ilumeny

Migration Analysis Report

banking complex testdata 7

Analyzed: 2026-06-07 09:38

KEY METRICS

Programs analyzed:	**11**
JCL Batch Jobs:	**2**
Risk HIGH:	**1 modules**
Risk MEDIUM:	**6 modules**
Risk LOW:	**4 modules**
Avg. Score:	**42.8 / 100**

TABLE OF CONTENTS

PROGRAM OVERVIEW

Risk assessment of all modules (sorted by score)

PROGRAM	TYP	CALLs	COPY	LINES	RISK
BUCHSERV	CORE SERVICE	2	3	301	**HIGH (77)**
UEBERWEIG	BATCH ENTRY	3	4	271	**MEDIUM (63)**
BATCHBCH	BATCH ENTRY	3	3	281	**MEDIUM (63)**
ZINSBERCH	BATCH ENTRY	1	3	265	**MEDIUM (59)**
LIMITPRU	SUBROUTINE	0	2	144	**MEDIUM (58)**
KONTOABFR	BATCH ENTRY	2	4	323	**MEDIUM (57)**
BENACHRI	SUBROUTINE	0	2	132	**MEDIUM (42)**
KTOSTRKC	COPYBOOK	0	0	52	**LOW (13)**
KUNDSTRKC	COPYBOOK	0	0	62	**LOW (13)**
BUCHSTRKC	COPYBOOK	0	0	65	**LOW (13)**
ERRHANDKC	COPYBOOK	0	0	57	**LOW (13)**

Risk Score Legend:
HIGH (70-100): Critical - many dependencies, entry-point, high migration risk
MEDIUM (40-69): Moderate risk - medium migration effort
LOW (0-39): Low risk - isolated, testable, recommended starting point

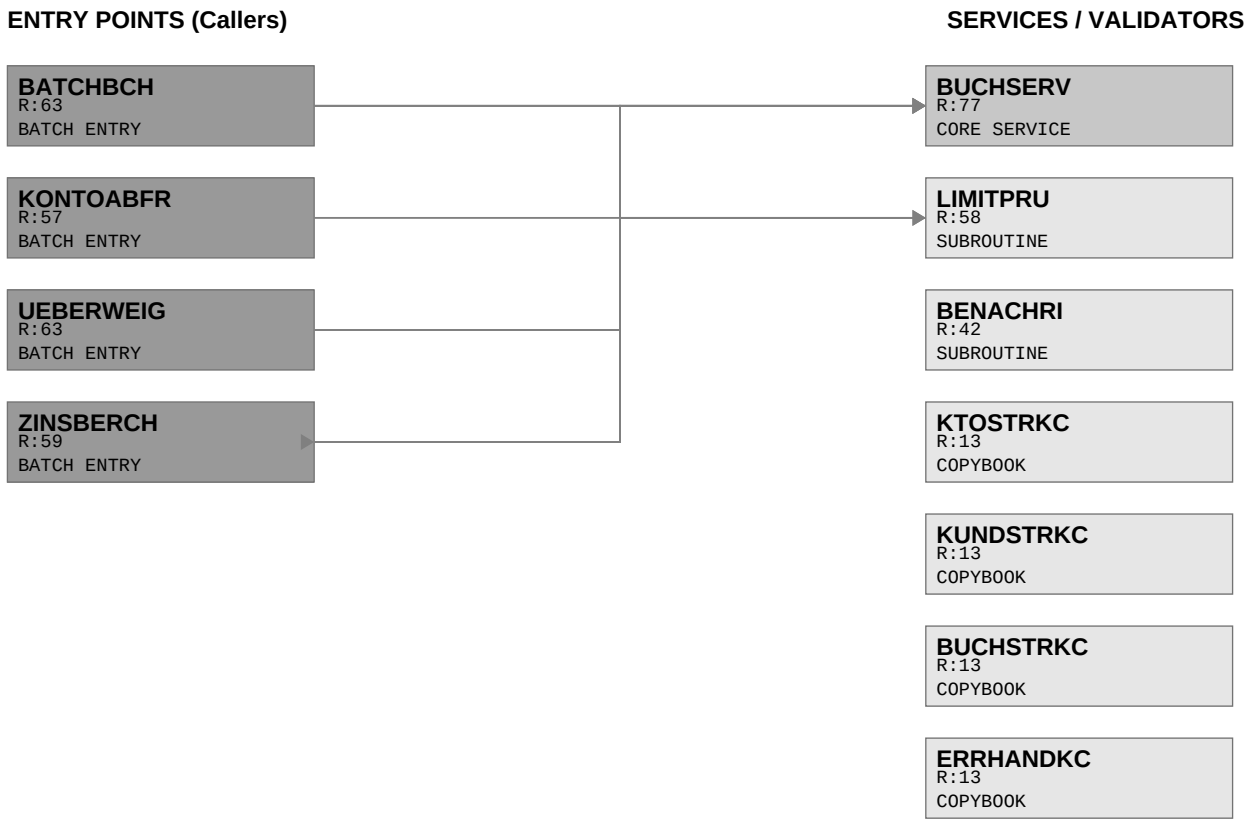
COPYBOOK USAGE

COPYBOOK	USED BY (client programs)	# CLIENTS
KUNDSTRKC.cpy	KONTOABFR, UEBERWEIG, BENACHRI	**3**
KTOSTRKC.cpy	KONTOABFR, BUCHSERV, UEBERWEIG ZINSBERCH, LIMITPRU, BATCHBCH	**6**
ERRHANDKC.cpy	KONTOABFR, BUCHSERV, UEBERWEIG ZINSBERCH, LIMITPRU, BATCHBCH BENACHRI	**7**
BUCHSTRKC.cpy	KONTOABFR, BUCHSERV, UEBERWEIG ZINSBERCH, BATCHBCH	**5**

CALL GRAPH VISUALIZATION

Dependencies and call relationships between modules

Left column = Callers (Entry Points). Right column = called modules.
Connecting lines show CALL relationships. Lighter boxes = lower risk.



NODE TYPE LEGEND:

- BATCH_ENTRY - Einstiegspunkt (direkt durch JCL EXEC aufgerufen)
- CORE_SERVICE - Kern-Buchungslogik (BUCHUNG)
- VALIDATOR - Pruefmodul (KONTOPRUEF, LIMITPRUEF)
- REPORTER - Ausgabemodul (REPORT)

CALL DEPENDENCIES (DETAIL)

CALLER	CALLS	ANZAHL	INCOMING CALLERS OF CALLEE
KONTOABFR	-> KONTOPRU	1x	1 programs call KONTOPRU (incoming)
KONTOABFR	-> BUCHSERV	1x	4 programs call BUCHSERV (incoming)
BUCHSERV	-> LIMITPRU	1x	3 programs call LIMITPRU (incoming)
BUCHSERV	-> BENACHRI	1x	1 programs call BENACHRI (incoming)
UEBERWEIG	-> IBANPRU	1x	1 programs call IBANPRU (incoming)
UEBERWEIG	-> LIMITPRU	1x	3 programs call LIMITPRU (incoming)
UEBERWEIG	-> BUCHSERV	1x	4 programs call BUCHSERV (incoming)
ZINSBERCH	-> BUCHSERV	1x	4 programs call BUCHSERV (incoming)
BATCHBCH	-> LIMITPRU	1x	3 programs call LIMITPRU (incoming)
BATCHBCH	-> ZINSBERCH	1x	1 programs call ZINSBERCH (incoming)
BATCHBCH	-> BUCHSERV	1x	4 programs call BUCHSERV (incoming)

JCL BATCH JOB ANALYSIS

Job chains, execution steps and dataset references

JOB: BATCHJCL "BATCH BUCHUNGEN" CLASS=A

Execution order (Steps):

STEP 1: STEP01
`EXEC PGM=IKJEFT01`
(no COBOL module found in ZIP)

\|/

STEP 2: STEP02
`EXEC PGM=IEBGENER`
(no COBOL module found in ZIP)

\|/

STEP 3: STEP03
`EXEC PGM=SORT`
(no COBOL module found in ZIP)

\|/

STEP 4: STEP04
`EXEC PGM=SORT`
(no COBOL module found in ZIP)

\|/

STEP 5: STEP05
`EXEC PGM=IKJEFT01`
(no COBOL module found in ZIP)

\|/

STEP 6: STEP06
`EXEC PGM=IEFBR14`
(no COBOL module found in ZIP)

\|/

STEP 7: STEP07
`EXEC PGM=IKJEFT01`
(no COBOL module found in ZIP)

\|/

STEP 8: STEP08
`EXEC PGM=IEBGENER`
(no COBOL module found in ZIP)

Referenced Datasets:

[OTHER] `BANKING.PROD.LOADLIB`
[OTHER] `BANKING.PROD.BATCH.PROTOKOLL`
[OTHER] `BANKING.PROD.BATCH.PROTOKOLL.BACKUP`
[OTHER] `BANKING.PROD.DAUERAUFTRAEGE.TAGES`
[OTHER] `BANKING.PROD.LASTSCHRIFTEN.EINGANG`
[OTHER] `BANKING.PROD.BATCH.ERGEBNIS.`
[OTHER] `BANKING.PROD.BATCH.FEHLER.`
[OTHER] `BANKING.PROD.BATCH.ARCHIV`

JOB: ZINSJCL (continued)

Execution order (Steps, continued):

STEP 1: STEP01
EXEC PGM=DSNUTILB
(no COBOL module found in ZIP)

\|/

STEP 2: STEP02
EXEC PGM=IKJEFT01
(no COBOL module found in ZIP)

\|/

STEP 3: STEP03
EXEC PGM=IKJEFT01
(no COBOL module found in ZIP)

\|/

STEP 4: STEP04
EXEC PGM=SORT
(no COBOL module found in ZIP)

\|/

STEP 5: STEP05
EXEC PGM=IKJEFT01
(no COBOL module found in ZIP)

\|/

STEP 6: STEP06
EXEC PGM=IEBGENER
(no COBOL module found in ZIP)

Referenced Datasets:

[OTHER] BANKING.PROD.LOADLIB
[OTHER] BANKING.PROD.BACKUP.KONTEN.
[OTHER] BANKING.PROD.BACKUP.BUCHUNGEN.
[OTHER] BANKING.PROD.ZINS.PROTOKOLL.
[OTHER] BANKING.PROD.ZINS.FEHLER.
[OTHER] BANKING.PROD.STEUER.BERICHT.
[OTHER] BANKING.PROD.ZINS.ARCHIV

DB2 SCHEMA ANALYSIS

11 Tables | 11 Indexes | 6 Sequences | 3 Relations

TABLES (CREATE TABLE)

KUNDEN (28 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
KUNDEN_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
ANREDE	VARCHAR(10)	String	VARCHAR(10)
TITEL	VARCHAR(15)	String	VARCHAR(15)
NACHNAME	VARCHAR(40)	String	NOT NULL.
VORNAME	VARCHAR(25)	String	NOT NULL.
GEBURTSDATUM	DATE	LocalDate	Date/Timestamp. NOT NULL.
GEBURTSORT	VARCHAR(40)	String	VARCHAR(40)
STAATSANGEHOER	CHAR(3)	String	Default: 'DEU',. NOT NULL.
AUSWEIS_NR	VARCHAR(20)	String	Unique ID/Number.
STRASSE	VARCHAR(50)	String	VARCHAR(50)
HAUSNUMMER	VARCHAR(10)	String	VARCHAR(10)
PLZ	CHAR(10)	String	CHAR(10)
ORT	VARCHAR(40)	String	VARCHAR(40)
LAND	CHAR(3)	String	Default: 'DEU',. NOT NULL.
ADRESSE_SEIT	DATE	LocalDate	DATE
TELEFON_PRIVAT	VARCHAR(20)	String	VARCHAR(20)
TELEFON_MOBIL	VARCHAR(20)	String	VARCHAR(20)
EMAIL	VARCHAR(60)	String	VARCHAR(60)
KONTAKT_ERLAUBT	CHAR(1)	String	Default: 'J',. NOT NULL.
SEGMENT_CODE	CHAR(4)	String	Default: 'PRIV',. NOT NULL.
SCHUFA_SCORE	SMALLINT	Integer	SMALLINT
RATING_INTERN	CHAR(2)	String	CHAR(2)
EINKOMMEN_NETTO	DECIMAL(11,2)	BigDecimal	DECIMAL(11,2)
BETREUER_NR	DECIMAL(6,0)	BigDecimal	Unique ID/Number.
FILIALE_NR	DECIMAL(4,0)	BigDecimal	Unique ID/Number.
ERSTELLT_AM	TIMESTAMP	LocalDateTime	Date/Timestamp. Default: CURRENT. NOT NULL.
GEAENDERT_AM	TIMESTAMP	LocalDateTime	Date/Timestamp.
STATUS	CHAR(1)	String	Status flag. Default: 'A',. NOT NULL.

```
JPA: @Entity @Table(name="KUNDEN") class Kunden { @Id Long id; ... }
```

KONTEN (24 columns) FK: KUNDEN

COLUMN	DB2 TYPE	JAVA	MEANING
KONTO_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
BLZ	CHAR(8)	String	Default: '70020270',. NOT NULL.
IBAN	CHAR(22)	String	NOT NULL.
BIC	CHAR(11)	String	Default: 'HYVEDEMMXXX',. NOT NULL.
KUNDEN_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
INHABER_NAME	VARCHAR(40)	String	NOT NULL.
INHABER_VORNAME	VARCHAR(25)	String	NOT NULL.
KONTO_TYP	CHAR(2)	String	Default: 'GK',. NOT NULL.
WAEHRUNG	CHAR(3)	String	Currency code (ISO 4217). Default: 'EUR',. NOT NUL..
STATUS	CHAR(1)	String	Status flag. Default: 'A',. NOT NULL.
SALDO	DECIMAL(15,2)	BigDecimal	Monetary amount - BigDecimal scale 2. Default: 0.0..
VERFUEGBAR	DECIMAL(15,2)	BigDecimal	Default: 0.00,. NOT NULL.
DISPO_LIMIT	DECIMAL(13,2)	BigDecimal	Default: 0.00,. NOT NULL.
GUTHABENGRENZE	DECIMAL(13,2)	BigDecimal	DECIMAL(13,2)
HABEN_ZINSSATZ	DECIMAL(7,4)	BigDecimal	Interest rate (%). Default: 0.0000,. NOT NULL.
SOLL_ZINSSATZ	DECIMAL(7,4)	BigDecimal	Interest rate (%). Default: 12.9500,. NOT NULL.
TAGES_LIMIT	DECIMAL(13,2)	BigDecimal	Default: 5000.00,. NOT NULL.
EINZEL_LIMIT	DECIMAL(13,2)	BigDecimal	Default: 2500.00,. NOT NULL.
AUSLAND_LIMIT	DECIMAL(13,2)	BigDecimal	Default: 1000.00,. NOT NULL.
EROEFFNUNGSDATUM	DATE	LocalDate	Date/Timestamp. NOT NULL.
KUENDIGUNGSDATUM	DATE	LocalDate	Date/Timestamp.
LETZTE_UMSATZ_DATUM	DATE	LocalDate	Date/Timestamp.
ERSTELLT_AM	TIMESTAMP	LocalDateTime	Date/Timestamp. Default: CURRENT. NOT NULL.
GEAENDERT_AM	TIMESTAMP	LocalDateTime	Date/Timestamp.

```
JPA: @Entity @Table(name="KONTEN") class Konten { @Id Long id; ... }
```

DB2 SCHEMA ANALYSIS (continued)

11 Tables | 11 Indexes | 6 Sequences | 3 Relations

TABLES (continued)

BUCHUNGEN (23 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
BUCHUNGS_ID	CHAR(36)	String	Unique ID/Number. NOT NULL.
KONTO_VON	DECIMAL(10,0)	BigDecimal	DECIMAL(10,0)
KONTO_NACH	DECIMAL(10,0)	BigDecimal	DECIMAL(10,0)
IBAN_VON	CHAR(22)	String	CHAR(22)
IBAN_NACH	CHAR(22)	String	CHAR(22)
BETRAG	DECIMAL(15,2)	BigDecimal	Monetary amount - BigDecimal scale 2. NOT NULL.
WAEHRUNG	CHAR(3)	String	Currency code (ISO 4217). Default: 'EUR',. NOT NUL..
BUCHUNGSTYP	CHAR(4)	String	NOT NULL.
KANAL	CHAR(4)	String	NOT NULL.
STATUS	CHAR(2)	String	Status flag. Default: 'PE',. NOT NULL.
BUCHUNGSDATUM	DATE	LocalDate	Date/Timestamp. NOT NULL.
VALUTADATUM	DATE	LocalDate	Date/Timestamp.
VERARBEITUNGSZEIT	TIMESTAMP	LocalDateTime	TIMESTAMP
VERWENDUNGSZWECK	VARCHAR(140)	String	VARCHAR(140)
AUFTRAGGEBER_REF	VARCHAR(35)	String	VARCHAR(35)
END_TO_END_ID	VARCHAR(35)	String	Unique ID/Number.
CLEARING_ID	VARCHAR(20)	String	Unique ID/Number.
INTERBANK_REF	VARCHAR(35)	String	VARCHAR(35)
BATCH_NR	DECIMAL(8,0)	BigDecimal	Unique ID/Number.
SEQUENZ_NR	DECIMAL(6,0)	BigDecimal	Unique ID/Number.
STORNO_DATUM	DATE	LocalDate	Date/Timestamp.
STORNO_REFERENZ	CHAR(36)	String	CHAR(36)
ERSTELLT_AM	TIMESTAMP	LocalDateTime	Date/Timestamp. Default: CURRENT. NOT NULL.

```
JPA: @Entity @Table(name="BUCHUNGEN") class Buchungen { @Id Long id; ... }
```

BUCHUNGS_JOURNAL (8 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
JOURNAL_ID	DECIMAL(12,0)	BigDecimal	Unique ID/Number. NOT NULL.
BUCHUNGS_ID	CHAR(36)	String	Unique ID/Number. NOT NULL.
PROGRAMM	CHAR(8)	String	NOT NULL.
ZEITSTEMPEL	TIMESTAMP	LocalDateTime	Default: CURRENT. NOT NULL.
BENUTZER	VARCHAR(8)	String	VARCHAR(8)
STATUS	CHAR(2)	String	Status flag. NOT NULL.
TERMINAL_ID	CHAR(4)	String	Unique ID/Number.
AKTION	VARCHAR(50)	String	VARCHAR(50)

```
JPA: @Entity @Table(name="BUCHUNGS_JOURNAL") class BuchungsJournal { @Id Long id; ... }
```

KAPITALERTRAGSTEUER (12 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
STEUER_ID	DECIMAL(12,0)	BigDecimal	Unique ID/Number. NOT NULL.
BUCHUNGSDATUM	DATE	LocalDate	Date/Timestamp. NOT NULL.
KUNDEN_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number.
KONTO_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number.
GESAMTBETRAG	DECIMAL(13,2)	BigDecimal	Monetary amount - BigDecimal scale 2. NOT NULL.
KEST_SATZ	DECIMAL(5,2)	BigDecimal	Default: 25.00,. NOT NULL.
KEST_BETRAG	DECIMAL(13,2)	BigDecimal	Monetary amount - BigDecimal scale 2.
SOLI_SATZ	DECIMAL(5,2)	BigDecimal	Default: 5.50,. NOT NULL.
SOLI_BETRAG	DECIMAL(11,2)	BigDecimal	Monetary amount - BigDecimal scale 2.
STEUER_JAHR	SMALLINT	Integer	NOT NULL.
PROGRAMM	CHAR(8)	String	CHAR(8)
ERSTELLT_AM	TIMESTAMP	LocalDateTime	Date/Timestamp. Default: CURRENT. NOT NULL.

```
JPA: @Entity @Table(name="KAPITALERTRAGSTEUER") class Kapitalertragsteuer { @Id Long id; ... }
```

ZINS_STAFFELN (7 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
STAFFEL_NR	SMALLINT	Integer	Unique ID/Number. NOT NULL.
KONTO_TYP	CHAR(2)	String	Default: 'GK',. NOT NULL.
BETRAG_BIS	DECIMAL(13,2)	BigDecimal	Monetary amount - BigDecimal scale 2. NOT NULL.
ZINSSATZ	DECIMAL(7,4)	BigDecimal	Interest rate (%). NOT NULL.
GUELTIG_VON	DATE	LocalDate	NOT NULL.
GUELTIG_BIS	DATE	LocalDate	DATE

COLUMN	DB2 TYPE	JAVA	MEANING
AKTIV	CHAR(1)	String	Default: 'J',. NOT NULL.

```
JPA: @Entity @Table(name="ZINS_STAFFELN") class ZinsStaffeln { @Id Long id; ... }
```

FREISTELLUNGSAUFTRAEGE (6 columns) FK: KUNDEN

COLUMN	DB2 TYPE	JAVA	MEANING
FREIST_ID	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
KUNDEN_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
FREISTELLUNGSBETRAG	DECIMAL(9,2)	BigDecimal	Monetary amount - BigDecimal scale 2. Default: 100..
STEUER_JAHR	SMALLINT	Integer	NOT NULL.
AKTIV	CHAR(1)	String	Default: 'J',. NOT NULL.
BEANTRAGT_AM	DATE	LocalDate	Date/Timestamp.

```
JPA: @Entity @Table(name="FREISTELLUNGSAUFTRAEGE") class Freistellungsauftraege { @Id Long id; ... }
```

RUECKLASTSCHRIFTEN (9 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
RUECK_ID	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
KONTO_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
IBAN_GLAEUBIGER	CHAR(22)	String	CHAR(22)
BETRAG	DECIMAL(11,2)	BigDecimal	Monetary amount - BigDecimal scale 2. NOT NULL.
MANDATS_REF	VARCHAR(35)	String	VARCHAR(35)
RUECK_DATUM	DATE	LocalDate	Date/Timestamp. NOT NULL.
GRUND	VARCHAR(80)	String	VARCHAR(80)
STATUS	VARCHAR(20)	String	Status flag. Default: 'OFFEN',. NOT NULL.
VERARBEITUNGSDATUM	DATE	LocalDate	Date/Timestamp.

```
JPA: @Entity @Table(name="RUECKLASTSCHRIFTEN") class Ruecklastschriften { @Id Long id; ... }
```

NACHRICHTEN_QUEUE (10 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
NACHRICHT_ID	DECIMAL(12,0)	BigDecimal	Unique ID/Number. NOT NULL.
TYP	VARCHAR(5)	String	NOT NULL.
EMPFAENGER	VARCHAR(60)	String	NOT NULL.
BETREFF	VARCHAR(80)	String	VARCHAR(80)
INHALT	VARCHAR(500)	String	NOT NULL.
PRIORITAET	SMALLINT	Integer	Default: 2,. NOT NULL.
STATUS	VARCHAR(10)	String	Status flag. Default: 'PENDING',. NOT NULL.
TIMESTAMP_EINGEST	TIMESTAMP	LocalDateTime	Default: CURRENT. NOT NULL.
TIMESTAMP_GESENDET	TIMESTAMP	LocalDateTime	TIMESTAMP
VERSUCHE	SMALLINT	Integer	Default: 0,. NOT NULL.

```
JPA: @Entity @Table(name="NACHRICHTEN_QUEUE") class NachrichtenQueue { @Id Long id; ... }
```

BATCH_LAUF_PROTOKOLL (9 columns)

COLUMN	DB2 TYPE	JAVA	MEANING
LAUF_ID	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
PROGRAMM_NAME	CHAR(8)	String	NOT NULL.
DATUM	DATE	LocalDate	Date/Timestamp. NOT NULL.
START_TIMESTAMP	TIMESTAMP	LocalDateTime	Default: CURRENT. NOT NULL.
ENDE_TIMESTAMP	TIMESTAMP	LocalDateTime	TIMESTAMP
STATUS	VARCHAR(10)	String	Status flag. Default: 'LAUF',. NOT NULL.
SAETZE_GESAMT	DECIMAL(8,0)	BigDecimal	DECIMAL(8,0)
SAETZE_BEARB	DECIMAL(8,0)	BigDecimal	DECIMAL(8,0)
SAETZE_FEHLER	DECIMAL(8,0)	BigDecimal	DECIMAL(8,0)

```
JPA: @Entity @Table(name="BATCH_LAUF_PROTOKOLL") class BatchLaufProtokoll { @Id Long id; ... }
```

NACHRICHT_PRAEFERENZEN (6 columns) FK: KUNDEN

COLUMN	DB2 TYPE	JAVA	MEANING
KUNDEN_NR	DECIMAL(10,0)	BigDecimal	Unique ID/Number. NOT NULL.
SMS_AKTIV	CHAR(1)	String	Default: 'N',. NOT NULL.
EMAIL_AKTIV	CHAR(1)	String	Default: 'J',. NOT NULL.
PUSH_AKTIV	CHAR(1)	String	Default: 'N',. NOT NULL.
SCHWELLWERT	DECIMAL(9,2)	BigDecimal	Default: 500.00,. NOT NULL.

```
JPA: @Entity @Table(name="NACHRICHT_PRAEFERENZEN") class NachrichtPraeferenzen { @Id Long id; ... }
```

DB2 SCHEMA: RELATIONS, INDEXES & MIGRATION

Foreign Keys, Performance Indexes, Sequences and DB2 ? PostgreSQL Migration Guide

FOREIGN KEYS

FROM TABLE	-> TO TABLE	CARDINALITY
KONTEN	KUNDEN	N:1 (KONTEN -> KUNDEN)
FREISTELLUNGSAUFTRAEGE	KUNDEN	N:1 (FREISTELLUNGSAUFTRAEGE -> KUNDEN)
NACHRICHT_PRAEFERENZEN	KUNDEN	N:1 (NACHRICHT_PRAEFERENZEN -> KUNDEN)

PERFORMANCE INDEXES

INDEX	TABLE	COLUMNS	PURPOSE
IX_KUNDEN_PK	KUNDEN	KUNDEN_NR ASC	Performance: key lookup
IX_KUNDEN_NAME	KUNDEN	NACHNAME ASC, VORNAM	Performance index for NACHNA
IX_KUNDEN_EMAIL	KUNDEN	EMAIL ASC	Performance index for EMAIL
IX_KONTEN_PK	KONTEN	KONTO_NR ASC	Performance: key lookup
IX_KONTEN_IBAN	KONTEN	IBAN ASC	Performance index for IBAN A
IX_KONTEN_KUNDEN	KONTEN	KUNDEN_NR ASC, KONTO	Performance: key lookup
IX_BUCHUNGEN_PK	BUCHUNGEN	BUCHUNGS_ID ASC	Performance index for BUCHUN
IX_BUCHUNGEN_KONTO_VON	BUCHUNGEN	KONTO_VON ASC, BUCHU	Performance: date range filt
IX_BUCHUNGEN_KONTO_NACH	BUCHUNGEN	KONTO_NACH ASC, BUCH	Performance: date range filt
IX_BUCHUNGEN_E2E	BUCHUNGEN	END_TO_END_ID ASC, B	Performance: date range filt
IX_BUCHUNGEN_DATUM	BUCHUNGEN	BUCHUNGSDATUM DESC,	Performance: date range filt

SEQUENCES (AUTO-INCREMENT)

JOURNAL_ID START: 1 INCREMENT: 1

JPA: @GeneratedValue(strategy=GenerationType.IDENTITY) // DB2: GENERATED ALWAYS AS IDENTITY

STEUER_ID START: 1 INCREMENT: 1

JPA: @GeneratedValue(strategy=GenerationType.IDENTITY) // DB2: GENERATED ALWAYS AS IDENTITY

FREIST_ID START: 1 INCREMENT: 1

JPA: @GeneratedValue(strategy=GenerationType.IDENTITY) // DB2: GENERATED ALWAYS AS IDENTITY

RUECK_ID START: 1 INCREMENT: 1

JPA: @GeneratedValue(strategy=GenerationType.IDENTITY) // DB2: GENERATED ALWAYS AS IDENTITY

NACHRICHT_ID START: 1 INCREMENT: 1

JPA: @GeneratedValue(strategy=GenerationType.IDENTITY) // DB2: GENERATED ALWAYS AS IDENTITY

LAUF_ID START: 1 INCREMENT: 1

JPA: @GeneratedValue(strategy=GenerationType.IDENTITY) // DB2: GENERATED ALWAYS AS IDENTITY

DB2 -> POSTGRESQL SYNTAX MAPPING

DB2 SYNTAX	POSTGRESQL/JPA	JAVA TYPE	EFFORT
DECIMAL(n,m)	NUMERIC(n,m)	BigDecimal	compatible
VARCHAR(n)	VARCHAR(n)	String	identical
DATE	DATE	LocalDate	identical
TIMESTAMP	TIMESTAMP	LocalDateTime	identical
CREATE SEQUENCE	@SequenceGenerator	Long	rewrite
EXEC SQL	Spring Data JPA	Repository	replace
DECLARE CURSOR	findAll()/Stream	Repository	replace
CURRENT DATE	CURRENT_DATE	LocalDate.now()	adapt
FOR UPDATE OF	FOR UPDATE	-	remove column
SQLCODE	DataAccessException	-	replace

BUSINESS RULES: BUCHSERV

Type: CORE_SERVICE | Risk: HIGH (77) | 301 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
BUCHSERV	CORE_SERVICE	HIGH (77)	301	CICS BMS Map (3270 terminal...

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (10)

- WS-KONTO-NR-IN
- LK-KONTO-NR
- KTO-IBAN
- KTO-INH-NAME
- KTO-INH-VORNAME
- ERR-USER-ID
- BCH-KONTO-VON
- BCH-KONTO-NACH
- ... and 2 more

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

1. Overview

Program Name	BUCHSERV
Type	CORE_SERVICE
Purpose	Central booking service for retail banking: handles single and batch bookings, updates account balances, prevents duplicates, writes booking journal and provides account turnover queries.
User Interface	CICS BMS Map (3270 terminal) -> REST API
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Perform booking or turnover query
Actor / User	Calling program (e.g., KontoAbfr, UeberweiG, BatchBch)
Description	Caller supplies account and parameters; BUCHSERV performs single/batch booking or turnover query, updates balances, writes journal entries, and returns result/rows.
Goal	Persist a booking transaction atomically and return success or error codes; or return up to 50 turnover rows.
Trigger	Procedure Division invoked with linkage fields including LK-KONTO-NR and LK-BUCHUNG and a mode in WS-MODUS.

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
LK-KONTO-NR	WS-KONTO-NR-IN	numeric(10)	Used to read and update KONTEN and filter BUCHUNGEN; must exist in KONTEN (read checked via SQL).
LK-DATUM-VON / LK-DATUM-BIS	WS-DATUM-VON-IN / WS-DATUM-BIS-IN	string(8) date	Used for BETWEEN filter in turnover query; assumed valid dates by caller.
LK-BUCHUNG	BCH-* fields (via copy BUCHSTRKC)	composite booking record	BCH-END-TO-END-ID used for duplicate check; BCH-BETRAG and BCH-TYP used for limit and saldo computation.

WS-MODUS (derived from caller)	WS-MODUS and 88-levels (MODUS-EINZEL/SAMMEL/STORNO/ABFR) string(4)	Must be one of defined 88 values or else program returns BCH-RC = 0003.

