Insufficient Balance and Chargebacks handling

Author : [Nishant Suri](mailto:nishant.suri@razorpay.com), [Ashish Khetani](mailto:ashish.khetani@razorpay.com)

# Quick Links

[Concept Note - Negative Balance](https://docs.google.com/document/d/1424MJnks58PDr3f6mRkCilXzVbCS3W9ZS5GnH6mLKUk/edit#heading=h.gjdgxs)  - Original concept note around negative balance.

<https://razorpay.com/docs/payments/dashboard/my-account/balances/> - Our current merchant facing documentation around the same.

# Background:

When a customer buys something and the customer is not happy with the process, product or services, they can raise a dispute to the issuing bank. If it's a Razorpay merchant the issuing bank deducts from Razorpay (happening for about 60% scenarios). Sometimes the merchant balance is not sufficient to deduct the charge bank so finops manually keep track of it and retry later. This process is complex and requires lots of human intervention.

Here’s a brief explanation of the chargeback cycle:

1. Customer raises a dispute with the issuing bank. The dispute amount could be partial or full.
2. Dispute comes to Razorpay, and the dispute amount gets deducted from Razorpay’s account.
3. If the merchant is on deduct at onset:
   1. Yes:
      1. Razorpay deducts an equivalent amount from the merchant’s account balance.
      2. If the merchant wins the dispute, then a reversal is done by the issuing bank to Razorpay and from Razorpay to the merchant
      3. If the merchant loses the dispute, then no reversal takes place.
   2. No:
      1. Razorpay doesn’t deduct this amount from the merchant’s account balance upfront.
      2. If the merchant wins the dispute then a reversal is done by the issuing bank to Razorpay
      3. If the merchant loses, then Razorpay deducts this amount from the merchant’s account balance.
4. Our problem space is 3.a and 3.b.iii when chargeback deduction is happening for the merchant account.

# Problem:

Consider this example: a customer raised a dispute of Rs. 100 and the merchant is on deduct at onset and hence, Razorpay will do an upfront debit.

1. Issuing bank deducted Rs. 100 from Razorpay.
2. Dispute Ops team uploads ([Sample file](https://docs.google.com/spreadsheets/d/19GN9GHTRyWkk-0ELoBnRX8jJV-0luPgujcaM0eUtLCM/edit#gid=1361028000))the dispute data received from the bank on the admin dashboard to create the disputes in the system.
3. While creating the dispute, the system attempts to deduct Rs. 100 from the merchant account balance, but let’s say that the account balance is Rs. 50.
4. At this point, the dispute creation will fail.
5. Dispute ops team will then turn off the flag of deduct at onset to false in the upload file for this dispute and reupload the file.
6. Dispute will be created without the deduction and the FinOps team will manually track this to do a debit in the future.
7. **This leads to three problems**:
   1. **P0 - Ops inefficiency** - Because the manual intervention is required from FinOps
   2. **P1 - Financial exposure** - Since we didn’t do upfront debit and if the merchant absconds, Razorpay is left with the loss.
   3. **Escrow reporting**

**Problem explained in numbers ( as posted by FinOps team)**

|  |  |  |  | INR Crores |
| --- | --- | --- | --- | --- |
| **Particulars** | **Mar'22** | **Apr'22** | **May'22** | **Jun'22** |
| GMV | 37,118 | 38,974 | 40,923 | 42,969 |
| Gross Chargebacks | 12.57 | 13.20 | 13.86 | 14.55 |
| Fallout | 7.22 | 7.58 | 7.96 | 8.36 |
| Fallout Recovery- Manual % | 22% |  |  |  |
| Fallout Recovery- Manual | 1.57 |  |  |  |
| Fallout recovery expected- with negative posting % |  | 40% | 40% | 40% |
| Estimated recovery |  | 3.03 | 3.18 | 3.34 |

# Solution

## Solution 1 - Let balance go negative for chargeback cases

Understanding of negative balance

There are two ways in which negative balance works in Razorpay.

