Specification

Specification for the TandaPay Smart Contract

5/31 Note: “Some minor changes have been made. You’ll see comments in relevant sections and a Q&A at the end that answers some questions. If you need any clarification about anything or have further questions, please feel free to reach out to me on Discord or Zulip” – Kevin

Introduction

The TandaPay Smart Contract will be the core of the application. It will contain all of the rules governing the operations of a TandaPay community. The smart contract will utilize the LUSD stablecoin. One useful analogy for understanding TandaPay is to think of it as being similar to insurance. The goal is to create a decentralized mechanism by which users can pay a premium each payment period, that premium is stored in an escrow, and the user maintains control over these funds until the moment a claim happens and they authorize their funds to go towards the claimant.

The purpose of this smart contract is to implement all of the logic relating to processing these payments and every potential user interaction within the system. The idea behind TandaPay is to create a system that eliminates fraudulent activity or collusion within the group by allowing users to retain control of their funds and to only authorize a claim that does not violate the charter (a document outlining the community’s principles).

The chief mechanism by which this is achieved is the death spiral mechanism, with users who believe a claim violated the community charter refusing to authorize their funds to go towards the claimant. This initial wave of users who refuse to authorize their funds to go towards the claimant are known as defectors. These users will have their funds reimbursed and will exit the community, but the overall coverage requirement will not decrease, causing the premiums for remaining members to increase to make up for the shortfall, encouraging more members to exit the community due to the increased premiums. This creates a feedback loop where upon a certain threshold of defectors being reached causes the community to collapse. In essence, only a small minority of members need to defect from their community in order to rapidly increase the probability that the community will collapse.

The smart contract will ensure that this system is maximally effective, because the more effective the death spiral mechanism is, the more powerful the TandaPay protocol itself is. The smart contract will change states dependent upon these community dynamics, introducing additional limitations for fractured communities, increasing the effectiveness of the death spiral mechanism.

Development Approach

Licensing

The goal of TandaPay is to be a fully decentralized and transparent whistleblowing protocol, where communities of people will band together to pool their resources in support of a specific cause outlined in their community charter. As was mentioned previously, this charter is enforced via the death spiral mechanism which this smart contract will implement. In order for this system to be effective, users must be able to put their full faith in the protocol, which requires transparency in the code and user control over their funds right up until the moment they are authorized to be paid out to a claimant. With these goals in mind, we would like to license this software under the GNU GPL v3.0 software license, keeping the software fully free and open source, auditable by any user or third party, and without outside agents being able to commercialize this code in a proprietary application.

Modeling this Software

We believe that the best way to model the TandaPay smart contract would be to use the state machine design pattern. This is a common model for smart contracts, meaning that they have different states in which they behave differently or in which different functions can be called. This is appropriate for TandaPay because depending on the community dynamics, the logic by which the smart contract operates should change in response. The state of the TandaPay smart contract may change depending on user interactions with their community, or sometimes automatically after a certain amount of time has passed and/or specific conditions are met.

External Interactions

An app will be created for TandaPay as a layer of abstraction between the smart contract layer and the users' interactions with their community. This app will need to communicate with the smart contract. It will have a built-in wallet and users will be able to perform actions within the app, which will operate programmatically by using their private key to sign transactions.

Users within the app will have different roles with different permissions which allow them to interact with the smart contract in differing ways. This means that the smart contract will need to keep track of all the members in a TandaPay community by their wallet address, and group them according to their role. It should reject transactions by any unknown wallet addresses or when a wallet attempts to interact with the smart contract in a way that is not permitted.

An instance of the smart contract will be deployed for each TandaPay community, upon initialization by a secretary who acts as the administrator of the community. This secretary is one such role with different permissions from any other community member. Development of the smart contract should be planned with this interaction in mind, ensuring that when the app is developed, the two will be able to integrate with each other seamlessly.

Adaptability

The smart contract should be developed with adaptability and modularity in mind where possible. Although we are planning to utilize the LUSD stablecoin, the smart contract should be built in such a way that this can be swapped out for another similar token with minimal code modifications for future deployments.

Overview

Modeling the TandaPay Community

In order to model the TandaPay community, the smart contract must first be aware of all community members and their roles. The member who initializes the community is the Secretary, who has all permissions related to administering the group. Everyone else is just a regular community member to the smart contract, which means they will have the default permissions involving signing their own transactions or choosing to defect.

In addition, TandaPay communities are divided into subgroups of users, which will have implications for the logic of the smart contract. The secretary is the member who will assign all of the other members to their respective subgroups. These subgroups should always be initialized to between 4 and 7 members, although they may drop below this threshold when users defect, which will be addressed later in the solutions section. The smart contract should keep track of which subgroup each community member is a part of.

