

System Failed to Create Deposit Checks when Transactions are Approved

08/25/2026 3:13 pm EDT

167971

System Failed to Create Deposit Checks when Transactions are Approved. Volume requires SQL resolution.

There is a situation that occurs occasionally where the deposit check object is not created when a transaction is approved by Forte. This will cause issues since the transaction will continue on as if nothing is wrong. However, the transaction will not appear in a deposit batch the way that it should, and without the deposit check to say that the invoice is satisfied, the customer could be charged for the same invoice down the line.

Normally, these can be resolved via a Manual Payment Batch. However, sometimes these occur in sufficient volume where the issue would take excessive time to resolve via the front end. Below are the set of scripts used to resolve this issue for API Alarm. The additional problem with this case was that it was caused by on-going server instability. That instability and the frequent timeouts made running the script for the entire contingent of transactions impossible. As such, there is a TOP(N) component of the script that configures the table for the number of records that can be run at one time.

Part 1 – Load the Temp Table

The first step is to run the below script to populate the temp table with the data used in Part 2.

First, run

--into [ZBG_CaseNumber_MMY] uncommented with just the select statement to ensure there's a copy of the data prior to making any changes.

There is a variable in the script you will need to fill in.

"and a.Submit_Date > '[Beginning_Date]' and a.Submit_Date < '[End_Date]'"

Change these variables to the beginning and end dates for when the issue occurred.

Before running the entire script, you should uncomment the below and use your example item to verify the process works.

--AND I.INVOICE_NUMBER = [Example_Item]"

You will need to run the script in Part 2 to create the check. Then use the select query in Part 3 to verify the process has worked. Once run and verified, comment this line back out to run the bulk version of the script.

If you receive any time out while running Part 2, you can alter the 'SELECT top(50)' at the top of the script to a lower number that doesn't time out.

```
/* This will show invoices, advance deposits, refunds, and unapplied cash. */
```

```
IF OBJECT_ID('tempdb..#tempt') IS NOT NULL
```

```
BEGIN
```

```
    DROP TABLE #tempt;
```

```
END;
```

```
SELECT TOP (50)
```

```
    a.ACH_Id,
```

```
    CAST(a.Submit_Date AS DATE) AS Submit_Date,
```

```
    IIF(
```

```
        a.Is_Credit = 'Y',
```

```
        -a.Amount,
```

```
        a.Amount
```

```
    ) AS Amount,
```

```
    a.Hold_Date,
```

```
    a.Customer_Id,
```

```
    a.Type_UAIM,
```

```
    IIF(
```

```
        a.Credit_Card <> 'Y',
```

```
        'ACH',
```

```
        IIF(a.Card_Type = 'AMER', 'AMER', 'CC')
```

```
    ) AS Trans_Type,
```

```
    a.Invoice_Id,
```

```
    a.Trans_Status,
```

```
    IIF(
```

```
        a.Credit_Card <> 'Y',
```

```
        'ACH',
```

```
        IIF(a.Card_Type = 'AMER', 'AMER', 'CC')
```

```
    ) + REPLACE(EOMONTH(a.Submit_Date), '-', '') AS Batch_Desc,
```

```
    EOMONTH(a.Submit_Date) AS [Date],
```

```
    a.Merchant_Id
```

```
-- INTO [ZBG_CaseNumber_MMY]
```

```
INTO #tempt
```

```
FROM AR_ACH AS a
```

```
INNER JOIN AR_Customer AS c
```

```
    ON c.Customer_Id = a.Customer_Id
```

```
INNER JOIN AR_Invoice AS i
```

```
    ON i.Invoice_Id = a.Invoice_Id
```

```
WHERE a.Submitted = 'Y'
```

```
AND a.Deposit_Check_Id = 1
```

```
AND a.Voided <> 'Y'
```

```
AND a.Inactive <> 'Y'
```

```
AND a.Trans_Status IN (
```

```
    'APPROVED',
```

```
    'REFUNDED',
```

```
    'SETTLED'
```

```
)
```

```
AND a.Response_Code IN (
```

```
    'A01',
```

```
    'S01'
```

```
)
```

```
AND a.Response_Type IN (
```

```
    'F'
```

```

        'A'
    )
    AND a.Second_Response_Type IN (
        'I',
        'F'
    )
    AND a.Submit_Date > '[Beginning_Date]'
    AND a.Submit_Date < '[End_Date]'
    AND a.Amount <> 0
    AND a.Trace_Number <> ''
    -- AND i.Invoice_Number = 7476951
    -- AND CASE a.Type_UAIM
    --     WHEN 'I' THEN i.Net_Due
    --     ELSE a.Amount
    --     END > 0
    ORDER BY a.ACH_Id;

GO

/* Get the open batches beginning with the minimum transaction submission date. */

IF OBJECT_ID('tempdb..#OpenBatches') IS NOT NULL
BEGIN
    DROP TABLE #OpenBatches;
END;

SELECT DISTINCT
    db.Deposit_Batch_Id,
    db.Description,
    db.Date AS BatchDate
INTO #OpenBatches
FROM AR_Deposit_Batch AS db
WHERE db.Date >= (
    SELECT MIN(CAST(Submit_Date AS DATE))
    FROM #tempt
)
AND db.Deposit_Id = 1
ORDER BY db.Deposit_Batch_Id;

GO

ALTER TABLE #tempt
ADD Deposit_Batch_Id INT;

GO

ALTER TABLE #tempt
ADD Used VARCHAR(2);

GO

/* Match transactions to existing batches. */

UPDATE t
SET
    t.Batch_Desc = ob.Description,
    t.Date = ob.BatchDate,