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Dispatch by mode: EVALUATE TRUE WHEN MODUS-EINZEL / WHEN MODUS-SAMMEL / WHEN MODUS-STORNO / WHEN MODUS-ABFRAGE WHEN OTHER -> When a mode is set, perform the corresponding paragraph (single booking, batch booking, storno, or turnover query). If none match, set BCH-RC = 0003 and BCH-FEHLERMSG = 'UNGUELTIGER MODUS'. (control-flow), 98% confidence
2	Initialization of program context: 1000-INITIALISIERUNG (MOVE and EXEC SQL SELECT CURRENT TIMESTAMP INTO :WS-TIMESTAMP) -> Set ERR-PROGRAMM = WS-PROGRAMM-NAME, BCH-RC = zeros, BCH-FEHLERMSG = spaces, copy linkage inputs to working storage, and populate WS-TIMESTAMP with CURRENT TIMESTAMP from DB. (initialization, Line 1000), 98% confidence
3	Duplicate booking detection query: 2010-DUPLIKAT-PRUEFEN: SELECT COUNT(*) INTO :WS-BUCH-ZAEHLER FROM BUCHUNGEN WHERE END_TO_END_ID = :BCH-END-TO-END-ID AND BUCHUNGSDATUM >= CURRENT DATE - 1 DAY -> Set WS-DUPLIKAT-FLAG = 'J' if WS-BUCH-ZAEHLER > 0; otherwise leave as 'N'. (search-filter, Line 2010), 97% confidence
4	Reject duplicate booking: IF DUPLIKAT-GEFUNDEN (WS-DUPLIKAT-FLAG = 'J') after duplicate check -> Set BCH-RC = 0004, set BCH-FEHLERMSG = 'DOPPELTE BUCHUNG ABGELEHNT' and EXIT PARAGRAPH (abort single booking flow). (validation, Line 2000), 98% confidence
5	Limit check invocation and failure mapping: 2020-LIMIT-PRUEFEN: CALL 'LIMITPRU' USING WS-KONTO-NR-IN, BCH-BETRAG, BCH-TYP, KONTO-RUECKGABE; IF NOT KTO-OK -> If external limit check returns NOT KTO-OK, set BCH-RC = 0001 and BCH-FEHLERMSG = KTO-FEHLERMSG (reject booking). (validation, Line 2020), 90% confidence
6	Read account saldo for update: 2030-KONTO-SALDO-LESEN: EXEC SQL SELECT SALDO, VERFUEGBAR INTO :WS-SALDO-ALT, :WS-VERFUEG-ALT FROM KONTEN WHERE KONTO_NR = :WS-KONTO-NR-IN FOR UPDATE -> Load current SALDO and VERFUEGBAR into working storage; on SQLCODE != 0 move SQLCODE to ERR-SQLCODE and PERFORM 9100-SQL-FEHLER. (file-io, Line 2030), 97% confidence
7	Compute new saldo depending on credit/debit: 2040-NEUEN-SALDO-BERECHNEN: IF BCH-GUTSCHRIFT THEN ... ELSE ... END-IF -> If BCH-GUTSCHRIFT is true add BCH-BETRAG to WS-SALDO-ALT and WS-VERFUEG-ALT to compute WS-SALDO-NEU and WS-VERFUEG-NEU; otherwise subtract. Then MOVE WS-SALDO-NEU TO BCH-BETRAG. (calculation, Line 2040), 96% confidence
8	Insert booking record and set return codes: 2050-BUCHUNG-SCHREIBEN: EXEC SQL INSERT INTO BUCHUNGEN ... END-EXEC followed by EVALUATE SQLCODE WHEN 0 WHEN -803 WHEN OTHER -> On SQLCODE 0 set BCH-RC = 0000; on -803 set BCH-RC = 0004 (duplicate key); on other SQLCODE move it to ERR-SQLCODE and PERFORM 9100-SQL-FEHLER. (file-io, Line 2050), 98% confidence
9	Update account balances and commit/rollback: 2060-SALDO-AKTUALISIEREN: EXEC SQL UPDATE KONTEN SET SALDO = :WS-SALDO-NEU, VERFUEGBAR = :WS-VERFUEG-NEU, LETZTE_UMSATZ_DATUM = CURRENT DATE WHERE KONTO_NR = :WS-KONTO-NR-IN END-EXEC IF SQLCODE NOT EQUAL 0 -> If update SQLCODE != 0 then EXEC SQL ROLLBACK, move SQLCODE to ERR-SQLCODE and PERFORM 9100-SQL-FEHLER; else EXEC SQL COMMIT. (file-io, Line 2060), 99% confidence
10	Write booking journal entry: 2070-JOURNAL-SCHREIBEN (after BCH-OK) -> Populate JRN fields from booking and program context and INSERT a row into BUCHUNGS_JOURNAL with STATUS 'GB' (no SQL error handling shown). (file-io, Line 2070), 90% confidence

BUSINESS RULES: BUCHSERV (continued)

Type: CORE_SERVICE | Risk: HIGH (77)

11	Notify downstream system on successful booking: IF BCH-OK after successful booking and saldo update -> CALL 'BENACHRI' USING BCH-KONTO-VON, BCH-BETRAG, BCH-TYP to notify other component. (control-flow, Line 2000), 80% confidence
12	Batch booking loop with stop-on-error: 2100-SAMMEL-BUCHUNG: PERFORM VARYING WS-BUCH-ZAEHLER FROM 1 BY 1 UNTIL WS-BUCH-ZAEHLER > 50 ... IF NOT BCH-OK EXEC SQL ROLLBACK EXIT PERFORM -> Perform up to 50 single bookings; if any booking fails (BCH-OK false) issue a ROLLBACK and stop processing further bookings. (control-flow, Line 2100), 97% confidence
13	Storno: mark booking as reversed and then reprocess: 2200-STORNO-BUCHUNG: EXEC SQL UPDATE BUCHUNGEN SET STATUS = 'ST', STORNO_DATUM = CURRENT DATE WHERE BUCHUNGS_ID = :BCH-BUCHUNGS-ID IF SQLCODE EQUAL 0 -> If update succeeds (SQLCODE = 0) perform 2000-EINZEL-BUCHUNG to create the reversal booking; otherwise on error move SQLCODE to ERR-SQLCODE and perform 9100-SQL-FEHLER. (control-flow, Line 2200), 95% confidence
14	Turnover query cursor selection: 3000-UMSATZ-ABFRAGE: DECLARE UMSATZ_CURSOR FOR SELECT ... WHERE (KONTO_VON = :WS-KONTO-NR-IN OR KONTO_NACH = :WS-KONTO-NR-IN) AND BUCHUNGSDATUM BETWEEN :WS-DATUM-VON-IN AND :WS-DATUM-BIS-IN ORDER BY BUCHUNGSDATUM DESC -> Prepare cursor to fetch bookings for given account and date range sorted descending by booking date. (search-filter, Line 3000), 98% confidence
15	Fetch up to 50 turnover rows or until no more rows: PERFORM UNTIL SQLCODE EQUAL 100 OR WS-UMSATZ-ANZ EQUAL 50 with FETCH into UL- arrays -> Increment WS-UMSATZ-ANZ and FETCH rows into UL arrays until cursor returns SQLCODE 100 (no more rows) or WS-UMSATZ-ANZ reaches 50; then CLOSE cursor and move WS-UMSATZ-ANZ to LK-ANZAHL. (control-flow, Line 3000), 98% confidence
16	Set output linkage for booking result: 9000-PROGRAMM-ENDE -> MOVE BUCHUNGS-RUECKGABE TO LK-RUECKGABE to return final booking/result data to caller. (data-transformation, Line 9000), 95% confidence
17	Central SQL error handling sets generic RC and rolls back: 9100-SQL-FEHLER or invoked when SQL errors detected (MOVE SQLCODE TO ERR-SQLCODE then PERFORM 9100-SQL-FEHLER) -> Set BCH-RC = 0099, BCH-FEHLERMSG = 'SQL FEHLER IN BUCHSERV' and EXEC SQL ROLLBACK. (error-handling, Line 9100), 99% confidence
18	Map SQL insert duplicate key to duplicate booking return code: EVALUATE SQLCODE WHEN -803 in 2050-BUCHUNG-SCHREIBEN -> Set BCH-RC = 0004 to indicate duplicate booking (duplicate key constraint violation). (validation, Line 2050), 98% confidence
19	Set default duplicate flag before checking: 2010-DUPLIKAT-PRUEFEN: MOVE 'N' TO WS-DUPLIKAT-FLAG before SQL SELECT COUNT -> Initialize WS-DUPLIKAT-FLAG to 'N' so it is only set to 'J' if a recent booking with same END_TO_END_ID is found. (initialization, Line 2010), 95% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- Linkage inputs moved into WS (account, dates, page, booking payload) - Current timestamp read from DB into WS-TIMESTAMP	- Mode dispatch (single, batch, storno, query) - For bookings: duplicate check -> limit check -> select account FOR UPDATE -> compute balances -> insert BUCHUNGEN -> update KONTEN -> commit -> write journal and notify - For batch: loop up to 50 single bookings, rollback on first failure - For storno: mark original booking ST, then re-run single booking flow - For turnover query: declare/open cursor, fetch up to 50 rows into UL arrays, close, and return count	- LK-RUECKGABE set from BUCHUNGS-RUECKGABE at program end - LK-ANZAHL set to number of fetched turnover rows - DB state changes: rows inserted into BUCHUNGEN and BUCHUNGS_JOURNAL, KONTEN updated (commit or rollback)

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Field mapping not available from static analysis. Provide source files for detailed mapping.

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
Duplicate END_TO_END_ID within last day found (WS-BUCH-ZAEHLER > 0)	Detected potential double booking	Set WS-DUPLIKAT-FLAG = 'J' then set BCH-RC = 0004 and BCH-FEHLERMSG = 'DOPPELTE BUCHUNG ABGELEHNT' and abort booking paragraph.	Throw DuplicateBookingException; do not perform insert/update; return specific error code 0004 to caller.
Limit check returns NOT KTO-OK	Booking violates account limits or checks	Set BCH-RC = 0001 and set BCH-FEHLERMSG = KTO-FEHLERMSG; abort booking flow.	Map to LimitExceededException with message from KTO-FEHLERMSG returned by LIMITPRU service.
SQL error on select/update/insert (SQLCODE not handled explicitly)	Database error occurred	Move SQLCODE to ERR-SQLCODE, PERFORM 9100-SQL-FEHLER which sets BCH-RC = 0099, sets BCH-FEHLERMSG and executes ROLLBACK.	Rollback transaction and throw DataAccessException containing SQL error code; return generic error 0099 to caller.
Insert BUCHUNGEN returns SQLCODE = -803	Unique constraint violation (duplicate key)	Treat as duplicate: set BCH-RC = 0004.	Map SQLIntegrityConstraintViolationException to DuplicateBookingException and return code 0004.
Batch booking: any booking results in BCH-OK = false	An item in batch failed	Execute SQL ROLLBACK and exit batch loop; do not continue with subsequent bookings.	Roll back entire batch transaction and return partial failure result; stop processing further items.

9. Technical Dependencies

Dependency Type	Name / Identifier	Direction	Notes / Migration Action
CICS Runtime	CICS Transaction Server	PLATFORM	Replace with Spring Boot @RestController. Remove EIB fields and DFHCOMMAREA.

10. Ready for Implementation ? Checklist

[ ] Define service interface with methods: performSingleBooking(BookingRequest), performBatchBooking(List<BookingRequest>), performStorno(String bookingId), queryTurnover(account, fromDate, toDate, maxRows=50).

[ ] Implement transaction management using Spring @Transactional; on SQL exceptions translate codes to domain error codes (0001, 0004, 0099).

[ ] Implement duplicate detection by querying count of bookings with same endToEndId in last 1 day before insert within same transaction/isolation level.

[ ] Limit the turnover query result to 50 rows and return count to caller in response object; preserve ordering by booking date desc.

[ ] Add comprehensive tests for duplicate, limit failure, SQL error rollback, batch rollback and storno flow.

BUSINESS RULES: BUCHSERV (continued)

Type: CORE_SERVICE | Risk: HIGH (77)

CRITICAL FINDINGS

Atomicity relies on manual COMMIT/ROLLBACK and FOR UPDATE; migration must ensure identical transaction isolation to avoid lost updates.

BCH-OK flag and other status flags come from copybooks (BUCHSTRKC, KTOSTRKC) ? ensure full semantics of those copybooks are replicated.

Journal insert (BUCHUNGS_JOURNAL) lacks explicit error handling; a failure there currently is not handled and could mask booking success/failure.

LIMITPRU and BENACHRI are external dependencies invoked via CALL; their failure modes determine booking acceptance and must be implemented or mocked.

Date handling uses CURRENT DATE and CURRENT TIMESTAMP in SQL; ensure Java-side timezone and date semantics align with DB.

BUSINESS RULES: UEBERWEIG

Type: BATCH_ENTRY | Risk: MEDIUM (63) | 271 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
UEBERWEIG	BATCH_ENTRY	MEDIUM (63)	271	No UI ? batch processing

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (18)

- WS-EMPF-IBAN
- WS-IBAN-VALID
- KD-NACHNAME
- KD-VORNAME
- KD-STRASSE
- KD-PLZ
- KD-ADRESSE-SEIT
- KD-TELEFON-PRIVAT
- ... and 10 more

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

1. Overview

Program Name	UEBERWEIG
Type	BATCH_ENTRY
Purpose	Handles CICS online domestic/SEPA transfers: collects and validates transfer input, checks limits, posts booking via BUCHSERV and returns confirmation or error to the terminal.
User Interface	No UI ? batch processing
Data Source	TBD ? see source code

2. Business Use Case

Use Case	CICS Online Transfer Entry and Processing
Actor / User	CICS terminal user (customer or teller)
Description	User enters transfer details on UEBWMAP, the program validates IBAN and amount, checks account and limits, posts booking via BUCHSERV and returns confirmation or error screens.
Goal	Successfully validate and post a domestic or SEPA transfer and present confirmation to the terminal user.
Trigger	User invokes transaction UEBW or returns to program with DFHCOMMAREA (follow-up call) and submits transfer (PF1 or ENTER).

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
WS-AUFTR-KTONR	WS-EINGABE-DATEN	numeric (PIC 9(10))	Exists in KONTEN table (SQL SELECT). If not found show 'KONTO NICHT GEFUNDEN'.
WS-EMPF-IBAN	WS-EINGABE-DATEN	string (PIC X(22))	Country code checked (first 2 chars). Validated via external CALL 'IBANPRU' which sets WS-IBAN-VALID or WS-FEHLERMSG-ANZ.
WS-BETRAG-INTERN	WS-EINGABE-DATEN	packed decimal S9(11)V99 COMP-3	Must be > 0 (sets WS-BETRAG-VALID), and checked via LIMITPRU against account limits.

BUSINESS RULES: UEBERWEIG (continued)

Type: BATCH_ENTRY | Risk: MEDIUM (63)

WS-WAEHRUNG-EING	WS-EINGABE-DATEN	string (PIC X(3))	Default 'EUR' set on first call; moved into BCH-WAEHRUNG when building booking.
WS-VERWENDUNG	WS-EINGABE-DATEN	string (PIC X(140))	Moved to BCH-VERWENDUNGSZWECK for booking; no syntactic validation in program.
WS-AUSFUEHR-DATUM	WS-EINGABE-DATEN	string (PIC X(8))	Moved to BCH-VALUTADATUM; no internal date format checks performed here.

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Program entry selects first or follow-up call: EVALUATE TRUE WHEN ERSTER-AUFRUF WHEN FOLGE-AUFRUF -> If ERSTER-AUFRUF perform 2000-ERSTEN-AUFRUF; if FOLGE-AUFRUF perform 3000-FOLGE-AUFRUF (control-flow, Line 54), 98% confidence
2	CICS initialization sets comarea and first-call flag: IF EIBCALEN > 0 -> Set WS-ERSTE-AUFRUF = 'N' and MOVE DFHCOMMAREA TO WS-EINGABE-DATEN; always move program/terminal/transaction IDs to ERR- variables (initialization, Line 66), 95% confidence
3	First call initializes screen and defaults: PERFORM 2100-FELDER-INITIALISIEREN (called from 2000-ERSTEN-AUFRUF) -> Clear BCH-BUCHUNGS-ID, clear WS-EMPF-IBAN and WS-EMPF-NAME, set WS-BETRAG-INTERN zeros, set WS-WAEHRUNG-EING 'EUR', set BCH-TYP 'UEBW' (initialization, Line 74), 97% confidence
4	Read ordering account from database: EXEC SQL SELECT ... FROM KONTEN WHERE KONTO_NR = :WS-AUFTR-KTONR; IF SQLCODE NOT EQUAL 0 -> On success populate KTO- fields; on SQLCODE <> 0 move 'KONTO NICHT GEFUNDEN' to WS-FEHLERMSG-ANZ and PERFORM 9200-FEHLER-ANZEIGEN (file-io, Line 85), 95% confidence
5	Reject if account is locked: IF KTO-GESPERRT -> Move 'KONTO GESPERRT' to WS-FEHLERMSG-ANZ and PERFORM 9200-FEHLER-ANZEIGEN (validation, Line 95), 90% confidence
6	Initial screen send to terminal: EXEC CICS SEND MAP('UEBWMAP1') MAPSET('UEBWMAP') ERASE CURSOR -> Send UEBWMAP1 to terminal with ERASE and set RESP in WS-RESP (file-io, Line 100), 99% confidence
7	Receive map and route by PF/ENTER keys: EXEC CICS RECEIVE MAP('UEBWMAP1') ...; EVALUATE EIBAID WHEN DFHPF1 WHEN DFHPF3 WHEN DFHPF5 WHEN DFHENTER WHEN OTHER -> DFHPF1 or DFHENTER => PERFORM 4000-UEBERWEISUNG-VERARBEITEN; DFHPF3 => program end; DFHPF5 => clear inputs; OTHER => show 'UNBEKANNTE TASTE' error (control-flow, Line 112), 98% confidence
8	Validation orchestrator sets overall result: In 4100-EINGABEN-VALIDIEREN: default MOVE 'N' TO WS-PRUEF-ERGEBNIS; PERFORM subchecks; IF IBAN-GUELTIG AND BETRAG-GUELTIG -> If both IBAN-GUELTIG and BETRAG-GUELTIG then set WS-PRUEF-ERGEBNIS = 'J' (ALLES-OK); otherwise remain 'N' (validation, Line 128), 98% confidence
9	IBAN country determines SEPA area and is validated via IBANPRU: IF WS-EMPF-IBAN(1:2) EQUAL 'DE' ELSE; CALL 'IBANPRU' USING WS-EMPF-IBAN WS-IBAN-VALID WS-FEHLERMSG-ANZ -> If first two chars = 'DE' set WS-SEPA-GEBIET='I' else 'E'; call IBANPRU which returns WS-IBAN-VALID and possibly sets WS-FEHLERMSG-ANZ (validation, Line 140), 95% confidence
10	Amount must be greater than zero: IF WS-BETRAG-INTERN > ZEROS -> If > 0 set WS-BETRAG-VALID = 'J'; else move 'BETRAG MUSS GROESSER NULL SEIN' to WS-FEHLERMSG-ANZ (validation, Line 150), 98% confidence

BUSINESS RULES: UEBERWEIG (continued)

Type: BATCH_ENTRY | Risk: MEDIUM (63)

11	Limit check via LIMITPRU service: CALL 'LIMITPRU' USING WS-AUFTR-KTONR WS-BETRAG-INTERN BCH-TYP KONTO-RUECKGABE; IF NOT KTO-OK -> If LIMITPRU indicates not OK move KTO-FEHLERMSG to WS-FEHLERMSG-ANZ and set WS-PRUEF-ERGEBNIS = 'N' (validation, Line 158), 92% confidence
12	Abort processing on validation failure: In 4000-UEBERWEISUNG-VERARBEITEN IF NOT ALLES-OK -> PERFORM 9200-FEHLER-ANZEIGEN and EXIT PARAGRAPH (stop further booking steps) (control-flow, Line 122), 97% confidence
13	Build booking record and unique booking id: 4200-BUCHUNGSDATEN-AUFBAUEN with EXEC SQL SELECT CHAR(GENERATE_UNIQUE()) ... -> Generate BCH-BUCHUNGS-ID via GENERATE_UNIQUE and move WS and KTO values into BCH- fields (konto-from, iban-from, iban-to, amount, currency, type 'UEBW', channel 'ONLN', usage, end-to-end id, valuta date) (data-transformation, Line 170), 95% confidence
14	Perform booking via BUCHSERV: CALL 'BUCHSERV' USING ... BUCHUNGS-SATZ BUCHUNGS-RUECKGABE -> Invoke BUCHSERV to post the booking; BUCHSERV returns status in BUCHUNGS-RUECKGABE (BCH-OK/BCH-FEHLERMSG) (file-io, Line 184), 94% confidence
15	Show confirmation on successful booking: IF BCH-OK (after BUCHSERV) -> PERFORM 4400-BESTAETIGUNG-ANZEIGEN which sends UEBWMAP2 DATAONLY to terminal (display, Line 188), 96% confidence
16	Show booking error on failure: ELSE after BUCHSERV (BCH not OK) -> Move BCH-FEHLERMSG to WS-FEHLERMSG-ANZ and PERFORM 9200-FEHLER-ANZEIGEN (send map1 as ALARM) (display, Line 191), 96% confidence
17	Clear inputs and refresh screen when PF5 pressed: 4100-EINGABEN-LOESCHEN (triggered by DFHPF5) -> Move SPACES to WS-EMPF-IBAN and WS-EMPF-NAME, ZEROS to WS-BETRAG-INTERN, SPACES to WS-VERWENDUNG and PERFORM 2300-MAP-SENDEN (data-transformation, Line 196), 98% confidence
18	Return to caller with COMMAREA on program end: 8000-PROGRAMM-ENDE EXEC CICS RETURN TRANSID('UEBW') COMMAREA(DFHCOMMAREA) LENGTH(300) -> Move WS-EINGABE-DATEN to DFHCOMMAREA and return to CICS with fixed length 300 (termination, Line 206), 98% confidence
19	Error display uses map send with ALARM: 9200-FEHLER-ANZEIGEN EXEC CICS SEND MAP('UEBWMAP1') ... DATAONLY ALARM -> Send error map UEBWMAP1 as DATAONLY with ALARM; RESP put into WS-RESP (display, Line 212), 97% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- Terminal map UEBWMAP1 fields received into WS-EINGABE-DATEN - WS-AUFTR-KTONR used to select KONTEN row - IBAN and amount from WS-EINGABE-DATEN used for validation calls	- On first call initialize defaults; on follow-up receive map and branch by PF/ENTER - Validate IBAN (IBANPRU), amount (>0) and limit (LIMITPRU) - If valid build booking record (generate unique id) and call BUCHSERV to post booking	- SEND MAP UEBWMAP1 (initial and error screens) and UEBWMAP2 (confirmation) via CICS - On program end RETURN with COMMAREA of length 300

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Field mapping not available from static analysis. Provide source files for detailed mapping.

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
SQLCODE <> 0 after SELECT KONTEN	Ordering account not found	Move 'KONTO NICHT GEFUNDEN' to WS-FEHLERMSG-ANZ and PERFORM 9200-FEHLER-ANZEIGEN (send map1 with ALARM)	Throw AccountNotFoundException; show error screen; do not proceed to booking.

BUSINESS RULES: UEBERWEIG (continued)
Type: BATCH_ENTRY | Risk: MEDIUM (63)

KTO-GESPERRT true	Account is locked	Move 'KONTO GESPERRT' to WS-FEHLERMSG-ANZ and PERFORM 9200-FEHLER-ANZEIGEN	Return error response indicating account locked; prevent booking.
IBANPRU sets WS-IBAN-VALID != 'J'	IBAN invalid	WS-FEHLERMSG-ANZ set by IBANPRU and PERFORM 9200-FEHLER-ANZEIGEN (stop processing)	Call IBAN validation library/service and return validation errors to UI.
WS-BETRAG-INTERN <= 0	Amount zero or negative	Set WS-FEHLERMSG-ANZ = 'BETRAG MUSS GROESSER NULL SEIN' and PERFORM 9200-FEHLER-ANZEIGEN	Validate amount > 0 on server side before calling booking service.
LIMITPRU indicates NOT KTO-OK	Insufficient funds or limit violation	Move KTO-FEHLERMSG into WS-FEHLERMSG-ANZ, set WS-PRUEF-ERGEBNIS='N' and show error map	Call limits/funds service; on failure return message to UI and do not post booking.
BUCHSERV returns not OK (BCH-OK false)	Booking service failed	Move BCH-FEHLERMSG to WS-FEHLERMSG-ANZ and PERFORM 9200-FEHLER-ANZEIGEN	Treat BUCHSERV failure as transactional failure, show error details and do not confirm.

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Define service interfaces for IBAN validation, limit check and booking (signatures matching CALL parameters).

[ ] Create DTOs for map data (UEBWMAP1/2) and map fields to WS-EINGABE-DATEN structure.

[ ] Implement DB access to KONTEN table and a method to generate unique booking IDs (DB or UUID).

[ ] Add server-side validation for amount (>0), IBAN format, and SEPA country detection (first two chars).

[ ] Implement error handling to propagate error messages to UI similarly to WS-FEHLERMSG-ANZ and maintain equivalent user flows for PF keys/Enter.

CRITICAL FINDINGS

Program relies on external CALLs (IBANPRU, LIMITPRU, BUCHSERV) and DB SELECT; these must be available when migrating.

COMMAREA fixed length 300 is used for input/output; losing this contract may break other programs unless adapted.

Several control flags and status bits (KTO-OK, BCH-OK, KTO-GESPERRT) come from COPYed structures ? ensure those definitions are preserved.

No explicit date format validation for WS-AUSFUEHR-DATUM; migration should add date parsing/validation.

Error messages are in German hardcoded; consider i18n if migrating to multi-language environment.

BUSINESS RULES: BATCHBCH

Type: BATCH_ENTRY | Risk: MEDIUM (63) | 281 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
BATCHBCH	BATCH_ENTRY	MEDIUM (63)	281	No UI ? batch processing

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (13)

- DAUA-KONTO-VON
- DAUA-IBAN-NACH
- LAST-KONTO-VON
- LAST-IBAN-NACH
- ERG-KONTO
- KTO-IBAN
- KTO-INH-NAME
- KTO-INH-VORNAME
- ... and 5 more

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

1. Overview

Program Name	BATCHBCH
Type	BATCH_ENTRY
Purpose	Nightly batch booking processor that executes standing orders, processes SEPA direct debits and collective transfers, records failed bookings/returns, performs day-end interest processing and updates a batch run log in the database.
User Interface	No UI ? batch processing
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Nightly batch booking run
Actor / User	Batch scheduler / System operator
Description	Reads standing orders and direct debit input files, attempts bookings via external services, logs results and errors to files and database, marks return-debits for later processing, performs interest calculation and commits changes.
Goal	Process all pending standing orders and direct debits for a run date and record outcomes and necessary follow-up actions
Trigger	Scheduled nightly invocation (JCL) that runs the BATCHBCH program

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
DAUA-KONTO-VON	DAUA-SATZ	PIC 9(10) numeric account number	Validated by LIMITPRU call; if LIMITPRU fails account is considered invalid/insufficient
DAUA-IBAN-NACH	DAUA-SATZ	PIC X(22) IBAN	Moved to BCH-IBAN-NACH for booking; no local format check
DAUA-BETRAG	DAUA-SATZ	PIC S9(9)V99 COMP-3 amount	Subject to LIMITPRU and booking service checks
LAST-KONTO-VON	LAST-SATZ	PIC 9(10) numeric account number	Used in booking; failures lead to RUECKLASTSCHRIFT insertion if BUCHSERV fails

LAST-IBAN-NACH	LAST-SATZ	PIC X(22) IBAN	No local validation; persisted to RUECKLASTSCHRIFTEN on failure
WS-LAUF-DATUM	WORKING-STORAGE	PIC X(8) run date (ISO)	Populated by an SQL SELECT of CURRENT DATE

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Program execution sequence: PROCEDURE DIVISION: 0000-HAUPTPROGRAMM PERFORM 1000-INITIALISIERUNG THEN PERFORM 2000... THEN STOP RUN -> Execute initialization, process standing orders, process direct debits, process return-debits, perform day-end, then perform closing/clean-up and stop (control-flow, Line 60), 99% confidence
2	Set error program name: MOVE WS-PROGRAMM-NAME TO ERR-PROGRAMM -> Populate error handling structure with the program name 'BATCHBCH' for downstream error/reporting (initialization, Line 82), 95% confidence
3	Open input and output files at startup: OPEN INPUT DAUERAUFTRAG-DATEI, OPEN INPUT LASTSCHRIFT-DATEI, OPEN OUTPUT ERGEBNIS-DATEI, OPEN OUTPUT FEHLER-DATEI -> Open two input files (standing orders, direct debits) and two output files (result file, error file) before processing (file-io, Line 84), 98% confidence
4	Determine run date: EXEC SQL SELECT CHAR(CURRENT DATE, ISO) INTO :WS-LAUF-DATUM FROM SYSIBM.SYSDUMMY1 -> Populate WS-LAUF-DATUM with current date (ISO format) for use throughout the run (date-time, Line 88), 98% confidence
5	Insert start run record: EXEC SQL INSERT INTO BATCH_LAUF_PROTOKOLL ... VALUES (:WS-PROGRAMM-NAME, :WS-LAUF-DATUM, CURRENT TIMESTAMP, 'LAUF') -> Create a database record marking the batch run as started with status 'LAUF' (file-io, Line 93), 97% confidence
6	Initial read of standing orders file sets EOF flag: READ DAUERAUFTRAG-DATEI AT END MOVE 'J' TO WS-DAUA-EOF -> On first read, if end-of-file is reached mark WS-DAUA-EOF='J' (DAUA-DATEI-ENDE will be true) (file-io, Line 106), 98% confidence
7	Loop over standing orders until EOF: PERFORM UNTIL DAUA-DATEI-ENDE -> For each DAUA record increment WS-DAUA-GESAMT, process booking (2100-DAUERAUFTRAG-BUCHEN), then read next record and set EOF when AT END (control-flow, Line 108), 99% confidence
8	Limit check before booking standing order: CALL 'LIMITPRU' USING DAUA-KONTO-VON, DAUA-BETRAG, BCH-TYP ...; IF KTO-OK -> If LIMITPRU returns KTO-OK true proceed to booking; otherwise increment WS-DAUA-FEHLER and write error protocol (2200) (validation, Line 132), 95% confidence
9	Attempt booking for standing orders: CALL 'BUCHSERV' ...; IF BCH-OK -> On BCH-OK true increment WS-DAUA-OK; on BCH-OK false increment WS-DAUA-FEHLER and perform 2200-FEHLER-PROTOKOLL (file-io, Line 137), 98% confidence
10	Write error record for failed standing order: 2200-FEHLER-PROTOKOLL (MOVE fields then WRITE FEH-SATZ FROM WS-ERG-SATZ) -> Compose ERG record with date, account, amount, type and KTO error message, set ERG-STATUS='FE' and write to FEHLER-DATEI (file-io, Line 154), 98% confidence
11	Initial read of direct debit file sets EOF flag: READ LASTSCHRIFT-DATEI AT END MOVE 'J' TO WS-LAST-EOF -> On first read, if end-of-file reached mark WS-LAST-EOF='J' (LAST-DATEI-ENDE will be true) (file-io, Line 170), 98% confidence
12	Loop over direct debits until EOF: PERFORM UNTIL LAST-DATEI-ENDE -> For each LAST record increment WS-LAST-GESAMT, perform booking (3100-LASTSCHRIFT-BUCHEN), then read next record and set EOF when AT END (control-flow, Line 174), 99% confidence

BUSINESS RULES: BATCHBCH (continued)

Type: BATCH_ENTRY | Risk: MEDIUM (63)

13	Prepare SEPA booking attributes for direct debit: In 3100-LASTSCHRIFT-BUCHEN MOVE LAST fields to BCH fields and MOVE 'LAST' TO BCH-TYP AND MOVE 'SEPA' TO BCH-KANAL -> Set booking context: type 'LAST' and channel 'SEPA', then call BUCHSERV to attempt booking (data-transformation, Line 194), 97% confidence
14	Handle direct debit booking success/failure: CALL 'BUCHSERV' ...; IF BCH-OK -> On success: increment WS-LAST-OK and set ERG-STATUS='OK'. On failure: increment WS-LAST-FEHLER, set ERG-STATUS='FE' and PERFORM 3200 to mark a return-debit (validation, Line 201), 98% confidence
15	Write result record for each direct debit: After booking in 3100: MOVE WS-LAUF-DATUM TO ERG-DATUM; WRITE ERG-SATZ FROM WS-ERG-SATZ -> Write a result line to ERGEBNIS-DATEI containing date, account, amount, type, status and info (file-io, Line 210), 98% confidence
16	Record return-debit for failed direct debit: 3200-RUECKLASTSCHRIFT-VORMERKEN (EXEC SQL INSERT INTO RUECKLASTSCHRIFTEN ... VALUES (:LAST-...,:WS-LAUF-DATUM,:BCH-FEHLERMSG)) -> Insert a row into RUECKLASTSCHRIFTEN with account, creditor IBAN, amount, mandate ref, run date and error reason; increment WS-RUECK-ANZ (file-io, Line 217), 97% confidence
17	Mark older open return-debits processed: EXEC SQL UPDATE RUECKLASTSCHRIFTEN SET STATUS='VERARBEITET', VERARBEITUNGSDATUM = :WS-LAUF-DATUM WHERE RUECK_DATUM < :WS-LAUF-DATUM AND STATUS = 'OFFEN' -> For all RUECKLASTSCHRIFTEN with RUECK_DATUM earlier than current run date and status 'OFFEN' set status to 'VERARBEITET' and set processing date (file-io, Line 233), 98% confidence
18	Day-end interest processing and commit: CALL 'ZINSBERCH' USING WS-LAUF-DATUM; EXEC SQL COMMIT -> Run interest calculation subprogram for the run date and commit database work (calculation, Line 252), 96% confidence
19	Close files and finalize batch log: 9000-ABSCHLUSS CLOSE ... EXEC SQL UPDATE BATCH_LAUF_PROTOKOLL SET ENDE_TIMESTAMP=CURRENT TIMESTAMP, STATUS='OK', SAETZE_GESAMT=:WS-DAUA-GESAMT, SAETZE_BEARB=:WS-DAUA-OK, SAETZE_FEHLER=:WS-DAUA-FEHLER WHERE PROGRAMM_NAME=:WS-PROGRAMM-NAME AND DATUM=:WS-LAUF-DATUM -> Close all files; update BATCH_LAUF_PROTOKOLL with end timestamp, status 'OK' and counters (note: only DAUA counters are written back) (termination, Line 264), 96% confidence
20	88-level EOF conditions: WS-DAUA-EOF PIC X(1) VALUE 'N' with 88 DAUA-DATEI-ENDE VALUE 'J'; WS-LAST-EOF similarly defines LAST-DATEI-ENDE -> Treat WS-DAUA-EOF='J' and WS-LAST-EOF='J' as boolean named conditions DAUA-DATEI-ENDE and LAST-DATEI-ENDE used to control loops (control-flow, Line 44), 99% confidence
21	File status variables declared but not checked: FILE-CONTROL clauses specify FILE STATUS IS WS-FILE-STATUS-1..4 but no subsequent IF checks on these fields -> Program relies on READ AT END and subprogram flags instead of explicit file status checks; file status variables are declared but not used for error handling (validation, Line 18), 80% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- DAUERAUFTRAG-DATEI (DAUA-SATZ records) - LASTSCHRIFT-DATEI (LAST-SATZ records) - Current date via SQL (WS-LAUF-DATUM)	- For each DAUA record: increment counters, LIMITPRU, call BUCHSERV, on failure write error record - For each LAST record: increment counters, set BCH type/channel, call BUCHSERV, on failure insert RUECKLASTSCHRIFTEN and record ERG record - Perform update of older RUECKLASTSCHRIFTEN to 'VERARBEITET' - Call interest calculation subprogram and COMMIT	- FEHLER-DATEI: error records for DAUA failures (FEH-SATZ) - ERGEBNIS-DATEI: result records for LAST processing (ERG-SATZ) - Database: INSERT into BATCH_LAUF_PROTOKOLL at start; UPDATE at end - Database: INSERT into RUECKLASTSCHRIFTEN for failed LAST bookings; UPDATE processed return-debits

BUSINESS RULES: BATCHBCH (continued)
Type: BATCH_ENTRY | Risk: MEDIUM (63)

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Field mapping not available from static analysis. Provide source files for detailed mapping.

