## Implementation Guide for Microsoft Access Frontend Designer

### Overview

This guide provides detailed instructions for implementing the new approach for handling MRL line items and their related fulfillment records in the Microsoft Access frontend. It covers the integration of views, stored procedures, and user interface design to ensure a seamless user experience.

### 1. Database Views

#### 1.1. Aggregated View

\*\*View Name:\*\* `vw\_line\_items\_aggregated`

\*\*Purpose:\*\* Display aggregated status information for MRL line items, summarizing the status of multiple fulfillment records.

\*\*SQL Definition:\*\*

```sql

CREATE VIEW vw\_line\_items\_aggregated AS

SELECT

m.order\_line\_item\_id,

m.jcn,

m.twcode,

m.nomenclature,

m.qty,

m.request\_date,

m.multiple\_shipments,

COALESCE(MAX(f.edd\_to\_ches), 'N/A') AS edd\_to\_ches,

COALESCE(MAX(f.carrier), 'N/A') AS carrier,

COALESCE(MAX(f.status), 'Pending') AS status,

COALESCE(MAX(f.comments), 'No comments') AS comments,

COUNT(f.fulfillment\_id) AS fulfillment\_count

FROM

MRL\_line\_items m

LEFT JOIN

fulfillment\_items f ON m.order\_line\_item\_id = f.order\_line\_item\_id

GROUP BY

m.order\_line\_item\_id, m.jcn, m.twcode, m.nomenclature, m.qty, m.request\_date, m.multiple\_shipments;

```

#### 1.2. Detailed Fulfillment View

\*\*View Name:\*\* `vw\_detailed\_fulfillment`

\*\*Purpose:\*\* Display detailed fulfillment records related to MRL line items.

\*\*SQL Definition:\*\*

```sql

CREATE VIEW vw\_detailed\_fulfillment AS

SELECT

m.jcn,

m.twcode,

f.fulfillment\_id,

f.edd\_to\_ches,

f.carrier,

f.status,

f.comments

FROM

MRL\_line\_items m

JOIN

fulfillment\_items f ON m.order\_line\_item\_id = f.order\_line\_item\_id;

```

### 2. Stored Procedures

#### 2.1. Update Line Item Status

\*\*Procedure Name:\*\* `update\_line\_item`

\*\*Purpose:\*\* Update the fulfillment status for a line item, ensuring transparency when there is only one fulfillment record.

\*\*SQL Definition:\*\*

```sql

CREATE OR REPLACE PROCEDURE update\_line\_item(

p\_order\_line\_item\_id INT,

p\_edd\_to\_ches DATE,

p\_carrier VARCHAR,

p\_status VARCHAR,

p\_comments TEXT,

p\_updated\_by INT

)

LANGUAGE plpgsql

AS $$

BEGIN

IF (SELECT multiple\_shipments FROM MRL\_line\_items WHERE order\_line\_item\_id = p\_order\_line\_item\_id) = FALSE THEN

UPDATE fulfillment\_items

SET edd\_to\_ches = p\_edd\_to\_ches,

carrier = p\_carrier,

status = p\_status,

comments = p\_comments,

updated\_by = p\_updated\_by,

updated\_at = CURRENT\_TIMESTAMP

WHERE order\_line\_item\_id = p\_order\_line\_item\_id;

ELSE

-- Logic for handling multiple fulfillment records

UPDATE fulfillment\_items

SET edd\_to\_ches = p\_edd\_to\_ches,

carrier = p\_carrier,

status = p\_status,

comments = p\_comments,

updated\_by = p\_updated\_by,

updated\_at = CURRENT\_TIMESTAMP

WHERE fulfillment\_id = (SELECT fulfillment\_id FROM fulfillment\_items WHERE order\_line\_item\_id = p\_order\_line\_item\_id LIMIT 1);

END IF;

-- Update overall status of the MRL line item

PERFORM update\_mrl\_status(p\_order\_line\_item\_id);

END;

$$;

```

#### 2.2. Update Fulfillment Status

\*\*Procedure Name:\*\* `update\_fulfillment\_status`

\*\*Purpose:\*\* Update the status of a specific fulfillment record and reflect the changes in the overall status of the MRL line item.