```

```

t.Deposit_Batch_Id = ob.Deposit_Batch_Id,
t.Used = 'N'
-- SELECT *
FROM #tempt AS t
CROSS APPLY (
    SELECT TOP (1)
        ob.*
    FROM #OpenBatches AS ob
    WHERE CAST(t.Merchant_Id AS VARCHAR(10)) =
        LEFT(ob.Description, LEN(t.Merchant_Id))

    -- Match the payment method type to the batch type
    AND LEFT(t.Trans_Type, 2) =
        SUBSTRING(
            ob.Description,
            CHARINDEX('_', ob.Description) + 1,
            2
        )

    -- Match the batch date to the transaction submission date
    AND ob.BatchDate >= t.Submit_Date
    ORDER BY
        ob.BatchDate,
        ob.Deposit_Batch_Id
) AS ob;

GO

IF OBJECT_ID('AchTempTab') IS NOT NULL
BEGIN
    DROP TABLE AchTempTab;
END;

GO

SELECT *
INTO AchTempTab
FROM #tempt;

GO

SELECT *
FROM AchTempTab;

```

Part 2 – Create the Deposit Checks

This script is rote and has no variables. The suggestion is to copy this script into a second query window in SSMS. Run Part 1 in its own window to populate the temp table. Then swap to the second window and run this script to make the checks. If it is the first time you are running the script, proceed to Part 3 for verification. Otherwise, return to Part 1 to populate the temp table with more data.

```

SET NOCOUNT ON;

DECLARE

```

```

@description      VARCHAR(30),
@cust_id          INT,
@reference         VARCHAR(20), -- Customer number
@usercode         VARCHAR(15) = 'SedonaEFT',
@deposit_batch_id INT,
@date             DATETIME,
@ach_idx          INT,      -- ACH ID
@amt              MONEY,
@branch_code      VARCHAR(35),
@postingDate      DATETIME = GETDATE(),
@collection_activity_id INT = 7, -- Payment posted
@credit_card      CHAR(1),
@payment_meth_code VARCHAR(35),
@last_4           VARCHAR(8),
@type_uaim        CHAR(1),
@job_number       VARCHAR(25),
@income_account_code VARCHAR(25) = 'N/A',
@invoice_id       INT,
@currentDate      DATE = CAST(GETDATE() AS DATE),
@RecordCount      INT = 0;

```

-- This is my activity loop

```

WHILE EXISTS (
    SELECT 1
    FROM AchTempTab AS t
    WHERE t.used <> 'Y'
)