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
LIMITPRU indicates failure (KTO-OK false)	Account limit or validation failed for a standing order	WS-DAUA-FEHLER incremented; 2200-FEHLER-PROTOKOLL performed and FEH-SATZ written with KTO-FEHLERMSG	Throw or return a validation error result; persist error details to an error table/audit log and increment counters; do not call booking service
BUCHSERV returns failure for DAUA (BCH-OK false)	Booking service rejected/stored booking failed for a standing order	WS-DAUA-FEHLER incremented and 2200-FEHLER-PROTOKOLL performed; error record written	Treat as business error: record failure with details and continue processing next record; do not abort whole run
BUCHSERV returns failure for LAST (BCH-OK false)	Direct debit booking failed and must be recorded as return-debit	WS-LAST-FEHLER incremented; 3200-RUECKLASTSCHRIFT-VORMERKEN inserts record into RUECKLASTSCHRIFTEN and increments WS-RUECK-ANZ; ERG-SATZ written with status 'FE'	Insert a return-debit entry in DB and increment counters; ensure error reason from booking service is stored
READ ... AT END sets WS-*-EOF	End of input file reached	Per-file loops terminate and processing moves to next section	Stop iterating when EOF detected; close resources cleanly
SQL INSERT/UPDATE or CALLs raise DB/service exceptions	Database or external service failure during run	Program does not show explicit SQL error handling in source; unspecified (likely exception bubbled to runtime or handled in copied ERRHANDKC)	Wrap DB operations in try/catch, ensure transactions and retries where appropriate, and set run-status to error; log and persist partial results

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Define interfaces for LIMITPRU, BUCHSERV and ZINSBERCH and implement mocks for unit testing.

[ ] Implement DTOs for DAUA-SATZ and LAST-SATZ with precise decimal mapping for COMP-3 amounts.

[ ] Reproduce PERFORM UNTIL EOF loops using streaming file readers with pre-read semantics (read first record, then loop until EOF).

[ ] Implement robust error handling for file I/O and SQL operations; map WS-FILE-STATUS checks to explicit exceptions/return codes.

[ ] Ensure batch-run logging preserves start/stop timestamps and counters; decide how LAST counters should be persisted in target schema.

CRITICAL FINDINGS

FILE STATUS variables (WS-FILE-STATUS-1..4) are declared but never checked ? file I/O errors beyond AT END are not handled explicitly.

WS-FEHLER-FLAG (and 88 LAUF-FEHLER) is defined but not used ? global run-level error signaling appears incomplete.

Final BATCH_LAUF_PROTOKOLL update writes only DAUA counters (WS-DAUA-GESAMT, WS-DAUA-OK, WS-DAUA-FEHLER) ? WS-LAST counters are not included which may misreport total processed records.

BUSINESS RULES: ZINSBERCH

Type: BATCH_ENTRY | Risk: MEDIUM (59) | 265 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
ZINSBERCH	BATCH_ENTRY	MEDIUM (59)	265	No UI ? batch processing

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (11)

- PROT-KONTO
- CURSOR-KONTO-NR
- CURSOR-KUNDEN-NR
- KTO-IBAN
- KTO-INH-NAME
- KTO-INH-VORNAME
- ERR-USER-ID
- BCH-KONTO-VON
- ... and 3 more

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

1. Overview

Program Name	ZINSBERCH
Type	BATCH_ENTRY
Purpose	Batch interest calculation for all active accounts: calculate daily interest (credit and debit), compute taxes for credit interest, post bookings via BUCHSERV, write protocol, and record tax totals to KAPITALERTRAGSTEUER.
User Interface	No UI ? batch processing
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Daily batch interest posting
Actor / User	Batch processing (scheduled job / JCL ZINSJCL)
Description	Calculate daily interest for all active GK and SK accounts, apply taxes for credit interest, post bookings via accounting service, write a protocol and record tax totals.
Goal	Accurately compute and post daily interest and record tax liabilities for the day
Trigger	Scheduled batch start (JCL ZINSJCL) which invokes program ZINSBERCH

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
KONTEN rows	CURSOR-KONTO-NR, CURSOR-SALDO, CURSOR-HABEN-ZINS, CURSOR-SOLL-ZINS, CURSOR-DISPO-LIMIT, CURSOR-KTO-TYP, CURSOR-KUNDEN-NR	database table rows (KONTEN) filtered by STATUS='A' and KONTO_TYP in ('GK','SK')	Fetch loop checks SQLCODE; only proceeds when SQLCODE = 0. Saldo sign determines credit or debit processing.
WS-ABRECHNUNGSDAT	WS-ABRECHNUNGSDAT	date string from DB CURRENT DATE	Loaded in initialization; used for booking dates and batch log update

Staffel entry	STF-BETRAG-BIS(1), STF-ZINSSATZ(1)	database lookup	Loaded from ZINS_STAFFELN where STAFFEL_NR=1 and AKTIV='J'; no further validation in code

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Program initialization: PERFORM 1000-INITIALISIERUNG -> Set ERR-PROGRAMM to WS-PROGRAMM-NAME, open ZINS-PROTOKOLL and FEHLER-DATEI for OUTPUT, load current date into WS-ABRECHNUNGSDAT, perform staffel and freistellorder loads (initialization, Line 1000), 95% confidence
2	Open protocol and error files: OPEN OUTPUT ZINS-PROTOKOLL / OPEN OUTPUT FEHLER-DATEI -> Open output files ZINS-PROTOKOLL and FEHLER-DATEI and set file status into WS-FILE-STATUS / WS-FEHLER-STATUS (file-io, Line 1000), 95% confidence
3	Set accounting date from database: EXEC SQL SELECT CHAR(CURRENT DATE, ISO) INTO :WS-ABRECHNUNGSDAT -> Populate WS-ABRECHNUNGSDAT with CURRENT DATE in ISO format (date-time, Line 1000), 98% confidence
4	Load interest tier (staffel) entry 1: EXEC SQL SELECT BETRAG_BIS, ZINSSATZ INTO :STF-BETRAG-BIS(1), :STF-ZINSSATZ(1) FROM ZINS_STAFFELN WHERE STAFFEL_NR = 1 AND AKTIV = 'J' -> Populate STF-BETRAG-BIS(1) and STF-ZINSSATZ(1) from ZINS_STAFFELN for active staffel 1 (data-transformation, Line 1100), 90% confidence
5	Load sum of freistellorders for tax year: EXEC SQL SELECT NVL(SUM(FREISTELLUNGSBETRAG), 0) INTO :WS-FREISTELLORDER FROM FREISTELLUNGSAUFTRAEGE WHERE STEUER_JAHR = :WS-STEUERJAHR AND AKTIV = 'J' -> Populate WS-FREISTELLORDER with the sum (or 0) of active freistellungsauftraege for WS-STEUERJAHR (data-transformation, Line 1200), 90% confidence
6	Select active accounts for processing: EXEC SQL DECLARE KONTO_CURSOR CURSOR FOR ... WHERE STATUS = 'A' AND KONTO_TYP IN ('GK', 'SK') -> Prepare cursor KONTO_CURSOR to iterate all active accounts of types GK and SK ordered by KUNDEN_NR, KONTO_NR for update of SALDO and VERFUEGBAR (search-filter, Line 2000), 98% confidence
7	Iterate accounts until no more rows: PERFORM UNTIL SQLCODE EQUAL 100 (and repeated EXEC SQL FETCH ...) -> Fetch rows from KONTO_CURSOR repeatedly; stop when SQLCODE = 100 (no more data) (control-flow, Line 2000), 99% confidence
8	Handle successful fetch: IF SQLCODE EQUAL 0 AFTER FETCH -> ADD 1 TO WS-KONTEN-GESAMT and PERFORM 2100-EINZEL-KONTO-VERARBEITEN (control-flow, Line 2000), 98% confidence
9	Close account cursor after processing: EXEC SQL CLOSE KONTO_CURSOR -> Close the KONTO_CURSOR after the fetch loop completes (file-io, Line 2000), 98% confidence
10	Branch account processing by saldo sign: EVALUATE TRUE WHEN CURSOR-SALDO > ZEROS WHEN CURSOR-SALDO < ZEROS WHEN OTHER -> If saldo > 0 perform credit interest calculation (2200); if saldo < 0 perform debit interest calculation (2300); otherwise do nothing (control-flow, Line 2100), 99% confidence
11	Compute daily interest rate for credit: COMPUTE WS-TAGESZINS = CURSOR-HABEN-ZINS / 100 / WS-JAHRESTAGE -> Set WS-TAGESZINS as CURSOR-HABEN-ZINS percent divided by WS-JAHRESTAGE (default 365) (calculation, Line 2200), 98% confidence
12	Compute interest amount for credit: COMPUTE WS-ZINSBETRAG = CURSOR-SALDO * WS-TAGESZINS -> Calculate WS-ZINSBETRAG = account SALDO * daily interest rate (calculation, Line 2200), 98% confidence
13	Only process and book credit interest if positive: IF WS-ZINSBETRAG > ZEROS -> Perform tax calculation (2400), set booking type to 'GUTS'/'ZINS', prepare BCH fields, CALL BUCHSERV to post net interest, increment WS-BUCHUNGEN-ANZ and write protocol (control-flow, Line 2200), 97% confidence

BUSINESS RULES: ZINSBERCH (continued)

Type: BATCH_ENTRY | Risk: MEDIUM (59)

14	Compute daily interest rate for debit: COMPUTE WS-TAGESZINS = CURSOR-SOLL-ZINS / 100 / WS-JAHRESTAGE -> Set WS-TAGESZINS for debit as CURSOR-SOLL-ZINS percent divided by WS-JAHRESTAGE (calculation, Line 2300), 98% confidence
15	Compute interest amount for debit and invert sign: COMPUTE WS-ZINSBETRAG = CURSOR-SALDO * WS-TAGESZINS * -1 -> Calculate WS-ZINSBETRAG from SALDO * daily rate and multiply by -1 to create charge amount (calculation, Line 2300), 97% confidence
16	Post debit interest booking: After computing WS-ZINSBETRAG in 2300 (no tax calculation invoked) -> Set BCH-TYP to 'LAST', move account to BCH-KONTO-VON, move WS-ZINSBETRAG to BCH-BETRAG, CALL BUCHSERV to post charge and increment WS-BUCHUNGEN-ANZ (file-io, Line 2300), 95% confidence
17	Compute tax and solidarity amounts for taxable interest: 2400-STEUER-BERECHNEN (called from credit path only) -> WS-STEUERBETRAG = WS-ZINSBETRAG * WS-KEST-SATZ / 100; WS-SOLI-BETRAG = WS-STEUERBETRAG * WS-SOLI-SATZ / 100; WS-NETTO-ZINS = WS-ZINSBETRAG - WS-STEUERBETRAG - WS-SOLI-BETRAG (calculation, Line 2400), 99% confidence
18	Write protocol entry after booking: 2500-PROTOKOLL-SCHREIBEN (called after successful booking) -> Populate PROT-SATZ fields (date, konto, typ, betrag, status, info) and WRITE PROT-SATZ to ZINS-PROTOKOLL (file-io, Line 2500), 96% confidence
19	Insert tax totals into KAPITALERTRAGSTEUER and commit: EXEC SQL INSERT INTO KAPITALERTRAGSTEUER ... VALUES (CURRENT DATE, :WS-STEUERBETRAG, :WS-SOLI-BETRAG, :WS-STEUERJAHR, :WS-PROGRAMM-NAME) -> Attempt to insert WS-STEUERBETRAG and WS-SOLI-BETRAG into KAPITALERTRAGSTEUER; if SQLCODE != 0 PERFORM 9100-SQL-FEHLER else EXEC SQL COMMIT (file-io, Line 3000), 98% confidence
20	Normal termination and batch log update: PERFORM 9000-ABSCHLUSS -> CLOSE ZINS-PROTOKOLL and FEHLER-DATEI; UPDATE BATCH_LAUF_PROTOKOLL set ENDE_TIMESTAMP CURRENT TIMESTAMP, STATUS 'OK' and counters: SAETZE_GESAMT, SAETZE_BEARB, SAETZE_FEHLER for this program and date (termination, Line 9000), 98% confidence
21	SQL error handling: IF SQLCODE NOT EQUAL 0 AFTER INSERT INTO KAPITALERTRAGSTEUER (or other SQL statements) -> PERFORM 9100-SQL-FEHLER which moves 'SQL FEHLER ZINSBERCH' into ERR-TEXT, moves SQLCODE to ERR-SQLCODE and WRITE FEHLER-SATZ to FEHLER-DATEI (validation, Line 9100), 95% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- Read active accounts from KONTEN via KONTO_CURSOR - Read current date and staffel and freistellorder via SQL selects	- For each account: determine credit vs debit by CURSOR-SALDO sign - Compute WS-TAGESZINS from HABEN_ZINSSATZ or SOLL_ZINSSATZ divided by 100 and WS-JAHRESTAGE - Compute WS-ZINSBETRAG = CURSOR-SALDO * WS-TAGESZINS (credit) or * -1 for debit - For positive credit WS-ZINSBETRAG compute tax and soli and net interest - Prepare BCH fields and CALL BUCHSERV to post booking, increment booking count - Write protocol record for each booking	- Write PROT-SATZ records to ZINS-PROTOKOLL - Write FEHLER-SATZ to FEHLER-DATEI on SQL error - Insert tax totals into KAPITALERTRAGSTEUER and COMMIT - Update BATCH_LAUF_PROTOKOLL with end timestamp and counters

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Field mapping not available from static analysis. Provide source files for detailed mapping.

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
SQLCODE != 0 after INSERT INTO KAPITALERTRAGSTEUER	Database insert failed when recording tax totals	PERFORM 9100-SQL-FEHLER: move error text and SQLCODE to error structure and WRITE FEHLER-SATZ; prevents COMMIT path	Throw SQLException or map to error handler; persist error details to an error table or log, avoid commit, or perform compensating actions
SQLCODE not equal 0 after other EXEC SQL statements (e.g., staffel load or cursor operations)	SQL operation failed (select/fetch/declare/open)	Fetch loop only processes when SQLCODE = 0; if SQLCODE becomes 100 loop terminates; other non-zero SQLCODEs are not explicitly handled except via 9100 in insert case	Check JDBC return/exception; on fetch exceptions log and continue or abort; implement explicit error handling and retries as needed
WS-ZINSBETRAG <= ZEROS for credit path	No positive interest to post	Do not compute taxes or call BUCHSERV; no booking or protocol entry is created	Skip booking for non-positive amounts; ensure zero/negative amounts are not persisted

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Implement JDBC-based retrieval of current date and staffel/freistellorder loads in initialization.

[ ] Implement streaming processing of accounts with transactional boundaries around BUCHSERV calls if booking must be atomic with DB.

[ ] Compute daily rate and interest using BigDecimal to avoid precision loss; use configured WS-JAHRESTAGE (default 365) and configurable tax rates (WS-KEST-SATZ, WS-SOLI-SATZ).

[ ] Ensure protocol and error file writing map to structured logging or persistent audit tables in the target system.

[ ] Add explicit handling for freistellorders usage or remove unused load to avoid confusion; add checks on BUCHSERV responses and handle retries/compensations.

CRITICAL FINDINGS

WS-FREISTELLORDER is loaded in initialization but never used in subsequent logic ? potential missing application of freistellorders to tax computation.

STAFFEL table is defined OCCURS 5 but only STAFFEL_NR = 1 is SELECTed into element (1); code never iterates over staffel table ? staffel logic seems incomplete.

Taxes (2400-STEUER-BERECHNEN) are only applied for positive credit interest; debit (Soll) interest does not compute taxes ? this appears intentional but should be confirmed as business rule.

BUCHSERV is called but its return BUCHUNGS-RUECKGABE is not checked; potential silent booking failures could occur.

SQL error handling is only explicitly performed for the tax INSERT; other SQL errors (e.g., staffel load, freistellorder load, cursor open/fetch) are not uniformly handled.

BUSINESS RULES: LIMITPRU

Type: SUBROUTINE | Risk: MEDIUM (58) | 144 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
LIMITPRU	SUBROUTINE	MEDIUM (58)	144	No UI ? batch processing

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (5)

- LK-KONTO-NR
- KTO-IBAN
- KTO-INH-NAME
- KTO-INH-VORNAME
- ERR-USER-ID

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

1. Overview

Program Name	LIMITPRU
Type	SUBROUTINE
Purpose	Performs central limit and amount checks for a bank account transaction: reads account limits and balances, checks daily total, single-transaction, dispo (overdraft) and foreign transfer limits, sets return code and message accordingly.
User Interface	No UI ? batch processing
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Limit and overdraft check for transaction
Actor / User	Calling program (e.g., BUCHSERV, UEBERWEIG, BATCHBCH)
Description	Validates whether a proposed booking (LK-BETRAG) for a given account (LK-KONTO-NR) violates daily, single, foreign-transfer, or overdraft limits and returns a code and message.
Goal	Approve or reject the transaction by setting return code and message (KTO-RC, KTO-FEHLERMSG) and returning structured response (LK-RUECKGABE).
Trigger	Procedure DIVISION invoked using LK-KONTO-NR, LK-BETRAG, LK-BUCHUNGSTYP, LK-RUECKGABE

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
LK-KONTO-NR	used in SQL WHERE and selects	numeric (PIC 9(10))	Must exist in KONTEN table; if not found SQLCODE != 0 triggers error KTO-RC=0099
LK-BETRAG	added to WS-TAGES-UMSATZ and compared to limits	packed decimal (PIC S9(13)V99 COMP-3)	Compared numerically to WS-TAGES-LIMIT, WS-EINZEL-LIMIT, WS-AUSLAND-LIMIT, WS-DISPO-GRENZE
LK-BUCHUNGSTYP	moved to WS-LIMIT-TYP and used to detect foreign transfer 'UEBW'	alphanumeric PIC X(4)	If equal 'UEBW' then foreign-transfer limit logic applies

Account fields from KONTEN	WS-TAGES-LIMIT, WS-EINZEL-LIMIT, WS-AUSLAND-LIMIT, WS-DISPO-GRENZE (initial), KTO-SALDO, KTO-VERFUEGBAR	various numeric packed/decimal	Assumed numeric; presence required for checks

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Main processing sequence: PERFORM 1000-INITIALISIERUNG PERFORM 2000-LIMIT-DATEN-LESEN PERFORM 3000-TAGES-UMSATZ-PRUEFEN PERFORM 4000-EINZEL-BETRAG-PRUEFEN PERFORM 5000-DISPO-PRUEFEN PERFORM 6000-ERGEBNIS-SETZEN GOBACK. -> Execute initialization, read limit data, perform daily-amount, single-amount and dispo checks, set result, then return to caller. (control-flow, Line 56), 99% confidence
2	Set error handler program name: PERFORM 1000-INITIALISIERUNG (1000-INITIALISIERUNG). -> MOVE WS-PROGRAMM-NAME TO ERR-PROGRAMM. (initialization, Line 62), 98% confidence
3	Initialize return code to zero: 1000-INITIALISIERUNG executed. -> MOVE ZEROS TO KTO-RC. (initialization, Line 63), 99% confidence
4	Initialize error message to blank: 1000-INITIALISIERUNG executed. -> MOVE SPACES TO KTO-FEHLERMSG. (initialization, Line 64), 99% confidence
5	Propagate booking type into working storage: 1000-INITIALISIERUNG executed. -> MOVE LK-BUCHUNGSTYP TO WS-LIMIT-TYP. (data-transformation, Line 65), 97% confidence
6	Read account limit and balance fields: EXEC SQL SELECT TAGES_LIMIT, EINZEL_LIMIT, AUSLAND_LIMIT, DISPO_LIMIT, SALDO, VERFUEGBAR INTO :WS-TAGES-LIMIT, :WS-EINZEL-LIMIT, :WS-AUSLAND-LIMIT, :WS-DISPO-GRENZE, :KTO-SALDO, :KTO-VERFUEGBAR FROM KONTEN WHERE KONTO_NR = :LK-KONTO-NR END-EXEC -> Populate working storage with account limits and balances from KONTEN for the given LK-KONTO-NR. (file-io, Line 78), 98% confidence
7	Handle missing account (SQL error): IF SQLCODE NOT EQUAL 0 -> MOVE 0099 TO KTO-RC, MOVE 'KONTO NICHT GEFUNDEN' TO KTO-FEHLERMSG, GOBACK (terminate and return to caller). (file-io, Line 84), 99% confidence
8	Calculate today's bookings sum for account: EXEC SQL SELECT NVL(SUM(BETRAG), 0) INTO :WS-TAGES-UMSATZ FROM BUCHUNGEN WHERE KONTO_VON = :LK-KONTO-NR AND BUCHUNGSDATUM = CURRENT DATE AND STATUS = 'GB' END-EXEC -> Set WS-TAGES-UMSATZ to the sum of today's bookings (status 'GB') for this account; 0 if none. (file-io, Line 96), 98% confidence
9	Include current transaction in daily total: COMPUTE WS-TAGES-UMSATZ = WS-TAGES-UMSATZ + LK-BETRAG -> Add the current transaction amount (LK-BETRAG) to WS-TAGES-UMSATZ. (calculation, Line 100), 99% confidence
10	Daily limit exceeded check: IF WS-TAGES-UMSATZ > WS-TAGES-LIMIT -> MOVE 'N' TO WS-TAGES-OK, MOVE 0003 TO KTO-RC, STRING 'TAGESLIMIT UEBERSCHRITTEN. LIMIT: ' WS-TAGES-LIMIT ' EUR' DELIMITED SIZE INTO KTO-FEHLERMSG. (validation, Line 102), 99% confidence
11	Single-transaction limit exceeded check: IF LK-BETRAG > WS-EINZEL-LIMIT -> MOVE 'N' TO WS-EINZEL-OK, MOVE 0003 TO KTO-RC, STRING 'EINZELBETRAGSLIMIT UEBERSCHRITTEN. LIMIT: ' WS-EINZEL-LIMIT ' EUR' DELIMITED SIZE INTO KTO-FEHLERMSG. (validation, Line 112), 99% confidence
12	Foreign transfer limit check for booking type UEBW: IF LK-BUCHUNGSTYP EQUAL 'UEBW' AND WS-AUSLAND-LIMIT > ZEROS AND LK-BETRAG > WS-AUSLAND-LIMIT -> MOVE 'N' TO WS-EINZEL-OK, MOVE 0003 TO KTO-RC, MOVE 'AUSLANDSUEBERWEISUNG LIMIT UEBERSCHRITTEN' TO KTO-FEHLERMSG. (validation, Line 118), 98% confidence

BUSINESS RULES: LIMITPRU (continued)

Type: SUBROUTINE | Risk: MEDIUM (58)

13	Compute dispo (overdraft) threshold: COMPUTE WS-DISPO-GRENZE = KTO-VERFUEGBAR + KTO-DISPOKREDITLIMIT -> Set WS-DISPO-GRENZE to available balance plus overdraft credit limit. (calculation, Line 130), 96% confidence
14	Dispo limit exceeded check: IF LK-BETRAG > WS-DISPO-GRENZE -> MOVE 'N' TO WS-DISPO-OK, MOVE 0001 TO KTO-RC, MOVE 'DISPOKREDITLIMIT UEBERSCHRITTEN' TO KTO-FEHLERMSG. (validation, Line 133), 99% confidence
15	Default approval flags: WS-PRUEF-ERGEBNIS data declaration -> Initialize WS-TAGES-OK, WS-EINZEL-OK, WS-DISPO-OK, WS-GESAMT-OK to 'J' by default. (initialization, Line 28), 98% confidence
16	Set overall success when all checks pass: IF WS-TAGES-OK EQUAL 'J' AND WS-EINZEL-OK EQUAL 'J' AND WS-DISPO-OK EQUAL 'J' -> MOVE 0000 TO KTO-RC and MOVE SPACES TO KTO-FEHLERMSG (clear error and set success return code). (control-flow, Line 140), 99% confidence
17	Return account response to caller: 6000-ERGEBNIS-SETZEN executed (end of processing). -> MOVE KONTO-RUECKGABE TO LK-RUECKGABE so caller receives account-specific structured response. (data-transformation, Line 144), 88% confidence
18	Only include bookings for current date: WHERE BUCHUNGSDATUM = CURRENT DATE in SELECT for WS-TAGES-UMSATZ -> Sum only today's bookings when calculating WS-TAGES-UMSATZ. (date-time, Line 97), 99% confidence
19	Only consider bookings with status 'GB' for daily sum: WHERE STATUS = 'GB' in SELECT for WS-TAGES-UMSATZ -> Include only bookings with status code 'GB' in daily total calculation. (search-filter, Line 97), 99% confidence
20	Program version is embedded as constant: WS-VERSION data declaration -> WS-VERSION initialized to value '2.0.3 '. (initialization, Line 16), 95% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- Receive LK-KONTO-NR, LK-BETRAG, LK-BUCHUNGSTYP from caller - Initialize WS flags and error structures	- SQL SELECT KONTEN -> populate WS limits and balances - SQL SELECT SUM(BETRAG) from BUCHUNGEN for current date and status 'GB' -> WS-TAGES-UMSATZ - Compute WS-TAGES-UMSATZ + LK-BETRAG - Compare against WS-TAGES-LIMIT, WS-EINZEL-LIMIT, WS-AUSLAND-LIMIT and WS-DISPO-GRENZE - Compute WS-DISPO-GRENZE = KTO-VERFUEGBAR + KTO-DISPOKREDITLIMIT - Set corresponding OK flags and KTO-RC/KTO-FEHLERMSG	- Set KTO-RC (return code) to 0000 on success, or 0001/0003/0099 on different failures - Set KTO-FEHLERMSG with human-readable message when rejected - MOVE KONTO-RUECKGABE TO LK-RUECKGABE returned to caller

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Field mapping not available from static analysis. Provide source files for detailed mapping.

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
SQLCODE NOT EQUAL 0 after selecting KONTEN	Account not found or DB error when reading account limits	Set KTO-RC=0099, KTO-FEHLERMSG='KONTO NICHT GEFUNDEN', then GOBACK (terminate)	Throw or return a specific AccountNotFoundException / set response code 0099; abort further checks

WS-TAGES-UMSATZ > WS-TAGES-LIMIT	Daily transaction total including current booking exceeds account daily limit	Set WS-TAGES-OK='N', KTO-RC=0003, set KTO-FEHLERMSG to include WS-TAGES-LIMIT message	Return validation failure with code 0003 and message; compute daily aggregated sum with DB SUM and include current transaction
LK-BETRAG > WS-EINZEL-LIMIT OR (UEBW and LK-BETRAG > WS-AUSLAND-LIMIT)	Single-transaction or foreign-transfer limit exceeded	Set WS-EINZEL-OK='N', KTO-RC=0003, set appropriate KTO-FEHLERMSG	Return validation failure code 0003; ensure decimal comparisons preserve scale
LK-BETRAG > WS-DISPO-GRENZE	Requested amount exceeds available + overdraft limit	Set WS-DISPO-OK='N', KTO-RC=0001, set KTO-FEHLERMSG 'DISPOKREDITLIMIT UEBERSCHRITTEN'	Return overdraft limit violation with code 0001; compute dispoThreshold = available + dispoCreditLimit

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Implement DB query to load account limits/balances and handle SQLCODE != 0 returning error 0099.

[ ] Implement query to sum today's bookings with status 'GB' and add current transaction amount before comparing to daily limit.

[ ] Implement numeric comparisons using BigDecimal to compare amounts against WS-TAGES-LIMIT, WS-EINZEL-LIMIT, WS-AUSLAND-LIMIT and dispoThreshold.

[ ] Compute dispoThreshold as available + dispoCreditLimit; ensure this matches expected business rule (overrides DB DISPO_LIMIT).

[ ] Return structured result containing return code, error message and account response equivalent to LK-RUECKGABE.

CRITICAL FINDINGS

WS-DISPO-GRENZE is read from the DB (DISPO_LIMIT) then overwritten by COMPUTE
WS-DISPO-GRENZE = KTO-VERFUEGBAR + KTO-DISPOKREDITLIMIT ? potential semantic conflict: initial DB value is ignored after compute.

Multiple checks set KTO-RC and KTO-FEHLERMSG ? later checks may overwrite earlier error messages/codes; last failing check wins unless short-circuited.

Comparisons mix packed decimals and numeric literals (ZEROS) ? ensure migrated comparisons handle sign and scale correctly.

LK-RUECKGABE is populated from KONTO-RUECKGABE (COPY) ? the source/format of KONTO-RUECKGABE must be resolved to map return structure properly.

BUSINESS RULES: KONTOABFR

Type: BATCH_ENTRY | Risk: MEDIUM (57) | 323 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
KONTOABFR	BATCH_ENTRY	MEDIUM (57)	323	No UI ? batch processing

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (17)

- WS-KONTO-NR
- KD-NACHNAME
- KD-VORNAME
- KD-STRASSE
- KD-PLZ
- KD-ADRESSE-SEIT
- KD-TELEFON-PRIVAT
- KD-TELEFON-MOBIL
- ... and 9 more

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

1. Overview

Program Name	KONTOABFR
Type	BATCH_ENTRY
Purpose	Online CICS account inquiry screen: shows account balance, available funds, customer data and transactions for a given account number; handles map send/receive, input validation, SQL reads and paging for transactions.
User Interface	No UI ? batch processing
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Interactive account inquiry (KABF)
Actor / User	Bank teller or online banking user via terminal
Description	User requests account information by entering account number and choosing options (balance, transactions, details, print). The program validates input, reads account/customer data, optionally fetches transactions, and displays results on a CICS map. Navigation supports paging and printing.
Goal	Display accurate account and customer information and transactions and allow printing/paging.
Trigger	User opens account inquiry map or presses a PF-key/ENTER on the KONABFM1 map.

