invoice-creation

Type: Combined Analysis (Multiple Files) **Generated**: 2026-01-29 12:15:41 **Processing Time**: 32035ms

---

```
# Business Logic for Invoice Creation

This document outlines the business rules that govern the selection and processing of orders for
invoice creation, 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 and how
the two programs interact and differ in their implementation.

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.

## 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 associated order lines are deleted.
    *   **File:** `FODTLR` (Order Detail File)
    *   **Field:** `FDLINE`
    *   **Condition:**
        - Both programs check if there are no lines associated with the order (`FDLINE` is empty).
If no lines are found, the order is deleted by calling the `FO410R` program.

4.  **Order Info Code Blocking Invoice (`FO602R`, `FO603R`)**
    *   **Logic:** If the order has an info code that blocks invoicing, the order is skipped.
    *   **File:** `VINFL1` (Order Info File)
    *   **Field:** `VAIFAK`
    *   **Condition:** If the `FOITYP` field on the order is not blank and `VAIFAK` is `'1'`, the
order is skipped.

5.  **Unauthorized User for Invoicing (`FO602R`, `FO603R`)**
    *   **Logic:** Only authorized users can initiate invoicing.
    *   **File:** `FUSRL1` (User Authorization File)
    *   **Field:** `FBKO08`
    *   **Condition:** If the user does not have authorization (`FBKO08` not equal to `'1'`) and
the process is for invoicing (`VAOAKK = 3`), the order is skipped.

6.  **Packing Slip Already Printed (`FO602R` only)**
    *   **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 `FO603R`.

7. **Deposit Not Paid (`FO602R`, `FO603R`)**
    *   **Logic:** Orders with unpaid deposit requirements 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` only)**
    *   **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 (`VAOOKO` and `VAOSTA` flags differ across selected types), the process is blocked. This check is not present in `FO602R`.

## Order Selection Rules

9. **Order Number Range Validation (`FO602R`, `FO603R`)**
    *   **Logic:** Ensures the "from" order number is less than or equal to the "to" order number.
    *   **File:** `FOHELF` (Order Header File)
    *   **Field:** `FONUMM`
    *   **Condition:**
        - In `FO602R`, the `D1FONR` (from order number) must not be `*ZERO`, and if `D1TONR` (to order number) is provided, it must be greater than or equal to `D1FONR`.
        - In `FO603R`, the same logic applies but uses `D9FONR` and `D9TONR`.

10. **Valid Order Dates (`FO602R`, `FO603R`)**
    *   **Logic:** Ensures the order date and packing slip date are valid.
    *   **File:** `FOHELF` (Order Header File)
    *   **Field:** `FOBDAT`, `FOLDAT`
    *   **Condition:**
        - Both programs validate the dates using the `TEST(D)` operation. If invalid, the order is skipped.

11. **Department Filtering (`FO602R`, `FO603R`)**
    *   **Logic:** Filters orders based on the department.
    *   **File:** `FOHELF` (Order Header File)
    *   **Field:** `FOAVDE`
    *   **Condition:** Orders are included or excluded based on the department (`FOAVDE`) and user-specified department filters.

12. **Customer and Project Validation (`FO602R`, `FO603R`)**
    *   **Logic:** Ensures the order matches the specified customer and project.
    *   **File:** `RKUNL1` (Customer File), `FKPRL1` (Customer Project File)
    *   **Field:** `FOKUND`, `FOKPRO`
    *   **Condition:** Orders are skipped if the customer or project does not match the specified values.

## Subprogram Calls Affecting Logic

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

1. **`FO410R` (Order Deletion)**
    *   **Trigger:** Called when an order has no lines.
    *   **Logic:** Deletes the order and logs the reason.

```
    *   **Impact:** Ensures only valid orders with lines are processed.

2.  **`FO709R` (Order Locking)**
    *   **Trigger:** Called when an order is marked as being processed.
    *   **Logic:** Updates the order header to indicate the order is locked for processing.
    *   **Impact:** Prevents concurrent processing of the same order.

3.  **`CO402R` (Collective Invoicing)**
    *   **Trigger:** Called in `FO602R` to determine if collective invoicing (debit/credit) is enabled.
    *   **Logic:** Sets the `U_SAML` flag if collective invoicing is allowed.
    *   **Impact:** Alters the processing logic to handle collective invoicing.

4.  **`FO605R` (Output Generation)**
    *   **Trigger:** Called in `FO603R` after order selection is complete.
    *   **Logic:** Generates the output for the selected orders.
    *   **Impact:** Finalizes the processing and prepares the output.

## Key Differences Between `FO602R` and `FO603R`

1. **Order Type Validation:**
    - `FO602R` uses a single order type (`D0FOTY`) for validation.
    - `FO603R` allows multiple order types using the `OTY` array.

2. **Mix of Order Types:**
    - `FO603R` includes additional checks to prevent processing orders with mixed types (`VAOOKO` and `VAOSTA` flags). This logic is absent in `FO602R`.

3. **Packing Slip Reprinting:**
    - `FO602R` blocks reprinting of packing slips (`FOPAKS` check), but this is commented out in `FO603R`.

4. **Collective Invoicing:**
    - `FO602R` explicitly calls `CO402R` to handle collective invoicing, while `FO603R` does not include this logic.

5. **Output Generation:**
    - `FO603R` calls `FO605R` to generate output after processing, while `FO602R` does not.

## Summary

The programs `FO602R` and `FO603R` share many core business rules for order selection and processing but differ in their handling of order types, collective invoicing, and output generation. Together, they ensure that only valid, authorized, and properly configured orders are processed for invoicing, with additional flexibility and enhancements introduced in `FO603R`.
```