

MBTIP Dynamic Source Table/Field Mapping Setup Guide

Date: 2025-07-20

Time: 19:22 EDT

Version: TriggerBuffered_v250720I

1. OVERVIEW

The MBTIP system processes data from a dynamic source table defined in ETPProHd (PHD) with field mappings in ETPProFM (PFM). A trigger (trigger_insert_prq) buffers inserts to ETPProBf, creates Ready batches in ETPProBt, and a Clarion timer embed runs TriggerProc.bat to process batches through PBatch3.bat-PBatch9.bat, applying masking and populating ETPProRt, ETPProTp, and etpproac. The etpprobt Clarion program automates configuration via a screen-driven interface, loading .ini files and setting range parameters.

2. DATABASE SETUP

2.1 Create Tables

Run PBatch0.sql to create required tables:

- etp_log: Logs events (LogOID BIGINT PRIMARY KEY, log_time VARCHAR(30), log_message TEXT).
- ETPProHd: Defines source table (DBTableName TEXT, DBName TEXT, TemplateNumber BIGINT, UserName TEXT, UniqueSourceColumn TEXT, Location TEXT).
- ETPProFM: Maps fields (DataFieldName TEXT, DataFieldNumber BIGINT, TemplateNumber BIGINT).
- ETPProBf: Buffers inserts (oid BIGINT PRIMARY KEY, guestbook TEXT, location TEXT, templatenum BIGINT, username TEXT, dbtablename TEXT, importid TEXT, datafieldnumber BIGINT, datafieldname TEXT, datavalue TEXT).
- ETPProBt: Tracks batches (oid BIGINT PRIMARY KEY, BatchNumber BIGINT, TemplateNumber BIGINT, UserName TEXT, DBTableName TEXT, RecordCount BIGINT, StartTime TIMESTAMP, EndTime TIMESTAMP, ExecutionTime VARCHAR(19), Status VARCHAR(10)).
- etpprorq: Request records (oid BIGINT PRIMARY KEY, importid BIGINT, templatenum BIGINT, batchnumber BIGINT).
- ETPProRt: Processing results (oid BIGINT PRIMARY KEY, importid BIGINT, acode TEXT, guestbook TEXT, templatenum BIGINT, username TEXT, dbtablename TEXT, batchnumber BIGINT).
- ETPProTp: Masking rules (oid BIGINT PRIMARY KEY, TemplateNumber BIGINT, FieldName TEXT, FieldNumber BIGINT, MaskType TEXT).
- etpproac: Final output (oid BIGINT PRIMARY KEY, importid BIGINT, acode TEXT, guestbook TEXT, templatenum BIGINT, username TEXT, dbtablename TEXT).

Command:

```
psql -U postgres -d postgres -f "PBatch\PBatch0.sql"
```

2.2 Configure ETPProHd

Insert source table and database details into ETPProHd (automated by etpprobt):

```
INSERT INTO ETPProHd (DBTableName, DBName, TemplateNumber, UserName,
UniqueSourceColumn, Location)
VALUES ('YourSourceTable', 'postgres', 100, 'user', 'source_id',
'default');
```

- DBTableName: Dynamic source table.
- DBName: Database (e.g., postgres).
- TemplateNumber: Links to ETPProFM and etpprorq.
- UserName: User identifier.
- UniqueSourceColumn: Source table column for importid (e.g., source_id).
- Location: Optional location field.
- Note: The etpprobt program configures this via a screen-driven interface, using .ini files with GetIni/PutIni.

2.3 Configure ETPProFM

Insert field mappings into ETPProFM (automated by etpprobt):

```
INSERT INTO ETPProFM (DataFieldName, DataFieldNumber, TemplateNumber)
VALUES ('data_info', 1, 100), ('field2', 2, 100), ('field3', 3, 100);
```

- DataFieldName: Source table column (e.g., data_info).
- DataFieldNumber: Field position for ETPProt.exe.
- TemplateNumber: Matches ETPProHd.TemplateNumber.
- Note: The etpprobt program populates ETPProFM via the UI, using .ini files.