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
WS-KONTO-NR	WS-KONTO-NR (PIC 9(10))	numeric (account number)	Must not be spaces; validated by external program KONTOPRU; SQL lookup requires STATUS <> 'L'
WS-AUSWAHL	WS-AUSWAHL (PIC X(1))	enumeration	88-levels: '1' saldo, '2' transactions, '3' detail, '4' print

EIBAID	EIBAID (CICS AID)	control-key	Mapped to DFHPF1..DFHPF8 and DFHENTER in EVALUATE to dispatch actions
DFHCOMMAREA	DFHCOMMAREA LINKAGE PIC X(200)	opaque commarea	Copied into WS-KOMMUNIKATION on follow-up and restored on return

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Dispatch first or subsequent invocation: EVALUATE TRUE WHEN ERSTER-AUFRUF WHEN FOLGE-AUFRUF WHEN OTHER -> If first call perform 2000-ERSTEN-AUFRUF; if follow-up perform 3000-FOLGE-AUFRUF; otherwise perform 9000-FEHLER-ALLGEMEIN (control-flow, Line 62), 99% confidence
2	Initialize CICS and environment fields: 1000-CICS-INIT / IF EIBCALEN > 0 -> Move EIBTRNID, EIBDATE, EIBTIME, EIBTRMID into WS fields; if EIBCALEN > 0 set WS-ERSTE-AUFRUF = 'N' and copy DFHCOMMAREA into WS-KOMMUNIKATION (initialization, Line 90), 98% confidence
3	Initialize display and working fields on first call: 2100-INIT-FELDER -> Clear KONTO-STAMMDATEN, KONTO-SALDEN, KUNDE-STAMMDATEN; set WS-KONTO-NR = 0; WS-ANZAHL-UMSTZE = 0; WS-SEITE = 1 (initialization, Line 140), 99% confidence
4	Send initial map to terminal and handle response: EXEC CICS SEND MAP('KONABFM1') ... RESP(WS-RESP) / IF WS-RESP NOT EQUAL DFHRESP(NORMAL) -> Send map 'KONABFM1' (erase, cursor). If response not NORMAL set ERR-TEXT and perform 9100-CICS-FEHLER (file-io, Line 150), 98% confidence
5	Receive map on follow-up and handle MAPFAIL: EXEC CICS RECEIVE MAP('KONABFM1') ... RESP(WS-RESP) RESP2(WS-RESP2) / EVALUATE WS-RESP WHEN DFHRESP(MAPFAIL) WHEN OTHER -> If RESP = NORMAL continue; if MAPFAIL perform 2200-MAP-SENDEN; otherwise set error text and perform 9100-CICS-FEHLER (file-io, Line 180), 97% confidence
6	AID handling dispatch: EVALUATE EIBAID WHEN DFHPF1 WHEN DFHPF2 WHEN DFHPF3 WHEN DFHPF4 WHEN DFHPF7 WHEN DFHPF8 WHEN DFHENTER WHEN OTHER -> PF1 or ENTER -> perform KONTOABFRAGE; PF2 -> perform UMSAETZE-ANZEIGEN; PF3 -> program end; PF4 -> start print; PF7 -> page back; PF8 -> page forward; other -> set ERR-TEXT and invalid input handling (control-flow, Line 197), 99% confidence
7	Account inquiry sequence: 4000-KONTOABFRAGE / IF KTO-RC EQUAL ZEROS -> Perform input validation; if KTO-RC = 0 then read account (4020), read customer (4030), format saldo (4040) and display results (4050) (control-flow, Line 220), 98% confidence
8	Account number required validation: IF WS-KONTO-NR-X EQUAL SPACES -> Set KTO-FEHLERMSG='KONTONUMMER EINGEBEN', KTO-RC=0001 and perform 9200-UNGUELTIGE-EINGABE (validation, Line 232), 99% confidence
9	Delegate account number validation to KONTOPRU: CALL 'KONTOPRU' USING KONTO-STAMMDATEN KONTO-RUECKGABE / IF NOT KTO-OK -> Call external validation routine; if return indicates NOT OK perform 9300-VALIDIERUNG-FEHLER (validation, Line 237), 97% confidence
10	Read account data from KONTEN table excluding closed accounts: EXEC SQL SELECT ... FROM KONTEN WHERE KONTO_NR = :WS-KONTO-NR AND STATUS <> 'L' -> Query KONTEN for account fields into KTO-* host variables only when STATUS not equal 'L' (exclude closed accounts) (file-io, Line 250), 99% confidence
11	Handle account SQL results: EVALUATE SQLCODE WHEN 0 WHEN 100 WHEN OTHER -> If SQLCODE=0 set KTO-RC=0000; if 100 set KTO-RC=0001 and KTO-FEHLERMSG='KONTO NICHT GEFUNDEN'; otherwise set KTO-RC=0099, move SQLCODE to ERR-SQLCODE and perform 9400-SQL-FEHLER (validation, Line 265), 99% confidence

BUSINESS RULES: KONTOABFR (continued)

Type: BATCH_ENTRY | Risk: MEDIUM (57)

12	Read customer data and error on SQL failure: EXEC SQL SELECT ... FROM KUNDEN WHERE KUNDEN_NR = :KTO-INH-KDNR / IF SQLCODE NOT EQUAL 0 -> Select customer record into KD-* variables; if SQLCODE <> 0 move SQLCODE to ERR-SQLCODE and perform 9400-SQL-FEHLER (file-io, Line 280), 98% confidence
13	Format saldo and available amount for display: 4040-SALDO-FORMATIEREN (MOVE KTO-SALDO TO WS-BETRAG-WS ... MOVE WS-BETRAG-WS TO WS-SALDO-ANZEIGE) -> Move packed/COMP-3 KTO-SALDO into WS-BETRAG-WS then to WS-SALDO-ANZEIGE for formatted presentation; same for KTO-VERFUEGBAR to WS-VERFUEG-ANZEIGE (data-transformation, Line 305), 98% confidence
14	Display results via CICS SEND MAP DATAONLY: EXEC CICS SEND MAP('KONABFM1') ... DATAONLY RESP(WS-RESP) -> Send data-only map to update screen with account/customer/Saldo/transactions data and check RESP (file-io, Line 315), 97% confidence
15	Retrieve transactions via BUCHSERV and error handling: CALL 'BUCHSERV' USING WS-KONTO-NR WS-DATUM-VON WS-DATUM-BIS WS-SEITE WS-UMSATZ-TABELLE WS-ANZAHL-UMSTZE BUCHUNGS-RUECKGABE / IF NOT BCH-OK -> Call transaction service to populate WS-UMSATZ-TABELLE and WS-ANZAHL-UMSTZE; if BCH-OK false perform 9000-FEHLER-ALLGEMEIN; afterwards perform 4050-ERGEBNIS-ANZEIGEN (file-io, Line 330), 96% confidence
16	Start print job for account: EXEC CICS START TRANSID('KDRK') FROM(WS-KONTO-NR) LENGTH(10) RESP(WS-RESP) -> Initiate asynchronous transaction 'KDRK' passing WS-KONTO-NR as payload to produce printout (file-io, Line 360), 95% confidence
17	Page back navigation: IF WS-SEITE > 1 -> If current page > 1 subtract 1 from WS-SEITE and perform 4100-UMSAETZE-ANZEIGEN to reload transactions for previous page (control-flow, Line 370), 98% confidence
18	Page forward navigation with max-page guard: IF WS-SEITE < WS-MAX-SEITEN -> If current page less than WS-MAX-SEITEN add 1 to WS-SEITE and perform 4100-UMSAETZE-ANZEIGEN to load next page (control-flow, Line 380), 97% confidence
19	Program return to caller with COMM area: 8000-PROGRAMM-ENDE / EXEC CICS RETURN TRANSID('KABF') COMMAREA(DFHCOMMAREA) LENGTH(200) -> Copy WS-KOMMUNIKATION back to DFHCOMMAREA and return to CICS with transaction id 'KABF' returning 200 bytes of commarea (termination, Line 400), 98% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- Terminal map KONABFM1 -> WS-KONTO-NR, EIBAID, WS-DATUM-VON, WS-DATUM-BIS, WS-SEITE	- Validate account via KONTOPRU; SQL SELECT from KONTEN (exclude STATUS='L'); SQL SELECT from KUNDEN; format amounts into display fields; call BUCHSERV to fetch transactions; handle paging and print start	- Send updated KONABFM1 map (DATAONLY) to terminal; send TEXT on general error; CICS START to 'KDRK' for printing; return commarea via EXEC CICS RETURN

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Field mapping not available from static analysis. Provide source files for detailed mapping.

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
WS-RESP NOT EQUAL DFHRESP(NORMAL) after SEND MAP	CICS map send failed	Set ERR-TEXT='MAP SEND FEHLER' and perform 9100-CICS-FEHLER which sends error text and returns	Throw MapSendException, log RESP codes, show user-friendly message and abort request

BUSINESS RULES: KONTOABFR (continued)

Type: BATCH_ENTRY | Risk: MEDIUM (57)

WS-RESP not NORMAL after RECEIVE MAP and not MAPFAIL	Unexpected map receive error	Set ERR-TEXT='MAP RECEIVE FEHLER' and perform 9100-CICS-FEHLER	Handle as receive error: log RESP/RESP2 and show error page; possibly re-render input form
WS-KONTO-NR-X EQUAL SPACES	No account number entered	Set KTO-FEHLERMSG and KTO-RC=0001 then perform 9200-UNGUELTIGE-EINGABE to resend map with alarm	Return validation error to UI (400 Bad Request) indicating account number required; highlight field
KONTOPRU returns NOT OK (NOT KTO-OK)	Account validation failed	Perform 9300-VALIDIERUNG-FEHLER which moves validation msg to error text and performs 9200-UNGUELTIGE-EINGABE	Call accountValidationService; on validation failure return validation errors; do not call DB reads
SQLCODE = 100 after account SELECT	Account not found	Set KTO-RC=0001 and KTO-FEHLERMSG='KONTO NICHT GEFUNDEN'	Map to 404/empty result: return 'account not found' message to user
SQLCODE other than 0 or 100 after account SELECT or any non-zero after customer SELECT	SQL/database error	Move SQLCODE to ERR-SQLCODE, set KTO-RC=0099 and perform 9400-SQL-FEHLER which sets ERR-TEXT and then 9000-FEHLER-ALLGEMEIN	Throw DataAccessException including SQL error code; present generic error page and log details
CALL 'BUCHSERV' returns not BCH-OK	Transaction service failed	Perform 9000-FEHLER-ALLGEMEIN	Handle transactionService failure: log and show error to user; fallback to showing account details without transactions
Attempt to page back when WS-SEITE <= 1	No previous page	Do nothing (page not changed)	Guard paging: if page <=1 do not decrement; return current page unchanged
Attempt to page forward when WS-SEITE >= WS-MAX-SEITEN	No next page	Do nothing (page not changed)	Guard paging using known maxPages; prevent increment and indicate end of pages to UI
UNKNOWN AID (WHEN OTHER in AID handling)	Unrecognized key pressed	Set ERR-TEXT='UNBEKANNTE TASTE' and perform 9200-UNGUELTIGE-EINGABE	Return invalid action error, keep user on same screen with message

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Create REST endpoints: GET /accounts/{accountId} for account details and POST /accounts/{accountId}/transactions/query for transaction pages.

[ ] Implement AccountService with methods validateAccount(accountId) calling existing validation logic (KONTOPRU) and throwing ValidationException for errors.

[ ] Implement DAO layer to execute SELECT from KONTEN with condition STATUS <> 'L' and map results to Account DTO; handle SQL 'not found' and other exceptions distinctly.

[ ] Implement TransactionService to encapsulate BUCHSERV behavior and return paginated results along with computed WS-MAX-SEITEN.

[ ] Implement error mapping: map legacy RC codes and ERR-TEXT to HTTP status codes and structured error responses; log SQLCODE and internal IDs for support.

CRITICAL FINDINGS

WS-MAX-SEITEN is initialized to 0 and never set in this program; page-forward guard (WS-SEITE < WS-MAX-SEITEN) may prevent any forward paging unless set by BUCHSERV or another component.

KONTOPRU is an external synchronous CALL used for validation ? migrating requires reimplementing and testing validation semantics, return codes and side effects.

BUSINESS RULES: BENACHRI

Type: SUBROUTINE | Risk: MEDIUM (42) | 132 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
BENACHRI	SUBROUTINE	MEDIUM (42)	132	No UI ? batch processing

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (11)

- WS-EMAIL-AKTIV
- LK-KONTO-NR
- KD-NACHNAME
- KD-VORNAME
- KD-STRASSE
- KD-PLZ
- KD-ADRESSE-SEIT
- KD-TELEFON-PRIVAT
- ... and 3 more

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

1. Overview

Program Name	BENACHRI
Type	SUBROUTINE
Purpose	Notify bank customers about account bookings by reading customer preferences and sending SMS, Email or Push notifications when appropriate.
User Interface	No UI ? batch processing
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Customer notification on account booking
Actor / User	Calling service (BUCHSERV) / Batch or transaction processor
Description	When a booking occurs on an account, call BENACHRI with account number, booking amount and booking type; BENACHRI reads customer contact/preferences and enqueues SMS, Email and/or Push notifications according to preferences and threshold.
Goal	Inform the customer of account activity via preferred channels, respecting a configurable monetary threshold.
Trigger	A booking event (call from BUCHSERV) providing LK-KONTO-NR, LK-BETRAG and LK-BUCHUNGSTYP

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
LK-KONTO-NR	none (linkage)	PIC 9(10) (account number)	Used to lookup customer and preferences in DB; no explicit format validation in code
LK-BETRAG	LK-BETRAG	PIC S9(13)V99 COMP-3 (signed packed decimal amount)	Compared to WS-SCHWELLWERT to determine suppression of SMS and Push; no other validation
LK-BUCHUNGSTYP	LK-BUCHUNGSTYP	PIC X(4) (booking type code)	Used verbatim in message text; no validation of allowed values

BUSINESS RULES: BENACHRI (continued)

Type: SUBROUTINE | Risk: MEDIUM (42)

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Program input parameters: PROCEDURE DIVISION USING LK-KONTO-NR LK-BETRAG LK-BUCHUNGSTYP -> Program expects LK-KONTO-NR, LK-BETRAG and LK-BUCHUNGSTYP as input linkage parameters and uses them for processing and message composition (initialization, Line 34), 99% confidence
2	Startup initialization of error and return code: PERFORM 1000-INITIALISIERUNG -> MOVE WS-PROGRAMM-NAME TO ERR-PROGRAMM and MOVE ZEROS TO KD-RC (initializes error context and return code) (initialization, Line 38), 98% confidence
3	Default channel activation flags: WS-KANAL-STEUERUNG definitions -> WS-SMS-AKTIV, WS-EMAIL-AKTIV and WS-PUSH-AKTIV are initialized to 'N' in working-storage before database read (initialization, Line 12), 98% confidence
4	Read customer and preference data from database: EXEC SQL SELECT ... FROM KONTEN KT JOIN KUNDEN K LEFT JOIN NACHRICHT_PRAEFERENZEN NP WHERE KT.KONTO_NR = :LK-KONTO-NR END-EXEC -> Populate KD-NACHNAME, KD-VORNAME, KD-EMAIL, KD-TELEFON-MOBIL and channel flags WS-SMS-AKTIV, WS-EMAIL-AKTIV, WS-PUSH-AKTIV and WS-SCHWELLWERT from the database for the given account number (file-io, Line 46), 96% confidence
5	Database NVL defaults for missing preferences: NVL(NP.SMS_AKTIV, 'N'), NVL(NP.EMAIL_AKTIV, 'J'), NVL(NP.PUSH_AKTIV, 'N'), NVL(NP.SCHWELLWERT, 500.00) in SELECT -> If preference fields are NULL in DB then defaults are applied: SMS='N', EMAIL='J', PUSH='N', SCHWELLWERT=500.00 and these values are moved into corresponding working-storage variables (validation, Line 50), 98% confidence
6	Abort on SQL error when reading preferences: IF SQLCODE NOT EQUAL 0 -> Perform GOBACK immediately (exit program) and do not proceed to threshold checks or sending notifications (control-flow, Line 62), 99% confidence
7	Threshold suppression of SMS and Push: IF LK-BETRAG < WS-SCHWELLWERT -> MOVE 'N' TO WS-SMS-AKTIV and MOVE 'N' TO WS-PUSH-AKTIV (disables SMS and Push for bookings below threshold) (validation, Line 74), 99% confidence
8	Conditional SMS send: IF WS-SMS-AKTIV EQUAL 'J' PERFORM 4000-SMS-SENDEN -> If WS-SMS-AKTIV equals 'J' then perform SMS creation and enqueue routine 4000-SMS-SENDEN (control-flow, Line 42), 98% confidence
9	Conditional Email send: IF WS-EMAIL-AKTIV EQUAL 'J' PERFORM 4100-EMAIL-SENDEN -> If WS-EMAIL-AKTIV equals 'J' then perform Email subject/body creation and enqueue routine 4100-EMAIL-SENDEN (control-flow, Line 43), 98% confidence
10	Conditional Push send: IF WS-PUSH-AKTIV EQUAL 'J' PERFORM 4200-PUSH-SENDEN -> If WS-PUSH-AKTIV equals 'J' then perform Push message enqueue routine 4200-PUSH-SENDEN (control-flow, Line 44), 98% confidence
11	Amount formatting for messages: MOVE LK-BETRAG TO WS-BETRAG-FORMAT -> Convert numeric LK-BETRAG into display format WS-BETRAG-FORMAT for inclusion in message text (data-transformation, Line 80), 95% confidence
12	Compose SMS body text: STRING 'Buchung ' LK-BUCHUNGSTYP ': ' WS-BETRAG-FORMAT ' EUR auf Konto ' LK-KONTO-NR DELIMITED SIZE INTO WS-TEXT -> Create WS-TEXT containing booking type, formatted amount with EUR and account number to be used as SMS/Push/Email body (data-transformation, Line 82), 99% confidence
13	Insert SMS into notification queue: EXEC SQL INSERT INTO NACHRICHTEN_QUEUE ... VALUES ('SMS', :KD-TELEFON-MOBIL, :WS-TEXT, 1, CURRENT TIMESTAMP) END-EXEC -> Enqueue a record of type 'SMS' with recipient KD-TELEFON-MOBIL, body WS-TEXT, priority 1 and TIMESTAMP_EINGEST set to CURRENT TIMESTAMP (file-io, Line 88), 98% confidence

BUSINESS RULES: BENACHRI (continued)

Type: SUBROUTINE | Risk: MEDIUM (42)

14	Compose Email subject: STRING 'Kontoauszug ' LK-KONTO-NR ': ' 'Neue Buchung eingegangen' DELIMITED SIZE INTO WS-BETREFF -> Create WS-BETREFF for email subject containing account number and fixed phrase (data-transformation, Line 100), 99% confidence
15	Insert Email into notification queue: EXEC SQL INSERT INTO NACHRICHTEN_QUEUE ... VALUES ('EMAIL', :KD-EMAIL, :WS-BETREFF, :WS-TEXT, 2, CURRENT TIMESTAMP) END-EXEC -> Enqueue a record of type 'EMAIL' with recipient KD-EMAIL, subject WS-BETREFF, body WS-TEXT, priority 2 and TIMESTAMP_EINGEST set to CURRENT TIMESTAMP (file-io, Line 106), 98% confidence
16	Insert Push into notification queue: EXEC SQL INSERT INTO NACHRICHTEN_QUEUE ... VALUES ('PUSH', :KD-KUNDENNUMMER, :WS-TEXT, 1, CURRENT TIMESTAMP) END-EXEC -> Enqueue a record of type 'PUSH' with recipient KD-KUNDENNUMMER, body WS-TEXT, priority 1 and TIMESTAMP_EINGEST set to CURRENT TIMESTAMP (file-io, Line 116), 90% confidence
17	Notification sending order: Sequence in 0000-HAUPTPROGRAMM: perform 1000, 2000, 3000 then IF checks for SMS, EMAIL, PUSH in that order -> Notifications are considered in a fixed order: SMS is evaluated first, then EMAIL, then PUSH; any that are active will be enqueued in that sequence (control-flow, Line 40), 97% confidence
18	Email uses WS-TEXT as body (may be empty): EMAIL INSERT uses :WS-TEXT in VALUES clause regardless of whether SMS routine previously populated it -> Email INHALT field is set to WS-TEXT; if WS-TEXT was not created (e.g. SMS not active) the email body may be empty or default null as passed to DB (data-transformation, Line 106), 92% confidence
19	Timestamp assignment on enqueue: INSERT statements use CURRENT TIMESTAMP in VALUES clauses -> All queued notification records receive current timestamp at insert time in TIMESTAMP_EINGEST (date-time, Line 88), 99% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- LK-KONTO-NR, LK-BETRAG, LK-BUCHUNGSTYP are passed in linkage to the program	- Initialize program error context and return code - Read customer and preference data from DB using LK-KONTO-NR - Apply NVL defaults from SELECT if DB fields are NULL - Compare LK-BETRAG to WS-SCHWELLWERT and disable SMS/Push if below threshold - Compose WS-TEXT and WS-BETREFF strings for messages - Conditionally INSERT notification records into NACHRICHTEN_QUEUE for SMS, EMAIL, PUSH	- INSERT records into NACHRICHTEN_QUEUE table with fields TYP, EMPFAENGER, BETREFF/INHALT, PRIORITAET and TIMESTAMP_EINGEST - No direct DISPLAY output; program terminates with GOBACK

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Field mapping not available from static analysis. Provide source files for detailed mapping.

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
SQLCODE NOT EQUAL 0 after preference SELECT	Database read failed or no matching account/DB error	Program executes GOBACK immediately and does not attempt to send notifications	Throw an exception or return an error status to caller; do not proceed to notification logic; log SQL error details

KD-EMAIL or KD-TELEFON-MOBIL or KD-KUNDENNUMMER is NULL/empty	No contact information available for a requested notification channel	Program still attempts to INSERT into queue using the NULL/empty recipient value (DB will receive NULL or empty string) ? no explicit guard in COBOL	Validate recipient fields before enqueue; if missing, skip enqueue and log a warning or route to error handling to avoid inserting invalid recipient rows
WS-TEXT not populated when EMAIL active	Email INHALT may be empty because WS-TEXT is only created in SMS routine	EMAIL INSERT uses WS-TEXT as body even if SMS routine wasn't run; result may be empty email body	Ensure email body is composed independently of SMS path; create message body before conditional sends

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Confirm and import COPY KUNDSTRKC and ERRHANDKC to identify KD-KUNDENNUMMER and KD-RC field definitions

[ ] Implement or map DB SELECT and INSERT operations in Java using parameterized queries and handle SQL errors/exceptions explicitly

[ ] Create precise mapping for COMP-3 packed decimals to BigDecimal with scale 2; implement amount formatting utility equivalent to WS-BETRAG-FORMAT

[ ] Refactor message body construction so each channel composes its own content or ensure WS-TEXT is created prior to conditional sends if intended

[ ] Add validation for recipient fields and skip or route invalid enrollments to an error queue instead of inserting NULL recipients

[ ] Decide and implement transactional semantics for multiple notification inserts (single transaction vs independent inserts)

[ ] Maintain default preference behavior: apply DB NVL defaults or set application defaults consistent with original SQL NVL semantics

[ ] Add logging and monitoring around SQL errors and queue inserts to allow operational visibility after migration

[ ] Create automated tests for threshold behavior (LK-BETRAG < WS-SCHWELLWERT), channel activation permutations and missing contact data

[ ] Document priority and timestamp mapping and ensure the target NACHRICHTEN_QUEUE schema is compatible

CRITICAL FINDINGS

KD-KUNDENNUMMER is referenced in 4200-PUSH-SENDEN but not declared in the displayed linkage or working-storage; it may be provided by an included copy (COPY KUNDSTRKC) ? verify presence and correct population

WS-TEXT is created in the SMS routine and reused for Email and Push; if SMS is not active WS-TEXT may be empty leading to empty email/push bodies

No validation or guard exists for NULL contact information before inserting into NACHRICHTEN_QUEUE; DB may receive records with NULL recipient fields

WS-EMAIL-AKTIV has a working-storage default of 'N' but the SELECT NVL default is 'J' ? email may be sent by default if DB has no preference row, which could cause unexpected emails

No explicit error handling for failed INSERTs into NACHRICHTEN_QUEUE (no SQLCODE checks after INSERT statements)

BUSINESS RULES: KTOSTRKC

Type: COPYBOOK | Risk: LOW (13) | 52 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
KTOSTRKC	COPYBOOK	LOW (13)	52	No UI ? shared copybook str...

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (3)

- KTO-IBAN
- KTO-INH-NAME
- KTO-INH-VORNAME

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

---

1. Overview

Program Name	KTOSTRKC
Type	COPYBOOK
Purpose	This copybook defines the account (KONTO) data structure for retail banking account management (giro, savings, fixed deposit, credit). It is used by programs such as KONTOABFR, BUCHSERV, ZINSBERCH, KONTOPRUEF and UEBERWEIG to provide a consistent in-memory layout for account master data, balances and return/error information for processing and messaging.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY KTOSTRKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
KONTO-STAMMDATEN	01 KONTO-STAMMDATEN	group	group container for account master data
KTO-NUMMER	05 within KONTO-STAMMDATEN	PIC 9(10)	numeric account number, up to 10 digits
KTO-BLZ	05 within KONTO-STAMMDATEN	PIC 9(8)	bank sort code (BLZ), 8 digits
KTO-IBAN	05 within KONTO-STAMMDATEN	PIC X(22)	IBAN as 22-character string (stored fixed length)
KTO-BIC	05 within KONTO-STAMMDATEN	PIC X(11)	BIC as 11-character string
KTO-INHABER	05 within KONTO-STAMMDATEN	group	group for account holder identity
KTO-INH-NAME	10 within KTO-INHABER	PIC X(40)	holder family name up to 40 chars
KTO-INH-VORNAME	10 within KTO-INHABER	PIC X(25)	holder given name up to 25 chars
KTO-INH-KDNR	10 within KTO-INHABER	PIC 9(10)	customer number, numeric up to 10 digits

BUSINESS RULES: KTOSTRKC (continued)
Type: COPYBOOK | Risk: LOW (13)

KTO-TYP	05 within KONTO-STAMMDATEN	PIC X(2)	88-levels: KTO-GIROKONTO='GK', KTO-SPARKONTO='SK', KTO-FESTGELD='FG', KTO-KREDITKONTO='KK'
KTO-WAEHRUNG	05 within KONTO-STAMMDATEN	PIC X(3)	currency 3-char code; 88-levels: KTO-EUR='EUR', KTO-USD='USD', KTO-CHF='CHF'
KTO-STATUS	05 within KONTO-STAMMDATEN	PIC X(1)	status flag; 88-levels: KTO-AKTIV='A', KTO-GESPERRT='G', KTO-GELOESCHT='L', KTO-PFAENDUNG='P'
KONTO-SALDEN	01 KONTO-SALDEN	group	group container for balance and limits
KTO-SALDO	05 within KONTO-SALDEN	PIC S9(13)V99 COMP-3	signed packed decimal, implied 2 decimals (balance)
KTO-VERFUEGBAR	05 within KONTO-SALDEN	PIC S9(13)V99 COMP-3	signed packed decimal, implied 2 decimals (available amount)
KTO-DISPOKREDITLIMIT	05 within KONTO-SALDEN	PIC S9(11)V99 COMP-3	signed packed decimal, implied 2 decimals (overdraft limit)
KTO-GUTHABENGRENZE	05 within KONTO-SALDEN	PIC S9(11)V99 COMP-3	signed packed decimal, implied 2 decimals (credit ceiling)
KTO-SOLLZINS-SATZ	05 within KONTO-SALDEN	PIC 9(3)V9(4) COMP-3	interest rate charged (debit) with 4 decimal places (scale=4)
KTO-HABENZINS-SATZ	05 within KONTO-SALDEN	PIC 9(3)V9(4) COMP-3	interest rate paid (credit) with 4 decimal places (scale=4)
KTO-LETZTER-UMSATZ	05 within KONTO-SALDEN	PIC X(8)	last transaction date or identifier as 8-char string (likely YYYYMMDD)
KTO-EROEFFFNUNGSDATUM	05 within KONTO-SALDEN	PIC X(8)	account opening date as 8-char string (likely YYYYMMDD)
KTO-KUENDIGUNGSDATUM	05 within KONTO-SALDEN	PIC X(8)	account termination date as 8-char string (likely YYYYMMDD) or blanks
KONTO-RUECKGABE	01 KONTO-RUECKGABE	group	group for return/response metadata
KTO-RC	05 within KONTO-RUECKGABE	PIC 9(4)	numeric return code; 88-levels: KTO-OK=0000, KTO-NICHT-GEFUNDEN=0001, KTO-GESPERRT-RC=0002, KTO-LIMIT-UEBER=0003, KTO-DB-FEHLER=0099
KTO-FEHLERMSG	05 within KONTO-RUECKGABE	PIC X(80)	free-form error message up to 80 chars

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Account type domain: 05 KTO-TYP PIC X(2); 88 values: 'GK','SK','FG','KK' -> KTO-TYP must be one of the defined codes (GK=Giro, SK=Sparkonto, FG=Festgeld, KK=Kreditkonto). Reject or map otherwise. (data-constraint, Line 12), 95% confidence

BUSINESS RULES: KTOSTRKC (continued)

Type: COPYBOOK | Risk: LOW (13)

2	Currency domain: 05 KTO-WAEHRUNG PIC X(3); 88 values: 'EUR','USD','CHF' -> KTO-WAEHRUNG must be a supported 3-letter currency code; treat unknown codes as invalid or require normalization. (data-constraint, Line 16), 90% confidence
3	Account status domain: 05 KTO-STATUS PIC X(1); 88 values: 'A','G','L','P' -> KTO-STATUS indicates lifecycle state (A=active, G=blocked, L=deleted, P=attached/levied). Business logic depends on this flag. (data-constraint, Line 19), 95% confidence
4	Monetary precision: COMP-3 fields: S9(13)V99 and S9(11)V99 -> Amounts have 2 decimal places implied; map to BigDecimal with scale=2 and adequate precision to avoid rounding errors. (data-format, Line 24), 98% confidence
5	Interest rate format: PIC 9(3)V9(4) COMP-3 for interest rates -> Interest rates have 4 decimal places (scale=4) and up to 3 integer digits; map to BigDecimal scale=4. (data-format, Line 28), 98% confidence
6	Return codes: KTO-RC PIC 9(4) with 88-levels: 0000,0001,0002,0003,0099 -> Interpret numeric return code to determine success or specific error conditions; propagate or translate to exceptions/messages. (business-rule, Line 37), 95% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
KONTO-STAMMDATEN	01 KONTO-STAMMDATEN	?	?	KontoStammdatenDto kontoStammdaten
KTO-NUMMER	05 within KONTO-STAMMDATEN (PIC 9(10))	?	?	Long ktoNummer
KTO-BLZ	05 within KONTO-STAMMDATEN (PIC 9(8))	?	?	Integer ktoBlz
KTO-IBAN	05 within KONTO-STAMMDATEN (PIC X(22))	?	?	String ktoIban
KTO-BIC	05 within KONTO-STAMMDATEN (PIC X(11))	?	?	String ktoBic
KTO-INHABER	05 within KONTO-STAMMDATEN	?	?	KtoInhaberDto ktoInhaber
KTO-INH-NAME	10 within KTO-INHABER (PIC X(40))	?	?	String ktoInhName
KTO-INH-VORNAME	10 within KTO-INHABER (PIC X(25))	?	?	String ktoInhVorname
KTO-INH-KDNR	10 within KTO-INHABER (PIC 9(10))	?	?	Long ktoInhKdnr
KTO-TYP	05 within KONTO-STAMMDATEN (PIC X(2))	?	?	String ktoTyp // allowed values: GK, SK, FG, KK
KTO-WAEHRUNG	05 within KONTO-STAMMDATEN (PIC X(3))	?	?	String ktoWaehrung // ISO currency code

BUSINESS RULES: KTOSTRKC (continued)
Type: COPYBOOK | Risk: LOW (13)

KTO-STATUS	05 within KONTO-STAMMDATEN (PIC X(1))	?	?	String ktoStatus // A/G/L/P
KONTO-SALDEN	01 KONTO-SALDEN	?	?	KontoSaldenDto kontoSalden
KTO-SALDO	05 within KONTO-SALDEN (PIC S9(13)V99 COMP-3)	?	?	java.math.BigDecimal ktoSaldo // scale=2
KTO-VERFUEGBAR	05 within KONTO-SALDEN (PIC S9(13)V99 COMP-3)	?	?	java.math.BigDecimal ktoVerfuegbar // scale=2
KTO-DISPOKREDITLIMIT	05 within KONTO-SALDEN (PIC S9(11)V99 COMP-3)	?	?	java.math.BigDecimal ktoDispoKreditLimit // scale=2
KTO-GUTHABENGRENZE	05 within KONTO-SALDEN (PIC S9(11)V99 COMP-3)	?	?	java.math.BigDecimal ktoGuthabenGrenze // scale=2
KTO-SOLLZINS-SATZ	05 within KONTO-SALDEN (PIC 9(3)V9(4) COMP-3)	?	?	java.math.BigDecimal ktoSollzinsSatz // scale=4
KTO-HABENZINS-SATZ	05 within KONTO-SALDEN (PIC 9(3)V9(4) COMP-3)	?	?	java.math.BigDecimal ktoHabenzinsSatz // scale=4
KTO-LETZTER-UMSATZ	05 within KONTO-SALDEN (PIC X(8))	?	?	String ktoLetzterUmsatz // YYYYMMDD or free-form 8 chars
KTO-EROEFFFNUNGSDATUM	05 within KONTO-SALDEN (PIC X(8))	?	?	String ktoEroeffnungsdatum // YYYYMMDD
KTO-KUENDIGUNGSDATUM	05 within KONTO-SALDEN (PIC X(8))	?	?	String ktoKuendigungsdatum // YYYYMMDD or blanks
KONTO-RUECKGABE	01 KONTO-RUECKGABE	?	?	KontoRueckgabeDto kontoRueckgabe
KTO-RC	05 within KONTO-RUECKGABE (PIC 9(4))	?	?	Integer ktoRc // interpret with defined 88-level codes
KTO-FEHLERMSG	05 within KONTO-RUECKGABE (PIC X(80))	?	?	String ktoFehlerMsg

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
0000	Successful lookup/operation		
0001	Account not found		
0002	Account is blocked		
0003	Limit exceeded		
0099	Database or internal error		

BUSINESS RULES: KTOSTRKC (continued)

Type: COPYBOOK | Risk: LOW (13)

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Define Java DTOs according to migrationHints and validate with sample data.

[ ] Implement conversion utilities for COMP-3 -> BigDecimal and for PIC numeric strings -> Long/Integer.

[ ] Create enums for KTO-TYP, KTO-WAEHRUNG, KTO-STATUS and mapping unit tests.

[ ] Add integration tests for serialization/deserialization between COBOL binary layout and Java DTO (if reading COBOL files).

CRITICAL FINDINGS

Dates are stored as PIC X(8) (likely YYYYMMDD) but not enforced; conversion errors possible if format differs.

Monetary fields are COMP-3 packed; migration must correctly unpack signed values and apply scale.

No explicit sign or overflow handling in copybook: application logic must enforce limits (e.g., S9(13)V99).

Fixed-length text fields (IBAN/BIC/names) may contain trailing spaces; trimming/normalization needed in Java.

BUSINESS RULES: KUNDSTRKC

Type: COPYBOOK | Risk: LOW (13) | 62 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
KUNDSTRKC	COPYBOOK	LOW (13)	62	No UI ? shared copybook str...