Solutions

Escrows

Within TandaPay, there are two different categories of escrow where funds are held. There are community escrows, and individual savings account escrows. Members of the TandaPay community retain custody and limited control over their funds while they are in these escrows, and their funds cannot be spent without their prior consent. If they choose to exit the community, they may do so and will receive a refund of all money in the system's escrows that is rightfully theirs.

Each period within TandaPay has an associated community escrow. During the last three days of any given period, members of the community are required to pay their premiums for the upcoming period. When they pay these premiums, they are placed into the upcoming period's community escrow. Since some individuals may choose not to pay their premiums, this can cause a shortfall in the community escrow, which is where the individual savings accounts come into play. When a shortfall occurs, individuals have their savings accounts debited based on various rules to ensure that the community escrow meets the coverage requirement. The base

premium requirement to fill up the community escrow is calculated as the coverage requirement divided by the number of community members.

Each individual within a TandaPay community has a savings account escrow, and each time their premium is calculated, the amount of money required to fill up their savings account is included. The only exception is when new members join the TandaPay community, because during the first period when a new member joins, they do not receive coverage and their payment for this period is calculated as the amount required to fill up (11/12)ths of their savings account escrow. After this, all calculations are performed as normal. The required amount of funds that each user has to maintain in their savings account is calculated each time the coverage requirement is set, and ONLY when the coverage requirement is set (e.g. it does not adjust based on members defecting or exiting the community, and premiums changing). The savings account requirement is calculated as 1.2x the base premium collected during the first period where the new coverage requirement goes into effect.

When claims occur, the funds that are in the community escrow are used to pay out the claimant(s). However, individuals who believe the claim violated the charter may choose to defect, which requires them to sign a transaction to defect during the claims payment window. When an individual defects, they are refunded all of their money that is currently in the community escrow from the period where the claim took place, as well as the balance of their savings account. Individuals who defect during the claims processing window likely would not have paid their premiums, but if they did for some reason, they would also receive these funds back. When claims do not occur, the funds that are in the community escrow are simply refunded back to the members who contributed those funds.

The idea is to create a logical lock box architecture where members of the TandaPay community retain self custody of their funds. This system of escrows describes a logical lock box because users are placing their funds into the smart contract which has some additional rules as to how these funds are spent and under what conditions they are returned to the individual who has custody of them. For instance, members cannot spend their money which is currently in the system, they would have to wait for a refund. If a member gets a refund either because no claim occurred, or because they chose to exit the community and get their money back, they will receive this refund on *day 4* of the period, which is when refunds are processed. However, users still retain self custody of these funds until the moment they are paid out to a claimant, because they always have the option to not pay out a claim and to defect, receiving their money back, or to otherwise exit the community and receive their money back. Users inherently must always consent to their funds being spent and taken out of their custody.

