



NAGAD – SOC – DATABASE RELATED WORK DOCUMENTATION

Table of Contents

<i>Adding Bank Branch</i>	2
<i>DOB Update</i>	4
<i>Partner Suspension</i>	6
<i>KYC QC Accept and Reject</i>	8
<i>User MNO Update</i>	11
<i>NID Blank Issue</i>	12
<i>Customer Trust Level Update</i>	13
<i>Customer Activation</i>	14
<i>Duplicate MAP ID Issue Fix</i>	14
<i>BIZ Code Update</i>	15
<i>Transaction Visibility Issue</i>	16
<i>DMS Insertion</i>	17
<i>MNO Campaign Data Check</i>	18
<i>RMS(Campaign) Promotion Pause</i>	19
<i>Transaction Data</i>	20

Adding Bank Branch

STEP 01 – Verification: Start by gathering the necessary data (Bank name, Routing number, Branch name, District name, and Scanned copy of cheque book) from the mail. Then, verify the routing number and bank branch name on the bank's official website. **If no bank branch matches the provided routing number, do not proceed further.** Send the mail back and ask for valid data.

STEP 02 – Check if it already exists in the system: Go to staging system portal >> Merchant Management >> Merchant Registration >> Bank Account Info and check if the branch already exists in the system. **If it already exists, do not proceed further.** Send the mail back saying that the branch is already available in the system.

You can also check from staging database using the following queries –

```
-- using bank name
SELECT *
FROM DMS.BANK_INFO
WHERE BANK_NAME LIKE '%SONALI%';

-- using district name
SELECT *
FROM DMS.BANK_BRANCH_INFO
WHERE DISTRICT LIKE '%DHAKA%';

-- using branch name
SELECT *
FROM DMS.BANK_BRANCH_INFO
WHERE BRANCH_NAME LIKE '%BANANI%';

-- using routing number
SELECT *
FROM DMS.BANK_BRANCH_INFO
WHERE ROUTING_NO IN ('120520460');

-- using bank code
SELECT *
FROM DMS.BANK_BRANCH_INFO
WHERE BANK_CODE IN ('120');
```

STEP 03 – Prepare Query: Use the following query to add new bank branch to the system

```
INSERT INTO DMS.BANK_BRANCH_INFO (
    ID,
    BRANCH_CODE,
    BRANCH_NAME,
    BANK_CODE,
    STATUS,
    CREATED,
    VERSION,
    DISTRICT,
    ROUTING_NO
)
VALUES (
    (SELECT MAX(ID) FROM DMS.BANK_BRANCH_INFO)+1,
    '120520460', -- Routing number
    'BASHABO', -- Branch name
    '120', -- First 3 digit of the routing number
    'ACTIVE',
    CURRENT_TIMESTAMP,
    '0',
    'DHAKA-SOUTH', -- District name
    '120520460'
);
```

STEP 04 – Staging DB Insert: Send the insert query to the chain mail *Staging DB Insert - Add New Bank Branch*, to be executed by the DB team.

STEP 05 – Check in Staging Portal: Follow Step 2 again. Now the branch should be available in the dropdown. If everything is alright, send the same query to the chain mail *Live DB Insert - Add New Bank Branch*, to be executed by the DB team.

After the execution, reply to the mail from CDIM team.

DOB Update

STEP 01 – Validation: Check for mobile number and NID number in the mail. Search in System portal >> Search Customer >> Search with NID >> Check if there is a mismatch in the DOB or the DOB is found empty.

STEP 02 – Encrypt NID Number: Encrypt the NID number from the App1 server.