2.4 Configure Range Parameters

- Set range parameters for record selection in the etpprobt UI (stored in ETPProBt or .ini files). The etpprobt program specifies which records to process from the source table.

2.5 Configure Trigger Enable

Set the trigger enable flag in ETPConfig (via etpprobt or manually):

```
INSERT INTO ETPConfig (Option_Name, Option_Value)
VALUES ('TriggerEnabled', 'true');
```

- Note: The etpprobt program may configure this via the UI.

3. TRIGGER SETUP

3.1 Create Trigger Function

Run CreateTriggerInsertPRQFunction_Auto.sql to create trigger_insert_prq():

```
CREATE OR REPLACE FUNCTION trigger_insert_prq()
RETURNS TRIGGER AS $$
DECLARE
    templatenum_local BIGINT;
    username_var TEXT;
    dbtablename_var TEXT;
    unique_column TEXT;
    field_rec RECORD;
    data_value TEXT;
    source_id BIGINT;
    import_id TEXT;
BEGIN
    SELECT TemplateNumber, UserName, DBTableName, UniqueSourceColumn
    INTO templatenum_local, username_var, dbtablename_var,
unique_column
    FROM ETPProHd
    LIMIT 1;
```

```

IF TG_TABLE_NAME != dbtablename_var THEN
    RETURN NEW;
END IF;
IF NOT EXISTS (
    SELECT 1 FROM ETPConfig
    WHERE option_name = 'TriggerEnabled' AND LOWER(option_value) =
'true'
) THEN
    RETURN NEW;
END IF;
EXECUTE format('SELECT ($1).%I::BIGINT', unique_column)
USING NEW INTO source_id;
import_id := source_id::TEXT;
FOR field_rec IN
    SELECT DataFieldName, DataFieldNumber
    FROM ETPProFM
    WHERE TemplateNumber = templatenum_local
LOOP
    EXECUTE format('SELECT ($1).%I::TEXT', field_rec.DataFieldName)
    USING NEW INTO data_value;
    INSERT INTO ETPProBf (
        guestbook,
        location,
        templatenum,
        username,
        dbtablename,
        importid,
        datafieldnumber,
        datafieldname,
        datavalue
    ) VALUES (
        NULL,
        (SELECT Location FROM ETPProHd WHERE DBTableName =
TG_TABLE_NAME LIMIT 1),
        templatenum_local,
        username_var,
        dbtablename_var,
        import_id,
        field_rec.DataFieldNumber,
        field_rec.DataFieldName,
        data_value
    );
END LOOP;
IF NOT EXISTS (
    SELECT 1 FROM ETPProBt
    WHERE status = 'Ready' AND dbtablename = dbtablename_var
) THEN
    INSERT INTO ETPProBt (
        BatchNumber,
        TemplateNumber,
        UserName,
        DBTableName,
        RecordCount,
        StartTime,
        Status
    ) VALUES (
        (SELECT COALESCE(MAX(BatchNumber), 0) + 1 FROM ETPProBt),
        templatenum_local,
        username_var,
        dbtablename_var,
        1,

```

```

        CURRENT_TIMESTAMP,
        'Ready'
    );
END IF;
RETURN NEW;
END $$ LANGUAGE plpgsql;

```

Command:

```

psql -U postgres -d postgres -f
"PBatch\CreateTriggerInsertPRQFunction_Auto.sql"

```

3.2 Attach Trigger Dynamically

Run SetupTriggerSystem_Auto.bat from the Clarion setup button:

```

@echo off
:: SetupTriggerSystem_Auto.bat
:: Installs the automatic PRQ trigger function and attaches it to the
configured table
set PGUSER=postgres
set PGPASSWORD=sa
set PGDATABASE=postgres
set PGHOST=localhost
set PGPORT=5432
set SQL_PATH=PBatch
set LOG_PATH=Logs
IF NOT EXIST "%LOG_PATH%" (
    mkdir "%LOG_PATH%"
)
echo [SetupTriggerSystem_Auto] Creating automatic trigger function...
psql -U %PGUSER% -h %PGHOST% -d %PGDATABASE% -f
"%SQL_PATH%\CreateTriggerInsertPRQFunction_Auto.sql" >
"%LOG_PATH%\CreateTriggerInsertPRQFunction_Auto.log" 2>&1
echo [SetupTriggerSystem_Auto] Attaching trigger to source table...
psql -U %PGUSER% -h %PGHOST% -d %PGDATABASE% -f
"%SQL_PATH%\AttachInsertPRQTrigger_Auto.sql" >
"%LOG_PATH%\AttachInsertPRQTrigger_Auto.log" 2>&1
echo [SetupTriggerSystem_Auto] Automatic PRQ trigger system installed
successfully.