The Payment Process

The Basics

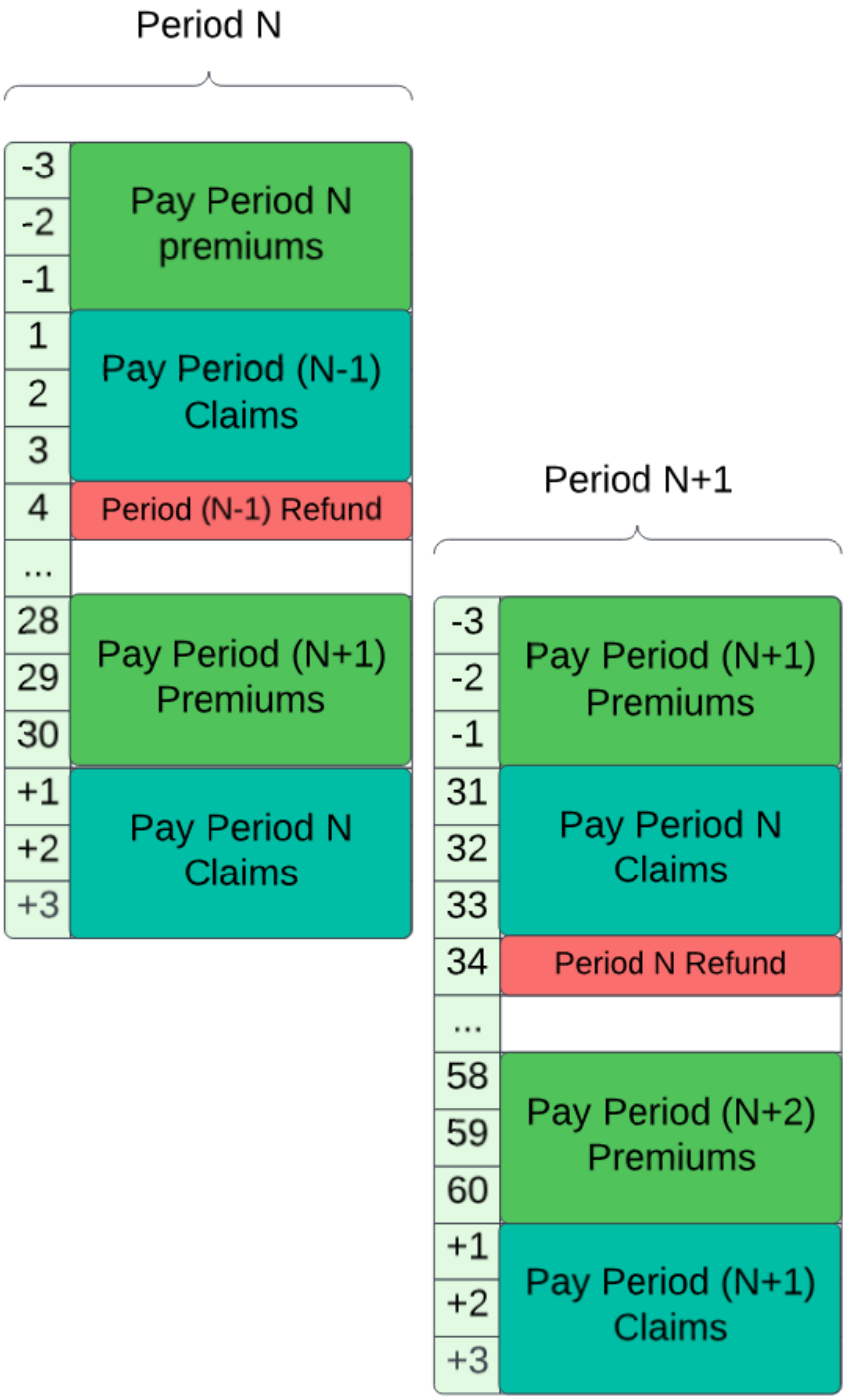
At the most basic level, members of a TandaPay community pay their premiums prior to each period, which grants them coverage during that period. These premiums go towards filling up

their savings account and towards the community escrow for that period, which is used to pay out claimants if claims occur. Since some individuals may not pay their premiums, this will create a shortfall in the community escrow, meaning that the amount of money in the community escrow is less than the coverage requirement. The savings accounts are intended to make up for this shortfall, with the remaining valid members who did pay their premiums having their savings accounts deducted.

If claim(s) are then whitelisted, members will have the opportunity to review these claims, choosing to defect if they believe it violates the charter. Although they paid their premiums and have contributed to the community escrow for the period in which the claim(s) took place, and have funds in their savings account, they are still in control of this money until the moment it is paid out to a claimant. They may choose to defect, which means that instead of their funds going towards the claimant(s), they will instead receive a refund and their fellow subgroup members' savings accounts will be deducted to make up for the defection shortfall.

All subgroups must be between four and seven members, so if defectors cause a subgroup to decrease in size to below four members, the remaining members will become *paid-invalid*, which means they will be refunded their contribution to the community escrow immediately, and will not receive coverage for the following period. Their savings accounts will still be deducted to make up for the defector shortfall like normal, however the refund they are issued will be deducted evenly from the remaining valid members' savings accounts.

To better illustrate the dynamics of the payment process, consider the following graphic that shows when various things happen during a TandaPay period:



So, to reiterate with this additional context:

- Individuals pay their premiums for the following period (N+1) during the last three days of the current period (N). If any individuals fail to pay their premium by the end of this window, they are marked as *unpaid-invalid* right away and the shortfall is deducted evenly from the savings accounts of the members who did pay.

- Any claims that were whitelisted during the current period (N) are paid after the first three days of the following period (N+1). During this three day window, individuals have the option to defect rather than having their funds go towards paying the claimant, which will create a defection shortfall. Defection shortfalls are special in that the shortfall is not divided among the entire community evenly, instead it is divided among the remaining valid members in their subgroup. If this shortfall is too great such that the subgroup's savings accounts cannot be deducted evenly to cover it, or in the case that all of the subgroup members defected, this excess shortfall is spread among the entire community.
- Refunds for the current period (N) are paid out on *day 4* of the following period (N+1). On this day, if no claims were whitelisted during period N, everyone who contributed to the community escrow and had coverage for period N will be refunded. Additionally, anyone who chose to defect or otherwise exit the community will receive a refund for all funds in the system which they have rightful ownership of.
- *Paid-invalid* members are a special case, in that they will actually be refunded their contribution to the period (N+1) community escrow during the period N refund window, meaning they will get a refund almost right away rather than having to wait a full period, and will not receive coverage for period (N+1).