```

BEGIN

-- Grab the first ACH_ID to work on

```

SET @ach_idx = (
    SELECT TOP (1) ACH_Id
    FROM AchTempTab
    WHERE used <> 'Y'
    ORDER BY ACH_Id
);

```

-- Test to make sure a check has not already been created

```

IF EXISTS (
    SELECT 1
    FROM AR_ACH
    WHERE ACH_Id = @ach_idx
    AND (
        Deposit_Check_Id > 1
        OR Voided = 'Y'
    )
)

```

)

BEGIN

```

    UPDATE AchTempTab
    SET used = 'Y'
    WHERE ACH_Id = @ach_idx;

```

END

ELSE

BEGIN

-- Get the posting date

```

SELECT
    @date = COALESCE(
        (

```

```

            SELECT t.Date
            FROM AchTempTab AS t

```

```

        FROM AchTempTab AS t
        WHERE t.ACH_Id = @ach_idx
        AND EXISTS (
            SELECT 1
            FROM GL_Accounting_Period AS glp
            WHERE t.Date BETWEEN glp.Start_Date
                AND glp.End_Date
            AND glp.Status IN ('O', 'R')
        )
    ),
    @currentDate
);

SELECT
    @description = (
        SELECT Batch_Desc
        FROM AchTempTab
        WHERE ACH_Id = @ach_idx
    );

SELECT
    @deposit_batch_id = Deposit_Batch_Id
FROM AR_Deposit_Batch
WHERE Description = @description;

SELECT
    @cust_id    = Customer_Id,
    @type_uaim  = Type_UAIM,
    @job_number = JobNumber,
    @invoice_id = Invoice_Id,
    @credit_card = Credit_Card,
    @last_4     = Last_Four_Digits,
    @amt        = CASE Is_Credit
        WHEN 'Y' THEN -Amount
        ELSE Amount
    END
FROM AR_ACH
WHERE ACH_Id = @ach_idx;

/*
    If Type_UAIM is 'I' and the amount is greater than the
    amount due on the invoice, turn it into unapplied cash.
*/
DECLARE @ZeroBallInvoice INT = 0;

SELECT
    @ZeroBallInvoice = COUNT(*)
FROM AR_ACH_Invoice AS aic
INNER JOIN AR_Invoice AS inv
    ON aic.Invoice_Id = inv.Invoice_Id
WHERE aic.ACH_Id = @ach_idx
    AND aic.Invoice_Id > 1
    AND aic.Amount - inv.Net_Due > 0.1;

SELECT
    @type_uaim = CASE
        WHEN @type_uaim = 'I'
        AND @ZeroBallInvoice > 0
        THEN 'U'

```

```

        ELSE @type_uaim
    END;

SELECT
    @reference = arc.Customer_Number
FROM AR_Customer AS arc
WHERE arc.Customer_Id = @cust_id;

SET @branch_code = (
    SELECT Branch_Code
    FROM AR_Branch
    WHERE Branch_Id = (
        SELECT Branch_Id
        FROM AR_Customer
        WHERE Customer_Id = @cust_id
    )
);

SET @payment_meth_code = (
    SELECT Payment_Method_Code
    FROM AR_Payment_Method
    WHERE Payment_Method_Id = (
        CASE @credit_card
            WHEN 'Y' THEN (
                SELECT TOP (1) Payment_Method_Id
                FROM AR_Customer_CC
                WHERE Customer_Id = @cust_id
                AND Last_Four_Digits = @last_4
                ORDER BY Customer_CC_Id DESC
            )
            ELSE (
                SELECT TOP (1) Payment_Method_Id
                FROM AR_Customer_Bank
                WHERE Customer_Id = @cust_id
                AND Last_Four_Digits = @last_4
                ORDER BY Customer_Bank_Id DESC
            )
        )
    END
);

SELECT
    @payment_meth_code = ISNULL(
        @payment_meth_code,
        (
            SELECT Payment_Method_Code
            FROM AR_Payment_Method
            WHERE Payment_Method_Id = 1
        )
    );

DECLARE @p5 INT;

EXEC dbo.Posting_Start
    @description,
    @usercode,
    @reference,
    'PYMT',
    @p5 OUTPUT; -- Posting ID

