The `AP250.OCL36` procedure orchestrates a series of RPG programs (`AP251.RPG36`, `AP255.RPG36`, `AP256A.RPG`, and `AP256.RPG36`) to process Accounts Payable (A/P) payments, update commission records, generate a cash disbursements journal, and produce vendor payment detail reports for ACH vendors. Below, I identify the **use cases** implemented by this call stack and provide a **Function Requirement Document** for the primary use case, structured as a large function that processes inputs to complete the A/P payment notification and reporting process.

Use Cases Implemented by AP250 OCL Call Stack

The `AP250.OCL36` call stack implements a single cohesive use case:

1. **Process Accounts Payable Payments and Generate ACH Vendor Payment Notifications with Journal Reporting**:
   - **Description**: This use case processes A/P payment records to update commission records, generate a summarized cash disbursements journal for general ledger integration, and produce detailed payment reports for ACH vendors, which are emailed via spoolflex. It involves preprocessing to count email addresses, updating commission tables, summarizing transactions, and generating vendor-specific reports with tailored messaging for crude and normal vendors.
   - **Components**:
     - `AP251.RPG36`: Updates the commission table (`APTORCY`) with payment details from the payment file (`APPAY`).
     - `AP255.RPG36`: Generates a cash disbursements journal report and updates a general ledger file (`TEMGEN`) with summarized A/P transactions.
     - `AP256A.RPG`: Preprocesses payment records to count valid ACH email addresses per vendor and populates the `APDTWSC` file.
     - `AP256.RPG36`: Generates up to four payment detail reports per ACH vendor, formatted for spoolflex emailing, with distinct messaging for crude and normal vendors.

This single use case encapsulates the entire A/P payment processing and reporting workflow, integrating multiple steps to ensure accurate financial updates and vendor notifications.

Function Requirement Document

Function Requirement Document: Process Accounts Payable Payments and Generate ACH Vendor Notifications

---

Function Name

`ProcessAPPaymentsAndNotifications`

Purpose

To process Accounts Payable (A/P) payment records, update commission records, generate a summarized cash disbursements journal for general ledger integration, and produce detailed payment reports for ACH vendors, emailed to up to four addresses per vendor with tailored messaging for crude and normal vendors.

Inputs

- **Payment Records** (`APPAY` equivalent):
  - Company number (`CONO`, 2 chars)
  - Vendor number (`VEND`, 5 chars)
  - Invoice description (`OPIN20`, 20 chars)
  - Gross amount (`OPGRAM`, packed, 6 digits)
  - Check number (`OPCKNO`, 6 chars)
  - Check date (`AXCKDT`, 6 chars, MMDDYY)
- **ACH Payment Records** (`APDTWS` equivalent):
  - Company number (`ADCO`, 2 chars)
  - Vendor number (`ADVEND`, 5 chars)
  - Company/vendor key (`ADCOVN`, 7 chars)
  - Invoice number (`ADINVN`, 20 chars)
  - Invoice amount (`ADINV$`, packed, 6 digits)
  - Discount (`ADDISC`, packed, 5 digits)
  - Payment amount (`ADLPAM`, packed, 6 digits)
  - Check date (`ADDATE`, 6 chars, MMDDYY)
  - Check number (`ADVCH#`, 5 chars)
- **Vendor Master** (`APVEND` equivalent):
  - Company number (`VNCO`, 2 chars)
  - Vendor number (`VNVEND`, 5 chars)
  - Vendor name (`VNNAME`, 30 chars)
  - Address lines (`VNADD1`–`VNADD4`, 30 chars each)
  - Vendor category (`VNCATG`, 6 chars, e.g., `CRDACT` for crude)
- **Vendor Email/Fax Details** (`APVNFMX` equivalent):
  - Company number (`AMCONO`, 2 chars)
  - Vendor number (`AMCVEN`, 5 chars)
  - Company/vendor key (`AMCOVN`, 7 chars)
  - Form type (`AMFMTY`, 4 chars, must be `ACHE`)
  - Email address (`AMEMLA`, 60 chars)
  - Send original flag (`AMFMYN`, 1 char, `Y` for valid)
  - Delete code (`AMDEL`, 1 char, `D` for deleted)
- **Company Master** (`APCONT` equivalent):
  - Company number (`ACCONO`, 2 chars)
  - Company name (`ACNAME`, 30 chars)
- **Cash Disbursements Journal Records** (`APCDJR` equivalent):
  - Company number (`CDCONO`, 2 chars)
  - Journal number (`CDJRNL`, 4 chars)
  - Credit/debit code (`CDCORD`, 1 char, `C` or `D`)
  - G/L number (`CDGLNO`, 8 chars)
  - Check number (`CDCHEK`, 6 chars)
  - Description (`CDDESC`, 24 chars)

  - Check date (`CDCKDT`, 6 chars, MMDDYY)
  - Amount (`CDAMT`, packed, 6 digits)
  - Vendor name (`CDNAME`, 30 chars)
  - Sequence number (`CDSEQ#`, 9 chars)
  - Transaction type (`CDTYPE`, 10 chars, e.g., `AP  `)
  - Year/period (`CDYYPD`, 4 chars)