STEP 03 – Ask for Data: Ask for DOB data using the following query

```
SELECT A.ID      MAP_ID,
       A.PHOTO_ID,
       A.DOB,
       B.ID      CUS_ID,
       B.PHOTO_ID,
       B.DOB,
       C.ID      CH_ID,
       C.PHOTO_ID,
       C.DOB,
       D.ID      MER_ID,
       D.PHOTO_ID,
       D.DOB,
       E.ID      KYC_ID,
       E.PHOTO_ID,
       E.DOB,
       E.KYC_STATUS,
       E.KYC_CATEGORY

  FROM MAP.WALLET_USER    A,
       DMS.CUSTOMER     B,
       DMS.CHANNEL_PARTNER C,
       DMS.MERCHANT      D,
       DMS.KYC_INFO       E

 WHERE A.PHOTO_ID = B.PHOTO_ID(+)
   AND A.PHOTO_ID = C.PHOTO_ID(+)
   AND A.PHOTO_ID = D.PHOTO_ID(+)
   AND A.PHOTO_ID = E.PHOTO_ID(+)
   AND A.PHOTO_ID IN ('4DB7057DDBC9BCC1F7F9BD31A636CF60')
 ORDER BY 1;
```

STEP 04 – Check Data: Once the data is shared, check which tables contain the data and identify any mismatches.

STEP 05 – Update DOB: Prepare a query only for the tables with available data. Then, send it to **Live DB Update – DOB Update** for execution. Here is the update query –

```
UPDATE DMS.CHANNEL_PARTNER
SET DOB = TO_DATE('01-Nov-1979','DD-MON-YYYY'),
UPDATED= SYSDATE
WHERE PHOTO_ID = '28088128B4A2302C83629CD691EB54C8';

UPDATE DMS.CUSTOMER
SET DOB = TO_DATE('01-Nov-1979','DD-MON-YYYY'),
UPDATED = SYSDATE
WHERE PHOTO_ID = '811CDAC0195F13B0A120D53FA903D0A1';

UPDATE DMS.KYC_INFO
SET DOB = TO_DATE('01-Nov-1979','DD-MON-YYYY'),
UPDATED = SYSDATE
WHERE PHOTO_ID = '811CDAC0195F13B0A120D53FA903D0A1';

UPDATE DMS.MERCHANT
SET DOB = TO_DATE('25-May-1965','DD-MON-YYYY'),
UPDATED = SYSDATE
WHERE PHOTO_ID = '811CDAC0195F13B0A120D53FA903D0A1';

UPDATE MAP.WALLET_USER
SET DOB = TO_DATE('29-Sep-1981','DD-MON-YYYY'),
UPDATED= SYSDATE
WHERE PHOTO_ID = '811CDAC0195F13B0A120D53FA903D0A1';
```

STEP 06 – Check in Portal: After the execution, check in the portal again and reply to the mail if everything is alright.

Partner Suspension

STEP 01 – Validation: Search the provided mobile number in the CC portal >> Search Partner >> Search the number >> Go to details >> Check if the suspend button is available or not.

If the suspend button is available, then suspend the partner and reply to the mail. Otherwise, if the wallet status is active then proceed to the next step.

STEP 02 – Encrypt the Number: Encrypt the mobile number from the App1 server.

STEP 03 – Ask for Data: Ask for the partner data using the following query

```
SELECT A.ID,
       A.USER_TYPE,
       A.STATUS          MAP_STATUS,
       B.ACCOUNT_STATUS DFS_ACCOUNT_STATUS,
       C.CARD_STATUS    IAS_CARD_STATUS,
       D.STATUS          DMS_STATUS,
       E.STATUS          CHANNELPARTNER_STATUS
  FROM MAP.WALLET_USER     A,
       DFS.DFS_ACCOUNT      B,
       IAS.IAS_CARD_LASTTX C,
       DMS.MERCHANT          D,
       DMS.CHANNEL_PARTNER   E

 WHERE A.ID = B.USER_ID(+)
   AND TO_CHAR(A.ID) = C.USER_ID(+)
   AND A.ID = D.MERCHANT_ID_WALLET(+)
   AND A.ID = E.PARTNER_ID_WALLET(+)
   AND A.MOBILE_NUMBER in
        ('2F1507F209DA9F1868A14FE5B2721DD2');
```

STEP 04 – Check Data: Once the data is shared, check which tables contain the data. We will only update those tables on which the status is ACTIVE.