At first this may seem unnecessarily complicated, but these design choices were specifically made to make the death spiral mechanism of TandaPay more effective. When an individual chooses to defect, the community will feel the pain of this at least twice, because someone who defects will not be contributing to the community escrow for the following period, and their subgroup members will also have to make up for the shortfall created by them being refunded for the period in which the claim was whitelisted.

Since defector shortfalls are split among their remaining valid subgroup members' savings accounts rather than the entire community's savings accounts, this will mean those members will have to pay a much higher premium during the next payment window, since they have to replenish their savings account. This creates a stronger incentive for those individuals to leave the community rather than paying this premium, creating even more pain for the community as a whole.

Additionally, when enough members within a subgroup defect to make that subgroup invalid, this means that the remaining members in that subgroup will become *paid-invalid*, and will receive a refund almost right away. This creates even more pain for the community, as this is yet another shortfall that will have to be divided evenly among the rest of the community members' savings accounts.

These factors combined can increase the amount of money the community members have to pay by a large sum, since not only do they have fewer members to divide the burden of meeting the coverage requirement, they also have to replenish their savings accounts during each premiums payment window. This creates a greater incentive for members to exit the community

rather than paying these higher premiums, which is at the core of the death spiral feedback loop.

Fractured Consensus and Queuing

In addition to the aforementioned mechanisms that enhance the effectiveness of the death spiral, when more than 12% of a TandaPay community chooses to defect, the community enters into the fractured state. This will be described in more detail during the section which goes over each possible state independently, but here we will discuss the effect it has on the payment process.

When a TandaPay community enters the fractured state, this creates some additional limitations on the community which makes it more difficult for them to avoid a collapse and further incentivizes members to exit the community. The primary way this is accomplished is via queuing, which delays refunds by an additional two periods. For instance, rather than receiving period N refunds during period (N+1) when no claim occurs, valid members will receive their period N refunds during period (N+3). However, if they choose to leave, they will receive their refunds during the next refunds window, further incentivizing members to exit the community.

It is possible for the community to recover from the fractured state, if they exhibit three consecutive periods of stability. A period of stability is defined as a period where nobody defects, leaves, or quits the community, regardless of whether a claim occurred. Once this happens, their queued refunds will be given to them during the next refunds window all at once, and they will return to receiving their refunds after only one period has elapsed.

In addition, new members may not be added to the community during the fractured state, and the coverage requirement may not be changed, which prevents the community from adding new members or changing the coverage requirement to mitigate the increased premiums, further increasing the effectiveness of the death spiral feedback loop.

The State Machine

A TandaPay community can be in four different states: initialization, default, fractured, or collapsed. Each state alters the smart contract logic accordingly.

Initialization State

When a TandaPay community is first created, and thus the smart contract is first deployed, it will begin in the initialization state for the first period. New members who are joining a TandaPay community do not receive coverage immediately, and must instead fill up 91.67% of their savings account requirement first. During the initialization state, every member who joins the community will be a new member, meaning nobody has coverage yet, and thus no claims may occur during this state. The premiums that are collected immediately when new members join

during the initialization state go towards their savings account escrows, and there is no community escrow for the initialization state since there are no claims to be paid out.

During this state, individuals join the community and are divided into subgroups. The coverage requirement is also set for the first time via a transaction signed by the secretary. After the community is divided into subgroups and the coverage requirement is set, and premiums are collected at the end of this state, the community will progress to the next period, which is the first period that members have coverage and claims may occur. The state will be updated to the default state at this point, and the community will never enter the initialization state again.

It is also worth noting that during the initialization state, if a member does not pay the initial premium that goes towards their savings account, they are simply kicked from the community. This is the behavior for any new members that join the community, and since they were not valid members who were expected to contribute to the community escrow, no shortfall will be generated and thus it will not have an affect on the community or be counted against them.

Default State

During the default state, the tandapay community functions as normal. There is no fractured consensus, so if a claim doesn't occur in the current period (N), individuals will receive their refunds during period (N+1). In the default state, new members may be added to the TandaPay community and the coverage requirement may be changed.

The community will enter the default state after their initialization period, and will only exit the default state if more than 12% of the community defects during any given period. If this happens, the community shifts into the fractured state, indicating that there is a fractured consensus, and making it more difficult for the community to survive. As a general rule of thumb, transactions that involve modifying the community are only allowed during the default or initialization state of TandaPay. This includes changing the coverage requirement and adding new members to the community.

During the default state, if an individual fails to pay their premium, they are marked as unpaid-invalid and the rest of the community will make up for the shortfall. The individual who did not pay will not receive coverage for the following period, but a grace period is granted and the individual will be able to pay their next premium to return to being a valid member.

Fractured State

If a claim occurs, there may be an initial wave of defectors, and if more than 12% of the community defects during this wave, the community shifts into the fractured state. The fractured state places limitations on the actions that the secretary may take, as well as delaying refunds to valid members via queueing.

