 0	L442 (MOVE 0 TO WS-ACCOUNT-TOTAL)	L439
WS-GRAND-TOTAL	S9(09)V99 VALUE 0	(never reset)	L451

Report Date Field

The report banner carries REPT-START-DATE X(10) and REPT-END-DATE X(10) (L182, L184). Both are loaded from WS-START-DATE / WS-END-DATE at L409–L410 inside the first-time gate of 1100-WRITE-TRANSACTION-REPORT. There is **no run-date / current-date stamp** on the report banner — [[ABSENT:WS-RUN-DATE|no run-date / system-date field is declared; the only intrinsic referenced by the program is MOD (notable_constructs.intrinsic_functions has a single entry)]]. The COBOL date-stamp intrinsic is unused by this program.

Carriage Control

The JCL DCB on TRANREPT specifies carriage_control: ASA (jcl_facts.tier1_core.dd_mappings.TRANREPT.dcb.carriage_control). Within the COBOL source, no WRITE ... ADVANCING clauses are used — 1111-WRITE-REPORT-REC (L475–L491) issues a bare WRITE FD-REPTFILE-REC (L477). The first byte of each 133-byte record is therefore the data, not a carriage-control character, and the dataset's interpretation of column 1 as ASA is a layout-time assumption rather than a program-time assertion.

s14 — Data Store Interaction

Not applicable — notable_constructs.absence_facts confirms no_sql (no embedded SQL) and no_ims_dli (no IMS DL/I). The program's data store is six MVS files (one sequential GDG input, three VSAM KSDS lookups, one sequential parameter file, one sequential GDG output). All data-store interaction is documented in *s03 Data Dictionary* and *s08 Error and Exception Handling*.

s15 — State Machine Documentation

Not applicable — batch program; no commarea, no pseudo-conversational state.

s16 — COMMAREA Structure

Not applicable — batch program; no COMMAREA.

s17 — PF Key Dispatch

Not applicable — batch program; no terminal interaction.

s18 — Copybook Inventory

Copybook List

Copybook	Inline location	Description	Cross-corpus consumer count
CVTRA05Y	L93 (DATA DIVISION, WORKING-	Transaction record layout (350	9 (incl. CBTRN03C)

CVTRA05Y	STORAGE)	bytes)	9 (incl. CBTRN03C)
CVACT03Y	L120 (WORKING-STORAGE)	Card cross-reference layout (50 bytes)	12
CVTRA03Y	L137 (WORKING-STORAGE)	Transaction type layout (60 bytes)	1 (CBTRN03C only)
CVTRA04Y	L153 (WORKING-STORAGE)	Transaction category layout (60 bytes)	1 (CBTRN03C only)
CVTRA07Y	L171 (WORKING-STORAGE)	Report layouts (banner, detail, headers, totals)	1 (CBTRN03C only)

Cross-program copybook sharing

`cross_corpus.copybook_consumers[]`:

- `CVTRA05Y` (TRAN-RECORD, 350-byte transaction record) is shared with 8 other programs: `CBACT04C`, `CBTRN01C`, `CBTRN02C`, `COBIL00C`, `CORPT00C`, `COTRN00C`, `COTRN01C`, `COTRN02C` (plus this program, total consumer_count 9).
- `CVACT03Y` (CARD-XREF-RECORD, 50-byte card cross-reference) is shared with 11 other programs: `CBACT03C`, `CBACT04C`, `CBSTM03A`, `CBTRN01C`, `CBTRN02C`, `COACTUPC`, `COACTVWC`, `COBIL00C`, `COPAUA0C`, `COPAUS0C`, `COTRN02C` (consumer_count 12).
- `CVTRA03Y`, `CVTRA04Y`, `CVTRA07Y` are **CBTRN03C-only** in the corpus (consumer_count 1 each) — modifications to these three copybooks affect only this program.