STEP 05 – Update Query: Prepare a query only for the tables with available data and the status is ACTIVE. Then, send it to **Live DB Update Query --- Partner Suspension** for execution. Here is the update query –

```
UPDATE MAP.WALLET_USER
SET STATUS='SUSPENDED',
UPDATED= SYSDATE
WHERE ID in ('27982750');

UPDATE DFS.DFS_ACCOUNT
SET ACCOUNT_STATUS='INACTIVE',
UPDATED= SYSDATE
WHERE USER_ID in ('27982750');

UPDATE IAS.IAS_CARD_LASTTX
SET CARD_STATUS='SUSPENDED',
UPDATED= SYSDATE
WHERE USER_ID in ('27982750');

UPDATE DMS.CHANNEL_PARTNER
SET STATUS='SUSPENDED',
UPDATED= SYSDATE
WHERE PARTNER_ID_WALLET in ('27982750');

UPDATE DMS.MERCHANT
SET STATUS='SUSPENDED',
UPDATED= SYSDATE
WHERE MERCHANT_ID_WALLET in ('26951384');
```

STEP 06 – Check in Portal: After the execution, check in the portal again if the wallet status is Suspended or not. Then reply to the mail.

KYC QC Accept and Reject

STEP 01 – Collect Serial Number: Collect the DKYC serial numbers (E.g. D055248878) for both KYC Accept and Reject.

STEP 02 – Ask for Data: Ask for the data using the following query

```
SELECT ID, KYC_SERIAL_NO, ACCOUNT_NO, PHOTO_ID, CUSTOMER_ID, KYC_STATUS,
       STATUS, FACE_MATCHING_STATUS, FACE_MATCHING_PERCENTAGE
  FROM DMS.KYC_INFO
 WHERE KYC_SERIAL_NO IN ('D048443252',
                         'D048448036',
                         'D048484851',
                         'D048586374',
                         'D048648217');

SELECT PHOTO_ID, COUNT (PHOTO_ID) FROM MAP.WALLET_USER
 WHERE USER_TYPE = '00' AND PHOTO_ID IN
   (SELECT PHOTO_ID FROM DMS.KYC_INFO
    WHERE KYC_SERIAL_NO IN
      ('D048443252',
       'D048448036',
       'D048484851',
       'D048586374',
       'D048648217'))
 GROUP BY PHOTO_ID;
```

STEP 03 – KYC Accept: Filter and select only those where the status is '**System Error - Need to Accept**' and proceed to the next steps.

STEP 04 – Check Data: Once the data is shared, we will focus only on the statuses marked as "System Error Need to Accept." We will use a VLOOKUP to compare the COUNT(PHOTO_ID) from the WALLET_USER table with the KYC_SERIAL_NO from the KYC_INFO table. **Any KYC_SERIAL_NO with more than one COUNT(PHOTO_ID) will not be accepted.**

STEP 05 – Check Face Matching Status: In the KYC_INFO table, filter FACE_MATCHING_STATUS = 1 and COUNT(PHOTO_ID) = 1. Then, collect KYC_SERIAL_NO and CUSTOMER_ID for the update query.

N.B: Except FACE_MATCHING_STATUS = 1 & COUNT(PHOTO_ID) = 1, do not proceed for others.

STEP 06 – KYC (Accept) Update Query: Prepare the KYC Accept Query and send to [Update Query for KYC Issue Resolution](#) for execution. Here is the query –

```
UPDATE DMS.KYC_INFO
SET KYC_STATUS = 'CS ACCEPTED',
    STATUS= 'ACCEPT',
    UPDATED = SYSDATE
WHERE KYC_SERIAL_NO IN (
    'D045792073',
    'D045792134',
    'D045795570',
    'D045795709',
    'D045796448');

UPDATE MAP.WALLET_USER
SET TRUST_LEVEL = '4', UPDATED = SYSDATE
WHERE ID in ('96211705',
             '96211646',
             '96218839',
             '96216854',
             '96219239');

UPDATE IAS.IAS_CARD_LASTTX
SET TRUST_LEVEL = '4', UPDATED = SYSDATE
WHERE USER_ID in ('96211705',
                  '96211646',
                  '96218839',
                  '96216854',
                  '96219239');

UPDATE DMS.CUSTOMER
SET TRUST_LEVEL = '4', UPDATED = SYSDATE
WHERE CUSTOMER_ID in ('96211705',
                      '96211646',
                      '96218839',
                      '96216854',
                      '96219239');
```

STEP 07 – KYC Rejection: Filter and select only those where the status is '[System Error - Need to Reject](#)' and proceed to the next steps.