To exit the fractured state and return to the default state, the community must exhibit three consecutive periods of stability. A period of stability is defined as a period in which all community members pay their premiums and none of them exit the community.

If an individual fails to pay their premium during the fractured state, they *leave* the community, receiving a refund of all money in the system which is rightfully theirs, including funds they contributed to the community escrow which have not been spent on a claim, as well as the balance of their savings account.

Additionally, if an individual becomes paid-invalid, and is not *reorged* into another subgroup, they *quit* and will also receive a refund of all money in the system that is rightfully theirs. An individual can fail to reorg if the secretary fails to assign them to another subgroup, if they do not approve of their new subgroup assignment, or if the members in the new subgroup they are being assigned to do not approve of their joining. This is the only instance in which members of a subgroup must approve of a new individual joining, and it requires one of them to sign a transaction stating their approval of this new individual joining.

In the fractured state, if the community escrow shortfall ever becomes too great for it to be eliminated via the normal rules, the secretary may have to intervene by injecting their own personal funds into the system to eliminate the shortfall, or by dividing the shortfall evenly among all valid members' savings accounts. If they choose to do the former, the community will survive simply because the coverage requirement was met due to the secretary's intervention. If they choose to do the ladder, the deduction from savings accounts will occur right before the claimant is paid out, giving members the opportunity to defect and have their funds returned, rather than having their savings account deducted to pay the claimant. When the money is deducted from the savings accounts, it must be done equitably, meaning that all remaining valid community members must have the same amount of funds deducted from their savings accounts. This makes it a risky play for the secretary, because if they sign this transaction in hopes that the savings accounts will be enough to make up for the shortfall, and then individuals defect such that the shortfall cannot be evenly divided among the community members, then the shortfall will not be eliminated and the community will collapse.

The condition by which a TandaPay community collapses is whenever the community escrow does not have enough funds to pay claimant(s) by the end of the claims payout window. If this happens, the community enters the collapsed state, and can no longer recover.

Collapsed State

The community shifts into the collapsed state when the community escrow during any given period fails to meet the coverage requirement needed to pay out claimants. In the collapsed state, all smart contract logic ceases and the community is dead. Any transactions are reverted, and the community can never recover.

Additional Details

Community Member Status Definitions

Members are assigned a different status depending on various community dynamics and the member's actions. Throughout this document, these statuses are referred to in order to be precise about exactly what has happened with a member and what the results will be. These include:

- **Valid**: A valid member of a TandaPay community is someone who currently has coverage. They paid their most recent premium and they haven't made the decision to defect or otherwise exit the community.
- **Paid-Invalid**: A member of the TandaPay community who paid their most recent premiums, but due to members within their subgroup defecting, their subgroup has decreased in size to less than four members. When a member is paid-invalid, they will be refunded their most recent premium payment without having to wait until the following period, and the remaining valid members will have their savings accounts deducted equally to make up for this shortfall in the community escrow.
- **Unpaid-Invalid**: This is a status that a member of the TandaPay community may receive if they fail to pay their premium, and the community is in the default state. They won't have coverage for the period that they didn't contribute a premium for, however, they will have the opportunity to pay their next premium and become a valid member again. The shortfall from this unpaid premium is covered by deducting the funds from the remaining valid members' savings accounts equally
- **Reorged**: This is a status that refers to a member who became paid-invalid, and then later joined a new subgroup within the TandaPay community. Individuals who reorg will need to pay a premium to contribute to the community escrow and fill their savings accounts, as usual. After that, they become valid members again.
- **User Left**: In discussion, this is often used as a verb, e.g. "User leaves the community," but it has a specific connotation. Specifically, a user who leaves the community is someone who, during the fractured state, doesn't pay their premium. They receive a refund of all money in the system that they are entitled to and exit the community.
- **Defected**: This refers to members of the community who signed the transaction to officially defect from the community, refusing to have their funds be used to pay out a claimant, exiting the community, and receiving a full refund of all funds they are entitled to within the system.
- **User Quit**: A quitter, or a user who quits the community, is someone who does not reorg after becoming paid-invalid. In this case, they receive a refund of all money they are entitled to within the system, and exit the community. A user may quit either because they choose not to reorg or because they can't reorg since no subgroups will have them.

Community Escrows Recap

Community escrows can seem somewhat confusing at first, this section is designed to reiterate ideas that may be more scattered throughout this document and to explain exactly how community escrows work.

