

# Batch Detail Design

## I. Functional Area

Tax Rate Changes

## II. Module Affected

Tifposdn.pc

## III. Design Overview

This batch program processes records off of the TIF\_EXPLODE table, calculate a tax rate total for every item/department/site combination and write this total to a POS flat file. The changes defined by this document are required to adapt tifposdn to the new item dialogue.

The process function drives the processing of the program. Data will be read from the tif\_explode table and each record on the tif\_explode table will be processed into a transaction record on the output file. This file will contain separate transactions for every store listed on the tif\_explode table.

This program needs to be modified to work in when the users can be in the system at any given time. The system\_options table will be modified to introduce a new indicator named Batch with Online Users (btch\_w\_usr\_ind). This indicator is set to 'Y' if users can be in the system at any time, otherwise it is set to 'N'.

In this program, the deletion of records done in post programs must be done after processing the records. This would eliminate the need to run the post programs in the batch schedule. The truncation of records done in prepost, specifically in tifposdn\_post function, will be incorporated in this batch program.

## IV. Stored Procedures / Shared Modules (Maintainability)

No changes.

## V. Input Specifications

No changes.

## VI. Output Specifications

Output file: Format will be as in table given below.

All character variables should be right-padded with blanks and left justified; all numerical variables should be left-padded with zeroes and right-justified. Missing variables should be blank.

Record	Variable	Field Type	Default Value	Description
FHEAD	File record descriptor	CHAR(5)	FHEAD	Identifies file record type.
	Line number	NUMBER(10)	0000000001	Identifies the line number
	Program descriptor	CHAR(8)	tifposdn	Identifies the

Document: tifposdn Master Batch Design.doc  
Created: 1/30/2004 2:18 AM by Eric Hendrickson  
Last modified: 03/31/04 10:44 AM by zzdalyc

Page  
1 of 4

Project Name: <insert project name>  
System ID: <insert system ID>  
Version #: <insert version #>



				program.
	Create date	CHAR(14)	File create date	YYYYMMDDHH 24MISS format
THEAD	File record descriptor	CHAR(5)	THEAD	Identifies Transaction Header
	Line Number	NUMBER(10)		Sequential Line Number
	Transaction Number	NUMBER(10)		Transaction Number
	Store	NUMBER(10)		Identifies store number
TDETL	File Record descriptor	CHAR(5)	TDETL	Identifies transaction detail
	Line Number	NUMBER(10)		Sequential Line Number
	Transaction Number	NUMBER(10)		Transaction Number
	Department	NUMBER(4)		Department number.
	item	NUMBER(25)		Identifies the item.
	Tax Rate	NUMBER(8)		A new sales tax associated with the item ( % = /5)
	Start Date	CHAR(8)		Start date for the new tax rate. (in YYYYMMDD format)
TTAIL	File Record Descriptor	CHAR(5)	TTAIL	Identifies transaction trailer.
	Line Number	NUMBER(10)		Sequential line number
	Transaction Number	NUMBER(10)		Transaction Number
	Detail Counter	NUMBER(6)		Number of detail records in transaction.
FTAIL	Line Number	NUMBER(10)		Sequential Line Number
	Detail Line Counter	NUMBER(6)		Total Number of detail records processed.

## VII. Function Level Description

Add a “#include <intrface.h>

Add #define RECORD\_LOCKED 54



Remove all "EXEC SQL VAR"s.

Remove the following #defines, as they are not used : LEN\_TAX\_JURIS, LEN\_TAX\_TYPE, LEN\_GEO\_LEVEL, NULL\_GEO\_LEVEL, LEN\_GEO\_ID, NULL\_TAX\_RATE, LEN\_REC\_NO, LEN\_REC\_TYPE, NULL\_REC\_NO.

All variables referring to "sku" need to be renamed and resized to "item" specifications.

Fetch the btch\_w\_usr\_ind from the system\_options table.

The C\_TIF\_EXPLODE cursor needs to be changed...where it selects SKU, it needs to select ITEM. Also, the NVL(ps\_restart\_sku, -999) needs to be changed to NVL(ps\_restart\_item, ""). In addition, select the row id from the tif\_explode table.

Bring all code up to the latest standards.

Change the following line, which can be found closely following the fetch of the c\_tif\_explode cursor, :

```
ll_num_records_fetched = NUM_RECORDS_PROCESSED - ll_prev_total;
```

To :

```
If(!(ll_num_records_fetched = NUM_RECORDS_PROCESSED - ll_prev_total)) break;
```

This will break out of the while loop if 0 records are brought back by the cursor.

Once records are fetched, it is then inserted into a global temporary table to check for bulk locks on all records fetched. If the record fetched is locked, it will fetch the next logical unit of work, otherwise it processes the fetched records.

Before committing the records, it will check for system\_options.btch\_w\_usr\_ind. If it is 'Y', it will delete the fetched records in the tif\_explode table, otherwise it will not delete records from the tif\_explode table.

## VIII. Scheduling Considerations

Processing Cycle:	Phase 4 (daily)
Scheduling Diagram:	N/A
Pre-Processing:	txrposdn.pc
Post-Processing:	prepost.pc (to TRUNCATE tif_explode table if system_options.btch_w_usr_ind is 'N')
Threading Scheme:	N/A

## IX. Locking Strategy

This program has been modified for bulk locking strategy for tif\_explode table. Before firing a delete, the records will be bulk locked with update for nowait clause. If batch is trying to acquire a lock that has been locked by some other application, the batch will skip the logical unit of work and continue processing other records.

## X. Restart/Recovery

LOGICAL UNIT OF WORK = item/department/store combination



## XI. Performance Considerations

No changes.

## XII. Security Considerations

No changes.

## XIII. Design Assumptions

## XIV. Outstanding Design Issues

Issue Description	Priority	Issue Log Updated?

## XV. Appendix

Program Flow :