STEP 08 – Rejection Data Check: Using the same data asked in STEP – 02, collect REFERENCE_ID & KYC_SERIAL_NO from KYC_INFO table.

STEP 09 – KYC (Reject) Update Query: Prepare the KYC Reject Query and send to [Live DB Query](#) --- **KYC Rejection** for execution. Here is the query –

```
INSERT INTO DMS.REJECTION_REASON_REFERENCE(
    ID,
    REFERENCE_TYPE,
    REFERENCE_ID,
    REASON_ID,
    REASON_CODE,
    REASON_MESSAGE,
    CREATED)
VALUES
    ((SELECT MAX(ID) FROM DMS.REJECTION_REASON_REFERENCE) + 1,
     'KYC_INFO',
     '2853572467',
     '7050',
     'DKYC-un-clear photo / NID',
     'DKYC-un-clear photo / NID',
     CURRENT_TIMESTAMP);

UPDATE DMS.KYC_INFO
    SET KYC_STATUS = 'CS REJECTED', STATUS = 'REJECT', UPDATED = SYSDATE
WHERE KYC_SERIAL_NO = 'D057071539';

INSERT INTO DMS.REJECTION_REASON_REFERENCE(
    ID,
    REFERENCE_TYPE,
    REFERENCE_ID,
    REASON_ID,
    REASON_CODE,
    REASON_MESSAGE,
    CREATED)
VALUES
    ((SELECT MAX(ID) FROM DMS.REJECTION_REASON_REFERENCE) + 1,
     'KYC_INFO',
     '2858266594',
     '6102',
     'Date Of Birth Mismatch/Missing',
     'Date Of Birth Mismatch/Missing',
     CURRENT_TIMESTAMP);

UPDATE DMS.KYC_INFO
    SET KYC_STATUS = 'CS REJECTED', STATUS = 'REJECT', UPDATED = SYSDATE
WHERE KYC_SERIAL_NO = 'D057165421';
```

STEP 10 Update File: Update the downloaded file and reply to the mail.

User MNO Update

STEP 01 – Ask for Data: Ask for the MAP.WALLET_USER, KNOTIFY.USER_MNO_INFO, and DMS.CUSTOMER data against the mobile numbers.

STEP 02 – Check Data: Check the shared data if there is any mismatch in MNO name. We will update only on those tables on which the mismatch is found.

The MNO names against the corresponding table is given below

EXPECTED_MNO	MAP_MNO	DMS_MNO	KNOTIFY_MNO
GRAMEENPHONE	GP	GRAMEENPHONE	GP
AIRTEL	AIRTEL	AIRTEL	AIRTEL
BANGLALINK	BLINK	BANGLALINK	BLINK
TELETALK	TTALK	TELETALK	TTALK
ROBI	ROBI	ROBI	ROBI

STEP 03 – Encrypt Mobile Number: Encrypt the mobile number from the App1 server.

STEP 04 – MNO Update Query: Prepare the MNO update query and send to [Live DB Update: User MNO Update](#) for execution. Here is a sample query

```
UPDATE MAP.WALLET_USER SET MNO_NAME='GP'  
WHERE MOBILE_NUMBER='3F788E1A5E24B51CCEEFDF6445FB3A7C';  
  
UPDATE DMS.CUSTOMER SET MNO='GRAMEENPHONE'  
WHERE ACCOUNT_NO='3F788E1A5E24B51CCEEFDF6445FB3A7C';  
  
UPDATE KNOTIFY.USER_MNO_INFO SET MNO_NAME='GP'  
WHERE MOBILE_NUMBER='3F788E1A5E24B51CCEEFDF6445FB3A7C';
```

After the execution, the issue is supposed to be resolved.

NID Blank Issue

STEP 01 – Verification: Search the number in the System portal. If two customer accounts are found, go to the details of the customer whose number starts with 1 (E.g. 117xxxxxxxx). Check if it is suspended or not. If the account is suspended, then get the MAP ID from the URL and proceed to the next steps.

