invoice-creation

Type: Combined Analysis (Multiple Files) **Generated**: 2026-01-29 11:53:59 **Processing Time**: 39787ms

_

Business Logic for Invoice Creation

This document outlines the business rules that govern the invoice creation process, based on an analysis of the RPG programs `FO602R` and `FO603R`. The primary focus is on the conditions that will **block** an order from being processed for invoicing, as well as the differences and interactions between the two programs.

The core logic for invoice creation is contained within the `oppord` subroutine in `FO602R` and the `oppord9` subroutine in `FO603R`. An order must pass all the following checks to be processed for invoicing.

Order Status and Header Rules

1. **Order Already Being Processed (`FO602R`, `FO603R`)**
* **Logic:** An order is skipped if it is already flagged as being under treatment.
* **File:** `FOHELF` (Order Header File)
* **Field:** `FOKODE`
* **Condition:** The process will not select an order if `FOKODE` is not equal to `*ZERO`. This indicates the order is locked by another process.
2. **Incorrect Order Type (`FO602R`, `FO603R`)**
* **Logic:** The order's type must match the type selected by the user for the batch job.
* **File:** `FOHELF` (Order Header File)
* **Field:** `FOOTYP`
* **Condition:** - In `FO602R`, the order's `FOOTYP` must match the `d0foty` parameter.
- In `FO603R`, the order's `FOOTYP` must be present in the `oty` array of selected order types.
3. **Order Without Lines (`FO602R`, `FO603R`)**
* **Logic:** Orders with no lines are deleted before processing.
* **File:** `FODTLR` (Order Detail File)
* **Field:** `FONUMM`, `FOSUFF`
* **Condition:** If no lines are found for the order, the order is deleted by calling the `FO410R` program. This logic is implemented in both programs.
4. **Order Info Code Blocking Invoice (`FO602R`, `FO603R`)**
* **Logic:** If the order has an info code that blocks invoicing, it is skipped.
* **File:** `VINFL1` (Order Info File)
* **Field:** `VAIFAK`

* **Condition:** If `VAIFAK` is `'1'` and the process is for invoicing (`VAOAKK = 3`), the order is skipped.
5. **User Authorization for Invoicing (`FO602R`, `FO603R`)**
* **Logic:** Only authorized users can initiate invoicing.
* **File:** `FUSRL1` (User Authorization File)
* **Field:** `FBKO08`
* **Condition:** If `FBKO08` is not equal to `1` and the process is for invoicing (`VAOAKK = 3`), the order is skipped.
6. **Packing Slip Reprinting (`FO602R`)**
* **Logic:** Prevents a packing slip that has already been generated from being printed again.
* **File:** `FOHELF` (Order Header File)
* **Field:** `FOPAKS`
* **Condition:** If the `FOPAKS` flag is `'1'` and the user is running the process to generate packing slips (`VAOAKK = 2`), the order is skipped. Note: This check is commented out in the newer `FO603R` program.
7. **Deposit Requirement (`FO602R`, `FO603R`)**
* **Logic:** Orders with unpaid deposits are skipped.
* **File:** `SDEPL1` (Deposit File)
* **Field:** `SKDEPB`, `SKDEPO`
* **Condition:** If the deposit balance (`SKDEPB`) is less than the required deposit (`SKDEPO`), the order is skipped.
8. **Mix of Order Types (`FO603R`)**
* **Logic:** Ensures that only orders of the same type are processed together.
* **File:** `VOTYL1` (Order Type File)
* **Field:** `VAOOKO`, `VAOSTA`
* **Condition:** If there is a mix of order types with different "OKO" or "STA" statuses, the process is blocked. This check is not present in `FO602R`.
Special Conditions (Program-Specific)
9. **Collective Invoicing (`FO602R`)**
* **Logic:** Allows collective invoicing of debit and credit orders.
* **File:** External program `CO402R`
* **Condition:** If `CO402R` returns `CO402_VERDI1 = '1'`, collective invoicing is enabled. This logic is not present in `FO603R`.
10. **Order Type Validation (`FO603R`)**
* **Logic:** Validates that the next order type is valid for processing.
* **File:** `VOTYL1` (Order Type File)
* **Field:** `VAONXT`
* **Condition:** The next order type (`VAONXT`) must match the expected

type. This validation is more extensive in `FO603R` compared to `FO602R`.

Subprogram Calls Affecting Logic

Beyond direct file checks, several external subprograms are called that play a significant role in the workflow.

1. **`FO410R` (Delete Orders with No Lines)**
* **Trigger:** Called when an order has no lines.
* **Logic:** Deletes the order and logs the reason.
* **Impact:** Ensures that only valid orders with lines are processed. This is implemented in both `FO602R` and `FO603R`.
2. **`FO709R` (Mark Orders as Under Processing)**
* **Trigger:** Called after an order passes all checks.
* **Logic:** Updates the order header to mark the order as being processed.
* **Impact:** Prevents duplicate processing of the same order. This is implemented in both programs.
3. **`CO402R` (Collective Invoicing)**
* **Trigger:** Called in `FO602R` to determine if collective invoicing is enabled.
* **Logic:** Returns a flag indicating whether debit and credit orders can be processed together.
* **Impact:** Allows for more flexible invoicing options. This is not called in `FO603R`.
4. **`FO605R` (Finalize Order Processing)**
* **Trigger:** Called at the end of processing in `FO603R`.
* **Logic:** Finalizes the processing of selected orders.
* **Impact:** Ensures that all selected orders are properly processed and logged. This is not explicitly called in `FO602R`.

Key Differences Between `FO602R` and `FO603R`

1. **Order Type Handling:**
- `FO602R` uses the `d0foty` parameter to validate order types.
- `FO603R` uses the `oty` array and performs additional checks to ensure consistency among order types.
2. **Packing Slip Reprinting:**
- `FO602R` blocks reprinting of packing slips (`FOPAKS` check).
- This check is commented out in `FO603R`.
3. **Collective Invoicing:**
- `FO602R` supports collective invoicing via `CO402R`.
- `FO603R` does not include this functionality.
4. **Mix of Order Types:**
- `FO603R` includes logic to block processing if there is a mix of incompatible order types.

- This logic is not present in `FO602R`.
5. **Next Order Type Validation:**
- `FO603R` performs more extensive validation of the next order type (`VAONXT`) compared to `FO602R`.
Summary
The programs `FO602R` and `FO603R` share a common goal of processing orders for invoicing, but they differ in their implementation and scope. `FO602R` includes legacy logic and additional features like collective invoicing, while `FO603R` introduces stricter validation and enhanced handling of order types. Both programs rely on external subprograms like `FO410R` and `FO709R` to manage order processing effectively.