```

AttachInsertPRQTrigger_Auto.sql:

```

DO $$
DECLARE
    active_table TEXT;
BEGIN
    SELECT DBTableName INTO active_table FROM ETProHd LIMIT 1;
    IF active_table IS NULL THEN
        RAISE EXCEPTION 'No active source table found in ETProHd!';
    END IF;
    IF NOT EXISTS (
        SELECT 1 FROM pg_trigger
        WHERE tgrelid = active_table::regclass
        AND tgname = format('prq_trigger_%s', active_table)
    ) THEN
        EXECUTE format(
            'CREATE TRIGGER prq_trigger_%I
            AFTER INSERT ON %I
            FOR EACH ROW
            EXECUTE FUNCTION trigger_insert_prq();',
            active_table, active_table

```

```

);
INSERT INTO etp_log (log_time, log_message)
VALUES (TO_CHAR(NOW(), 'YYYY-MM-DD HH24:MI:SS'),
'AttachInsertPRQTrigger_Auto: Created trigger for table ' || active_table);
ELSE
INSERT INTO etp_log (log_time, log_message)
VALUES (TO_CHAR(NOW(), 'YYYY-MM-DD HH24:MI:SS'),
'AttachInsertPRQTrigger_Auto: Trigger already exists for table ' ||
active_table);
END IF;
END $$;

```

Clarion Call:

In the Clarion setup button embed:

```
RUN('SetupTriggerSystem_Auto.bat',1)
```

4. CLARION TIMER EMBED SETUP

Configure the Clarion app timer embed:

- Save the timer embed code:

```

IF TriggerOn = TRUE
    DISABLE(?Btn_ManualBatch) ; DISABLE(?GROUPManualBatch)
    BatchProcessed = FALSE
    CLEAR(ETPProBt)
    ETPBt:Status = 'Ready'
    SET(ETPBt:ByStatusOID, ETPBt:ByStatusOID)
    LOOP
        IF ACCESS:ETPProBt.NEXT() = Level:Benign
            IF ETPBt:Status = 'Ready' AND NOT BatchProcessed
                BatchNum = ETPBt:BatchNumber
                TxtLastDetected = 'Detected Batch: ' & BatchNum
                DISPLAY(?TxtLastDetected)
                RUN('psql -d postgres -h localhost -p 5432 -U postgres -c
"INSERT INTO etp_log (log_time, log_message) VALUES (TO_CHAR(NOW(),
''YYYY-MM-DD HH24:MI:SS''), ''Timer: Detected Ready batch ' & BatchNum &
''');"')

                CLEAR(ETPConfig)
                ETPCF:Option_Name = 'BufferMode'
                IF ACCESS:ETPConfig.Fetch(ETPCF:KeyOptionName) =
Level:Benign

                    IF LOWER(ETPCF:Option_Value) = 'true'
                        IF EXISTS('StartFlushBuffered.bat')
                            RUN('StartFlushBuffered.bat', 1)
                            RUN('psql -d postgres -h localhost -p 5432 -U
postgres -c "INSERT INTO etp_log (log_time, log_message) VALUES (TO_CHAR(NOW(),
''YYYY-MM-DD HH24:MI:SS''), ''Timer: Flushed ETPProBf to ETPProRq for batch '
& BatchNum & ''');"')

                        ELSE
                            STOP('Flush batch file missing!')
                        END
                    END
                END
                ETPBt:Status = 'Processing'
                ACCESS:ETPProBt.Update()
                RUN('psql -d postgres -h localhost -p 5432 -U postgres -c
"INSERT INTO etp_log (log_time, log_message) VALUES (TO_CHAR(NOW(),
''YYYY-MM-DD HH24:MI:SS''), ''Timer: Batch ' & BatchNum & ' set to
Processing'');"')

                CLEAR(ETPParam)
                ETPPar:OID = 1