Without reserve balance

Negative balance as a feature needs to be enabled on the merchant. There is a limit upto which negative balance can be there. This feature can be enabled by ops team

[Aayush Sood](mailto:aayush.sood@razorpay.com): To add data points around merchant live on the feature and limits for each merchant.

Once negative balance feature is enabled , merchants can use negative balance for selective features such as e-mandate registrations and processing refunds.

With reserve balance

Merchant can keep topping up the reserve balance and in this case , the reserve balanced is the negative balance limit for the merchant.

[Doc link](https://razorpay.com/docs/payments/dashboard/my-account/balances/)

Proposed solution

System should always deduct the amount from the balance for chargeback even if the balance is going negative. System should not check if the merchant is enabled on the negative balance feature or not.

**Note : This is only for this chargeback case , where the check “ if the merchant is on negative balance or not for the balance going negative” will be overridden. For all cases , we follow status quo**

Examples

Case 1 :

Negative balance feature is not enabled upon the merchant. Current merchant balance is 0 . Chargeback is received worth 30K

Here , the merchant balance will go -30K , irrespective of whether the feature is enabled or not.

Ledger entry : Debit: 30K, Credit 0

Now assume merchant receives payment worth 20K

Here , merchant balance will go -10K.

Ledger entry : Debit: 0 , Credit 20K. Net settlement to merchant is 0

Lets assume mechant further receives payment worth 25K

Merchant balance now becomes : 15K.

Ledger entry : Debit 0, Credit : 25K, Merchant balance: 15K

Now since the balance is +ve , settlement can be made to the merchant as per his settlement cycle.

Assume settlement of 15K is done.

Merchant balance become 0.

Ledger entry : Debit 15K, Credit 0

Case 2 :

Merchant is already enabled on negative balance.

In this case , merchant can issues refunds upto his negative balance limit . Assume it is 50K

Merchant balance is 0 and chargeback worth 80K comes in.

Merchant balance becomes -80K.

Ledger entry : Debit : 80K, credit 0

Now if the merchant tries to issue refund, it will fail because his merchant balance is lower than the negative balance limit

Now, if the merchant gets payment worth 40K

Merchant balance becomes -40K.

Leger entry : Debit 0, Credit 40K

Here , the merchant can still issue refunds upto 10K (since his negative balance limit is 50K)