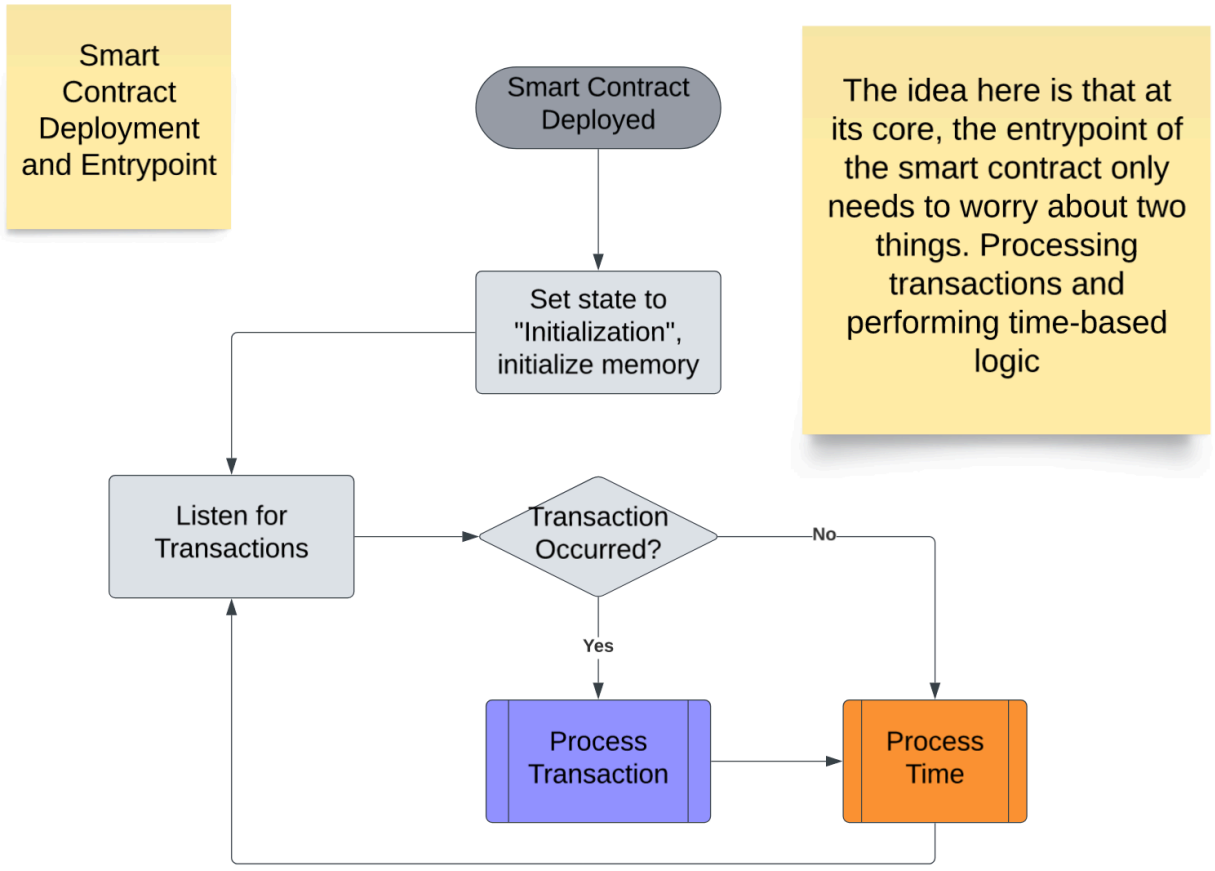
Each period within TandaPay has an associated community escrow. The community escrow for any given period is used to pay out claims that are whitelisted during that period. At the end of a period, when members pay their premiums, those funds go towards the community escrow for the next period. The community escrow for the current period is either used to pay out a claimant, or if no claims occurred, are simply refunded to the members who contributed.

This means that after the claims payout window and the refunds window, the community escrow for previous periods will always be emptied out, whether it be via refunds or paying out claimants. Funds do not accrue over time within the TandaPay protocol, which is what we mean when we refer to it as a zero reserve architecture.

Implementation

Overview

The implementation of the TandaPay Smart Contract logic is broadly classified into two categories: time-based logic and transaction-based logic. Upon being deployed, the smart contract will initialize all necessary variables and transition into the “initialization” state. Subsequently, it will begin listening for any incoming transactions and managing time-related activities, executing automatic actions as necessary depending on the time that has elapsed. A simple representation of this is shown in the following flowchart diagram:



Transactions

User interactions within the TandaPay community will be performed via several different transactions, each of which has its own associated rules and functionality within the system. The behavior of the smart contract in response to these transactions may depend on the context, including who sent the transaction, what type of transaction it is, and when the transaction was sent. Each transaction that may be used to interact with the TandaPay smart contract is listed briefly below:

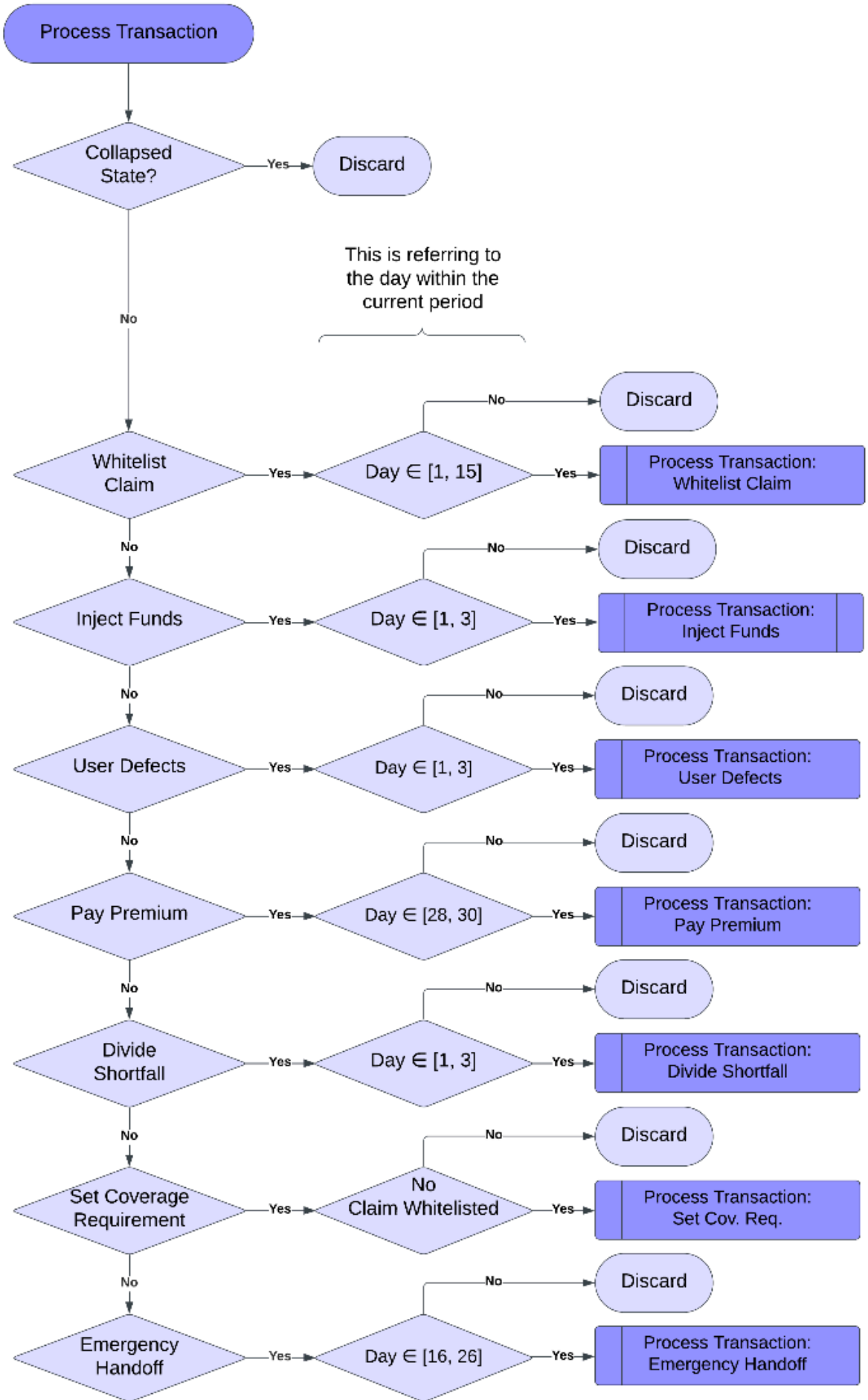
1. **A member is added to the community**: A transaction sent only by the secretary that adds a new user to the community. It accomplishes this by adding the user's wallet address to the list of community members and assigning them the status of new member so that their premiums may be calculated appropriately for them to meet all of the necessary criteria to become a valid member and receive coverage.
2. **A member is assigned to a subgroup**: A transaction sent only by the secretary that assigns a member to a subgroup. For the member to actually be added to that subgroup, another transaction from the member in question is required in order to approve this subgroup assignment.