```

```

        IF ACCESS:ETPPParam.Fetch(ETPPPar:ByOID) = Level:Benign
            ETPPar:batch_oid = BatchNum
            ACCESS:ETPPParam.Update()
        END
        RUN('TriggerProc.bat', 1)
        BatchProcessed = TRUE
    END
ELSE
    BREAK
END
END
ELSE
    ENABLE(?Btn_ManualBatch) ; ENABLE(?GROUPManualBatch)
    ENABLE(?LOC:min_id)
    ENABLE(?LOC:max_id)
END

```

- Run SetupTriggerSystem_Auto.bat from the Clarion setup button.
- Set TriggerOn = TRUE.
- Ensure StartFlushBuffered.bat and TriggerProc.bat are in the same directory as the Clarion executable.

5. BATCH PROCESSING

TriggerProc.bat processes the batch:

- Claim Batch: Runs ClaimBatch_ReturnBatchNumber.sql to select a Ready batch from ETPProBt.
- Steps:
 - PBatch3.bat: Runs ETPProt.exe /SQL, using ETPProHd.DBTableName and ETPProFM mappings to populate ETPProRt.
 - PBatch4.bat: Verifies ETPProRt and updates ETPProBt.
 - PBatch6.bat: Populates ETPProTp with masking rules from ETPProHd.
 - PBatch7.bat: Updates ETPProBt with completion time/status.
 - PBatch8.bat: Applies masking to the source table.
 - PBatch9.bat: Populates etpproac from ETPProRt.

Command:

```
cmd /k TriggerProc.bat
```

6. TESTING

- Configure via etpprobt:
 - Use the etpprobt UI to set ETPProHd (DBTableName, UniqueSourceColumn), ETPProFM, range parameters, and TriggerEnabled.
- Insert Records:


```
INSERT INTO YourSourceTable (source_id, templatenum, data_info, field2, field3)
VALUES (1, 100, 'Data 1', 'Value 2', 'Value 3'), (2, 100, 'Data 2', 'Value 2', 'Value 3');
```

 - Replace YourSourceTable and source_id with ETPProHd.DBTableName and UniqueSourceColumn.
- Run Clarion App:
 - Run SetupTriggerSystem_Auto.bat from the setup button.
 - Set TriggerOn = TRUE.
 - Verify etp_log for AttachInsertPRQTrigger_Auto, Timer, and TriggerProc messages.
- Check Results:

- etp_log: SELECT LogOID, log_time, log_message FROM etp_log WHERE log_message LIKE 'AttachInsertPRQTrigger_Auto%' OR log_message LIKE 'Timer%' OR log_message LIKE 'TriggerProc%';
- ETPProBf: Confirm field-level data (datafieldname, datavalue).
- ETPProBt: Status = 'Processing', then Completed.
- etpprorq, ETPProRt, ETPProTp, etpproac: Populated.
- Logs: Logs.log, Logs_Auto.log, Logs_Auto.log.
- Clarion: Browse etp_log (ByLogOID), etpprobt (ByOID), etpproac.

7. NOTES

- Dynamic Mapping: ETPProHd.DBTableName and UniqueSourceColumn set the source table and ID column. ETPProFM maps fields dynamically.
- Trigger: trigger_insert_prq uses TG_TABLE_NAME to match ETPProHd.DBTableName, inserting field-level data into ETPProBf. SetupTriggerSystem_Auto.bat attaches the trigger.
- User Interface: The etpprobt program automates ETPProHd, ETPProFM, range parameters, and TriggerEnabled via a screen-driven interface, using .ini files with GetIni/PutIni.
- OID Reset: Optionally reset etpprorq.oid:

```
TRUNCATE etpprorq RESTART IDENTITY;
```