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (8)

- KD-NACHNAME
- KD-VORNAME
- KD-STRASSE
- KD-PLZ
- KD-ADRESSE-SEIT
- KD-TELEFON-PRIVAT
- KD-TELEFON-MOBIL
- KD-EMAIL

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

---

1. Overview

Program Name	KUNDSTRKC
Type	COPYBOOK
Purpose	This copybook defines the customer master and related CRM account-relationship data (name, address, contact, credit and segment info) used by multiple banking programs. It is reused by programs like KONTOABFR, UEBERWEIG, BUCHSERV, AUSKUNFT and KREDITPRU to provide a consistent on-wire/in-memory layout for customer data and return codes.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY KUNDSTRKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
KUNDE-STAMMDATEN	01 KUNDE-STAMMDATEN	group	group container for customer master fields
KD-KUNDENNUMMER	05 in KUNDE-STAMMDATEN	PIC 9(10)	numeric 10 digits
KD-ANREDE	05 in KUNDE-STAMMDATEN	PIC X(10)	text up to 10 chars (title/salutation)
KD-TITEL	05 in KUNDE-STAMMDATEN	PIC X(15)	text up to 15 chars
KD-NACHNAME	05 in KUNDE-STAMMDATEN	PIC X(40)	text up to 40 chars (surname)
KD-VORNAME	05 in KUNDE-STAMMDATEN	PIC X(25)	text up to 25 chars (given name)
KD-GEBURTSDATUM	05 in KUNDE-STAMMDATEN	PIC X(8)	date as 8-char string (format not enforced here)

BUSINESS RULES: KUNDSTRKC (continued)

Type: COPYBOOK | Risk: LOW (13)

KD-GEBURTSORT	05 in KUNDE-STAMMDATEN	PIC X(40)	text up to 40 chars (place of birth)
KD-STAATSANGEHOER	05 in KUNDE-STAMMDATEN	PIC X(3)	3-char country code
KD-AUSWEIS-NR	05 in KUNDE-STAMMDATEN	PIC X(20)	ID document number up to 20 chars
KUNDE-ADRESSE	01 KUNDE-ADRESSE	group	group container for address fields
KD-STRASSE	05 in KUNDE-ADRESSE	PIC X(50)	street name up to 50 chars
KD-HAUSNUMMER	05 in KUNDE-ADRESSE	PIC X(10)	house number up to 10 chars
KD-PLZ	05 in KUNDE-ADRESSE	PIC X(10)	postal code up to 10 chars
KD-ORT	05 in KUNDE-ADRESSE	PIC X(40)	city up to 40 chars
KD-LAND	05 in KUNDE-ADRESSE	PIC X(3)	3-char country code
KD-ADRESSE-SEIT	05 in KUNDE-ADRESSE	PIC X(8)	since-date as 8-char string
KUNDE-KONTAKT	01 KUNDE-KONTAKT	group	group container for contact fields
KD-TELEFON-PRIVAT	05 in KUNDE-KONTAKT	PIC X(20)	private phone up to 20 chars
KD-TELEFON-MOBIL	05 in KUNDE-KONTAKT	PIC X(20)	mobile phone up to 20 chars
KD-EMAIL	05 in KUNDE-KONTAKT	PIC X(60)	email address up to 60 chars
KD-KONTAKT-ERLAUBT	05 in KUNDE-KONTAKT	PIC X(1)	flag allowed values via 88-levels: 'J' or 'N'
KD-KONTAKT-JA	88 in KUNDE-KONTAKT (relates to KD-KONTAKT-ERLAUBT)	VALUE 'J'	enumerated value representing yes
KD-KONTAKT-NEIN	88 in KUNDE-KONTAKT (relates to KD-KONTAKT-ERLAUBT)	VALUE 'N'	enumerated value representing no
KUNDE-BONITAET	01 KUNDE-BONITAET	group	group container for creditworthiness fields
KD-SCHUFA-SCORE	05 in KUNDE-BONITAET	PIC 9(3)	numeric 3 digits (SCHUFA score)
KD-RATING-INTERN	05 in KUNDE-BONITAET	PIC X(2)	internal rating codes; allowed values via 88-levels AA/BB/CC/DD
KD-RATING-AAA	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'AA'	rating constant
KD-RATING-GUT	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'BB'	rating constant
KD-RATING-OK	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'CC'	rating constant
KD-RATING-SCHLECHT	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'DD'	rating constant
KD-EINKOMMEN-NETTO	05 in KUNDE-BONITAET	PIC S9(9)V99 COMP-3	signed packed decimal with 2 implied decimals; scale 2
KD-BESTEHENDEVERBINDL	05 in KUNDE-BONITAET	PIC S9(9)V99 COMP-3	signed packed decimal with 2 implied decimals; scale 2
KUNDE-SEGMENT	01 KUNDE-SEGMENT	group	group container for segmentation and branch fields
KD-SEGMENTCODE	05 in KUNDE-SEGMENT	PIC X(4)	segment code - allowed values via 88-levels PRIV/PREM/GSCH/VVIP

BUSINESS RULES: KUNDSTRKC (continued)
Type: COPYBOOK | Risk: LOW (13)

KD-PRIVATKUNDE	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'PRIV'	segment constant
KD-PREMIUM	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'PREM'	segment constant
KD-GESCHAEFTSKD	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'GSCH'	segment constant
KD-VIP	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'VVIP'	segment constant
KD-BETREUER-NR	05 in KUNDE-SEGMENT	PIC 9(6)	numeric 6 digits (account manager id)
KD-FILIALE-NR	05 in KUNDE-SEGMENT	PIC 9(4)	numeric 4 digits (branch number)
KUNDE-RUECKGABE	01 KUNDE-RUECKGABE	group	group container for return code and message
KD-RC	05 in KUNDE-RUECKGABE	PIC 9(4)	4-digit return code; mapped 88-levels define semantic codes
KD-OK	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0000	return code OK
KD-NICHT-GEFUNDEN	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0001	return code not found
KD-GESPERRT	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0002	return code blocked
KD-DB-FEHLER	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0099	return code database error
KD-FEHLERMSG	05 in KUNDE-RUECKGABE	PIC X(80)	error message text up to 80 chars

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	KD-KUNDENNUMMER numeric length: PIC 9(10) -> Customer number must be a 10-digit numeric identifier; store as numeric (no sign, fixed width) (data-constraint, Line 7), 90% confidence
2	KD-KONTAKT-ERLAUBT allowed values: PIC X(1) with 88-levels KD-KONTAKT-JA='J' and KD-KONTAKT-NEIN='N' -> Contact allowed flag must be 'J' or 'N' (use validation during input/output) (data-constraint, Line 20), 95% confidence
3	KD-RATING-INTERN enumerated values: PIC X(2) with 88-levels AA|BB|CC|DD -> Internal rating must be one of 'AA','BB','CC','DD' representing qualitative ratings (data-constraint, Line 27), 95% confidence
4	Packed decimal financial fields: PIC S9(9)V99 COMP-3 -> EINKOMMEN-NETTO and BESTEHENDEVERBINDL are signed packed decimals with 2 implied decimals; map to BigDecimal with scale 2 (data-constraint, Line 30), 95% confidence
5	KD-SEGMENTCODE enumerated values: PIC X(4) with 88-levels PRIV|PREM|GSCH|VVIP -> Segment code must be one of defined codes indicating customer segment (data-constraint, Line 35), 95% confidence

BUSINESS RULES: KUNDSTRKC (continued)

Type: COPYBOOK | Risk: LOW (13)

6	KD-RC return codes: PIC 9(4) with 88-level values 0000,0001,0002,0099 -> Return code indicates status: 0000 OK, 0001 Not Found, 0002 Blocked, 0099 DB error (data-constraint, Line 44), 98% confidence
7	String length limits: Multiple PIC X(n) fields (names, address, email etc.) -> Enforce maximum lengths on textual fields (e.g., KD-EMAIL 60 chars, KD-NACHNAME 40 chars) to avoid truncation (data-constraint, Line 9), 90% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
KD-KUNDENNUMMER	PIC 9(10)	?	?	Long kundenNummer
KD-ANREDE	PIC X(10)	?	?	String anrede
KD-TITEL	PIC X(15)	?	?	String titel
KD-NACHNAME	PIC X(40)	?	?	String nachname
KD-VORNAME	PIC X(25)	?	?	String vorname
KD-GEBURTSDATUM	PIC X(8)	?	?	String geburtsdatum
KD-GEBURTSORT	PIC X(40)	?	?	String geburtsort
KD-STAATSANGEHOER	PIC X(3)	?	?	String staatsangehoerigkeit
KD-AUSWEIS-NR	PIC X(20)	?	?	String ausweisNr
KD-STRASSE	PIC X(50)	?	?	String strasse
KD-HAUSNUMMER	PIC X(10)	?	?	String hausnummer
KD-PLZ	PIC X(10)	?	?	String plz
KD-ORT	PIC X(40)	?	?	String ort
KD-LAND	PIC X(3)	?	?	String land
KD-ADRESSE-SEIT	PIC X(8)	?	?	String adresseSeit
KD-TELEFON-PRIVAT	PIC X(20)	?	?	String telefonPrivat
KD-TELEFON-MOBIL	PIC X(20)	?	?	String telefonMobil
KD-EMAIL	PIC X(60)	?	?	String email
KD-KONTAKT-ERLAUBT	PIC X(1)	?	?	String kontaktErlaubt
KD-SCHUFA-SCORE	PIC 9(3)	?	?	Integer schuFaScore
KD-RATING-INTERN	PIC X(2)	?	?	String ratingIntern
KD-EINKOMMEN-NETTO	PIC S9(9)V99 COMP-3	?	?	BigDecimal einkommenNetto
KD-BESTEHENDEVERBINDL	PIC S9(9)V99 COMP-3	?	?	BigDecimal bestehendeVerbindlichkeiten
KD-SEGMENTCODE	PIC X(4)	?	?	String segmentCode
KD-BETREUER-NR	PIC 9(6)	?	?	Integer betreuerNr
KD-FILIALE-NR	PIC 9(4)	?	?	Integer filialeNr
KD-RC	PIC 9(4)	?	?	Integer rc
KD-FEHLERMSG	PIC X(80)	?	?	String fehlerMsg

BUSINESS RULES: KUNDSTRKC (continued)

Type: COPYBOOK | Risk: LOW (13)

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
General error	Processing error	Log error to SYSOUT. Set non-zero RC.	catch Exception -> log.error -> step FAILED.

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Create KUNDSTRKCDto Java class

[ ] PIC 9(n) -> Integer/Long depending on length (use Long for 10 digits), PIC X(n) -> String

[ ] COMP-3 (packed decimal) -> BigDecimal with appropriate scale (S9(9)V99 -> scale 2)

[ ] Map 88-levels to enums or constant values in Java; keep parent field (PIC) as the source of truth

[ ] Enforce string length validations in DTO setters or during mapping to avoid truncation

[ ] Treat date-like X(8) fields consistently (e.g., use LocalDate with explicit parsing if format known)

[ ] Return codes (KD-RC) should be mapped to an enum with explicit numeric values

BUSINESS RULES: BUCHSTRKC

Type: COPYBOOK | Risk: LOW (13) | 65 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
BUCHSTRKC	COPYBOOK	LOW (13)	65	No UI ? shared copybook str...

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (4)

- BCH-KONTO-VON
- BCH-KONTO-NACH
- BCH-IBAN-VON
- BCH-IBAN-NACH

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

---

1. Overview

Program Name	BUCHSTRKC
Type	COPYBOOK
Purpose	This copybook defines the booking/transaction record (Buchungssatz) used across multiple banking programs. It is used by BUCHSERV, UEBERWEIG, ZINSBERCH, BATCHBCH, KONTOABFR to provide a consistent layout for bookings including parties, amounts, clearing and return/status data, and to support ISO20.022,00 ?/SEPA compatible fields.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY BUCHSTRKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
BUCHUNGS-SATZ	01 BUCHUNGS-SATZ	group	
BCH-BUCHUNGS-ID	05 within 01 BUCHUNGS-SATZ	PIC X(36)	
BCH-KONTO-VON	05 within 01 BUCHUNGS-SATZ	PIC 9(10)	
BCH-KONTO-NACH	05 within 01 BUCHUNGS-SATZ	PIC 9(10)	
BCH-IBAN-VON	05 within 01 BUCHUNGS-SATZ	PIC X(22)	
BCH-IBAN-NACH	05 within 01 BUCHUNGS-SATZ	PIC X(22)	
BCH-BETRAG	05 within 01 BUCHUNGS-SATZ	PIC S9(13)V99 COMP-3	
BCH-WAEHRUNG	05 within 01 BUCHUNGS-SATZ	PIC X(3)	

BUSINESS RULES: BUCHSTRKC (continued)

Type: COPYBOOK | Risk: LOW (13)

BCH-BUCHUNGSDATUM	05 within 01 BUCHUNGS-SATZ	PIC X(8)	
BCH-VALUTADATUM	05 within 01 BUCHUNGS-SATZ	PIC X(8)	
BCH-VERWENDUNGSZWECK	05 within 01 BUCHUNGS-SATZ	PIC X(140)	
BCH-AUFTRAGGEBER-REF	05 within 01 BUCHUNGS-SATZ	PIC X(35)	
BCH-END-TO-END-ID	05 within 01 BUCHUNGS-SATZ	PIC X(35)	
BUCHUNGS-KONTROLLE	01 BUCHUNGS-KONTROLLE	group	
BCH-TYP	05 within 01 BUCHUNGS-KONTROLLE	PIC X(4)	88-levels: BCH-GUTSCHRIFT='GUTS', BCH-LASTSCHRIFT='LAST', BCH-UEBERWEISUNG='UEBW', BCH-DAUERAUFTRAG='DAUA', BCH-GEBUEHR='GEBR', BCH-ZINS='ZINS', BCH-STORNO='STOR'
BCH-KANAL	05 within 01 BUCHUNGS-KONTROLLE	PIC X(4)	88-levels: BCH-ONLINE='ONLN', BCH-FILIALE='FILI', BCH-BATCH='BATC', BCH-SEPA='SEPA'
BCH-STATUS	05 within 01 BUCHUNGS-KONTROLLE	PIC X(2)	88-levels: BCH-PENDING='PE', BCH-GEBUCHT='GB', BCH-STORNIERT='ST', BCH-ABGELEHNT='AB'
BCH-AUTORISIERT	05 within 01 BUCHUNGS-KONTROLLE	PIC X(1)	88-levels: BCH-AUTH-OK='J', BCH-AUTH-NEIN='N'
BUCHUNGS-CLEARING	01 BUCHUNGS-CLEARING	group	
BCH-CLEARING-ID	05 within 01 BUCHUNGS-CLEARING	PIC X(20)	
BCH-INTERBANK-REF	05 within 01 BUCHUNGS-CLEARING	PIC X(35)	
BCH-TARGET2-REF	05 within 01 BUCHUNGS-CLEARING	PIC X(35)	
BCH-VERARBEITUNGSZEIT	05 within 01 BUCHUNGS-CLEARING	PIC X(6)	
BCH-BATCH-NR	05 within 01 BUCHUNGS-CLEARING	PIC 9(8)	
BCH-SEQUENZ-NR	05 within 01 BUCHUNGS-CLEARING	PIC 9(6)	
BUCHUNGS-RUECKGABE	01 BUCHUNGS-RUECKGABE	group	
BCH-RC	05 within 01 BUCHUNGS-RUECKGABE	PIC 9(4)	88-levels: BCH-OK=0000, BCH-LIMIT-FEHLER=0001, BCH-KONTO-FEHLER=0002, BCH-BETRAG-FEHLER=0003, BCH-DOPPELT=0004, BCH-DB-FEHLER=0099
BCH-FEHLERMSG	05 within 01 BUCHUNGS-RUECKGABE	PIC X(80)	
BCH-WARN-FLAG	05 within 01 BUCHUNGS-RUECKGABE	PIC X(1)	

BUSINESS RULES: BUCHSTRKC (continued)
Type: COPYBOOK | Risk: LOW (13)

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Amount storage and scale: BCH-BETRAG PIC S9(13)V99 COMP-3 -> Signed packed decimal with 2 implied decimal places; must be interpreted as monetary value (scale 2). (data-constraint), 90% confidence
2	Transaction type enumerations: BCH-TYP PIC X(4) with 88-levels (GUTS, LAST, UEBW, DAUA, GEBR, ZINS, STOR) -> Only those values are valid business types; drive processing logic (credit, debit, transfer, standing order, fee, interest, reversal). (data-constraint), 90% confidence
3	Channel enumerations: BCH-KANAL PIC X(4) with 88-levels (ONLN, FILI, BATC, SEPA) -> Indicates originating channel and may control routing/authorization rules. (data-constraint), 90% confidence
4	Status enumerations: BCH-STATUS PIC X(2) with 88-levels (PE, GB, ST, AB) -> Defines processing lifecycle: pending, booked, reversed, rejected. (data-constraint), 90% confidence
5	Authorization flag: BCH-AUTORISIERT PIC X(1) with 88-levels (J,N) -> Indicates whether transaction was authorized (J) or not (N); likely required for booking. (data-constraint), 90% confidence
6	Return code enumerations: BCH-RC PIC 9(4) with 88-levels (0000,0001,0002,0003,0004,0099) -> Numeric response codes for booking result; 0000 = OK, other codes identify specific error types. (data-constraint), 90% confidence
7	IBAN and identifier lengths: PIC X(22) for IBANs, PIC X(35) for references -> Fixed-length fields; values should be trimmed/padded when mapping to variable-length DB columns and validated for allowed characters and formats (IBAN checksum externally). (data-constraint), 80% confidence
8	Date fields format: BCH-BUCHUNGSDATUM, BCH-VALUTADATUM PIC X(8) -> Stored as 8-character strings, expected format is likely YYYYMMDD or DDMMYYYY depending on system conventions - confirm before converting to LocalDate. (data-constraint), 60% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
BUCHUNGS-SATZ	01	?	?	BuchungsSatzDto buchungsSatz
BCH-BUCHUNGS-ID	05 PIC X(36)	?	?	String bchBuchungsId
BCH-KONTO-VON	05 PIC 9(10)	?	?	Long bchKontoVon
BCH-KONTO-NACH	05 PIC 9(10)	?	?	Long bchKontoNach
BCH-IBAN-VON	05 PIC X(22)	?	?	String bchIbanVon
BCH-IBAN-NACH	05 PIC X(22)	?	?	String bchIbanNach
BCH-BETRAG	05 PIC S9(13)V99 COMP-3	?	?	BigDecimal bchBetrag
BCH-WAEHRUNG	05 PIC X(3)	?	?	String bchWaehrung
BCH-BUCHUNGSDATUM	05 PIC X(8)	?	?	String bchBuchungsdatum

BCH-VALUTADATUM	05 PIC X(8)	?	?	String bchValutadatum
BCH-VERWENDUNGSZWECK	05 PIC X(140)	?	?	String bchVerwendungszweck
BCH-AUFTRAGGEBER-REF	05 PIC X(35)	?	?	String bchAuftraggeberRef
BCH-END-TO-END-ID	05 PIC X(35)	?	?	String bchEndToEndId
BUCHUNGS-KONTROLLE	01	?	?	BuchungsKontrolleDto buchungsKontrolle
BCH-TYP	05 PIC X(4)	?	?	String bchTyp
BCH-KANAL	05 PIC X(4)	?	?	String bchKanal
BCH-STATUS	05 PIC X(2)	?	?	String bchStatus
BCH-AUTORISIERT	05 PIC X(1)	?	?	String bchAutorisiert
BUCHUNGS-CLEARING	01	?	?	BuchungsClearingDto buchungsClearing
BCH-CLEARING-ID	05 PIC X(20)	?	?	String bchClearingId
BCH-INTERBANK-REF	05 PIC X(35)	?	?	String bchInterbankRef
BCH-TARGET2-REF	05 PIC X(35)	?	?	String bchTarget2Ref
BCH-VERARBEITUNGSZEIT	05 PIC X(6)	?	?	String bchVerarbeitungsZeit
BCH-BATCH-NR	05 PIC 9(8)	?	?	Integer bchBatchNr
BCH-SEQUENZ-NR	05 PIC 9(6)	?	?	Integer bchSequenzNr
BUCHUNGS-RUECKGABE	01	?	?	BuchungsRueckgabeDto buchungsRueckgabe
BCH-RC	05 PIC 9(4)	?	?	Integer bchRc
BCH-FEHLERMSG	05 PIC X(80)	?	?	String bchFehlerMsg
BCH-WARN-FLAG	05 PIC X(1)	?	?	String bchWarnFlag

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
General error	Processing error	Log error to SYSOUT. Set non-zero RC.	catch Exception -> log.error -> step FAILED.

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Create BUCHSTRKCDto Java class (and nested DTOs for groups: BuchungsSatzDto, BuchungsKontrolleDto, BuchungsClearingDto, BuchungsRueckgabeDto).

[ ] PIC 9(n) -> Integer/Long depending on length (9(10) -> Long; 9(8)/9(6) -> Integer). PIC X(n) -> String.

[ ] COMP-3 (packed decimal) with implied decimals -> BigDecimal (apply scale=2 for V99).

[ ] 88-levels -> Java enums or constants (map to Enum types: Typ, Kanal, Status, Rc, AuthFlag).

[ ] Date fields PIC X(8) -> map to java.time.LocalDate after determining format (likely YYYYMMDD).

[ ] Fixed-length fields -> trim trailing spaces when converting to variable-length DB/VARCHAR columns.

[ ] Consider validation for IBAN (checksum) and currency (ISO 4217) during migration.

BUSINESS RULES: ERRHANDKC

Type: COPYBOOK | Risk: LOW (13) | 57 lines | [AI-generated | GDPR data detected]

Program	Type	Risk	Lines	Interface
ERRHANDKC	COPYBOOK	LOW (13)	57	No UI ? shared copybook str...

GDPR-RELEVANT

This program processes personal data according to GDPR Art. 4(1).

Affected Fields (1)

ERR-USER-ID

Analysis Method: AI_POWERED

AI analysis completed. GDPR fields detected — field names only, no personal data values.

---

1. Overview

Program Name	ERRHANDKC
Type	COPYBOOK
Purpose	This copybook defines a shared error handling and logging data structure used across retail banking programs to standardize error metadata, abend codes, SQLCA fields and logging information. It is COPY-ed into many programs to provide a consistent in-memory layout for passing and recording error, database and logging context.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY ERRHANDKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
ERROR-STRUKTUR	01 ERROR-STRUKTUR (group)	GROUP	Container for error fields
ERR-PROGRAMM	01 ERROR-STRUKTUR / 05 ERR-PROGRAMM	PIC X(8)	8-char program identifier
ERR-PARAGRAPH	01 ERROR-STRUKTUR / 05 ERR-PARAGRAPH	PIC X(30)	30-char paragraph/routine name
ERR-CODE	01 ERROR-STRUKTUR / 05 ERR-CODE	PIC 9(8)	Numeric code up to 8 digits, unsigned
ERR-SQLCODE	01 ERROR-STRUKTUR / 05 ERR-SQLCODE	PIC S9(8) COMP	Signed binary numeric (DB2 SQLCODE); consider deprecated per pattern dictionary
ERR-CICS-RESP	01 ERROR-STRUKTUR / 05 ERR-CICS-RESP	PIC S9(8) COMP	Signed binary CICS response code
ERR-CICS-RESP2	01 ERROR-STRUKTUR / 05 ERR-CICS-RESP2	PIC S9(8) COMP	Secondary signed binary CICS response code
ERR-TEXT	01 ERROR-STRUKTUR / 05 ERR-TEXT	PIC X(120)	120-char free text description
ERR-TIMESTAMP	01 ERROR-STRUKTUR / 05 ERR-TIMESTAMP	PIC X(26)	26-char timestamp (format to be determined)

BUSINESS RULES: ERRHANDKC (continued)

Type: COPYBOOK | Risk: LOW (13)

ERR-TERMINAL-ID	01 ERROR-STRUKTUR / 05 ERR-TERMINAL-ID	PIC X(4)	4-char terminal identifier
ERR-TRANSAKTIONS-ID	01 ERROR-STRUKTUR / 05 ERR-TRANSAKTIONS-ID	PIC X(4)	4-char transaction identifier
ERR-USER-ID	01 ERROR-STRUKTUR / 05 ERR-USER-ID	PIC X(8)	8-char user identifier
ABEND-CODES	01 ABEND-CODES (group)	GROUP	Container for abend code constants
ABEND-NORMAL	01 ABEND-CODES / 05 ABEND-NORMAL	PIC X(4) VALUE '0000'	Constant value '0000' representing normal abend code
ABEND-SQL-FEHLER	01 ABEND-CODES / 05 ABEND-SQL-FEHLER	PIC X(4) VALUE 'ASQL'	Constant 'ASQL' for SQL errors
ABEND-CICS-FEHLER	01 ABEND-CODES / 05 ABEND-CICS-FEHLER	PIC X(4) VALUE 'ACIC'	Constant 'ACIC' for CICS errors
ABEND-DATEN-FEHLER	01 ABEND-CODES / 05 ABEND-DATEN-FEHLER	PIC X(4) VALUE 'ADAT'	Constant 'ADAT' for data errors
ABEND-AUTH-FEHLER	01 ABEND-CODES / 05 ABEND-AUTH-FEHLER	PIC X(4) VALUE 'AAUT'	Constant 'AAUT' for auth errors
ABEND-SYSTEM-FEHLER	01 ABEND-CODES / 05 ABEND-SYSTEM-FEHLER	PIC X(4) VALUE 'ASYS'	Constant 'ASYS' for system errors
SQLCA	01 SQLCA (group)	GROUP	Standard SQLCA structure container
SQLCAID	01 SQLCA / 05 SQLCAID	PIC X(8)	8-char SQLCA identifier
SQLCABC	01 SQLCA / 05 SQLCABC	PIC S9(9) COMP	Binary numeric (SQLCA length) signed
SQLCODE	01 SQLCA / 05 SQLCODE	PIC S9(9) COMP	DB2 SQL return code (signed binary) - pattern dictionary says remove and use exceptions
SQLERRM	01 SQLCA / 05 SQLERRM (group)	GROUP	SQL error message components
SQLERRML	01 SQLCA / 05 SQLERRM / 49 SQLERRML	PIC S9(4) COMP	Length of SQLERRMC (binary 4-digit)
SQLERRMC	01 SQLCA / 05 SQLERRM / 49 SQLERRMC	PIC X(70)	70-char SQL error message text
SQLERRP	01 SQLCA / 05 SQLERRP	PIC X(8)	8-char SQL error procedure name
SQLERRD	01 SQLCA / 05 SQLERRD	OCCURS 6 TIMES PIC S9(9) COMP	Six binary numeric diagnostic fields
SQLWARN	01 SQLCA / 05 SQLWARN (group)	GROUP	SQL warnings bitmap as characters
SQLWARN0	01 SQLCA / 05 SQLWARN / 10 SQLWARN0	PIC X(1)	Single char SQL warning flag 0
SQLWARN1	01 SQLCA / 05 SQLWARN / 10 SQLWARN1	PIC X(1)	Single char SQL warning flag 1
SQLWARN2	01 SQLCA / 05 SQLWARN / 10 SQLWARN2	PIC X(1)	Single char SQL warning flag 2
SQLWARN3	01 SQLCA / 05 SQLWARN / 10 SQLWARN3	PIC X(1)	Single char SQL warning flag 3
SQLWARN4	01 SQLCA / 05 SQLWARN / 10 SQLWARN4	PIC X(1)	Single char SQL warning flag 4
SQLWARN5	01 SQLCA / 05 SQLWARN / 10 SQLWARN5	PIC X(1)	Single char SQL warning flag 5

BUSINESS RULES: ERRHANDKC (continued)

Type: COPYBOOK | Risk: LOW (13)

SQLWARN6	01 SQLCA / 05 SQLWARN / 10 SQLWARN6	PIC X(1)	Single char SQL warning flag 6
SQLWARN7	01 SQLCA / 05 SQLWARN / 10 SQLWARN7	PIC X(1)	Single char SQL warning flag 7
SQLEXT	01 SQLCA / 05 SQLEXT	PIC X(8)	8-char SQL extension field
LOGGING-STRUKTUR	01 LOGGING-STRUKTUR (group)	GROUP	Container for logging level and message
LOG-LEVEL	01 LOGGING-STRUKTUR / 05 LOG-LEVEL	PIC X(5)	5-char logging level; associated 88-level condition names
LOG-DEBUG	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-DEBUG	88 VALUE 'DEBUG'	Condition name: LOG-LEVEL == 'DEBUG'
LOG-INFO	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-INFO	88 VALUE 'INFO '	Condition name: LOG-LEVEL == 'INFO ' (note trailing space)
LOG-WARN	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-WARN	88 VALUE 'WARN '	Condition name: LOG-LEVEL == 'WARN ' (note trailing space)
LOG-ERROR	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-ERROR	88 VALUE 'ERROR'	Condition name: LOG-LEVEL == 'ERROR'
LOG-NACHRICHT	01 LOGGING-STRUKTUR / 05 LOG-NACHRICHT	PIC X(200)	200-char log message
LOG-KORRELATIONS-ID	01 LOGGING-STRUKTUR / 05 LOG-KORRELATIONS-ID	PIC X(36)	36-char correlation id (UUID-length)

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Fixed-length text fields: PIC X(n) -> Map to fixed-length String in DTO; trim/pad at boundaries; ensure encoding UTF-8 (data-constraint), 90% confidence
2	Binary numeric (COMP) fields: PIC S9(n) COMP -> Map to native numeric (Integer/Long/Short) in DTO; beware platform binary representation and signedness (data-constraint), 90% confidence
3	Value-constrained abend codes: PIC X(4) VALUE '....' -> These are constants and should be represented as static final constants or enums in Java (e.g. ABEND_SQL_FEHLER='ASQL') (data-constraint), 95% confidence
4	Condition names for LOG-LEVEL: 88-levels under LOG-LEVEL with VALUE -> Represent LOG-LEVEL as enum or String in DTO; 88-levels become derived boolean helpers or enum values (DEBUG/INFO/WARN/ERROR) (data-constraint), 95% confidence
5	Array length constraint: OCCURS 6 TIMES -> SQLERRD is an array of 6 numeric elements; map to List<Integer> or int[] with fixed length 6 (data-constraint), 95% confidence
6	SQLCODE handling: SQLCODE present in SQLCA (PIC S9(9) COMP) and field comment: remove -> Do not rely on SQLCODE in migrated Java code - use Spring DataAccessException hierarchy; map but mark deprecated or remove from business DTOs (business-rule), 98% confidence

BUSINESS RULES: ERRHANDKC (continued)
Type: COPYBOOK | Risk: LOW (13)

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
ERROR-STRUKTUR	01 ERROR-STRUKTUR (group)	?	?	ErrorStrukturDto errorStruktur
ERR-PROGRAMM	05 ERR-PROGRAMM PIC X(8)	?	?	String errProgramm
ERR-PARAGRAPH	05 ERR-PARAGRAPH PIC X(30)	?	?	String errParagraph
ERR-CODE	05 ERR-CODE PIC 9(8)	?	?	Integer errCode
ERR-SQLCODE	05 ERR-SQLCODE PIC S9(8) COMP	?	?	Integer errSqlCode /* deprecated: use exceptions */
ERR-CICS-RESP	05 ERR-CICS-RESP PIC S9(8) COMP	?	?	Integer errCicsResp
ERR-CICS-RESP2	05 ERR-CICS-RESP2 PIC S9(8) COMP	?	?	Integer errCicsResp2
ERR-TEXT	05 ERR-TEXT PIC X(120)	?	?	String errText
ERR-TIMESTAMP	05 ERR-TIMESTAMP PIC X(26)	?	?	String errTimestamp /* consider mapping to java.time.OffsetDateTime with parsing */
ERR-TERMINAL-ID	05 ERR-TERMINAL-ID PIC X(4)	?	?	String errTerminalId
ERR-TRANSAKTIONS-ID	05 ERR-TRANSAKTIONS-ID PIC X(4)	?	?	String errTransaktionsId
ERR-USER-ID	05 ERR-USER-ID PIC X(8)	?	?	String errUserId
ABEND-CODES	01 ABEND-CODES (group)	?	?	AbendCodesDto abendCodes
ABEND-NORMAL	05 ABEND-NORMAL PIC X(4) VALUE '0000'	?	?	static final String ABEND_NORMAL = "0000"
ABEND-SQL-FEHLER	05 ABEND-SQL-FEHLER PIC X(4) VALUE 'ASQL'	?	?	static final String ABEND_SQL_FEHLER = "ASQL"
ABEND-CICS-FEHLER	05 ABEND-CICS-FEHLER PIC X(4) VALUE 'ACIC'	?	?	static final String ABEND_CICS_FEHLER = "ACIC"
ABEND-DATEN-FEHLER	05 ABEND-DATEN-FEHLER PIC X(4) VALUE 'ADAT'	?	?	static final String ABEND_DATEN_FEHLER = "ADAT"
ABEND-AUTH-FEHLER	05 ABEND-AUTH-FEHLER PIC X(4) VALUE 'AAUT'	?	?	static final String ABEND_AUTH_FEHLER = "AAUT"

BUSINESS RULES: ERRHANDKC (continued)

Type: COPYBOOK | Risk: LOW (13)

ABEND-SYSTEM-FEHLER	05 ABEND-SYSTEM-FEHLER PIC X(4) VALUE 'ASYS'	?	?	static final String ABEND_SYSTEM_FEHLER = "ASYS"
SQLCA	01 SQLCA (group)	?	?	SqlcaDto sqlca
SQLCAID	05 SQLCAID PIC X(8)	?	?	String sqlcaId
SQLCABC	05 SQLCABC PIC S9(9) COMP	?	?	Integer sqlcabc
SQLCODE	05 SQLCODE PIC S9(9) COMP	?	?	Integer sqlCode /* prefer to remove and rely on Spring DataAccessException */
SQLERRM	05 SQLERRM (group)	?	?	SqlerrmDto sqlerrm
SQLERRML	49 SQLERRML PIC S9(4) COMP	?	?	Integer sqlerrml
SQLERRMC	49 SQLERRMC PIC X(70)	?	?	String sqlerrmc
SQLERRP	05 SQLERRP PIC X(8)	?	?	String sqlerrp
SQLERRD	05 SQLERRD OCCURS 6 TIMES PIC S9(9) COMP	?	?	List<Integer> sqlerrd /* length 6 */
SQLWARN	05 SQLWARN (group)	?	?	List<String> sqlwarn /* SQLWARN0..7 as single-char flags */
SQLWARN0	10 SQLWARN0 PIC X(1)	?	?	String sqlwarn0
SQLWARN1	10 SQLWARN1 PIC X(1)	?	?	String sqlwarn1
SQLWARN2	10 SQLWARN2 PIC X(1)	?	?	String sqlwarn2
SQLWARN3	10 SQLWARN3 PIC X(1)	?	?	String sqlwarn3
SQLWARN4	10 SQLWARN4 PIC X(1)	?	?	String sqlwarn4
SQLWARN5	10 SQLWARN5 PIC X(1)	?	?	String sqlwarn5
SQLWARN6	10 SQLWARN6 PIC X(1)	?	?	String sqlwarn6
SQLWARN7	10 SQLWARN7 PIC X(1)	?	?	String sqlwarn7
SQLEXT	05 SQLEXT PIC X(8)	?	?	String sqlext
LOGGING-STRUKTUR	01 LOGGING-STRUKTUR (group)	?	?	LoggingStrukturDto loggingStruktur
LOG-LEVEL	05 LOG-LEVEL PIC X(5)	?	?	String logLevel /* prefer enum LogLevel {DEBUG, INFO, WARN, ERROR} */
LOG-DEBUG	88 LOG-DEBUG VALUE 'DEBUG'	?	?	boolean isLogDebug() /* derived from logLevel == 'DEBUG' */
LOG-INFO	88 LOG-INFO VALUE 'INFO '	?	?	boolean isLogInfo() /* derived from logLevel == 'INFO ' */
LOG-WARN	88 LOG-WARN VALUE 'WARN '	?	?	boolean isLogWarn() /* derived from logLevel == 'WARN ' */
LOG-ERROR	88 LOG-ERROR VALUE 'ERROR'	?	?	boolean isLogError() /* derived from logLevel == 'ERROR' */
LOG-NACHRICHT	05 LOG-NACHRICHT PIC X(200)	?	?	String logNachricht

LOG-KORRELATIONS-ID	05 LOG-KORRELATIONS-ID PIC X(36)	?	?	String logKorrelationsId /* UUID length 36 recommended */

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
General error	Processing error	Log error to SYSOUT. Set non-zero RC.	catch Exception -> log.error -> step FAILED.