\*\*SQL Definition:\*\*

```sql

CREATE OR REPLACE PROCEDURE update\_fulfillment\_status(

p\_fulfillment\_id INT,

p\_edd\_to\_ches DATE,

p\_carrier VARCHAR,

p\_status VARCHAR,

p\_comments TEXT,

p\_updated\_by INT

)

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE fulfillment\_items

SET edd\_to\_ches = p\_edd\_to\_ches,

carrier = p\_carrier,

status = p\_status,

comments = p\_comments,

updated\_by = p\_updated\_by,

updated\_at = CURRENT\_TIMESTAMP

WHERE fulfillment\_id = p\_fulfillment\_id;

-- Update the overall status of the MRL line item based on fulfillment records

PERFORM update\_mrl\_status((SELECT order\_line\_item\_id FROM fulfillment\_items WHERE fulfillment\_id = p\_fulfillment\_id));

END;

$$;

```

### 3. Functions and Triggers

#### 3.1. Update MRL Line Item Status Function

\*\*Function Name:\*\* `update\_mrl\_status`

\*\*Purpose:\*\* Update the overall status of an MRL line item based on its related fulfillment records.

\*\*SQL Definition:\*\*

```sql

CREATE OR REPLACE FUNCTION update\_mrl\_status(p\_order\_line\_item\_id INT)

RETURNS VOID AS $$

BEGIN

UPDATE MRL\_line\_items

SET status = (SELECT CASE

WHEN COUNT(f.fulfillment\_id) = 0 THEN 'Pending'

WHEN COUNT(f.fulfillment\_id) = 1 THEN MAX(f.status)

ELSE 'Multiple Statuses'

END

FROM fulfillment\_items f

WHERE f.order\_line\_item\_id = p\_order\_line\_item\_id)

WHERE order\_line\_item\_id = p\_order\_line\_item\_id;

END;

$$ LANGUAGE plpgsql;

```

#### 3.2. Trigger to Update Fulfillment Status

\*\*Trigger Name:\*\* `trigger\_update\_fulfillment\_status`

\*\*Purpose:\*\* Ensure the fulfillment status is updated correctly and reflected in the overall MRL line item status.

\*\*SQL Definition:\*\*

```sql

CREATE TRIGGER trigger\_update\_fulfillment\_status

BEFORE INSERT OR UPDATE ON fulfillment\_items

FOR EACH ROW EXECUTE FUNCTION update\_fulfillment\_status();

```

### 4. User Interface Integration

#### 4.1. Main Form Design

- \*\*Data Source:\*\* `vw\_line\_items\_aggregated`

- \*\*Fields to Display:\*\*

- `jcn`

- `twcode`

- `nomenclature`

- `qty`

- `request\_date`

- `status`

- `fulfillment\_count`

- \*\*Features:\*\*

- Indicate if there are multiple fulfillment records using `fulfillment\_count`.

- Allow updates directly on this form for single fulfillment records.

#### 4.2. Sub-Form for Detailed Fulfillment

- \*\*Data Source:\*\* `vw\_detailed\_fulfillment`

- \*\*Fields to Display:\*\*

- `jcn`

- `twcode`

- `fulfillment\_id`

- `edd\_to\_ches`

- `carrier`

- `status`

- `comments`

- \*\*Linking:\*\*

- Link the sub-form to the main form by `order\_line\_item\_id`.

#### 4.3. Update Process

- \*\*Main Form:\*\*

- Present a unified view using `vw\_line\_items\_aggregated`.

- If `multiple\_shipments` is `false`, updates are directly applied to the related fulfillment record.

- If `multiple\_shipments` is `true`, provide an option to drill down into the sub-form for detailed updates.

- \*\*Sub-Form:\*\*

- Provide detailed view and update functionality for individual fulfillment records.

- Ensure updates are processed via the `update\_fulfillment\_status` stored procedure.

### 5. User Instructions

#### 5.1. Viewing Line Items

- Open the main form to view all MRL line items.

- Check the `fulfillment\_count` to see if there are multiple shipments.

- For single shipment items, update fields directly on the main form.

#### 5.2. Updating Line Items

- For single shipment items, make updates directly on the main form.

- For items with multiple shipments, open the sub-form to view and update each fulfillment record individually.

#### 5.3. Drill-Down for Multiple Shipments

- Click on the indicator or link to open the sub-form.

- View and manage detailed fulfillment records.

By following this implementation guide, the Microsoft Access frontend designer can create a seamless and user-friendly interface for managing MRL line items and their related fulfillment records. This approach ensures that end users can interact with the system efficiently, with clear indications and transparent updates for both single and multiple fulfillment scenarios.