```

```
DECLARE @checknumber VARCHAR(25) = 'ACH Processing';
DECLARE @p15 INT;
```

```
EXEC dbo.Deposit_Check_ADD
    @deposit_batch_id,
    @date,
    @payment_meth_code,
    'C',
    @reference,
    "",
    "",
    "",
    @checknumber,
    @date,
    @amt,
    @description,
    @branch_code,
    'N',
    @p15 OUTPUT; -- Deposit check ID
```

```
UPDATE AR_Deposit_Check
SET ACH_Id = @ach_idx
WHERE Deposit_Check_Id = @p15;
```

```
DECLARE
    @adv_dep_id INT = 1,
    @una_cash_id INT = 1,
    @inv_number INT = 0,
    @p10 INT; -- Deposit check detail ID
```

```
IF @type_uaim = 'A'
BEGIN
    EXEC Advance_Deposit_ADD
        @p15,
        @reference,
        @date,
        @job_number,
        @amt,
        0,
        @adv_dep_id OUTPUT;
END;
```

```
IF @type_uaim = 'U'
BEGIN
    EXEC Unapplied_Cash_ADD
        @p15,
        @reference,
        @date,
        @amt,
        0,
        @una_cash_id OUTPUT;
END;
```

```
IF @type_uaim = 'M'
```

```

BEGIN
    SELECT
        @income_account_code = (
            SELECT ISNULL(g.Account_Code, 'N/A')
            FROM GL_Account AS g
            INNER JOIN AR_ACH AS a
                ON g.Account_Id = a.Misc_Account_Id
            WHERE a.ACH_Id = @ach_idx
        );
END;

IF @type_uaim = 'I'
BEGIN
    DECLARE
        @ach_inv_id INT,
        @invamt MONEY;

    WHILE EXISTS (
        SELECT 1
        FROM AR_ACH_Invoice AS ACHINV
        INNER JOIN AR_Invoice AS ARINV
            ON ACHINV.Invoice_Id = ARINV.Invoice_Id
        WHERE ACHINV.ACH_Id = @ach_idx
        AND ACHINV.ACH_Invoice_Id > 1
        AND ACHINV.Invoice_Id > 1
        AND ACHINV.Amount <> 0
        AND ACHINV.Deposit_Check_Detail_Id = 1
    )
    BEGIN
        SELECT TOP (1)
            @inv_number = ARINV.Invoice_Number,
            @ach_inv_id = ACHINV.ACH_Invoice_Id,
            @invamt = ACHINV.Amount
        FROM AR_ACH_Invoice AS ACHINV
        INNER JOIN AR_Invoice AS ARINV
            ON ACHINV.Invoice_Id = ARINV.Invoice_Id
        WHERE ACHINV.ACH_Id = @ach_idx
        AND ACHINV.ACH_Invoice_Id > 1
        AND ACHINV.Invoice_Id > 1
        AND ACHINV.Amount <> 0
        AND ACHINV.Deposit_Check_Detail_Id = 1
        ORDER BY ACHINV.ACH_Invoice_Id;

        EXEC dbo.Deposit_Check_Detail_ADD
            @deposit_batch_id,
            @p15,
            @type_uaim,
            @una_cash_id,
            @adv_dep_id,
            @inv_number,
            @income_account_code,
            @invamt,
            'N/A',
            @p10 OUTPUT; -- Deposit check detail ID

        EXEC dbo.SEFT_Update_ACH_Invoice
            @ach_inv_id,
            @p10;
    END;
END;

