Functional Tips for Print Envoy

This month's tip for getting more from your Print Envoy implementation!

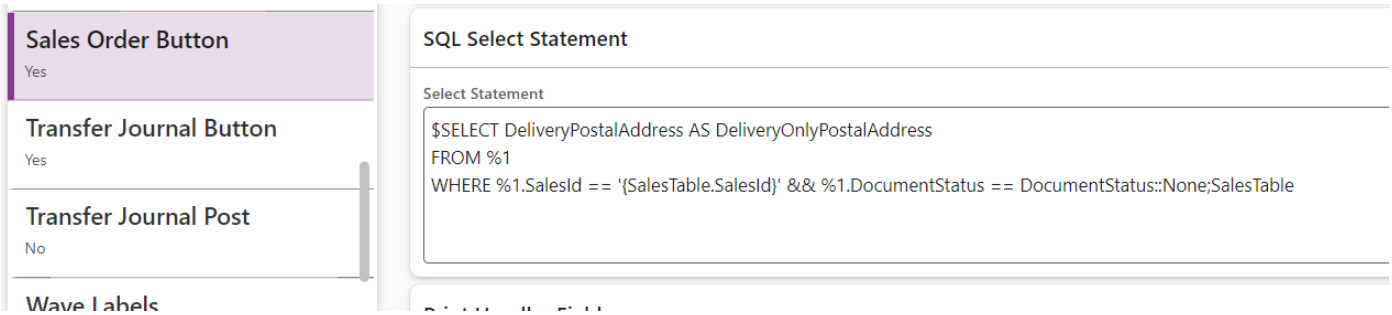
Extend your Print Envoy label data files using SQL

-By Olivia Johnson Olivia.Johnson@cloudinventory.com

Have you ever wanted to add additional data from D365 to your labels? Every Print Envoy handler allows for optional extension by SQL query, giving you the ability to bring information into your label file based on information that is already provided in the Print Handler you are using. This month's Functional Tip is meant to help you get started writing your own amazing queries to get the best out of Print Envoy.

Important notes:

Microsoft Dynamics 365 uses Transact-SQL (T-SQL), an extension of SQL that includes procedural programming features. While the core SQL syntax remains consistent, T-SQL in D365 introduces additional functionalities like stored procedures, triggers, and functions. Select statements on the Print Envoy handler forms should always start with an $, and don't need a semicolon (;) at the end. Keep in mind that every record you query will appear on the label data file, and querying multiple lines from the same column will generate multiple labels! Take a look at the syntax below.



SQL SELECT Basics:

Your select statement features three primary components: The fields you want to query, the tables from where they originate, and the conditions upon which those records will be selected.

$**SELECT** column1, column2, ... **FROM** table **WHERE** conditions;

- **SELECT**: Specifies the columns to retrieve. (Use * for all columns)
- **FROM**: Specifies the table from which to retrieve data.
- **WHERE** (optional): Specifies conditions for filtering records.

SQL Statement Construction: Examples

1. Basic Retrieval

To retrieve one or more records from the **InventSum** table, you can use the following SQL statement:

$SELECT * FROM InventSum;

This statement will return all columns for every row in the **InventSum** table. Replace the * with a specific field or list of fields from InventSum to select only those columns. While this is a very simple select statement, this data set will generate out a label file for every line in your InventSum table. We don't want that! Let's work on narrowing it down.

2. Filtered Retrieval

If you only need specific records based on certain conditions, you can use the WHERE clause:

```
$SELECT * FROM InventSum WHERE ItemId = 'ABC123';
```

This statement will retrieve all columns for the records where the **ItemId** is 'ABC123'.

Retrieving Records from the InventSum and InventDim Tables (Joined on InventDimId)

Let's explore how to retrieve records from both the InventSum and InventDim tables by joining them on the InventDimId column.

3. Inner Join

An inner join combines records from both the InventSum and InventDim tables based on the InventDimId column. It only returns rows where there is a matching InventDimId in both tables.

```
$SELECT * FROM InventSum INNER JOIN InventDim ON InventSum.InventDimId = InventDim.InventDimId;
```

This statement will retrieve all columns for the matched records between the two tables.

4. Using Substitution Variables

You can incorporate substitution variables directly into joined queries. For example, if you want to parameterize the InventDimId, you can do the following:

```
$SELECT * FROM InventSum INNER JOIN InventDim ON InventSum.InventDimId = InventDim.InventDimId WHERE InventDim.InventDimId = :InventDimId;
```

This allows for dynamic filtering based on the value provided for InventDimId.

Retrieving Records from the InventSum, InventDim, and InventTrans Tables (Joined on InventDimId)

Let's take it a step further by retrieving records from the InventSum, InventDim, and InventTrans tables, all joined on the InventDimId column.

5. Complex Join

```
$SELECT * FROM InventSum INNER JOIN InventDim ON InventSum.InventDimId = InventDim.InventDimId INNER JOIN InventTrans ON InventDim.InventDimId = InventTrans.InventDimId;
```

This statement combines records from all three tables.

6. Using Substitution Variables

You can apply substitution variables in complex joins as well. For instance, if you want to parameterize the InventDimId, you can do the following:

```
$SELECT * FROM InventSum INNER JOIN InventDim ON InventSum.InventDimId = InventDim.InventDimId INNER JOIN InventTrans ON InventDim.InventDimId = InventTrans.InventDimId WHERE InventDim.InventDimId = :InventDimId;
```

This allows for dynamic filtering based on the value provided for InventDimId.

Syntax for Substitution Variables:

In D365, substitution variables can be used directly in queries. Here are some syntax options:

- Using a colon (:) followed by the variable name, like **:VariableName**.
- Using a question mark **?** followed by the variable index, like **?1**.

Other Valid Operators and Symbols in SQL:

SQL supports a variety of operators and symbols for different operations. Some of the common ones include:

- Comparison operators: **=**, **!=**, **>**, **<**, **>=**, **<=**
- Logical operators: **AND**, **OR**, **NOT**
- Arithmetic operators: **+**, **-**, *****, **/**
- Wildcard characters: **%**, **_**
- Parentheses: **()**

Conclusion

Thank you for walking through this tutorial on SQL for Print Envoy! I hope this brief overview can help you get started thinking about how you can use SQL to get more out of your labels. There's so much more that you can do with this feature that I couldn't fit in this article, so please don't hesitate to reach out to me at Olivia.Johnson@cloudinventory.com with your questions and ideas. I can't wait to help you leverage Print Envoy's SQL capability for your business' unique needs.