Outputs

- **Updated Commission Table** (`APTORCY` equivalent):
    - Invoice number (`ATINV`, 20 chars)
    - Payment amount (`ATAPMT`, packed, 5 digits)
    - Payment status (`ATSTAT`, 1 char, set to `P`)
    - Check number (`ATCHK#`, 6 chars)
- **General Ledger File** (`TEMGEN` equivalent):
    - Company number (`CDCONO`, 2 chars)
    - G/L number (`CDGLNO`, 8 chars)
    - Journal number (`CDJRNL`, 4 chars)
    - Journal reference number (`JRREF#`, 4 digits)
    - Credit/debit code (`CORD`, 1 char)
    - Check number (`CDCHEK`, 6 chars) or check date (`CDCKDT`, 6 chars, summarized)
    - Description (`CDDESC`, 24 chars, or `-SUMMARIZED A/P`)
    - Amount (`JRAMT`, packed, 11 digits)
    - Vendor name (`CDNAME`, 30 chars)
    - Formatted check date (`CYMD`, 8 chars, YYYYMMDD)
- **Cash Disbursements Journal Report** (`APPRINT` equivalent):
    - Printed report with company name, journal totals (debit/credit), and detailed/summarized A/P entries.
- **ACH Vendor Payment Reports** (`REPORT1`–`REPORT4` equivalent):
    - Up to four reports per vendor, each emailed to a unique address, containing:
        - Vendor name and address
        - Payment details (invoice number, invoice amount, discount, payment amount)
        - Totals per vendor
        - Tailored messages (crude vs. normal vendors)
- **ACH Vendor Control File** (`APDTWSC` equivalent):
    - Company/vendor key (`ADCOVN`, 7 chars)
    - Email address count (`ACECNT`, packed, 2 digits)

Process Steps

1. **Count ACH Email Addresses**:

    - Read ACH payment records and vendor email details.
    - Count valid email addresses (`AMFMYN = 'Y'`, `AMDEL ≠ 'D'`, `AMFMTY = 'ACHE'`) per vendor.
    - Write/update `APDTWSC` with company/vendor key and email count (`ACECNT`).

2. **Update Commission Table**:

  - Read payment records.
  - Match records to commission table using company number, vendor number, and invoice description.
  - Update matching records with invoice number, gross amount, payment status (`P`), and check number.

3. **Generate Cash Disbursements Journal**:

   - Read journal records, identifying A/P transactions (`CDTYPE = 'AP  '`) and debit entries (`CDCORD = 'D'`).
   - Accumulate amounts per vendor; switch negative amounts to positive with adjusted credit/debit code.
   - Write detailed and summarized entries to general ledger file.
   - Produce a printed report with company name, journal number, debit/credit totals, and period/year (if applicable).

4. **Generate ACH Vendor Payment Reports**:

   - Read ACH payment records and vendor master data.
   - Retrieve email count from `APDTWSC` and up to four email addresses from `APVNFMX`.
   - Calculate payment amount as invoice amount minus discount.
   - Generate up to four reports per vendor, each including:
     - Vendor name, address, and check date.
     - Invoice details (number, amount, discount, payment).
     - Vendor-level totals.
     - Messages: normal vendors use standard messages; crude vendors (`VNCATG = 'CRDACT'`) use specific messages referencing `crudestatements@amref.com`.
   - Output reports to spoolflex queues for emailing.

Business Rules

1. **Commission Updates**:

   - Update commission records only for matching company, vendor, and invoice description.
   - Set payment status to `P` and record gross amount and check number.

2. **Journal Processing**:

   - Process only A/P transactions (`CDTYPE = 'AP  '`) for summarization.
   - Convert check dates to YYYYMMDD format with Y2K compliance (century `19` or `20` based on year ≥ 80).
   - Summarize amounts at the vendor level; switch negative amounts to positive with appropriate credit/debit code.

3. **ACH Vendor Notifications**:

   - Generate 1–4 reports per vendor based on email count (`ACECNT` from `APDTWSC`).
   - Include only valid email addresses (`AMFMYN = 'Y'`, `AMDEL ≠ 'D'`, `AMFMTY = 'ACHE'`).
   - Use crude-specific messages for vendors with `VNCATG = 'CRDACT'`; otherwise, use standard messages.

- Payment amount = invoice amount - discount.
- Paginate reports if exceeding 62 lines.

4. **Error Handling**:
   - Skip non-matching or invalid records without logging errors.
   - Handle missing records (vendor, company, email) gracefully, proceeding with available data.

Calculations

- **Payment Amount**: `ADPYM$ = ADINV$ - ADDISC` (ACH payment reports).
- **Journal Amount**: Accumulate `CDAMT` per vendor; if negative, switch sign and adjust credit/debit code (`CORD`).
- **Email Count**: Count `APVNFMX` records where `ADCOVN = AMCOVN`, `AMFMTY = 'ACHE'`, `AMDEL ≠ 'D'`, `AMFMYN = 'Y'`, up to 4.
- **Date Formatting**: Convert MMDDYY to YYYYMMDD (prefix `19` or `20` based on year comparison with `80`).

Assumptions

- Input files are pre-sorted by company and vendor.
- `APDTWSC` is populated by preprocessing step (`AP256A`).
- Spoolflex system handles email delivery from output queues (`QUSRSYS/APACHOUTQ` or `QUSRSYS/TESTOUTQ`).
- Y2K compliance uses century `19` or `20` based on year threshold (`80`).

Notes

- **Single Use Case**: The `AP250.OCL36` call stack integrates multiple programs to perform a unified A/P payment processing and notification function, treated as a single use case due to its cohesive workflow.
- **OCL Context**: The OCL sets up files (`APDTWSC`) and output queues, ensuring proper data flow and report distribution.
- **Modifications**:
  - `AP256.RPG36`: Added support for four email addresses (5-13-19) and crude-specific messaging (5-01-24).
- **Error Handling**: The programs rely on indicators to skip invalid records, ensuring robust processing without detailed error logging.
- **Business Context**: The function supports ACH vendor payments with automated notifications, tailored for crude and normal vendors, and integrates with general ledger systems via journal reporting.

This document concisely captures the business requirements, process steps, and calculations for the `AP250.OCL36` call stack, structured as a single function. Let me know if you need further clarification or additional details!