```

```

END;
ELSE
BEGIN
    EXEC dbo.Deposit_Check_Detail_ADD
        @deposit_batch_id,
        @p15,
        @type_uaim,
        @una_cash_id,
        @adv_dep_id,
        @inv_number,
        @income_account_code,
        @amt,
        'N/A',
        @p10 OUTPUT; -- Deposit check detail ID
END;

EXEC Customer_Age
    @reference,
    @status = 0;

DECLARE @register_number INT;

EXEC dbo.Get_Register_For_Deposit_Check
    @p15,
    @register_number OUTPUT;

DECLARE @errorcode INT;
SET @errorcode = 0;

DECLARE @errordescription NVARCHAR(255);
SET @errordescription = '';

EXEC dbo.Register_Balance_Check
    @register_number,
    @errorcode OUTPUT,
    @errordescription OUTPUT;

SELECT
    @errorcode,
    @errordescription;

EXEC dbo.ACH_Post
    @ach_idx,
    @p15;

DECLARE @collection_event_id INT;

EXEC dbo.Collection_Event_ADD
    @reference,
    'N/A',
    'Posted Payment',
    @collection_activity_id,
    @postingDate,
    'Administrator',
    'EFT Payment Posted',
    0,
    @amt,
    @collection_event_id OUTPUT;

EXEC dbo.Posting End

```

```

    @p5;

UPDATE d
SET
    d.Tape_Total = t.amt,
    d.Entered_Amount = t.amt
FROM (
    SELECT SUM(Amount) AS amt
    FROM AR_Deposit_Check
    WHERE Deposit_Batch_Id = @deposit_batch_id
) AS t
INNER JOIN AR_Deposit_Batch AS d
    ON d.Deposit_Batch_Id = @deposit_batch_id;

UPDATE AchTempTab
SET used = 'Y'
WHERE ACH_Id = @ach_idx;
END;

SELECT
    @RecordCount = @RecordCount + 1;

SELECT
    CONCAT('Processed record count: ', @RecordCount);
END;

```

Part 3 – Verify Function, Check for Errors

Once Part 2 completes, you can change the where clause, 'where a.Ach_id = [Example]', to have the relevant ID for your example and verify that the deposit check exists and is the appropriate batch. Then return to Part 1 and run the script for items in bulk. If you ever get a time out, you can modify this query to be for the last item before time out to verify it created a check.

```
SELECT cu.Customer_Number, a.ACH_Id, a.Type_UAIM, a.Entered_Date, a.Submit_Date, a.Amount [ACH Amount], a.Deposit_Check_Id, a.Trans_Status , a.Voided
,c.Transaction_Date, c.Check_Number, c.Amount [Check Amount]
, gc.Register_Number, gc.Register_Id, gc.Account_Id, gc.Register_Type_Id, gc.Credit_Or_Debit, gc.Amount [GLC_Amount]
, cd.Deposit_Check_Detail_Id, cd.Type_UAIM, case cd.Type_UAIM when 'U' then cd.Unapplied_Cash_Id when 'A' then cd.Advance_Deposit_Id when 'M' then cd.Misc_Inc_Account_Id when 'I' then cd.Invoice_Id else 0 end as [Applied_Id]
, gd.Register_Id, gd.Account_Id, gd.Register_Type_Id, gd.Credit_Or_Debit, gd.Amount [GLCD_Amount]
, iv.invoice_number, iv.Invoice_Date, Net_Due, iv.Has_Pending_EFT

FROM AR_ACH a

INNER JOIN AR_Customer cu ON cu.Customer_Id = a.Customer_Id
left join ar_deposit_check c on c.deposit_check_id = a.deposit_check_id or c.ACH_Id = a.ACH_Id
left join AR_Deposit_Check_Detail cd on c.deposit_check_id = cd.deposit_check_id
left Join GL_Register gc on gc.Register_Id = c.Register_Id
left join GL_Register gd on gd.Register_Id = cd.Register_Id
left join AR_Invoice iv on iv.Invoice_Id = cd.Invoice_Id and cd.Invoice_Id > 1
left Join AR_Unapplied_Cash u on u.Unapplied_Cash_Id = cd.Unapplied_Cash_Id and cd.Unapplied_Cash_Id > 1
left Join AR_Unapplied_Cash_Detail ud on ud.Unapplied_Cash_Id = u.Unapplied_Cash_Id
left Join AR_Advance_Deposit ad on ad.Advance_Deposit_Id = cd.Advance_Deposit_Id and cd.Advance_Deposit_Id > 1
left Join AR_Advance_Deposit_Detail adt on ad.Advance_Deposit_Id = adt.Advance_Deposit_Id

where a.Ach_id = [Example]
```