STEP 02 – NID Blank Query: Prepare the query and send it to **LIVE DB UPDATE - NID BLANK** for execution. It will untag the NID and update the wallet to RESTRICTED profile. Sample query -

```
-- Frees the NID
UPDATE MAP.WALLET_USER
SET PHOTO_ID = NULL, UPDATED = SYSDATE
WHERE ID IN ('27055207','20061584','25592187');

UPDATE DMS.CUSTOMER
SET PHOTO_ID = NULL, UPDATED = SYSDATE, INTRO_NAME = 'NRB'
WHERE CUSTOMER_ID IN ('27055207','20061584','25592187');

UPDATE DMS.KYC_INFO
SET PHOTO_ID = NULL, UPDATED = SYSDATE, INTRO_NAME = 'NRB'
WHERE CUSTOMER_ID IN ('27055207','20061584','25592187');

-- Updates the profile to RESTRICTED
UPDATE MAP.WALLET_USER
SET TRUST_LEVEL = 2, UPDATED = SYSDATE
WHERE ID IN ('27055207','20061584','25592187');

UPDATE IAS.IAS_CARD_LASTTX
SET TRUST_LEVEL = 2
WHERE USER_ID IN ('27055207','20061584','25592187');

UPDATE DMS.CUSTOMER
SET TRUST_LEVEL = 2, UPDATED = SYSDATE
WHERE CUSTOMER_ID IN ('27055207','20061584','25592187');
```

After the execution of the query, the NID would be free, and the wallet would be in RESTRICTED profile.

Customer Trust Level Update

A quick overview of customer trust level against the customer profile type –

TRUST_LEVEL	DESCRIPTION
1	Limited Profile
2	Restricted Profile
4	Full Profile
8	Micro Business
32	Preliminary Profile
256	Disbursement Profile

Prepare the following query and send it to [Live DB Query: Customer Trust Level Update](#) for execution to update the customer trust level. The sample query updates the customer trust level to 4 (Full profile). Here the ID is customer MAP ID.

```
UPDATE MAP.WALLET_USER
SET TRUST_LEVEL = '4',
UPDATED = SYSDATE
WHERE ID in ('33104334','100487788','65682206');

UPDATE IAS.IAS_CARD_LASTTX
SET TRUST_LEVEL = '4',
UPDATED = SYSDATE
WHERE USER_ID in ('33104334','100487788','65682206');

UPDATE DMS.CUSTOMER
SET TRUST_LEVEL = '4',
UPDATED = SYSDATE
WHERE CUSTOMER_ID in ('33104334','100487788','65682206');
```

After the execution of the query, the customer trust level will be updated.

Customer Activation

Prepare the following query and send it to [Live DB Update Query --- Customer Activation](#) for execution to ACTIVE the customer wallet. Here the ID is customer MAP ID.

```
UPDATE MAP.WALLET_USER
SET STATUS = 'ACTIVE'
WHERE ID IN ('61628971','96891198');

UPDATE IAS.IAS_CARD_LASTTX
SET CARD_STATUS = 'ACTIVE'
WHERE USER_ID IN ('61628971','96891198');

UPDATE DFS.DFS_ACCOUNT
SET ACCOUNT_STATUS = 'ACTIVE'
WHERE USER_ID IN ('61628971','96891198');
```

After the execution of the query, the customer wallet status will be ACTIVE.

Duplicate MAP ID Issue Fix

Use the following query and send it to [LIVE DB Update Query - Duplicate Map ID Issue Fix](#) for execution to update the duplicate customer status to DELETED. Here the ID is customer MAP ID.

```
UPDATE MAP.WALLET_USER
SET
    STATUS = 'DELETED',
    UPDATED = SYSDATE,
    VERSION = VERSION + 1
WHERE ID IN ('106465495');
```

After the execution of the query, the duplicate customer wallet will be in DELETED status.