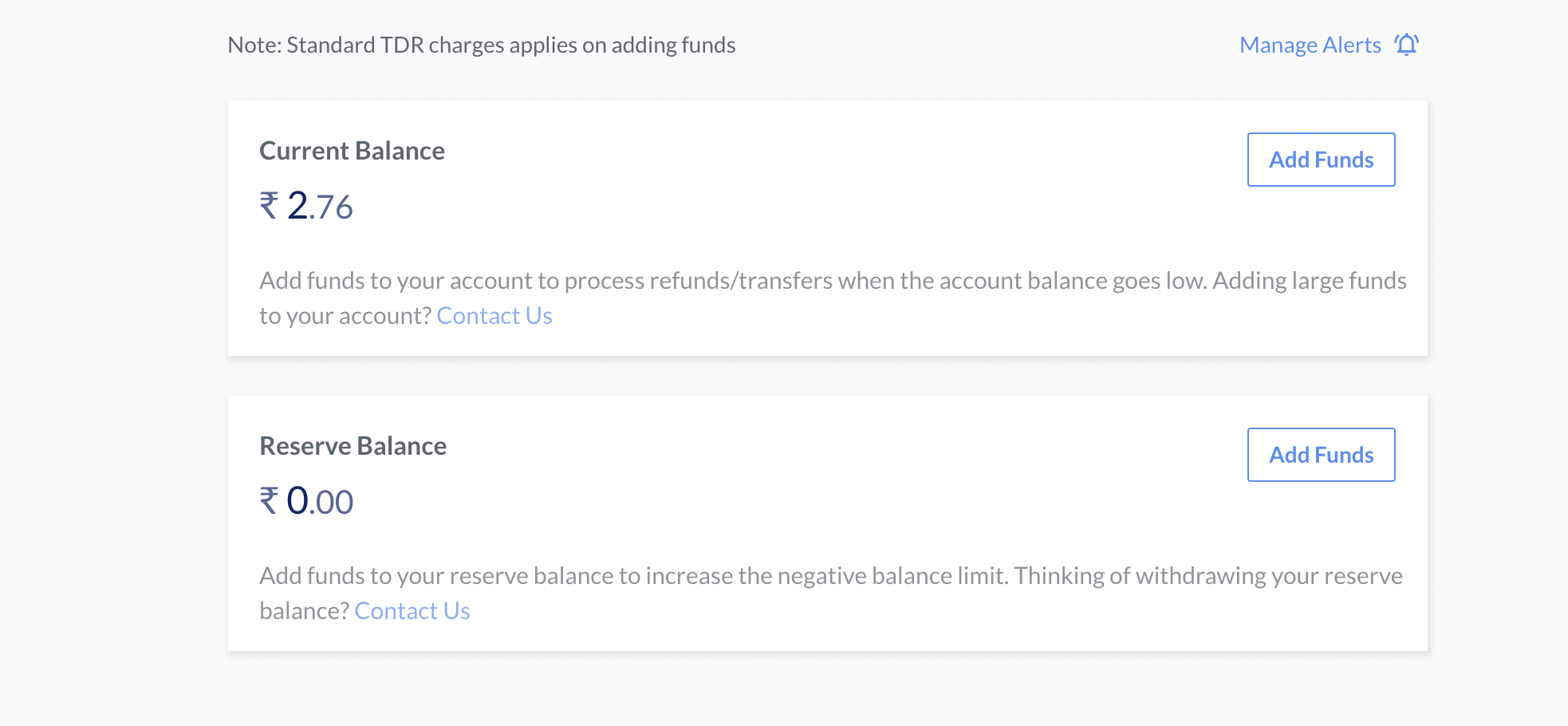
Settlements to the merchant account will only happen once the balance starts becoming positive.

What if merchant wins the dispute:

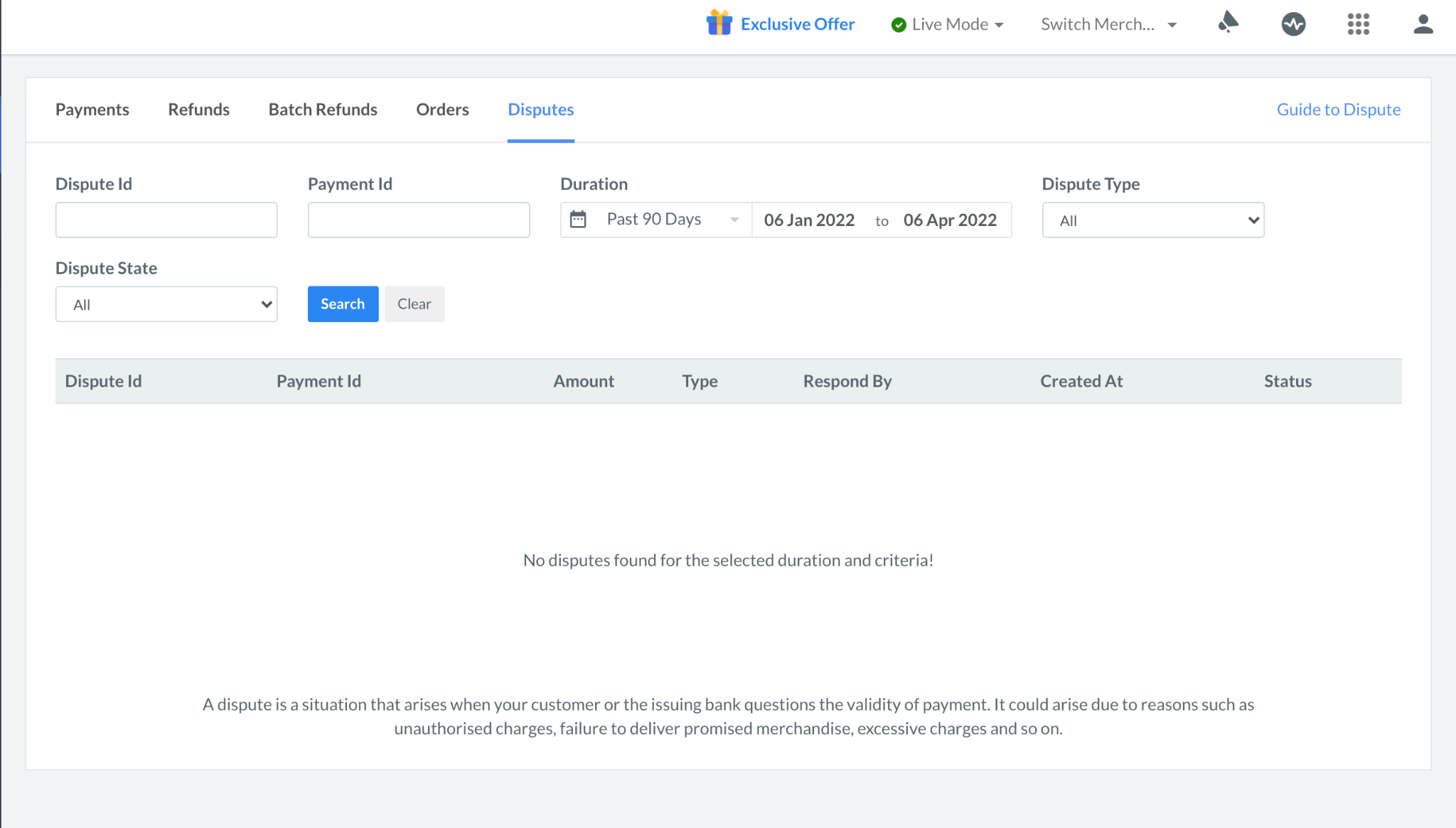
* If the merchant wins the dispute then Razorpay will be adding the dispute amount to the merchants live balance

**UI Consideration**

* Negative balance should be clearly shown to the merchant on the merchant dashboard.



* Results of the dispute will be visible in disputes tab



## Solution 2 - Retry failed dispute related adjustments periodically

Currently, the creation of the dispute has failed because adjustment cannot be created.

Adjustment cannot be created because of insufficient balance. This leads to problems outlined in the first section.

In this solution, we are proposing not to fail the adjustment creation incase of insufficient balance. Instead, the adjustment will be created in “created” status.

We will retry periodically (every X mins for Y days) to move the adjustment from “created” to “processed” status. This transition will happen only if there is sufficient balance for that adjustment in the primary balance of the merchant.

### Implementation details

1. Whenever we create an adjustment against the dispute, it should initially be in “created” status.
2. If sufficient balance, move the adjustment to “processed” status.
3. If insufficient balance, create a reminder against the adjustment.
4. Reminders will follow a pattern of checking at intervals of X mins upto Y days.
5. If any reminder succeeds in recovering the amount, then adjustment is moved to processed and amount is recovered
6. If no reminder succeeds, then adjustment is moved to failed status.