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Generate Java DTOs reflecting groups and fields; use nested DTOs for SQLCA,
ABEND-CODES and LOGGING-STRUKTUR.

[ ] Implement mapping layer that converts COBOL layouts to DTOs, handling COMP binary
numeric decoding, OCCURS arrays and trimming PIC X fields.

[ ] Introduce enums/constants for abend codes and log levels; implement derived
boolean helpers for 88-levels.

[ ] Remove or mark SQLCODE as deprecated in DTOs and update data access code to use
Spring DataAccessException mapping.

CRITICAL FINDINGS

SQLCODE exists in SQLCA and is defined as PIC S9(9) COMP, but pattern dictionary
advises removal in favor of Spring exceptions ? migration must decide to deprecate
or drop this field.

Several numeric fields use COMP (binary) format. Endianness/platform differences
must be handled when reading legacy files or converting raw byte streams.

LOG-LEVEL 88-values include padded values ('INFO ' and 'WARN ') ?
trimming/preservation of trailing spaces matters when matching conditions.

Timestamps are stored as PIC X(26) with unspecified format; mapping requires
agreed parsing/format rules.

No explicit nullability flags provided ? migration should define defaults and null
handling for each field.

BUSINESS RULES: KUNDSTRKC.cpy

Type: COPYBOOK | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
KUNDSTRKC.cpy	COPYBOOK	?	?	No UI ? shared copybook str...

1. Overview

Program Name	KUNDSTRKC.cpy
Type	COPYBOOK
Purpose	This copybook defines the customer master and related CRM account-relationship data (name, address, contact, credit and segment info) used by multiple banking programs. It is reused by programs like KONTOABFR, UEBERWEIG, BUCHSERV, AUSKUNFT and KREDITPRU to provide a consistent on-wire/in-memory layout for customer data and return codes.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY KUNDSTRKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
KUNDE-STAMMDATEN	01 KUNDE-STAMMDATEN	group	group container for customer master fields
KD-KUNDENNUMMER	05 in KUNDE-STAMMDATEN	PIC 9(10)	numeric 10 digits
KD-ANREDE	05 in KUNDE-STAMMDATEN	PIC X(10)	text up to 10 chars (title/salutation)
KD-TITEL	05 in KUNDE-STAMMDATEN	PIC X(15)	text up to 15 chars
KD-NACHNAME	05 in KUNDE-STAMMDATEN	PIC X(40)	text up to 40 chars (surname)
KD-VORNAME	05 in KUNDE-STAMMDATEN	PIC X(25)	text up to 25 chars (given name)
KD-GEBURTSDATUM	05 in KUNDE-STAMMDATEN	PIC X(8)	date as 8-char string (format not enforced here)
KD-GEBURTSORT	05 in KUNDE-STAMMDATEN	PIC X(40)	text up to 40 chars (place of birth)
KD-STAATSANGEHOER	05 in KUNDE-STAMMDATEN	PIC X(3)	3-char country code
KD-AUSWEIS-NR	05 in KUNDE-STAMMDATEN	PIC X(20)	ID document number up to 20 chars
KUNDE-ADRESSE	01 KUNDE-ADRESSE	group	group container for address fields
KD-STRASSE	05 in KUNDE-ADRESSE	PIC X(50)	street name up to 50 chars
KD-HAUSNUMMER	05 in KUNDE-ADRESSE	PIC X(10)	house number up to 10 chars
KD-PLZ	05 in KUNDE-ADRESSE	PIC X(10)	postal code up to 10 chars
KD-ORT	05 in KUNDE-ADRESSE	PIC X(40)	city up to 40 chars
KD-LAND	05 in KUNDE-ADRESSE	PIC X(3)	3-char country code
KD-ADRESSE-SEIT	05 in KUNDE-ADRESSE	PIC X(8)	since-date as 8-char string

BUSINESS RULES: KUNDSTRKC.cpy (continued)
Type: COPYBOOK

KUNDE-KONTAKT	01 KUNDE-KONTAKT	group	group container for contact fields
KD-TELEFON-PRIVAT	05 in KUNDE-KONTAKT	PIC X(20)	private phone up to 20 chars
KD-TELEFON-MOBIL	05 in KUNDE-KONTAKT	PIC X(20)	mobile phone up to 20 chars
KD-EMAIL	05 in KUNDE-KONTAKT	PIC X(60)	email address up to 60 chars
KD-KONTAKT-ERLAUBT	05 in KUNDE-KONTAKT	PIC X(1)	flag allowed values via 88-levels: 'J' or 'N'
KD-KONTAKT-JA	88 in KUNDE-KONTAKT (relates to KD-KONTAKT-ERLAUBT)	VALUE 'J'	enumerated value representing yes
KD-KONTAKT-NEIN	88 in KUNDE-KONTAKT (relates to KD-KONTAKT-ERLAUBT)	VALUE 'N'	enumerated value representing no
KUNDE-BONITAET	01 KUNDE-BONITAET	group	group container for creditworthiness fields
KD-SCHUFA-SCORE	05 in KUNDE-BONITAET	PIC 9(3)	numeric 3 digits (SCHUFA score)
KD-RATING-INTERN	05 in KUNDE-BONITAET	PIC X(2)	internal rating codes; allowed values via 88-levels AA/BB/CC/DD
KD-RATING-AAA	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'AA'	rating constant
KD-RATING-GUT	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'BB'	rating constant
KD-RATING-OK	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'CC'	rating constant
KD-RATING-SCHLECHT	88 in KUNDE-BONITAET (relates to KD-RATING-INTERN)	VALUE 'DD'	rating constant
KD-EINKOMMEN-NETTO	05 in KUNDE-BONITAET	PIC S9(9)V99 COMP-3	signed packed decimal with 2 implied decimals; scale 2
KD-BESTEHENDEVERBINDL	05 in KUNDE-BONITAET	PIC S9(9)V99 COMP-3	signed packed decimal with 2 implied decimals; scale 2
KUNDE-SEGMENT	01 KUNDE-SEGMENT	group	group container for segmentation and branch fields
KD-SEGMENTCODE	05 in KUNDE-SEGMENT	PIC X(4)	segment code - allowed values via 88-levels PRIV/PREM/GSCH/VVIP
KD-PRIVATKUNDE	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'PRIV'	segment constant
KD-PREMIUM	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'PREM'	segment constant
KD-GESCHAEFTSKD	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'GSCH'	segment constant
KD-VIP	88 in KUNDE-SEGMENT (relates to KD-SEGMENTCODE)	VALUE 'VVIP'	segment constant
KD-BETREUER-NR	05 in KUNDE-SEGMENT	PIC 9(6)	numeric 6 digits (account manager id)
KD-FILIALE-NR	05 in KUNDE-SEGMENT	PIC 9(4)	numeric 4 digits (branch number)

KUNDE-RUECKGABE	01 KUNDE-RUECKGABE	group	group container for return code and message
KD-RC	05 in KUNDE-RUECKGABE	PIC 9(4)	4-digit return code; mapped 88-levels define semantic codes
KD-OK	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0000	return code OK
KD-NICHT-GEFUNDEN	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0001	return code not found
KD-GESPERRT	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0002	return code blocked
KD-DB-FEHLER	88 in KUNDE-RUECKGABE (relates to KD-RC)	VALUE 0099	return code database error
KD-FEHLERMSG	05 in KUNDE-RUECKGABE	PIC X(80)	error message text up to 80 chars

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	KD-KUNDENNUMMER numeric length: PIC 9(10) -> Customer number must be a 10-digit numeric identifier; store as numeric (no sign, fixed width) (data-constraint, Line 7), 90% confidence
2	KD-KONTAKT-ERLAUBT allowed values: PIC X(1) with 88-levels KD-KONTAKT-JA='J' and KD-KONTAKT-NEIN='N' -> Contact allowed flag must be 'J' or 'N' (use validation during input/output) (data-constraint, Line 20), 95% confidence
3	KD-RATING-INTERN enumerated values: PIC X(2) with 88-levels AA|BB|CC|DD -> Internal rating must be one of 'AA','BB','CC','DD' representing qualitative ratings (data-constraint, Line 27), 95% confidence
4	Packed decimal financial fields: PIC S9(9)V99 COMP-3 -> EINKOMMEN-NETTO and BESTEHENDEVERBINDL are signed packed decimals with 2 implied decimals; map to BigDecimal with scale 2 (data-constraint, Line 30), 95% confidence
5	KD-SEGMENTCODE enumerated values: PIC X(4) with 88-levels PRIV|PREM|GSCH|VVIP -> Segment code must be one of defined codes indicating customer segment (data-constraint, Line 35), 95% confidence
6	KD-RC return codes: PIC 9(4) with 88-level values 0000,0001,0002,0099 -> Return code indicates status: 0000 OK, 0001 Not Found, 0002 Blocked, 0099 DB error (data-constraint, Line 44), 98% confidence
7	String length limits: Multiple PIC X(n) fields (names, address, email etc.) -> Enforce maximum lengths on textual fields (e.g., KD-EMAIL 60 chars, KD-NACHNAME 40 chars) to avoid truncation (data-constraint, Line 9), 90% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
KD-KUNDENNUMMER	PIC 9(10)	?	?	Long kundenNummer
KD-ANREDE	PIC X(10)	?	?	String anrede
KD-TITEL	PIC X(15)	?	?	String titel

KD-NACHNAME	PIC X(40)	?	?	String nachname
KD-VORNAME	PIC X(25)	?	?	String vorname
KD-GEBURTSDATUM	PIC X(8)	?	?	String geburtsdatum
KD-GEBURTSORT	PIC X(40)	?	?	String geburtsort
KD-STAATSANGEHOER	PIC X(3)	?	?	String staatsangehoerigkeit
KD-AUSWEIS-NR	PIC X(20)	?	?	String ausweisNr
KD-STRASSE	PIC X(50)	?	?	String strasse
KD-HAUSNUMMER	PIC X(10)	?	?	String hausnummer
KD-PLZ	PIC X(10)	?	?	String plz
KD-ORT	PIC X(40)	?	?	String ort
KD-LAND	PIC X(3)	?	?	String land
KD-ADRESSE-SEIT	PIC X(8)	?	?	String adresseSeit
KD-TELEFON-PRIVAT	PIC X(20)	?	?	String telefonPrivat
KD-TELEFON-MOBIL	PIC X(20)	?	?	String telefonMobil
KD-EMAIL	PIC X(60)	?	?	String email
KD-KONTAKT-ERLAUBT	PIC X(1)	?	?	String kontaktErlaubt
KD-SCHUFA-SCORE	PIC 9(3)	?	?	Integer schuFaScore
KD-RATING-INTERN	PIC X(2)	?	?	String ratingIntern
KD-EINKOMMEN-NETTO	PIC S9(9)V99 COMP-3	?	?	BigDecimal einkommenNetto
KD-BESTEHENDEVERBINDL	PIC S9(9)V99 COMP-3	?	?	BigDecimal bestehendeVerbindlichkeiten
KD-SEGMENTCODE	PIC X(4)	?	?	String segmentCode
KD-BETREUER-NR	PIC 9(6)	?	?	Integer betreuerNr
KD-FILIALE-NR	PIC 9(4)	?	?	Integer filialeNr
KD-RC	PIC 9(4)	?	?	Integer rc
KD-FEHLERMSG	PIC X(80)	?	?	String fehlerMsg

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
General error	Processing error	Log error to SYSOUT. Set non-zero RC.	catch Exception -> log.error -> step FAILED.

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Create KUNDSTRKCDto Java class

[ ] PIC 9(n) -> Integer/Long depending on length (use Long for 10 digits), PIC X(n) -> String

[ ] COMP-3 (packed decimal) -> BigDecimal with appropriate scale (S9(9)V99 -> scale 2)

[ ] Map 88-levels to enums or constant values in Java; keep parent field (PIC) as the source of truth

[ ] Enforce string length validations in DTO setters or during mapping to avoid truncation

[ ] Treat date-like X(8) fields consistently (e.g., use LocalDate with explicit parsing if format known)

BUSINESS RULES: KUNDSTRKC.cpy (continued)

Type: COPYBOOK

[ ] Return codes (KD-RC) should be mapped to an enum with explicit numeric values

BUSINESS RULES: KTOSTRKC.cpy

Type: COPYBOOK | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
KTOSTRKC.cpy	COPYBOOK	?	?	No UI ? shared copybook str...

1. Overview

Program Name	KTOSTRKC.cpy
Type	COPYBOOK
Purpose	This copybook defines the account (KONTO) data structure for retail banking account management (giro, savings, fixed deposit, credit). It is used by programs such as KONTOABFR, BUCHSERV, ZINSBERCH, KONTOPRUEF and UEBERWEIG to provide a consistent in-memory layout for account master data, balances and return/error information for processing and messaging.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY KTOSTRKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
KONTO-STAMMDATEN	01 KONTO-STAMMDATEN	group	group container for account master data
KTO-NUMMER	05 within KONTO-STAMMDATEN	PIC 9(10)	numeric account number, up to 10 digits
KTO-BLZ	05 within KONTO-STAMMDATEN	PIC 9(8)	bank sort code (BLZ), 8 digits
KTO-IBAN	05 within KONTO-STAMMDATEN	PIC X(22)	IBAN as 22-character string (stored fixed length)
KTO-BIC	05 within KONTO-STAMMDATEN	PIC X(11)	BIC as 11-character string
KTO-INHABER	05 within KONTO-STAMMDATEN	group	group for account holder identity
KTO-INH-NAME	10 within KTO-INHABER	PIC X(40)	holder family name up to 40 chars
KTO-INH-VORNAME	10 within KTO-INHABER	PIC X(25)	holder given name up to 25 chars
KTO-INH-KDNR	10 within KTO-INHABER	PIC 9(10)	customer number, numeric up to 10 digits
KTO-TYP	05 within KONTO-STAMMDATEN	PIC X(2)	88-levels: KTO-GIROKONTO='GK', KTO-SPARKONTO='SK', KTO-FESTGELD='FG', KTO-KREDITKONTO='KK'
KTO-WAEHRUNG	05 within KONTO-STAMMDATEN	PIC X(3)	currency 3-char code; 88-levels: KTO-EUR='EUR', KTO-USD='USD', KTO-CHF='CHF'

BUSINESS RULES: KTOSTRKC.cpy (continued)
Type: COPYBOOK

KTO-STATUS	05 within KONTO-STAMMDATEN	PIC X(1)	status flag; 88-levels: KTO-AKTIV='A', KTO-GESPERRT='G', KTO-GELOESCHT='L', KTO-PFAENDUNG='P'
KONTO-SALDEN	01 KONTO-SALDEN	group	group container for balance and limits
KTO-SALDO	05 within KONTO-SALDEN	PIC S9(13)V99 COMP-3	signed packed decimal, implied 2 decimals (balance)
KTO-VERFUEGBAR	05 within KONTO-SALDEN	PIC S9(13)V99 COMP-3	signed packed decimal, implied 2 decimals (available amount)
KTO-DISPOKREDITLIMIT	05 within KONTO-SALDEN	PIC S9(11)V99 COMP-3	signed packed decimal, implied 2 decimals (overdraft limit)
KTO-GUTHABENGRENZE	05 within KONTO-SALDEN	PIC S9(11)V99 COMP-3	signed packed decimal, implied 2 decimals (credit ceiling)
KTO-SOLLZINS-SATZ	05 within KONTO-SALDEN	PIC 9(3)V9(4) COMP-3	interest rate charged (debit) with 4 decimal places (scale=4)
KTO-HABENZINS-SATZ	05 within KONTO-SALDEN	PIC 9(3)V9(4) COMP-3	interest rate paid (credit) with 4 decimal places (scale=4)
KTO-LETZTER-UMSATZ	05 within KONTO-SALDEN	PIC X(8)	last transaction date or identifier as 8-char string (likely YYYYMMDD)
KTO-EROEFFFNUNGSDATUM	05 within KONTO-SALDEN	PIC X(8)	account opening date as 8-char string (likely YYYYMMDD)
KTO-KUENDIGUNGSDATUM	05 within KONTO-SALDEN	PIC X(8)	account termination date as 8-char string (likely YYYYMMDD) or blanks
KONTO-RUECKGABE	01 KONTO-RUECKGABE	group	group for return/response metadata
KTO-RC	05 within KONTO-RUECKGABE	PIC 9(4)	numeric return code; 88-levels: KTO-OK=0000, KTO-NICHT-GEFUNDEN=0001, KTO-GESPERRT-RC=0002, KTO-LIMIT-UEBER=0003, KTO-DB-FEHLER=0099
KTO-FEHLERMSG	05 within KONTO-RUECKGABE	PIC X(80)	free-form error message up to 80 chars

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Account type domain: 05 KTO-TYP PIC X(2); 88 values: 'GK','SK','FG','KK' -> KTO-TYP must be one of the defined codes (GK=Giro, SK=Sparkonto, FG=Festgeld, KK=Kreditkonto). Reject or map otherwise. (data-constraint, Line 12), 95% confidence
2	Currency domain: 05 KTO-WAEHRUNG PIC X(3); 88 values: 'EUR','USD','CHF' -> KTO-WAEHRUNG must be a supported 3-letter currency code; treat unknown codes as invalid or require normalization. (data-constraint, Line 16), 90% confidence
3	Account status domain: 05 KTO-STATUS PIC X(1); 88 values: 'A','G','L','P' -> KTO-STATUS indicates lifecycle state (A=active, G=blocked, L=deleted, P=attached/levied). Business logic depends on this flag. (data-constraint, Line 19), 95% confidence

BUSINESS RULES: KTOSTRKC.cpy (continued)
Type: COPYBOOK

4	Monetary precision: COMP-3 fields: S9(13)V99 and S9(11)V99 -> Amounts have 2 decimal places implied; map to BigDecimal with scale=2 and adequate precision to avoid rounding errors. (data-format, Line 24), 98% confidence
5	Interest rate format: PIC 9(3)V9(4) COMP-3 for interest rates -> Interest rates have 4 decimal places (scale=4) and up to 3 integer digits; map to BigDecimal scale=4. (data-format, Line 28), 98% confidence
6	Return codes: KTO-RC PIC 9(4) with 88-levels: 0000,0001,0002,0003,0099 -> Interpret numeric return code to determine success or specific error conditions; propagate or translate to exceptions/messages. (business-rule, Line 37), 95% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
KONTO-STAMMDATEN	01 KONTO-STAMMDATEN	?	?	KontoStammdatenDto kontoStammdaten
KTO-NUMMER	05 within KONTO-STAMMDATEN (PIC 9(10))	?	?	Long ktoNummer
KTO-BLZ	05 within KONTO-STAMMDATEN (PIC 9(8))	?	?	Integer ktoBlz
KTO-IBAN	05 within KONTO-STAMMDATEN (PIC X(22))	?	?	String ktoIban
KTO-BIC	05 within KONTO-STAMMDATEN (PIC X(11))	?	?	String ktoBic
KTO-INHABER	05 within KONTO-STAMMDATEN	?	?	KtoInhaberDto ktoInhaber
KTO-INH-NAME	10 within KTO-INHABER (PIC X(40))	?	?	String ktoInhName
KTO-INH-VORNAME	10 within KTO-INHABER (PIC X(25))	?	?	String ktoInhVorname
KTO-INH-KDNR	10 within KTO-INHABER (PIC 9(10))	?	?	Long ktoInhKdnr
KTO-TYP	05 within KONTO-STAMMDATEN (PIC X(2))	?	?	String ktoTyp // allowed values: GK, SK, FG, KK
KTO-WAEHRUNG	05 within KONTO-STAMMDATEN (PIC X(3))	?	?	String ktoWaehrung // ISO currency code
KTO-STATUS	05 within KONTO-STAMMDATEN (PIC X(1))	?	?	String ktoStatus // A/G/L/P
KONTO-SALDEN	01 KONTO-SALDEN	?	?	KontoSaldenDto kontoSalden
KTO-SALDO	05 within KONTO-SALDEN (PIC S9(13)V99 COMP-3)	?	?	java.math.BigDecimal ktoSaldo // scale=2

KTO-VERFUEGBAR	05 within KONTO-SALDEN (PIC S9(13)V99 COMP-3)	?	?	java.math.BigDecimal ktoVerfuegbar // scale=2
KTO-DISPOKREDITLIMIT	05 within KONTO-SALDEN (PIC S9(11)V99 COMP-3)	?	?	java.math.BigDecimal ktoDispoKreditLimit // scale=2
KTO-GUTHABENGRENZE	05 within KONTO-SALDEN (PIC S9(11)V99 COMP-3)	?	?	java.math.BigDecimal ktoGuthabenGrenze // scale=2
KTO-SOLLZINS-SATZ	05 within KONTO-SALDEN (PIC 9(3)V9(4) COMP-3)	?	?	java.math.BigDecimal ktoSollzinsSatz // scale=4
KTO-HABENZINS-SATZ	05 within KONTO-SALDEN (PIC 9(3)V9(4) COMP-3)	?	?	java.math.BigDecimal ktoHabenzinsSatz // scale=4
KTO-LETZTER-UMSATZ	05 within KONTO-SALDEN (PIC X(8))	?	?	String ktoLetzterUmsatz // YYYYMMDD or free-form 8 chars
KTO-EROEFFFNUNGSDATUM	05 within KONTO-SALDEN (PIC X(8))	?	?	String ktoEroeffnungsdatum // YYYYMMDD
KTO-KUENDIGUNGSDATUM	05 within KONTO-SALDEN (PIC X(8))	?	?	String ktoKuendigungsdatum // YYYYMMDD or blanks
KONTO-RUECKGABE	01 KONTO-RUECKGABE	?	?	KontoRueckgabeDto kontoRueckgabe
KTO-RC	05 within KONTO-RUECKGABE (PIC 9(4))	?	?	Integer ktoRc // interpret with defined 88-level codes
KTO-FEHLERMSG	05 within KONTO-RUECKGABE (PIC X(80))	?	?	String ktoFehlerMsg

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
0000	Successful lookup/operation		
0001	Account not found		
0002	Account is blocked		
0003	Limit exceeded		
0099	Database or internal error		

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Define Java DTOs according to migrationHints and validate with sample data.

[ ] Implement conversion utilities for COMP-3 -> BigDecimal and for PIC numeric strings -> Long/Integer.

[ ] Create enums for KTO-TYP, KTO-WAEHRUNG, KTO-STATUS and mapping unit tests.

BUSINESS RULES: KTOSTRKC.cpy (continued)

Type: COPYBOOK

[ ] Add integration tests for serialization/deserialization between COBOL binary layout and Java DTO (if reading COBOL files).

CRITICAL FINDINGS

Dates are stored as PIC X(8) (likely YYYYMMDD) but not enforced; conversion errors possible if format differs.

Monetary fields are COMP-3 packed; migration must correctly unpack signed values and apply scale.

No explicit sign or overflow handling in copybook: application logic must enforce limits (e.g., S9(13)V99).

Fixed-length text fields (IBAN/BIC/names) may contain trailing spaces; trimming/normalization needed in Java.

BUSINESS RULES: ERRHANDKC.cpy

Type: COPYBOOK | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
ERRHANDKC.cpy	COPYBOOK	?	?	No UI ? shared copybook str...

1. Overview

Program Name	ERRHANDKC.cpy
Type	COPYBOOK
Purpose	This copybook defines a shared error handling and logging data structure used across retail banking programs to standardize error metadata, abend codes, SQLCA fields and logging information. It is COPY-ed into many programs to provide a consistent in-memory layout for passing and recording error, database and logging context.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY ERRHANDKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
ERROR-STRUKTUR	01 ERROR-STRUKTUR (group)	GROUP	Container for error fields
ERR-PROGRAMM	01 ERROR-STRUKTUR / 05 ERR-PROGRAMM	PIC X(8)	8-char program identifier
ERR-PARAGRAPH	01 ERROR-STRUKTUR / 05 ERR-PARAGRAPH	PIC X(30)	30-char paragraph/routine name
ERR-CODE	01 ERROR-STRUKTUR / 05 ERR-CODE	PIC 9(8)	Numeric code up to 8 digits, unsigned
ERR-SQLCODE	01 ERROR-STRUKTUR / 05 ERR-SQLCODE	PIC S9(8) COMP	Signed binary numeric (DB2 SQLCODE); consider deprecated per pattern dictionary
ERR-CICS-RESP	01 ERROR-STRUKTUR / 05 ERR-CICS-RESP	PIC S9(8) COMP	Signed binary CICS response code
ERR-CICS-RESP2	01 ERROR-STRUKTUR / 05 ERR-CICS-RESP2	PIC S9(8) COMP	Secondary signed binary CICS response code
ERR-TEXT	01 ERROR-STRUKTUR / 05 ERR-TEXT	PIC X(120)	120-char free text description
ERR-TIMESTAMP	01 ERROR-STRUKTUR / 05 ERR-TIMESTAMP	PIC X(26)	26-char timestamp (format to be determined)
ERR-TERMINAL-ID	01 ERROR-STRUKTUR / 05 ERR-TERMINAL-ID	PIC X(4)	4-char terminal identifier
ERR-TRANSAKTIONS-ID	01 ERROR-STRUKTUR / 05 ERR-TRANSAKTIONS-ID	PIC X(4)	4-char transaction identifier
ERR-USER-ID	01 ERROR-STRUKTUR / 05 ERR-USER-ID	PIC X(8)	8-char user identifier
ABEND-CODES	01 ABEND-CODES (group)	GROUP	Container for abend code constants

BUSINESS RULES: ERRHANDKC.cpy (continued)
Type: COPYBOOK

ABEND-NORMAL	01 ABEND-CODES / 05 ABEND-NORMAL	PIC X(4) VALUE '0000'	Constant value '0000' representing normal abend code
ABEND-SQL-FEHLER	01 ABEND-CODES / 05 ABEND-SQL-FEHLER	PIC X(4) VALUE 'ASQL'	Constant 'ASQL' for SQL errors
ABEND-CICS-FEHLER	01 ABEND-CODES / 05 ABEND-CICS-FEHLER	PIC X(4) VALUE 'ACIC'	Constant 'ACIC' for CICS errors
ABEND-DATEN-FEHLER	01 ABEND-CODES / 05 ABEND-DATEN-FEHLER	PIC X(4) VALUE 'ADAT'	Constant 'ADAT' for data errors
ABEND-AUTH-FEHLER	01 ABEND-CODES / 05 ABEND-AUTH-FEHLER	PIC X(4) VALUE 'AAUT'	Constant 'AAUT' for auth errors
ABEND-SYSTEM-FEHLER	01 ABEND-CODES / 05 ABEND-SYSTEM-FEHLER	PIC X(4) VALUE 'ASYS'	Constant 'ASYS' for system errors
SQLCA	01 SQLCA (group)	GROUP	Standard SQLCA structure container
SQLCAID	01 SQLCA / 05 SQLCAID	PIC X(8)	8-char SQLCA identifier
SQLCABC	01 SQLCA / 05 SQLCABC	PIC S9(9) COMP	Binary numeric (SQLCA length) signed
SQLCODE	01 SQLCA / 05 SQLCODE	PIC S9(9) COMP	DB2 SQL return code (signed binary) - pattern dictionary says remove and use exceptions
SQLERRM	01 SQLCA / 05 SQLERRM (group)	GROUP	SQL error message components
SQLERRML	01 SQLCA / 05 SQLERRM / 49 SQLERRML	PIC S9(4) COMP	Length of SQLERRMC (binary 4-digit)
SQLERRMC	01 SQLCA / 05 SQLERRM / 49 SQLERRMC	PIC X(70)	70-char SQL error message text
SQLERRP	01 SQLCA / 05 SQLERRP	PIC X(8)	8-char SQL error procedure name
SQLERRD	01 SQLCA / 05 SQLERRD	OCCURS 6 TIMES PIC S9(9) COMP	Six binary numeric diagnostic fields
SQLWARN	01 SQLCA / 05 SQLWARN (group)	GROUP	SQL warnings bitmap as characters
SQLWARN0	01 SQLCA / 05 SQLWARN / 10 SQLWARN0	PIC X(1)	Single char SQL warning flag 0
SQLWARN1	01 SQLCA / 05 SQLWARN / 10 SQLWARN1	PIC X(1)	Single char SQL warning flag 1
SQLWARN2	01 SQLCA / 05 SQLWARN / 10 SQLWARN2	PIC X(1)	Single char SQL warning flag 2
SQLWARN3	01 SQLCA / 05 SQLWARN / 10 SQLWARN3	PIC X(1)	Single char SQL warning flag 3
SQLWARN4	01 SQLCA / 05 SQLWARN / 10 SQLWARN4	PIC X(1)	Single char SQL warning flag 4
SQLWARN5	01 SQLCA / 05 SQLWARN / 10 SQLWARN5	PIC X(1)	Single char SQL warning flag 5
SQLWARN6	01 SQLCA / 05 SQLWARN / 10 SQLWARN6	PIC X(1)	Single char SQL warning flag 6
SQLWARN7	01 SQLCA / 05 SQLWARN / 10 SQLWARN7	PIC X(1)	Single char SQL warning flag 7
SQLEXT	01 SQLCA / 05 SQLEXT	PIC X(8)	8-char SQL extension field
LOGGING-STRUKTUR	01 LOGGING-STRUKTUR (group)	GROUP	Container for logging level and message
LOG-LEVEL	01 LOGGING-STRUKTUR / 05 LOG-LEVEL	PIC X(5)	5-char logging level; associated 88-level condition names

LOG-DEBUG	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-DEBUG	88 VALUE 'DEBUG'	Condition name: LOG-LEVEL == 'DEBUG'
LOG-INFO	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-INFO	88 VALUE 'INFO '	Condition name: LOG-LEVEL == 'INFO ' (note trailing space)
LOG-WARN	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-WARN	88 VALUE 'WARN '	Condition name: LOG-LEVEL == 'WARN ' (note trailing space)
LOG-ERROR	01 LOGGING-STRUKTUR / 05 LOG-LEVEL / 88 LOG-ERROR	88 VALUE 'ERROR'	Condition name: LOG-LEVEL == 'ERROR'
LOG-NACHRICHT	01 LOGGING-STRUKTUR / 05 LOG-NACHRICHT	PIC X(200)	200-char log message
LOG-KORRELATIONS-ID	01 LOGGING-STRUKTUR / 05 LOG-KORRELATIONS-ID	PIC X(36)	36-char correlation id (UUID-length)