BIZ Code Update

A quick overview of BIZ code against the Account Status –

BIZ_CODE	DESCRIPTION
10	ACTIVE
13	IN BLOCK
14	OUT BLOCK
15	CO BLOCK
16	PAYMENT BLOCK
18	CS BAR

Use the following query and send it to [Live DB Update Query --- BIZ Code Update](#) for execution to update the Account Status of the user. The sample query updates the customer account status to 14 (Out Block). Here the ID is customer MAP ID.

```
UPDATE MAP.WALLET_USER
SET BIZ_CODE = '14'
WHERE ID IN ('1597054');

UPDATE DMS.CUSTOMER
SET ACCOUNT_STATUS = '14'
WHERE CUSTOMER_ID IN ('1597054');

UPDATE IAS.IAS_CARD_LASTTX
SET BIZ_CODE = '14'
WHERE USER_ID IN ('1597054');
```

After the execution of the query, the user Account Status will be updated.

Transaction Visibility Issue

STEP 01 – Ask for Data: Ask for the IAS.IAS_DFS_TXN_LOG and EXTCH.EIP_LOG data for the given transaction ID.

STEP 02 – Check Data: Check the shared data. We will go for the update query only if there is 01 (Success) status is IAS and 50 (Non-Reversible transaction. Reason failed occurred in external channel) status is EIP.

STEP 03 – Update Query: Prepare the following query and send it to [Live DB Update Query - EXTCH.eip_log Update](#) for execution. It will update the transaction status is Core DB.

```
UPDATE EXTCH.EIP_LOG E
SET
    E.REASON = '000_200',
    E.MESSAGE = 'Manual Successful',
    E.STATUS = '00',
    E.UPDATED = SYSDATE
WHERE E.ID IN ( '255647311' );
```

After the execution of the query, the transaction status in EIP will be updated.

DMS Insertion

When a customer can't submit their KYC after setting up their PIN because the data isn't inserted into the DMS table, we need to manually update the table using the query.

Before insertion, we need to free the Free ID's using the following query

```
SELECT LEVEL+( MIN_A +1) "Unused_ID"
FROM (SELECT MIN(id) MIN_A, MAX(id) MAX_A FROM DMS.CUSTOMER)
CONNECT BY LEVEL <= 100000 MINUS SELECT ID FROM DMS.CUSTOMER;
```

Use to following query and send it to **Live DB Update Query: DMS.customer** for execution.

```
INSERT INTO DMS.CUSTOMER (
    ID,
    ACCOUNT_NO,
    DOB,
    CUSTOMER_ID_MAP,
    CUSTOMER_ID,
    VERSION,
    MNO,
    TRUST_LEVEL,
    ACCOUNT_STATUS,
    ACCOUNT_APPROVAL_STATUS,
    ACCOUNT_TYPE,
    CREATED,
    UPDATED,
    FIRST_SUBMITTED_BY,
    LAST_SUBMITTED_BY,
    APPROVAL_STATUS
)
VALUES (
    1542,
    '138FDC7A1F392FAF396A02E951CE563D',
    '',
    '104687654',
    '104687654',
    1,
    'GP',
    2,
    10,
    'APPROVED',
    'BASIC',
```

```
CURRENT_TIMESTAMP,  
' ',  
'uatdemo03@gmail.com',  
'uatdemo03@gmail.com',  
'APPROVED');
```