### Pros

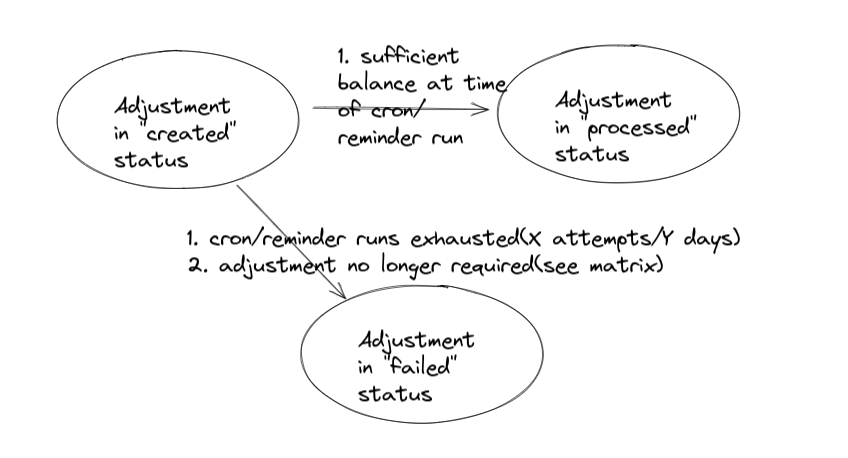
1. Will reduce tracking overhead of ops and reduce manual steps needed for them

### Cons

1. It's not going to be 100% efficient. In the following sequence of events, even though balance was available for 10 units of time, because the reminder did not run, we are unable to recover.

| Time | Event |
| --- | --- |
| 100 | Adjustment in “created” status, balance = 0  Adjustment amount = 10 |
| 110 | Reminder runs, unable to recover |
| 120 | Balance = 20 |
| 130 | Balance = 0 |
| 140 | Reminder runs, unable to recover |

* 1. The above issue can be reduced by
     1. Reducing the intervals at which reminder runs.



### Impact on dispute + adjustment entities in proposed flow

| **Scenario** | **Sufficient balance at time of dispute creation** | **Insufficient balance at time of dispute creation** |
| --- | --- | --- |
| Deduct at onset creation | 1. adjustment=processed(existing behavior) | adjustment=created, **//new**  dispute=created |
| Deduct at onset merchant won - adjustment was processed | Add positive adjustment to merchant | Add positive adjustment to merchant |
| Deduct at onset merchant won dispute - adjustment was not processed | Not applicable | Don’t add positive adjustment  Move adjustment to failed status  Update dispute statuses as required |
| Deduct at onset merchant lost dispute - adjustment was processed | No action needed | No action needed |
| Deduct at onset merchant lost - adjustment was not processed | No action needed | No action needed(reminder/cron will take care of recovery) |
| Non deduct at onset creation | No changes from existing flow | No changes from existing flow |
| Non deduct at onset merchant won | No changes from existing flow | No changes from existing flow |
| Non deduct at onset merchant lost | No changes from existing flow | 1. We need to take a call here. Currently the dispute goes into internal\_status=lost merchant not debited. 2. We can choose to use the new flow of adjustment retry, if needed |
| Adjustments unrelated to disputes | No change | 1. Based on use case, they may utilize the periodic retry feature. This will be upto implementer of the use case. |

## Recommended Approach

We are going ahead with solution 1 because of the following reasons :

* Edge cases around recovery cron and settlement to merchant
  + Say there is a scenario where merchant balance is 100 and chargeback amount is 200. Now in this case , if we go via solution 2 , chargeback will fail. Assume the next retry is supposed to happen after 8 hours and in that time the balance is settled to the merchant or the merchant avails early settlement.Then again we will have to wait for the merchant to have the money in his account when the next cron is run
* Support for partial recovery
  + Similar to the case above , support for partial recovery has to be built in as well.   
    Assume merchant balance is 4200 , and we receive 5 chargeback worth 1000 each. Here the first four chargeback will go though but the fifth one will fail and we can’t do partial recovery as well in this case ( or it will have be built)
* Manual ops work in solution 2
  + There will always be some ops effort involved in solution (2). For eg , the cron will retry every X mins for Y days, but if recovery needs to be done after that , then we will have to manually do it. In general, there is expected to more operational overhead for solution (2)
* Escrow reporting requirements
  + One of the requirements from a regulatory perspective is that merchant live balance should always represent what is due to and due from the merchant.