Implication for change impact: a layout change to `CVTRA05Y` (transaction record) is the highest-blast-radius modification — it touches 9 programs in the corpus. `CVACT03Y` (card xref) is even broader at 12 programs. The three CBTRN03C-only copybooks (`CVTRA03Y`/`CVTRA04Y`/`CVTRA07Y`) can be changed without coordinating with other consumers in the corpus.

s19 — Cross-Program Data Flow

Job Chain Position

`jcl_facts.tier2_enhanced.step_sequence` shows this program is order 3 of 3 in the `TRANREPT` job:

1. `STEP05R.PRC001` — IDCAMS backup copy (`AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS` → `AWS.M2.CARDDEMO.TRANSACT.BKUP(+1)`).
2. `STEP05R` — SORT (`TRANSACT.BKUP(+1)` → `TRANSACT.DALY(+1)`), date-window filtered and card-key sorted (see *s02 Pipeline Step Sequence*).
3. `STEP10R` — this program (`PGM=CBTRN03C`).

`pipeline.upstream_steps[0]` names `STEP05R / SORT` with `output_dd: SORTOUT` feeding this step's `TRANFILE` (the same dataset `AWS.M2.CARDDEMO.TRANSACT.DALY(+1)`). `pipeline.downstream_steps[]` is empty — there is no in-job successor.

File Flow

`cross_corpus.dataset_xref[]` (corpus-wide DSN sharing — accessor_count column shows how many programs touch each DSN):

DSN	Access by this program	Other corpus accessors	Notes

AWS.M2.CARDDEMO.TRANSACT.DALY	read (via TRANSACT-FILE/TRANFILE)	(none)	Pre-sorted GDG; this program is the sole reader of the DALY generation
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS	read (via XREF-FILE/CARDXREF)	CBACT03C, CBTRN02C	Card cross-reference; 3 readers in the corpus
AWS.M2.CARDDEMO.TRANTYPE.VSAM.KSDS	read (via TRANTYPE-FILE/TRANTYPE)	(none)	Transaction-type catalog; this program is the sole accessor in the corpus
AWS.M2.CARDDEMO.TRANCATG.VSAM.KSDS	read (via TRANCATG-FILE/TRANCATG)	(none)	Transaction-category catalog; this program is the sole accessor in the corpus
AWS.M2.CARDDEMO.DATEPARM	read (via DATE-PARMS-FILE/ DATEPARM)	(none)	Parameter file; this program is the sole accessor in the corpus
AWS.M2.CARDDEMO.TRANREPT	write (via REPORT-FILE/TRANREPT)	(none)	Output report GDG; this program is the sole writer in the corpus

`cross_corpus.program_transfers.transfers_to[]` is empty (no CICS XCTL/LINK) and `cross_corpus.program_transfers.invoked_by[]` is empty (no corpus program CALLs CBTRN03C — it is invoked directly by JCL `EXEC PGM=`).

s20 — Subprogram Interface

Return Codes

The program has no LINKAGE SECTION (`data_division.has_linkage_section = false`) and no `PROCEDURE DIVISION USING` clause — it is not called as a subprogram. `cross_corpus.return_codes[]` is empty.

The program terminates in one of two ways:

1. **Normal:** `GOBACK` at L349 after the close-files sequence. The implicit RC is the value of the COBOL special register `RETURN-CODE` (untouched by this program — i.e., 0 by default).
2. **Abend:** `CALL 'CEE3ABD' USING ABCODE, TIMING.` at L762 with `ABCODE = 999` and `TIMING = 0`. Per IBM Language Environment semantics, this raises user abend U0999 (the value of `ABCODE`) with no delay; LE terminates the process without returning to the COBOL caller.

The 18 distinct error sites in the program (12 `IF *-STATUS` non-'00' branches in OPEN/CLOSE paragraphs, 3 `INVALID KEY` clauses in the lookup paragraphs, 2 EVALUATE-`WHEN OTHER` branches in DATEPARM-READ and TRANFILE-GET-NEXT, plus the WRITE-error branch in `1111-WRITE-REPORT-REC`) all funnel into the same `9910-DISPLAY-IO-STATUS` + `9999-ABEND-PROGRAM` pair — there is one and only one abend code, U0999, regardless of the originating error site.