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Fixed-length text fields: PIC X(n) -> Map to fixed-length String in DTO; trim/pad at boundaries; ensure encoding UTF-8 (data-constraint), 90% confidence
2	Binary numeric (COMP) fields: PIC S9(n) COMP -> Map to native numeric (Integer/Long/Short) in DTO; beware platform binary representation and signedness (data-constraint), 90% confidence
3	Value-constrained abend codes: PIC X(4) VALUE '....' -> These are constants and should be represented as static final constants or enums in Java (e.g. ABEND_SQL_FEHLER='ASQL') (data-constraint), 95% confidence
4	Condition names for LOG-LEVEL: 88-levels under LOG-LEVEL with VALUE -> Represent LOG-LEVEL as enum or String in DTO; 88-levels become derived boolean helpers or enum values (DEBUG/INFO/WARN/ERROR) (data-constraint), 95% confidence
5	Array length constraint: OCCURS 6 TIMES -> SQLERRD is an array of 6 numeric elements; map to List<Integer> or int[] with fixed length 6 (data-constraint), 95% confidence
6	SQLCODE handling: SQLCODE present in SQLCA (PIC S9(9) COMP) and field comment: remove -> Do not rely on SQLCODE in migrated Java code - use Spring DataAccessException hierarchy; map but mark deprecated or remove from business DTOs (business-rule), 98% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
ERROR-STRUKTUR	01 ERROR-STRUKTUR (group)	?	?	ErrorStrukturDto errorStruktur
ERR-PROGRAMM	05 ERR-PROGRAMM PIC X(8)	?	?	String errProgramm

BUSINESS RULES: ERRHANDKC.cpy (continued)
Type: COPYBOOK

ERR-PARAGRAPH	05 ERR-PARAGRAPH PIC X(30)	?	?	String errParagraph
ERR-CODE	05 ERR-CODE PIC 9(8)	?	?	Integer errCode
ERR-SQLCODE	05 ERR-SQLCODE PIC S9(8) COMP	?	?	Integer errSqlCode /* deprecated: use exceptions */
ERR-CICS-RESP	05 ERR-CICS-RESP PIC S9(8) COMP	?	?	Integer errCicsResp
ERR-CICS-RESP2	05 ERR-CICS-RESP2 PIC S9(8) COMP	?	?	Integer errCicsResp2
ERR-TEXT	05 ERR-TEXT PIC X(120)	?	?	String errText
ERR-TIMESTAMP	05 ERR-TIMESTAMP PIC X(26)	?	?	String errTimestamp /* consider mapping to java.time.OffsetDateTime with parsing */
ERR-TERMINAL-ID	05 ERR-TERMINAL-ID PIC X(4)	?	?	String errTerminalId
ERR-TRANSAKTIONS-ID	05 ERR-TRANSAKTIONS-ID PIC X(4)	?	?	String errTransaktionsId
ERR-USER-ID	05 ERR-USER-ID PIC X(8)	?	?	String errUserId
ABEND-CODES	01 ABEND-CODES (group)	?	?	AbendCodesDto abendCodes
ABEND-NORMAL	05 ABEND-NORMAL PIC X(4) VALUE '0000'	?	?	static final String ABEND_NORMAL = "0000"
ABEND-SQL-FEHLER	05 ABEND-SQL-FEHLER PIC X(4) VALUE 'ASQL'	?	?	static final String ABEND_SQL_FEHLER = "ASQL"
ABEND-CICS-FEHLER	05 ABEND-CICS-FEHLER PIC X(4) VALUE 'ACIC'	?	?	static final String ABEND_CICS_FEHLER = "ACIC"
ABEND-DATEN-FEHLER	05 ABEND-DATEN-FEHLER PIC X(4) VALUE 'ADAT'	?	?	static final String ABEND_DATEN_FEHLER = "ADAT"
ABEND-AUTH-FEHLER	05 ABEND-AUTH-FEHLER PIC X(4) VALUE 'AAUT'	?	?	static final String ABEND_AUTH_FEHLER = "AAUT"
ABEND-SYSTEM-FEHLER	05 ABEND-SYSTEM-FEHLER PIC X(4) VALUE 'ASYS'	?	?	static final String ABEND_SYSTEM_FEHLER = "ASYS"
SQLCA	01 SQLCA (group)	?	?	SqlcaDto sqlca
SQLCAID	05 SQLCAID PIC X(8)	?	?	String sqlcaId
SQLCABC	05 SQLCABC PIC S9(9) COMP	?	?	Integer sqlcabc
SQLCODE	05 SQLCODE PIC S9(9) COMP	?	?	Integer sqlCode /* prefer to remove and rely on Spring DataAccessException */
SQLERRM	05 SQLERRM (group)	?	?	SqlerrmDto sqlerrm

BUSINESS RULES: ERRHANDKC.cpy (continued)
Type: COPYBOOK

SQLERRML	49 SQLERRML PIC S9(4) COMP	?	?	Integer sqlerrml
SQLERRMC	49 SQLERRMC PIC X(70)	?	?	String sqlerrmc
SQLERRP	05 SQLERRP PIC X(8)	?	?	String sqlerrp
SQLERRD	05 SQLERRD OCCURS 6 TIMES PIC S9(9) COMP	?	?	List<Integer> sqlerrd /* length 6 */
SQLWARN	05 SQLWARN (group)	?	?	List<String> sqlwarn /* SQLWARN0..7 as single-char flags */
SQLWARN0	10 SQLWARN0 PIC X(1)	?	?	String sqlwarn0
SQLWARN1	10 SQLWARN1 PIC X(1)	?	?	String sqlwarn1
SQLWARN2	10 SQLWARN2 PIC X(1)	?	?	String sqlwarn2
SQLWARN3	10 SQLWARN3 PIC X(1)	?	?	String sqlwarn3
SQLWARN4	10 SQLWARN4 PIC X(1)	?	?	String sqlwarn4
SQLWARN5	10 SQLWARN5 PIC X(1)	?	?	String sqlwarn5
SQLWARN6	10 SQLWARN6 PIC X(1)	?	?	String sqlwarn6
SQLWARN7	10 SQLWARN7 PIC X(1)	?	?	String sqlwarn7
SQLEXT	05 SQLEXT PIC X(8)	?	?	String sqlext
LOGGING-STRUKTUR	01 LOGGING-STRUKTUR (group)	?	?	LoggingStrukturDto loggingStruktur
LOG-LEVEL	05 LOG-LEVEL PIC X(5)	?	?	String logLevel /* prefer enum LogLevel {DEBUG, INFO, WARN, ERROR} */
LOG-DEBUG	88 LOG-DEBUG VALUE 'DEBUG'	?	?	boolean isLogDebug() /* derived from logLevel == 'DEBUG' */
LOG-INFO	88 LOG-INFO VALUE 'INFO '	?	?	boolean isLogInfo() /* derived from logLevel == 'INFO ' */
LOG-WARN	88 LOG-WARN VALUE 'WARN '	?	?	boolean isLogWarn() /* derived from logLevel == 'WARN ' */
LOG-ERROR	88 LOG-ERROR VALUE 'ERROR'	?	?	boolean isLogError() /* derived from logLevel == 'ERROR' */
LOG-NACHRICHT	05 LOG-NACHRICHT PIC X(200)	?	?	String logNachricht
LOG-KORRELATIONS-ID	05 LOG-KORRELATIONS-ID PIC X(36)	?	?	String logKorrelationsId /* UUID length 36 recommended */

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
General error	Processing error	Log error to SYSOUT. Set non-zero RC.	catch Exception -> log.error -> step FAILED.

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Generate Java DTOs reflecting groups and fields; use nested DTOs for SQLCA, ABEND-CODES and LOGGING-STRUKTUR.

[ ] Implement mapping layer that converts COBOL layouts to DTOs, handling COMP binary numeric decoding, OCCURS arrays and trimming PIC X fields.

[ ] Introduce enums/constants for abend codes and log levels; implement derived boolean helpers for 88-levels.

[ ] Remove or mark SQLCODE as deprecated in DTOs and update data access code to use Spring DataAccessException mapping.

CRITICAL FINDINGS

SQLCODE exists in SQLCA and is defined as PIC S9(9) COMP, but pattern dictionary advises removal in favor of Spring exceptions ? migration must decide to deprecate or drop this field.

Several numeric fields use COMP (binary) format. Endianness/platform differences must be handled when reading legacy files or converting raw byte streams.

LOG-LEVEL 88-values include padded values ('INFO ' and 'WARN ') ? trimming/preservation of trailing spaces matters when matching conditions.

Timestamps are stored as PIC X(26) with unspecified format; mapping requires agreed parsing/format rules.

No explicit nullability flags provided ? migration should define defaults and null handling for each field.

BUSINESS RULES: BUCHSTRKC.cpy

Type: COPYBOOK | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
BUCHSTRKC.cpy	COPYBOOK	?	?	No UI ? shared copybook str...

1. Overview

Program Name	BUCHSTRKC.cpy
Type	COPYBOOK
Purpose	This copybook defines the booking/transaction record (Buchungssatz) used across multiple banking programs. It is used by BUCHSERV, UEBERWEIG, ZINSBERCH, BATCHBCH, KONTOABFR to provide a consistent layout for bookings including parties, amounts, clearing and return/status data, and to support ISO20.022,00 ?/SEPA compatible fields.
User Interface	No UI ? shared copybook structure
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Shared data structure
Actor / User	Programs that COPY this copybook
Description	This copybook defines shared data used by dependent programs.
Goal	Provide consistent data layout.
Trigger	COPY BUCHSTRKC statement

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
BUCHUNGS-SATZ	01 BUCHUNGS-SATZ	group	
BCH-BUCHUNGS-ID	05 within 01 BUCHUNGS-SATZ	PIC X(36)	
BCH-KONTO-VON	05 within 01 BUCHUNGS-SATZ	PIC 9(10)	
BCH-KONTO-NACH	05 within 01 BUCHUNGS-SATZ	PIC 9(10)	
BCH-IBAN-VON	05 within 01 BUCHUNGS-SATZ	PIC X(22)	
BCH-IBAN-NACH	05 within 01 BUCHUNGS-SATZ	PIC X(22)	
BCH-BETRAG	05 within 01 BUCHUNGS-SATZ	PIC S9(13)V99 COMP-3	
BCH-WAEHRUNG	05 within 01 BUCHUNGS-SATZ	PIC X(3)	
BCH-BUCHUNGSDATUM	05 within 01 BUCHUNGS-SATZ	PIC X(8)	
BCH-VALUTADATUM	05 within 01 BUCHUNGS-SATZ	PIC X(8)	
BCH-VERWENDUNGSZWECK	05 within 01 BUCHUNGS-SATZ	PIC X(140)	
BCH-AUFTRAGGEBER-REF	05 within 01 BUCHUNGS-SATZ	PIC X(35)	
BCH-END-TO-END-ID	05 within 01 BUCHUNGS-SATZ	PIC X(35)	

BUSINESS RULES: BUCHSTRKC.cpy (continued)
Type: COPYBOOK

BUCHUNGS-KONTROLLE	01 BUCHUNGS-KONTROLLE	group	
BCH-TYP	05 within 01 BUCHUNGS-KONTROLLE	PIC X(4)	88-levels: BCH-GUTSCHRIFT='GUTS', BCH-LASTSCHRIFT='LAST', BCH-UEBERWEISUNG='UEBW', BCH-DAUERAUFTRAG='DAUA', BCH-GEBUEHR='GEBR', BCH-ZINS='ZINS', BCH-STORNO='STOR'
BCH-KANAL	05 within 01 BUCHUNGS-KONTROLLE	PIC X(4)	88-levels: BCH-ONLINE='ONLN', BCH-FILIALE='FILI', BCH-BATCH='BATC', BCH-SEPA='SEPA'
BCH-STATUS	05 within 01 BUCHUNGS-KONTROLLE	PIC X(2)	88-levels: BCH-PENDING='PE', BCH-GEBUCHT='GB', BCH-STORNIERT='ST', BCH-ABGELEHNT='AB'
BCH-AUTORISIERT	05 within 01 BUCHUNGS-KONTROLLE	PIC X(1)	88-levels: BCH-AUTH-OK='J', BCH-AUTH-NEIN='N'
BUCHUNGS-CLEARING	01 BUCHUNGS-CLEARING	group	
BCH-CLEARING-ID	05 within 01 BUCHUNGS-CLEARING	PIC X(20)	
BCH-INTERBANK-REF	05 within 01 BUCHUNGS-CLEARING	PIC X(35)	
BCH-TARGET2-REF	05 within 01 BUCHUNGS-CLEARING	PIC X(35)	
BCH-VERARBEITUNGSZEIT	05 within 01 BUCHUNGS-CLEARING	PIC X(6)	
BCH-BATCH-NR	05 within 01 BUCHUNGS-CLEARING	PIC 9(8)	
BCH-SEQUENZ-NR	05 within 01 BUCHUNGS-CLEARING	PIC 9(6)	
BUCHUNGS-RUECKGABE	01 BUCHUNGS-RUECKGABE	group	
BCH-RC	05 within 01 BUCHUNGS-RUECKGABE	PIC 9(4)	88-levels: BCH-OK=0000, BCH-LIMIT-FEHLER=0001, BCH-KONTO-FEHLER=0002, BCH-BETRAG-FEHLER=0003, BCH-DOPPELT=0004, BCH-DB-FEHLER=0099
BCH-FEHLERMSG	05 within 01 BUCHUNGS-RUECKGABE	PIC X(80)	
BCH-WARN-FLAG	05 within 01 BUCHUNGS-RUECKGABE	PIC X(1)	

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	Amount storage and scale: BCH-BETRAG PIC S9(13)V99 COMP-3 -> Signed packed decimal with 2 implied decimal places; must be interpreted as monetary value (scale 2). (data-constraint), 90% confidence

2	Transaction type enumerations: BCH-TYP PIC X(4) with 88-levels (GUTS, LAST, UEBW, DAUA, GEBR, ZINS, STOR) -> Only those values are valid business types; drive processing logic (credit, debit, transfer, standing order, fee, interest, reversal). (data-constraint), 90% confidence
3	Channel enumerations: BCH-KANAL PIC X(4) with 88-levels (ONLN, FILI, BATC, SEPA) -> Indicates originating channel and may control routing/authorization rules. (data-constraint), 90% confidence
4	Status enumerations: BCH-STATUS PIC X(2) with 88-levels (PE, GB, ST, AB) -> Defines processing lifecycle: pending, booked, reversed, rejected. (data-constraint), 90% confidence
5	Authorization flag: BCH-AUTORISIERT PIC X(1) with 88-levels (J,N) -> Indicates whether transaction was authorized (J) or not (N); likely required for booking. (data-constraint), 90% confidence
6	Return code enumerations: BCH-RC PIC 9(4) with 88-levels (0000,0001,0002,0003,0004,0099) -> Numeric response codes for booking result; 0000 = OK, other codes identify specific error types. (data-constraint), 90% confidence
7	IBAN and identifier lengths: PIC X(22) for IBANs, PIC X(35) for references -> Fixed-length fields; values should be trimmed/padded when mapping to variable-length DB columns and validated for allowed characters and formats (IBAN checksum externally). (data-constraint), 80% confidence
8	Date fields format: BCH-BUCHUNGSDATUM, BCH-VALUTADATUM PIC X(8) -> Stored as 8-character strings, expected format is likely YYYYMMDD or DDMMYYYY depending on system conventions - confirm before converting to LocalDate. (data-constraint), 60% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
BUCHUNGS-SATZ	01	?	?	BuchungsSatzDto buchungsSatz
BCH-BUCHUNGS-ID	05 PIC X(36)	?	?	String bchBuchungsId
BCH-KONTO-VON	05 PIC 9(10)	?	?	Long bchKontoVon
BCH-KONTO-NACH	05 PIC 9(10)	?	?	Long bchKontoNach
BCH-IBAN-VON	05 PIC X(22)	?	?	String bchIbanVon
BCH-IBAN-NACH	05 PIC X(22)	?	?	String bchIbanNach
BCH-BETRAG	05 PIC S9(13)V99 COMP-3	?	?	BigDecimal bchBetrag
BCH-WAEHRUNG	05 PIC X(3)	?	?	String bchWaehrung
BCH-BUCHUNGSDATUM	05 PIC X(8)	?	?	String bchBuchungsdatum
BCH-VALUTADATUM	05 PIC X(8)	?	?	String bchValutadatum
BCH-VERWENDUNGSZWECK	05 PIC X(140)	?	?	String bchVerwendungszweck
BCH-AUFTRAGGEBER-REF	05 PIC X(35)	?	?	String bchAuftraggeberRef
BCH-END-TO-END-ID	05 PIC X(35)	?	?	String bchEndToEndId
BUCHUNGS-KONTROLLE	01	?	?	BuchungsKontrolleDto buchungsKontrolle
BCH-TYP	05 PIC X(4)	?	?	String bchTyp
BCH-KANAL	05 PIC X(4)	?	?	String bchKanal
BCH-STATUS	05 PIC X(2)	?	?	String bchStatus

BCH-AUTORISIERT	05 PIC X(1)	?	?	String bchAutorisiert
BUCHUNGS-CLEARING	01	?	?	BuchungsClearingDto buchungsClearing
BCH-CLEARING-ID	05 PIC X(20)	?	?	String bchClearingId
BCH-INTERBANK-REF	05 PIC X(35)	?	?	String bchInterbankRef
BCH-TARGET2-REF	05 PIC X(35)	?	?	String bchTarget2Ref
BCH-VERARBEITUNGSZEIT	05 PIC X(6)	?	?	String bchVerarbeitungsZeit
BCH-BATCH-NR	05 PIC 9(8)	?	?	Integer bchBatchNr
BCH-SEQUENZ-NR	05 PIC 9(6)	?	?	Integer bchSequenzNr
BUCHUNGS-RUECKGABE	01	?	?	BuchungsRueckgabeDto buchungsRueckgabe
BCH-RC	05 PIC 9(4)	?	?	Integer bchRc
BCH-FEHLERMSG	05 PIC X(80)	?	?	String bchFehlerMsg
BCH-WARN-FLAG	05 PIC X(1)	?	?	String bchWarnFlag

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
General error	Processing error	Log error to SYSOUT. Set non-zero RC.	catch Exception -> log.error -> step FAILED.

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Create BUCHSTRKCDto Java class (and nested DTOs for groups: BuchungsSatzDto, BuchungsKontrolleDto, BuchungsClearingDto, BuchungsRueckgabeDto).

[ ] PIC 9(n) -> Integer/Long depending on length (9(10) -> Long; 9(8)/9(6) -> Integer). PIC X(n) -> String.

[ ] COMP-3 (packed decimal) with implied decimals -> BigDecimal (apply scale=2 for V99).

[ ] 88-levels -> Java enums or constants (map to Enum types: Typ, Kanal, Status, Rc, AuthFlag).

[ ] Date fields PIC X(8) -> map to java.time.LocalDate after determining format (likely YYYYMMDD).

[ ] Fixed-length fields -> trim trailing spaces when converting to variable-length DB/VARCHAR columns.

[ ] Consider validation for IBAN (checksum) and currency (ISO 4217) during migration.

BUSINESS RULES: BATCHJCL

Type: JCL-JOB | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
BATCHJCL	JCL-JOB	?	?	No UI ? JCL batch job

1. Overview

Program Name	BATCHJCL
Type	JCL-JOB
Purpose	This nightly batch job orchestrates end-of-day booking (posting) processes for the retail banking system: it prepares standing orders and incoming direct debits, runs the core batch posting program against DB2, and archives/result/logs outputs. It also creates backups of the run/protocol and sends notifications on error conditions.
User Interface	No UI ? JCL batch job
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Execute JCL Batch Job BATCHJCL
Actor / User	Batch scheduler / operations team
Description	Batch job BATCHJCL orchestrates 8 steps.
Goal	Complete all steps successfully (RC=0).
Trigger	JCL job scheduler. Description: BATCH BUCHUNGEN

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
JOBLIB / BANKING.PROD.LOADLIB	JOBLIB DD	PDS (loadlib)	Must be cataloged and accessible for program load
STEP01: SYSTSIN DB2 SYSTEM	TSO/IKJEFT01 runtime	DB2 subsystem connection	DB2 subsystem DB2P reachable; credentials/config available
BANKING.PROD.BATCH.PROTOKOLL	SYSUT1 in STEP02	Sequential (FB)	Must exist for backup copy
BANKING.PROD.BATCH.PROTOKOLL.BACKUP	SYSUT2 in STEP02	Sequential (FB, NEW)	Will be created; catalog/storage available
BANKING.PROD.DAUERAUFTRAEGE.STAGES	SORTIN in STEP03	Sequential (FB)	Must exist and be readable
&&DAUAUFT	SORTOUT in STEP03 and DD in STEP05	Temporary sequential (cataloged transient)	Created in STEP03; must be passed to STEP05 and then deleted
BANKING.PROD.LASTSCHRIFTEN.EINGANG	SORTIN in STEP04	Sequential (FB)	Must exist and be readable
&&LASTDAT	SORTOUT in STEP04 and DD in STEP05	Temporary sequential (cataloged transient)	Created in STEP04; must be passed to STEP05 and then deleted
BANKING.PROD.BATCH.ERGEBNIS.&&DATE	BATCHERG DD in STEP05; SYSUT1 in STEP08	Sequential (FB, dynamic name)	Generated by STEP05; used by STEP08 for archiving
BANKING.PROD.BATCH.FEHLER.&&DATE	BATCHFEH DD in STEP05	Sequential (FB, dynamic name)	Generated by STEP05 for errors
BANKING.PROD.BATCH.ARCHIV	SYSUT2 in STEP08	Tape (MOD)	Tape unit available and writeable; archive retention policy

BANKING.PROD.RUNLIB.LOAD	LIB referenced in SYSTSIN RUN commands	PDS (load/runlib)	Contains DB2-run programs like DSNTEP2, BATCHBCH, BENACHRI
STEPLIB / BANKING.PROD.LOADLIB	STEPLIB DD in STEP05	PDS (loadlib)	Used to load BATCHBCH at runtime

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	STEP01 - DB2 connect: No COND specified -> Always executed at job start to establish DB2 environment (control-flow, Line 1), 98% confidence
2	STEP02 - Backup protocol: No COND specified -> Always executed to copy BANKING.PROD.BATCH.PROTOKOLL to backup dataset (control-flow, Line 2), 98% confidence
3	STEP03 - Prepare standing orders (DAUERAUFTRAEGE): No COND specified -> Always executed; produces temporary dataset &&DAUAUFT (control-flow, Line 3), 98% confidence
4	STEP04 - Prepare direct debits (LASTSCHRIFTEN): No COND specified -> Always executed; produces temporary dataset &&LASTDAT (control-flow, Line 4), 98% confidence
5	STEP05 - Main batch processing (BATCHBCH) suppression rule: COND=(0,LT,STEP03) -> STEP05 is bypassed if RC of STEP03 < 0 (COND true). In practice RC<0 is rare, so STEP05 will run in normal cases. (control-flow, Line 5), 90% confidence
6	STEP06 - Check result placeholder: COND=(0,NE,STEP05) -> STEP06 (IEFBR14) is bypassed if RC of STEP05 != 0. Therefore STEP06 runs only when STEP05 RC == 0. (control-flow, Line 6), 95% confidence
7	STEP07 - Notification on errors: COND=(4,LT,STEP05) -> STEP07 (BENACHRI) is bypassed if RC of STEP05 < 4. It executes only when RC of STEP05 >= 4 (treat as error/severity threshold). (control-flow, Line 7), 95% confidence
8	STEP08 - Archive protocol: COND=(0,LT,STEP05) -> STEP08 is bypassed if RC of STEP05 < 0. As negative RC is unusual, STEP08 will normally run and append results to tape archive. (control-flow, Line 8), 90% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- BANKING.PROD.DAUERAUFTRAEGE.TAGESS - BANKING.PROD.LASTSCHRIFTEN.EINGANG - BANKING.PROD.BATCH.PROTOKOLL - BANKING.PROD.LOADLIB (programs) - DB2 subsystem DB2P	STEP02 copies protokoll to backup STEP03 sorts/filters standing orders into &&DAUAUFT - STEP04 sorts/filters direct debits into &&LASTDAT - STEP05 runs BATCHBCH (DB2-bound) consuming &&DAUAUFT and &&LASTDAT, producing BATCHERG and BATCHFEH - STEP07 runs post-processing notification when error threshold reached - STEP08 appends BATCHERG to tape archive	- BANKING.PROD.BATCH.PROTOKOLL.BACKUP - &&DAUAUFT (transient) - &&LASTDAT (transient) - BANKING.PROD.BATCH.ERGEBNIS.&DATE - BANKING.PROD.BATCH.FEHLER.&DATE - BANKING.PROD.BATCH.ARCHIV (tape)

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
STEP01 - DB2 connect (DSNTEP2 via IKJEFT01)	STEP01	TBD	TBD	step:db2Connect

BUSINESS RULES: BATCHJCL (continued)
Type: JCL-JOB

STEP02 - Copy protocol (IEBGENER SYSUT1->SYSUT2)	STEP02	TBD	TBD	step:copyProtocol
STEP03 - Sort standing orders (SORTIN->&&DAUAUFT)	STEP03	TBD	TBD	step:prepareStandingOrders
STEP04 - Sort direct debits (SORTIN->&&LASTDAT)	STEP04	TBD	TBD	step:prepareDirectDebits
STEP05 - Main batch program (BATCHBCH) consumes &&DAUAUFT &&LASTDAT	STEP05	TBD	TBD	step:batchBchProcessing
STEP07 - Notification (BENACHRI)	STEP07	TBD	TBD	step:sendNotification
STEP08 - Archive (IEBGENER SYSUT1->Tape)	STEP08	TBD	TBD	step:archiveProtocol

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
RC = 0	Success	Normal completion	ExitStatus.COMPLETED
RC = 4	Warning/Recoverable error threshold (treated as error for notification)	Triggers STEP07 notification when RC >=4	ExitStatus.WARNING or ExitStatus.FAILED depending on policy
RC > 4	Failure / severe error	Job should surface failure and halt downstream processing where COND semantics indicate	ExitStatus.FAILED
Abend/negative RC	System abend/abnormal termination	Immediate job failure; downstream steps usually bypassed per COND	throw JobExecutionException

9. Technical Dependencies

Dependency Type	Name / Identifier	Direction	Notes / Migration Action
Scheduler	JCL Job Scheduler	INVOKES	Replace with Spring Batch @Scheduled or CI/CD pipeline trigger.

10. Ready for Implementation ? Checklist

[ ] Define @Scheduled cron to run nightly (23:00) or leave to external scheduler and expose as Spring Batch job

[ ] Implement Job as Spring Batch @Job composed of sequential @Step beans mirroring STEP02..STEP08

[ ] Implement FlatFileItemReader/Writer for each file with precise LRECL/RECFM mapping; use temp files for && names and JobParameters for dynamic output names (&DATE)

[ ] Implement a StepExecutionListener/JobExecutionListener that maps mainframe RCs to ExitStatus and stores RCs in JobExecutionContext for conditional flows

[ ] Re-implement or adapt DB2-bound logic from BATCHBCH into a Spring Batch ItemProcessor or delegating service using JDBC/stored procedures; if not possible, provide an integration/wrapping strategy to call the mainframe program

BUSINESS RULES: BATCHJCL (continued)

Type: JCL-JOB

[ ] Replace tape/archive with configurable archive writer (S3/Object
storage/enterprise archive) and implement retention/append behavior

[ ] Implement a Decider to model COND semantics (e.g., run notification when rc >=4)
and route job flow accordingly

CRITICAL FINDINGS

BUSINESS RULES: ZINSJCL

Type: JCL-JOB | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
ZINSJCL	JCL-JOB	?	?	No UI ? JCL batch job

1. Overview

Program Name	ZINSJCL
Type	JCL-JOB
Purpose	This job orchestrates the monthly interest calculation (ZINSBERECHNUNG) for all accounts: crediting interest, debiting interest, and computing capital gains tax reporting. It performs database backups, runs DB2 programs to load and validate data, executes the interest calculation, validates results, produces tax reports, and archives protocol/output files.
User Interface	No UI ? JCL batch job
Data Source	TBD ? see source code

2. Business Use Case

Use Case	Execute JCL Batch Job ZINSJCL
Actor / User	Batch scheduler / operations team
Description	Batch job ZINSJCL orchestrates 6 steps.
Goal	Complete all steps successfully (RC=0).
Trigger	JCL job scheduler. Description: ZINSBERECHNUNG

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
JOBLIB / STEPLIB	//JOBLIB DD DISP=SHR,DSN=BANKING.PROD.LOADLIB //STEPLIB DD DISP=SHR,DSN=BANKING.PROD.LOADLIB	PDS/LOADLIB	Must be accessible to load DB2 run-time and application programs (BANKING.PROD.LOADLIB)
KONTEN_BK	//KONTEN_BK DD DISP=(NEW,CATLG),DSN=BANKING.PROD.BACKUP.KONTEN.&DATE,UNIT=TAPE	Sequential (TAPE)	Created by STEP01; must be successfully cataloged
BUCH_BK	//BUCH_BK DD DISP=(NEW,CATLG),DSN=BANKING.PROD.BACKUP.BUCHUNGEN.&DATE,UNIT=TAPE	Sequential (TAPE)	Created by STEP01; must be successfully cataloged
ZINSPROT	//ZINSPROT DD DISP=(NEW,CATLG,DELETE),DSN=BANKING.PROD.ZINS.PROTOKOLL.&DATE,UNIT=SYSDA,SPACE=(CYL,(20,10)),DCB=(RECFM=FB,LRECL=200,BLKSIZE=20000)	Sequential (DISK)	Created by STEP03 (interest calculation log); must be available for STEP04/STEP06
ZINSFEHLER	//ZINSFEHLER DD DISP=(NEW,CATLG,DELETE),DSN=BANKING.PROD.ZINS.FEHLER.&DATE,UNIT=SYSDA,SPACE=(CYL,(2,1)),DCB=(RECFM=FB,LRECL=120,BLKSIZE=12000)	Sequential (DISK)	Created by STEP03 to capture errors

BUSINESS RULES: ZINSJCL (continued)
Type: JCL-JOB

SORTIN / SORTOUT	//SORTIN DD DISP=SHR,DSN=BANKING.PROD.ZINS.PROTOKOLL.&DATE //SORTOUT DD SYSOUT=*	Sequential (DISK)	SORTIN must reference the ZINSPROT created in STEP03
STEUERBERICHT	//STEUERBERICHT DD DISP=(NEW,CATLG),DSN=BANKING.PROD.STEUER.BERICHT.&DATE,UNIT=SYSDA,SPACE=(CYL,(5,2)),DCB=(RECFM=FB,LRECL=200,BLKSIZE=20000)	Sequential (DISK)	Created by STEP05 when tax reporting runs
SYSUT2 (ARCHIVE)	//SYSUT2 DD DISP=MOD,DSN=BANKING.PROD.ZINS.ARCHIV,UNIT=TAPE,LABEL=(,SL)	Sequential (TAPE)	Append-only archive target for STEP06; must be writable
RUNLIB	LIB('BANKING.PROD.RUNLIB.LOAD') referenced in SYSTSIN for IKJEFT01 runs	PDS/LOADLIB	Must contain DSNTEP2 and ZINSBERCH load modules

4. Database Access

SQL ?

```
COPY TABLESPACE BANKING.KONTEN_TS COPYDDN KONTEN_BK PARALLEL 2; COPY TABLESPACE BANKING.BUCHUNGEN_TS COPYDDN BUCH_BK
```

SQL ?

```
SELECT COUNT(*) AS AKTIVE_FREISTELLUNGEN, SUM(FREISTELLUNGSBETRAG) AS GESAMTBETRAG FROM FREISTELLUNGSAUFTRAEGE WHERE A
```

SQL ?

```
SELECT STEUER_JAHR, SUM(GESAMTBETRAG) AS KEST_GESAMT, SUM(SOLI_BETRAG) AS SOLI_GESAMT, COUNT(*) AS ANZAHL_BUCHUNGEN FR
```

5. Business Rules (derived from source code)

Rule	Description
1	Backup before processing: Always before calculation (STEP01 executes unconditionally at job start) -> Create copies of KONTEN_TS and BUCHUNGEN_TS to KONTEN_BK and BUCH_BK (tape datasets) (data-protection, Line 1), 99% confidence
2	Freistellungsauftraege validation gating: None (STEP02 runs unconditionally after STEP01) -> Run DB2 select to count active exemptions and sum amounts; results determine business visibility but do not automatically abort (validation, Line 2), 97% confidence
3	Run interest calculation only when STEP02 RC >= 4: COND=(4,LT,STEP02) on STEP03 (skip STEP03 if RC(STEP02) < 4) -> STEP03 is bypassed when STEP02 return code is less than 4; STEP03 runs only when STEP02 RC >= 4 (control-flow, Line 3), 92% confidence
4	Validation and reporting execute regardless of success criteria (practically always): STEP04 uses COND=(0,LT,STEP03) which is never true for normal RC values >=0 -> STEP04 (SORT error extraction) will run in normal cases; practically always executed unless negative RC semantics present (control-flow, Line 4), 85% confidence
5	Tax reporting and archive run only when STEP03 RC >= 4: STEP05 and STEP06 have COND=(4,LT,STEP03) and COND=(0,LT,STEP03) respectively; STEP05 uses COND=(4,LT,STEP03) (skip if STEP03 RC <4) -> STEP05 (tax report) will be bypassed if STEP03 RC <4; STEP06 uses COND=(0,LT,STEP03) which effectively allows execution in normal cases (control-flow, Line 5), 88% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- BANKING.PROD.LOADLIB (JOBLIB/STEPLIB) - BANKING.PROD.KONTEN_TS (source DB2 tablespace) - BANKING.PROD.BUCHUNGEN_TS (source	- STEP01: DB2 utility copies tablespaces to tape (KONTEN_BK, BUCH_BK) - STEP02: Run DB2 SELECT to validate freistellungsauftraege	- BANKING.PROD.BACKUP.KONTEN.&DATE (tape) - BANKING.PROD.BACKUP.BUCHUNGEN.&DATE (tape) - BANKING.PROD.ZINS.PROTOKOLL.&DATE

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
STEP01: KONTEN_BK BUCH_BK	//KONTEN_BK, //BUCH_BK	BANKING.KONTEN_TS / BANKING.BUCHUNGEN_TS (tablespaces)	N/A (full tablespace copy)	step01.backupFiles (List<String>)
STEP02: FREISTELLUNGSAUFTRAEGE SELECT	SYSTSIN SYSIN for IEFBR14	FREISTELLUNGSAUFTRAEGE	AKTIV, FREISTELLUNGSBETRAG, STEUER_JAHR	step02.freistellungCounts.totalActive, step02.freistellungCounts.totalAmount
STEP03: ZINSPROT / ZINSFEHLER outputs	//ZINSPROT, //ZINSFEHLER	Various accounting tables updated by ZINSBERCH	Various booking/result fields (TBD in application spec)	step03.protocolFilePath, step03.errorFilePath
STEP04: SORTIN -> SORTOUT error extract	//SORTIN, //SORTOUT	N/A (protocol file parsing)	Record positions (e.g. error code at pos 49,2)	step04.errorSummary
STEP05: STEUERBERICHT generation	//STEUERBERICHT	KAPITALERTRAGSTEUER	STEUER_JAHR, GESAMTBETRAG, SOLI_BETRAG, BUCHUNGSDATUM	step05.taxReport
STEP06: Archive append	//SYSUT2	N/A (tape archive)	N/A	step06.archiveAppendResult

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
RC = 0	Success	Normal completion	ExitStatus.COMPLETED
RC = 8	Failure	Steps flushed	Spring Batch step FAILED
RC > 8	Abend	Job abends	throw JobExecutionException
COND=(4,LT,STEP02) evaluated true	STEP03 skipped because STEP02 RC < 4	Interest calculation not executed; downstream behavior depends on other COND settings	Skip step by not launching step bean; record as ExitStatus.NOOP
Missing or inaccessible datasets (e.g., JOBLIB, RUNLIB, target datasets)	Allocation failure	Step abends with non-zero RC	fail JobExecution with resource allocation exception

9. Technical Dependencies

Dependency Type	Name / Identifier	Direction	Notes / Migration Action
Scheduler	JCL Job Scheduler	INVOKES	Replace with Spring Batch @Scheduled or CI/CD pipeline trigger.