MNO Campaign Data Check

The following query is used to check the MNO Campaign data -

```
SELECT Z.RATE_RANGE_MIN, (  
    SELECT RATE_RANGE_MIN  
    FROM RMS.RATE_RANGE  
    WHERE RATE_ID = Z.RATE_ID  
        AND RATE_RANGE_MIN > Z.RATE_RANGE_MIN  
    ORDER BY 1  
    FETCH FIRST ROWS ONLY  
) - Z.RATE_RANGE_MIN NO_OF_DENO,  
Z.RATE_VALUE_FIXED,  
Z.RATE_VALUE_PERCENT,  
Z.RATE_VALUE_MIN,  
Z.RATE_VALUE_MAX,  
(  
    SELECT ID  
    FROM RMS.PROMOTION  
    WHERE BENEFIT_DEFINITION_ID IN (  
        SELECT ID  
        FROM RMS.BENEFIT_DEFINITION  
        WHERE ACCUMULATION_RATE_ID = Z.RATE_ID  
    )  
) PROMOTION_ID,  
(  
    SELECT NAME  
    FROM RMS.PROMOTION  
    WHERE BENEFIT_DEFINITION_ID IN (  
        SELECT ID  
        FROM RMS.BENEFIT_DEFINITION  
        WHERE ACCUMULATION_RATE_ID = Z.RATE_ID  
    )  
) PROMOTION_NAME  
FROM RMS.RATE_RANGE Z  
WHERE NVL(RATE_VALUE_FIXED, 0) + NVL(RATE_VALUE_PERCENT, 0) + NVL(RATE_VALUE_MIN,  
0) + NVL(RATE_VALUE_MAX, 0) != 0
```

```
AND RATE_ID IN (
    SELECT ACCUMULATION_RATE_ID
    FROM RMS.BENEFIT_DEFINITION
    WHERE ID IN (
        SELECT BENEFIT_DEFINITION_ID
        FROM RMS.PROMOTION
        WHERE ID = ( 203780 )
    )
) ORDER BY 7,
Z.RATE_RANGE_MIN;
```

RMS(Campaign) Promotion Pause

The following query is used to pause the promotion – Live DB Update - RMS Promotion

```
UPDATE RMS.PROMOTION E
SET
E.STATUS = '4',
E.UPDATED = SYSDATE
WHERE E.ID IN ('211706','212706','212708','211709');
```

Quick Overview

The Updated List is available here - [Transaction data.xlsx](#)

Sl.	TXN Name	Related Table				Remarks
1	Mobile Recharge Via Nagad	ias.ias_dfs_transaction	extch.air_recharge_log			Status: 00 && 01 = Success, 50=Failed
2	Mobile Recharge Via Ecom	ias.ias_dfs_transaction	rpg.purchase_info			Status: 00 && 01 = Success, 50=Failed
3	Add Money	ias.dfs trx log	cs.tb charge data			Status: 00 && 01 = Success, 50=Failed and in tb status=completed
4	Transfer Money	ias.dfs trx log	cs.tb charge data			
5	Bill Payment	ias.dfs_txn	extch.eip_data			
6	OTP Issue for app login	map.wallet_user	dms.customer	ias.ias_card_last_tx	knotify.user_mno_inf	DMS.KYC_INFO
7	Customer Data	dms.Customer	map.wallet_user	ias.ias_card_last_tx		
8	DOB Mismatch	MAP.WALLET_USER	DMS.CUSTOMER	DMS.CHANNEL_PARTNER	DMS.MERCHANT	DMS.KYC_INFO
9	Partner Suspension	MAP.WALLET_USER	DFS.DFS_ACCOUNT	IAS.IAS_CARD_LASTTX	DMS.MERCHANT	DMS.CHANNEL_PARTNER
10	Bank Branch	DMS.bank_branch_inf				
11	Trust level update	IAS.IAS_CARD_LASTTX	MAP.WALLET_USER	DMS.CUSTOMER		Trust level check and update with
12	For MNO Status Check:	DMS.customer	KNOTIFY.user_mno_inf	MAP.wallet_user	DMS.kyc_info	
13	MNO Update	knotify.user_mno_info				
14	KYC info update	DMS.kyc_info	map.wallet_user			FACE_MATCHING_STATUS <60% or 0= Face match % is below threshold value ; Status>60% or 1= Issue Resolved ; photoid count in map.wallet_user
15	DMS Insertion	dms.Customer	map.wallet_user	ias.ias_card_last_tx		If no data found in dms.Customer table, then DMS Insertion needed.
16	Transaction Visibility data	IAS.ias_dfs_txn_log	EXTCH.eip_log			Status: 01 = Success, 50=Failed; if status=01 in ias and 50 in eip then update query run by db
17	SMS info check	KNOTIFY.sms_info_log				
18	Trust Level Check:	map.wallet_user	dms.customer	ias.ias_card_last_tx		
19	For Biller Transaction Check:	ias.ias.dfs.txn_log	extch.eip_log			
20	Against Order ID:	ias.ias_dfs_transaction	rpg_purchase			