10. Ready for Implementation ? Checklist

[ ] Define @Scheduled cron to trigger monthly on last banking day or wire into scheduler as needed.

[ ] Implement JobExecutionListener to capture and persist step return codes and emulate JCL COND semantics to decide step execution.

[ ] Create Tasklets/Steps: backup (DB2 COPY), validation SELECT, interest calculation (reimplement ZINSBERCH), protocol processing (filter/sort), tax reporting SELECT, and archive writer.

BUSINESS RULES: ZINSJCL (continued)
Type: JCL-JOB

CRITICAL FINDINGS

COND logic appears unusual: STEP03 has COND=(4,LT,STEP02), meaning STEP03 is skipped when STEP02 RC < 4 (including RC=0). This likely inverts intended gating and must be confirmed with business/ops.

STEP04 and STEP06 use COND=(0,LT,STEP03) which is effectively always false for non-negative RCs, so they will almost always run ? confirm intended behavior.

Use of tape datasets (UNIT=TAPE, DISP=MOD/CATLG) implies archival and backup processes that require special handling in target environment (object store or long-term storage).

ZINSBERCH is an external compiled program invoked under IKJEFT01; migration requires access to source or reimplementation in Java to run within Spring Batch.

SORT step relies on fixed-position fields (e.g., include=(49,2,CH,EQ,C'FE')) ? record layout must be documented to correctly implement filters in Spring Batch.

BUSINESS RULES: KONTOMAP

Type: BMS-MAP | 14 lines | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
KONTOMAP	BMS-MAP	?	14	CICS BMS Map (3270 terminal...

1. Overview

Program Name	KONTOMAP
Type	BMS-MAP
Purpose	This screen is used by bank clerks or terminal users to view and edit account master data (account number, owner, balances, limits) and to trigger actions (search, transactions, print). It collects identifying data (date/time, terminal id, customer/account numbers, IBAN, currency) and monetary fields needed to display or update account status.
User Interface	CICS BMS Map (3270 terminal) -> REST API
Data Source	TBD ? see source code

2. Business Use Case

Use Case	CICS Screen Dialog: KONTOMAP
Actor / User	Bank clerk / terminal user
Description	User enters data on the KONTOMAP screen and receives results.
Goal	Provide interactive dialog for the KONTOMAP business function.
Trigger	CICS transaction triggered by terminal user

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
DFHMDF NAME=DATUMF	WS_DATUMF	PIC X(10)	Date format DD.MM.YYYY or ISO YYYY-MM-DD; max length 10; required if operation needs date
DFHMDF NAME=UHRZEITF	WS_UHRZEITF	PIC X(8)	Time format HH:mm:ss or HHmmss; max length 8
DFHMDF NAME=TERMID	WS_TERMID	PIC 9(4)	Numeric only; max length 4; mapable to Long
DFHMDF NAME=KTONRF	WS_KTONRF	PIC 9(10)	Numeric only; max length 10; mapable to Long
DFHMDF NAME=WAEHRUF	WS_WAEHRUF	PIC X(3)	3-letter currency code A-Z; max length 3
DFHMDF NAME=INHABERF	WS_INHABERF	PIC X(50)	Free text; max length 50; no control characters
DFHMDF NAME=KDNRF	WS_KDNRF	PIC 9(10)	Numeric only; max length 10; mapable to Long
DFHMDF NAME=IBANF	WS_IBANF	PIC X(22)	Alphanumeric; validate IBAN structure where possible; max length 22
DFHMDF NAME=KTOTYPF	WS_KTOTYPF	PIC X(15)	Account type; max length 15; alphanumeric and punctuation allowed
DFHMDF NAME=SALDOF	WS_SALDOF	PIC numeric width 17	Monetary numeric; parseable to BigDecimal; allow sign and decimal separator; max width 17

BUSINESS RULES: KONTOMAP (continued)
Type: BMS-MAP

DFHMDF NAME=VERFUEGF	WS_VERFUEGF	PIC X(17)	Available amount; numeric or placeholder; if numeric parseable to BigDecimal; max length 17
DFHMDF NAME=DISPOF	WS_DISPOF	PIC X(17)	Disp limit; numeric or placeholder; if numeric parseable to BigDecimal; max length 17
DFHMDF NAME=MELDUNGF	WS_MELDUNGF	PIC X(79)	PF-key commands or messages; allow PF1..PF4 tokens; free text up to 79 chars
DFHMDF NAME=ERRORF	WS_ERRORF	PIC X(79)	Error message field; max length 79; no control characters

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	DATUMF Field Validation: DFHMDF NAME=DATUMF ATTRB=UNPROT - PIC X(10); must be a valid date in DD.MM.YYYY or ISO YYYY-MM-DD representation and length <=10 -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
2	UHRZEITF Field Validation: DFHMDF NAME=UHRZEITF ATTRB=UNPROT - PIC X(8); must match time format HH:mm:ss (24h) or HHmmss; length = 8 -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
3	TERMID Field Validation: DFHMDF NAME=TERMID ATTRB=UNPROT - PIC 9(4); numeric only; length <=4; map to Long -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
4	KTONRF Field Validation: DFHMDF NAME=KTONRF ATTRB=UNPROT - PIC 9(10); numeric only; length <=10; no leading non-digits -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
5	WAEHRUF Field Validation: DFHMDF NAME=WAEHRUF ATTRB=UNPROT - PIC X(3); must be 3-letter ISO currency code A-Z; length =3 -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
6	INHABERF Field Validation: DFHMDF NAME=INHABERF ATTRB=UNPROT - PIC X(50); text field; max length 50; disallow control chars; trim leading/trailing spaces -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
7	KDNRF Field Validation: DFHMDF NAME=KDNRF ATTRB=UNPROT - PIC 9(10); numeric only; length <=10; map to Long -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
8	IBANF Field Validation: DFHMDF NAME=IBANF ATTRB=UNPROT - PIC X(22); alphanumeric; validate as IBAN-like: starts with 2 letters followed by digits/letters; length <=22 (map to String); optionally run IBAN checksum where possible -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 85% confidence
9	KTOTYPF Field Validation: DFHMDF NAME=KTOTYPF ATTRB=UNPROT - PIC X(15); account type description; max length 15; allow alphanumeric and common punctuation -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
10	SALDOF Field Validation: DFHMDF NAME=SALDOF ATTRB=UNPROT - PIC 9/EDIT like width 17; numeric with optional sign and decimal separator (point or comma); parseable to BigDecimal; total characters <=17 -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence

BUSINESS RULES: KONTOMAP (continued)
Type: BMS-MAP

11	VERFUEGF Field Validation: DFHMDF NAME=VERFUEGF ATTRB=UNPROT - PIC X(17); expected numeric amount (available) or '-' placeholders; if numeric: parseable to BigDecimal; max length 17 -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 85% confidence
12	DISPOF Field Validation: DFHMDF NAME=DISPOF ATTRB=UNPROT - PIC X(17); expected numeric limit (may include sign/decimal) or '---'; if numeric: parseable to BigDecimal; max length 17 -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 85% confidence
13	MELDUNGF Field Validation: DFHMDF NAME=MELDUNGF ATTRB=UNPROT - PIC X(79); used for PF-key commands and messages; max length 79; if contains PF command only allow PF1..PF4 tokens or free text for messages -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 80% confidence
14	ERRORF Field Validation: DFHMDF NAME=ERRORF ATTRB=UNPROT - PIC X(79); reserved for error messages; max length 79; should not contain control characters -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation), 90% confidence
15	Initial Screen Display: Screen entry/first render; EIBCALEN unknown - initial display should present empty or default values -> On initial display populate fields with defaults; do not call backend services unless user triggers action. (screen-flow), 90% confidence
16	EIBCALEN=0 First Call Init: EIBCALEN == 0 (first call) - treat as form init -> Initialize WS_ variables, clear MELDUNGF/ERRORF, set focus to primary input; do not call downstream service. (screen-flow), 90% confidence
17	Service Call Trigger: User presses PF-key (e.g., PF1=search PF2=transactions PF3=end PF4=print) or submits form; inputs must pass validation before service invocation -> If validation passes, map WS_ fields to service DTO and call backend service; on return populate screen fields; on validation failure reject submission and populate MELDUNGF/ERRORF. (screen-flow), 90% confidence
18	Error Display Rule: Any downstream error or validation failure -> Populate ERRORF/MELDUNGF with human-readable message, highlight offending field(s), and do not proceed to further service calls. (screen-flow), 90% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- BMS input fields (14 UNPROT) - DFHCOMMAREA	- EIBCALEN check - Move input to WS - Perform per-field validation rules - If valid, map to KontomapDto and call backend service - Populate output fields and MELDUNGF/ERRORF as needed	- BMS display fields (0 PROT) - FEHLER message field (ERRORF / MELDUNGF)

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
DFHMDF NAME=DATUMF	WS_DATUMF	?	?	datumf (LocalDate) in KontomapDto
DFHMDF NAME=UHRZEITF	WS_UHRZEITF	?	?	uhrzeitf (String) in KontomapDto
DFHMDF NAME=TERMID	WS_TERMID	?	?	termid (Long) in KontomapDto
DFHMDF NAME=KTONRF	WS_KTONRF	?	?	ktonrf (Long) in KontomapDto
DFHMDF NAME=WAEHRUF	WS_WAEHRUF	?	?	waehruf (String) in KontomapDto
DFHMDF NAME=INHABERF	WS_INHABERF	?	?	inhaberf (String) in KontomapDto

BUSINESS RULES: KONTOMAP (continued)
Type: BMS-MAP

DFHMDF NAME=KDNRF	WS_KDNRF	?	?	kdnrf (Long) in KontomapDto
DFHMDF NAME=IBANF	WS_IBANF	?	?	ibanf (String) in KontomapDto
DFHMDF NAME=KTOTYPF	WS_KTOTYPF	?	?	ktotypf (String) in KontomapDto
DFHMDF NAME=SALDOF	WS_SALDOF	?	?	saldof (BigDecimal) in KontomapDto
DFHMDF NAME=VERFUEGF	WS_VERFUEGF	?	?	verfuegf (String) in KontomapDto
DFHMDF NAME=DISPOF	WS_DISPOF	?	?	dispof (String) in KontomapDto
DFHMDF NAME=MELDUNGF	WS_MELDUNGF	?	?	meldungf (String) in KontomapDto
DFHMDF NAME=ERRORF	WS_ERRORF	?	?	errorf (String) in KontomapDto

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
SQLCODE=0	Found	Show result	HTTP 200
SQLCODE=+100	Not found	Show FEHLER	HTTP 404
SQLCODE<0	DB error	Show error	HTTP 500
Invalid input	Validation error	Highlight field	HTTP 400

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Implement KontomapDto

[ ] Write REST controller

BUSINESS RULES: UEBWMAP

Type: BMS-MAP | 13 lines | [AI-generated | Azure OpenAI (EU)]

Program	Type	Risk	Lines	Interface
UEBWMAP	BMS-MAP	?	13	CICS BMS Map (3270 terminal...

1. Overview

Program Name	UEBWMAP
Type	BMS-MAP
Purpose	This screen supports initiating and confirming a bank transfer (Überweisung). A terminal user (bank clerk or customer via terminal) enters account, beneficiary, amount, execution date and usage information; the screen collects transfer details and displays status/error messages.
User Interface	CICS BMS Map (3270 terminal) -> REST API
Data Source	TBD ? see source code

2. Business Use Case

Use Case	CICS Screen Dialog: UEBWMAP
Actor / User	Bank clerk / terminal user
Description	User enters data on the UEBWMAP screen and receives results.
Goal	Provide interactive dialog for the UEBWMAP business function.
Trigger	CICS transaction triggered by terminal user

3. Input Data

Field Name (COBOL)	Working Storage Variable	Type / Format	Validation Rule
DFHMDF NAME=AUFTKTNF	WS_AUFTKTNF	PIC X(10)	String; max length 10; required; alphanumeric; no control chars
DFHMDF NAME=AUFTNAMF	WS_AUFTNAMF	PIC X(40)	String; max length 40; required; printable chars
DFHMDF NAME=AUFTSALF	WS_AUFTSALF	PIC X(25)	String; max length 25; printable chars
DFHMDF NAME=EMPFNAMF	WS_EMPFNAMF	PIC X(40)	String; max length 40; required; printable chars
DFHMDF NAME=EMPFIBANF	WS_EMPFIBANF	PIC X(22)	String; max length 22; IBAN format validation; uppercase letters and digits; checksum validated
DFHMDF NAME=EMPFBICF	WS_EMPFBICF	PIC X(11)	String; max length 11; optional; if present must match BIC/SWIFT pattern (8 or 11 uppercase alnum)
DFHMDF NAME=BETRAGF	WS_BETRAGF	PIC 9(12)V99	BigDecimal; numeric; up to 12 integer digits and 2 decimals; non-negative; scale 2
DFHMDF NAME=AUSFDATF	WS_AUSFDATF	PIC 9(8)	String/Date; format YYYYMMDD; valid calendar date; business-date rules apply
DFHMDF NAME=VERWZWKF	WS_VERWZWKF	PIC X(70)	String; max length 70; printable chars; free text purpose
DFHMDF NAME=STATUSF	WS_STATUSF	PIC X(79)	String; max length 79; contains PF-key / command tokens; validate allowed commands (AUSFUEHREN, ABBRECHEN, LOESCHEN) or blank

BUSINESS RULES: UEBWMAP (continued)
Type: BMS-MAP

DFHMDF NAME=ERRORF	WS_ERRORF	PIC X(79)	String; max length 79; message field; printable chars or blank
DFHMDF NAME=BUCHIDAF	WS_BUCHIDAF	PIC X(50)	String; max length 50; confirmation message; printable chars
DFHMDF NAME=BETCHKF	WS_BETCHKF	PIC X(25)	String; max length 25; optional batch/check identifier; printable chars

4. Database Access

No DB2 SQL statements detected in source analysis.

5. Business Rules (derived from source code)

Rule	Description
1	AUFTKTNF Field Validation: DFHMDF NAME=AUFTKTNF ATTRB=UNPROT - PIC X(10); max length 10; required; alphanumeric account identifier; no control characters -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 1), 90% confidence
2	AUFTNAMF Field Validation: DFHMDF NAME=AUFTNAMF ATTRB=UNPROT - PIC X(40); max length 40; required; printable characters only -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 2), 90% confidence
3	AUFTSALF Field Validation: DFHMDF NAME=AUFTSALF ATTRB=UNPROT - PIC X(25); max length 25; optional/structured sender details; printable characters -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 3), 90% confidence
4	EMPFNAMF Field Validation: DFHMDF NAME=EMPFNAMF ATTRB=UNPROT - PIC X(40); max length 40; required; beneficiary name; printable characters -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 4), 90% confidence
5	EMPFIBANF Field Validation: DFHMDF NAME=EMPFIBANF ATTRB=UNPROT - PIC X(22); max length 22; expected IBAN format (country code + up to 34 alphanumerics; here constrained to 22); uppercase letters and digits; checksum validated -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 5), 90% confidence
6	EMPFBICF Field Validation: DFHMDF NAME=EMPFBICF ATTRB=UNPROT - PIC X(11); max length 11; optional field; if present must match BIC/SWIFT pattern (8 or 11 uppercase alnum) -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 6), 90% confidence
7	BETRAGF Field Validation: DFHMDF NAME=BETRAGF ATTRB=UNPROT - PIC 9(12)V99 (display width 14); numeric; up to 12 integer digits and 2 fractional digits; non-negative; scale 2 for cents -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 7), 90% confidence
8	AUSFDATF Field Validation: DFHMDF NAME=AUSFDATF ATTRB=UNPROT - PIC 9(8) (display width 8); expected date in YYYYMMDD; must be a valid calendar date; may be checked against business rules (not in past if required) -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 8), 90% confidence
9	VERWZWKF Field Validation: DFHMDF NAME=VERWZWKF ATTRB=UNPROT - PIC X(70); max length 70; printable characters; may be segmented/structured free text for usage/purpose -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 9), 90% confidence

BUSINESS RULES: UEBWMAP (continued)
Type: BMS-MAP

10	STATUSF Field Validation: DFHMDF NAME=STATUSF ATTRB=UNPROT - PIC X(79); max length 79; expected to contain PF-key indicators or command tokens (allowed tokens: PF1/ENTER=AUSFUEHREN, PF3=ABBRECHEN, PF5=LOESCHEN) or be blank; validate that any command is in allowed set -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 10), 90% confidence
11	ERRORF Field Validation: DFHMDF NAME=ERRORF ATTRB=UNPROT - PIC X(79); max length 79; message field; allow blank or printable characters; do not accept control chars -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 11), 90% confidence
12	BUCHIDAF Field Validation: DFHMDF NAME=BUCHIDAF ATTRB=UNPROT - PIC X(50); max length 50; confirmation/message field; allow printable characters; typically read-only but here UNPROT so accept but validate length -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 12), 90% confidence
13	BETCHKF Field Validation: DFHMDF NAME=BETCHKF ATTRB=UNPROT - PIC X(25); max length 25; optional check/batch identifier; printable characters only -> Reject submission; populate MELDUNGF/ERRORF with error message; do not call downstream service. (validation, Line 13), 90% confidence
14	Initial Screen Display: Screen first rendered for transaction; EIBCALEN may be 0 on initial call -> Populate default values in WS_*; display UEBWMAP with empty or default fields; do not invoke backend service until user action (flow, Line 14), 95% confidence
15	EIBCALEN=0 First Call Init: EIBCALEN=0 indicates first invocation; no DFHCOMMAREA present -> Initialize working storage (WS_*) and UI controls; set focus to first input field (AUFTKTNF); return screen without service call (flow, Line 15), 95% confidence
16	Service Call Trigger: User submits ENTER or PF1/selection mapped to 'AUSFUEHREN' and all field validations pass -> Call downstream service/API with mapped DTO; if service succeeds populate BUCHIDAF and STATUSF; on failure populate ERRORF and return to screen (flow, Line 16), 95% confidence
17	Error Display Rule: Any validation failure or backend error -> Reject submission; populate ERRORF (or MELDUNGF) with user-friendly error text and highlight offending field(s); do not call downstream service when validation fails; return HTTP 400 for validation, map service errors to 500/404 as appropriate (flow, Line 17), 95% confidence

6. Data Flow (Input -> Processing -> Output)

INPUT	PROCESSING	OUTPUT
- BMS input fields (13 UNPROT) - DFHCOMMAREA	- EIBCALEN check - Move input to WS - Service call - Populate output	- BMS display fields (0 PROT) - FEHLER message field

7. Field Mapping (Source / Screen -> Working Storage -> DB Column)

Source Field (Screen/JCL)	Working Storage	DB Table	DB Column	Java DTO Field
DFHMDF NAME=AUFTKTNF	WS_AUFTKTNF	?	?	auftktnf (String) in UebwmapDto
DFHMDF NAME=AUFTNAMF	WS_AUFTNAMF	?	?	auftnamf (String) in UebwmapDto
DFHMDF NAME=AUFTSALF	WS_AUFTSALF	?	?	auftsalf (String) in UebwmapDto
DFHMDF NAME=EMPFNAMF	WS_EMPFNAMF	?	?	empfnamf (String) in UebwmapDto
DFHMDF NAME=EMPFIBANF	WS_EMPFIBANF	?	?	empfibanf (String) in UebwmapDto
DFHMDF NAME=EMPFBICF	WS_EMPFBICF	?	?	empfbicf (String) in UebwmapDto

BUSINESS RULES: UEBWMAP (continued)

Type: BMS-MAP

DFHMDF NAME=BETRAGF	WS_BETRAGF	?	?	betragf (BigDecimal) in UebwmapDto
DFHMDF NAME=AUSFDATF	WS_AUSFDATF	?	?	ausfdatf (String) in UebwmapDto
DFHMDF NAME=VERWZWKF	WS_VERWZWKF	?	?	verwzwkf (String) in UebwmapDto
DFHMDF NAME=STATUSF	WS_STATUSF	?	?	statusf (String) in UebwmapDto
DFHMDF NAME=ERRORF	WS_ERRORF	?	?	errorf (String) in UebwmapDto
DFHMDF NAME=BUCHIDAF	WS_BUCHIDAF	?	?	buchidaf (String) in UebwmapDto
DFHMDF NAME=BETCHKF	WS_BETCHKF	?	?	betchkf (String) in UebwmapDto

8. Error Cases & SQLCODE Handling

Condition / SQLCODE	Meaning	Mainframe Behavior	Java Migration
SQLCODE=0	Found	Show result	HTTP 200
SQLCODE=+100	Not found	Show FEHLER	HTTP 404
SQLCODE<0	DB error	Show error	HTTP 500
Invalid input	Validation error	Highlight field	HTTP 400

9. Technical Dependencies

10. Ready for Implementation ? Checklist

[ ] Implement UebwmapDto

[ ] Write REST controller

DEAD CODE ANALYSIS | Unused Artefact Detection

0 UNUSED (loeschen) | 3 REVIEW (pruefen) | 7 EXTERN (dokumentieren)

REVIEW - MANUAL VERIFICATION REQUIRED

REVIEW: KONTOABFR

REVIEW: Pruefen ob dieses Programm in einer JCL ausserhalb des ZIP vorkommt.

REVIEW: UEBERWEIG

REVIEW: Pruefen ob dieses Programm in einer JCL ausserhalb des ZIP vorkommt.

REVIEW: BATCHBCH

REVIEW: Pruefen ob dieses Programm in einer JCL ausserhalb des ZIP vorkommt.

EXTERNAL DEPENDENCIES - NOT IN ZIP

EXTERN: KONTOABFR -> CALL 'KONTOPRU'

Aufgerufenes Programm 'KONTOPRU' ist nicht im analysierten ZIP vorhanden.

EXTERN: UEBERWEIG -> CALL 'IBANPRU'

Aufgerufenes Programm 'IBANPRU' ist nicht im analysierten ZIP vorhanden.

EXTERN: BATCHJCL/STEP03 -> PGM=SORT

JCL-Step ruft Programm 'SORT' auf, das nicht im ZIP vorhanden ist.

EXTERN: BATCHJCL/STEP04 -> PGM=SORT

JCL-Step ruft Programm 'SORT' auf, das nicht im ZIP vorhanden ist.

EXTERN: BATCHJCL/STEP06 -> PGM=IEFBR14

JCL-Step ruft Programm 'IEFBR14' auf, das nicht im ZIP vorhanden ist.

EXTERN: ZINSJCL/STEP01 -> PGM=DSNUTILB

JCL-Step ruft Programm 'DSNUTILB' auf, das nicht im ZIP vorhanden ist.

EXTERN: ZINSJCL/STEP04 -> PGM=SORT

JCL-Step ruft Programm 'SORT' auf, das nicht im ZIP vorhanden ist.

BMS-MAP: KONTOMAP

24x80 Bildschirm | 14 Eingabefelder | 0 Anzeigefelder | DTO: KontomapDto

PURPOSE AND FUNCTION

CICS BMS-Map 'KONTOMAP' definiert den Bildschirm-Aufbau des 3270-Terminal-Screens. Datei: KONTOMAP.bms. Jedes Eingabefeld entspricht einem REST-API Parameter oder DTO-Feld in der Migration.

SCREEN FIELDS (DFHMDF definitions)

FIELD NAME	POS	LEN	TYP	JAVA TYPE	MEANING
DATUMF	2/1	10	INPUT	LocalDate	Beschriftung: 'KABF'. Eingabefeld (UNPROT). Laenge: 10 ..
UHRZEITF	2/12	8	INPUT	String	Eingabefeld (UNPROT). Laenge: 8 Zeichen.
TERMID	2/79	4	INPUT	Long	Beschriftung: 'TERMINAL-ID:'. Eingabefeld (UNPROT). Lae..
KTONRF	5/17	10	INPUT	Long	Beschriftung: 'KONTONUMMER'. Eingabefeld (UNPROT). Laen..
WAEHRUF	5/46	3	INPUT	String	Beschriftung: 'WAEHRUNG:'. Eingabefeld (UNPROT). Laenge..
INHABERF	7/17	50	INPUT	String	Beschriftung: 'KONTOINHABER'. Eingabefeld (UNPROT). Lae..
KDNRF	8/17	10	INPUT	Long	Beschriftung: 'KUNDEN-NR'. Eingabefeld (UNPROT). Laenge..
IBANF	10/17	22	INPUT	String	Beschriftung: 'IBAN'. Eingabefeld (UNPROT). Laenge: 22 ..
KTOTYPF	11/17	15	INPUT	String	Beschriftung: 'KONTOTYP'. Eingabefeld (UNPROT). Laenge:..
SALDOF	14/17	17	INPUT	BigDecimal	Beschriftung: 'AKTUELLER SALDO'. Eingabefeld (UNPROT). ..
VERFUEGF	15/17	17	INPUT	String	Beschriftung: 'VERFUEGBAR'. Eingabefeld (UNPROT). Laeng..
DISPOF	16/17	17	INPUT	String	Beschriftung: 'DISPO-LIMIT'. Eingabefeld (UNPROT). Laen..
MELDUNGF	23/1	79	INPUT	String	Beschriftung: 'PF1=SUCHEN PF2=UMSAETZE PF3=BEENDEN P..
ERRORF	24/1	79	INPUT	String	Eingabefeld (UNPROT). Laenge: 79 Zeichen.

MIGRATION HINTS (CICS -> Spring Boot)

- CICS-Screen als REST-Endpoint: @PostMapping mit @RequestBody KontomapDto.
- DTO generieren mit 14 @NotNull/@Valid Feldern fuer Eingaben.
- Anzeigefelder als Response-DTO (kein Setter noetig, nur @JsonProperty).
- Fehlermeldungsfeld (COLOR=RED) durch @ExceptionHandler oder BindingResult ersetzen.
- PF-Tasten (PF3=Ende, PF5=Abfrage) als separate REST-Endpoints oder als 'action'-Parameter.

BMS-MAP: UEBWMAP

24x80 Bildschirm | 13 Eingabefelder | 0 Anzeigefelder | DTO: UebwmapDto

PURPOSE AND FUNCTION

CICS BMS-Map 'UEBWMAP' definiert den Bildschirm-Aufbau des 3270-Terminal-Screens.
Datei: UEBWMAP.bms. Jedes Eingabefeld entspricht einem REST-API Parameter oder DTO-Feld
in der Migration.

SCREEN FIELDS (DFHMDF definitions)

FIELD NAME	POS	LEN	TYP	JAVA TYPE	MEANING
AUFTKTNF	5/16	10	INPUT	String	Beschriftung: 'KONTO-NR'. Eingabefeld (UNPROT). Laenge:..
AUFTNAMF	6/16	40	INPUT	String	Eingabefeld (UNPROT). Laenge: 40 Zeichen.
AUFTSALF	7/1	25	INPUT	String	Eingabefeld (UNPROT). Laenge: 25 Zeichen.
EMPFNAMF	10/16	40	INPUT	String	Beschriftung: 'EMPF. NAME'. Eingabefeld (UNPROT). Laeng..
EMPFIBANF	11/16	22	INPUT	String	Beschriftung: 'IBAN'. Eingabefeld (UNPROT). Laenge: 22 ..
EMPFBICF	12/16	11	INPUT	String	Beschriftung: 'BIC (OPT.)'. Eingabefeld (UNPROT). Laeng..
BETRAGF	15/16	14	INPUT	BigDecimal	Beschriftung: 'BETRAG (EUR)'. Eingabefeld (UNPROT). Lae..
AUSFDATF	15/46	8	INPUT	String	Beschriftung: 'AUSFUEHREN'. Eingabefeld (UNPROT). Laeng..
VERWZWKF	18/1	70	INPUT	String	Beschriftung: 'VERWENDUNGSZWECK'. Eingabefeld (UNPROT)...
STATUSF	23/1	79	INPUT	String	Beschriftung: 'PF1/ENTER=AUSFUEHREN PF3=ABBRECHEN P..
ERRORF	24/1	79	INPUT	String	Eingabefeld (UNPROT). Laenge: 79 Zeichen.
BUCHIDAF	7/1	50	INPUT	String	Beschriftung: 'IHRE UEBERWEISUNG WURDE ERFOLGREICH AUSG..
BETCHKF	9/1	25	INPUT	String	Eingabefeld (UNPROT). Laenge: 25 Zeichen.

MIGRATION HINTS (CICS -> Spring Boot)

- CICS-Screen als REST-Endpoint: @PostMapping mit @RequestBody UebwmapDto.
- DTO generieren mit 13 @NotNull/@Valid Feldern fuer Eingaben.
- Anzeigefelder als Response-DTO (kein Setter noetig, nur @JsonProperty).
- Fehlermeldungsfeld (COLOR=RED) durch @ExceptionHandler oder BindingResult ersetzen.
- PF-Tasten (PF3=Ende, PF5=Abfrage) als separate REST-Endpoints oder als 'action'-Parameter.

RECOMMENDATIONS

Migration Order, Risks and Technology Stack

MIGRATION ORDER

1	**MODULE: KONTOPRUEF, LIMITPRUEF** Isolierte Validatoren ohne ausgehende CALLs - einfachster Startpunkt. Action: Spring Boot REST-Endpoint. Testbar ohne Abhaengigkeiten. **Effort: ~1 Sprint** **Risk: LOW**
2	**MODULE: BUCHUNG** Kern-Buchungsservice. Wird von ZAHLUNG1 und ZAHLUNG2 aufgerufen. Action: Spring Boot + PostgreSQL mit @Transactional. Interface-Vertrag beachten. **Effort: ~2 Sprints** **Risk: MEDIUM**
3	**MODULE: REPORT** Ausgabe-Modul. 3 JCL-Jobs rufen es auf, keine ausgehenden CALLs. Action: Spring Boot Reporting-Service. Apache PDFBox / POI fuer Exports. **Effort: ~1 Sprint** **Risk: LOW**
4	**MODULE: ZAHLUNG1 + ZAHLUNG2** Entry-Points mit identischer CALL-Kette. ZAHLUNG2 hat zusaetzl. Verwendungstext. Action: Spring Batch Job. Beide zusammenfuehren - ZAHLUNG2 als Erweiterung. **Effort: ~3 Sprints** **Risk: HIGH**

TECHNICAL RISKS

KONTO.cpy:	KONTO-SALDO PIC 9(15)V99 -> als BigDecimal in JPA-Entity abbilden.
KUNDE.cpy:	Von 2 Programmen genutzt -> zentrale Customer-Entitaet definieren.
LIMITPRUEF:	Hardcodierter Grenzwert 100000.00 -> in application.properties auslagern.
Dyn. CALLs:	CALL WS-VARIABLE kann nicht statisch aufgeloest werden -> manuelle Pruefung.

TECHNOLOGY STACK

Backend:	Java 17, Spring Boot 3.x, Spring Batch
Datenbank:	PostgreSQL 15 (ersetzt DB2)
ORM:	Spring Data JPA / Hibernate
PDF-Export:	Apache PDFBox 3.x
Excel-Export:	Apache POI 5.x
Container:	Docker, Docker Compose (On-Premise, kein Cloud-Zwang)
Tests:	JUnit 5, AssertJ, Spring Boot Test
Java-Paket:	com.sz.discovery.* | Projekt: mainframe_discovery