3. **A member approves their subgroup**: A transaction sent by a member being assigned to a subgroup to approve of the decision. For a member to be added to a subgroup, the secretary must assign them, and they must also sign a transaction to approve this decision.
4. **An existing member of a subgroup approves of an individual who is joining**: A transaction sent by a member who is already in a subgroup and not moving or being reorged, to approve of an individual who is joining their subgroup. This transaction only occurs when the individual who is joining is joining due to a reorg or is otherwise already a member of the tandapay community, not if the individual who is joining is new to the community entirely.
5. **A claim is whitelisted**: A transaction sent only by the secretary to whitelist a claim. The coverage requirement will be paid out to all claimants, divided evenly among them if there are multiple claims in a single period.
6. **A member defects**: A transaction sent by a member with the intent to defect from the community because they believe a claim that was whitelisted violates the charter. This will result in all funds this user currently has rightful ownership of in the community being refunded to them, rather than making a contribution to paying out the claimant.
7. **A premium is paid**: A transaction sent by a member of the community to pay their premium for the upcoming period. Failure to pay the premium will result in the user being marked as unpaid-invalid, and thus not receiving coverage for the upcoming period. Two periods in a row of non-payment will cause an individual to be kicked from the community.
8. **The coverage requirement is changed**: A transaction sent only by the secretary to change the coverage requirement of the community. This transaction has several special rules and requirements for it to be accepted, to avoid its abuse by the secretary to prevent the protocol from working as intended.
9. **Secretary successor(s) defined**: A transaction which is sent only by the secretary to define the chain of command, essentially who will be the next individuals in charge of the community if something were to happen to the secretary to prevent them from being able to complete their duties.
10. **Secretary Handoff**: A transaction in which the secretary can choose to hand off their role to one of their successors. They sign this transaction and specify one of their successors to take over, and that successor must sign a transaction accepting the handoff in order to take over. This results in two possibilities, either a smooth handoff where the secretary hands their role over to a successor they defined, and that individual accepts it, or a one-sided handoff, where the successor the secretary specified does not sign the transaction accepting the handoff, in which case the first individual in line simply becomes the new secretary.
11. **Successor Accepts Handoff**: The transaction a successor may sign to accept a secretary handoff. If the secretary specified them to take over and they sign this transaction within 24 hours, they become the new secretary.
12. **Emergency Handoff**: A special transaction that a pair of two successors may sign to replace the secretary. This is primarily meant to be used in cases where the secretary

becomes unable to complete their duties because of external circumstances. Two successors must sign this transaction within a 24 hour window to replace the secretary.

13. **Secretary Injects Funds**: A transaction which is sent only by the secretary to inject their own personal funds into the community, which can be used to pay out a claimant if the community is incapable of doing so due to a lack of funds and a highly fractured consensus.
14. **Secretary Divides Shortfall**: A transaction which is sent only by the secretary, which can be used to pay out a claimant if the community is incapable of doing so under the normal rules due to a large shortfall and a highly fractured consensus. It divides the shortfall evenly among savings balances of remaining valid members in the community.
15. **Secretary Adds a day**: The secretary may add a 24 hour delay at the end of any given period before the next period begins. This has no effect on the logic, and is simply a mechanism to assist the secretary in maintaining alignment between the premiums due date and the calendar months.
16. **Secretary Triggers Manual Collapse**: The secretary has the ability to shut down the TandaPay community by shifting it into a collapsed state manually. If the secretary signs this transaction, the community will shift into the collapsed state at the end of the next claims payout window. All members will receive refunds for all of the funds in the system that they are entitled to at this point.
17. **Secretary Cancels Manual Collapse**: The secretary also has the ability to cancel a manual collapse before the community shifts into the collapsed state. This can be useful if they change their mind, or if the secretary is replaced before the community collapses and the new secretary wishes to cancel the previous secretary's manual collapse.

Each transaction additionally has a specific time window in which it may be executed, which is shown below in the form of a flowchart diagram:


Process Transaction
Collapsed State?
Yes
Discard
No
This is referring to the day within the current period
Whitelist Claim
Yes
Day ∈ [1, 15]
No
Discard
Yes
Process Transaction: Whitelist Claim
No
Inject Funds
Yes
Day ∈ [1, 3]
No
Discard
Yes
Process Transaction: Inject Funds
No
User Defects
Yes
Day ∈ [1, 3]
No
Discard
Yes
Process Transaction: User Defects
No
Pay Premium
Yes
Day ∈ [28, 30]
No
Discard
Yes
Process Transaction: Pay Premium
No
Divide Shortfall
Yes
Day ∈ [1, 3]
No
Discard
Yes
Process Transaction: Divide Shortfall
No
Set Coverage Requirement
Yes
No Claim Whitelisted
No
Discard
Yes
Process Transaction: Set Cov. Req.
No
Emergency Handoff
Yes
Day ∈ [16, 26]
No
Discard
Yes
Process Transaction: Emergency Handoff

A member is added to the community

Explanation

When a member is added to the community, they will be treated just like new members who are added during the initialization state. In the period when they are first added, they will have to fill up 11/12ths (91.67%) of their savings account balance and will not have coverage for that period. For the second period they are added, they will have to fill the remainder of their savings account and pay the premium for that period, and they will receive coverage for that period, becoming a regular valid member of the community.

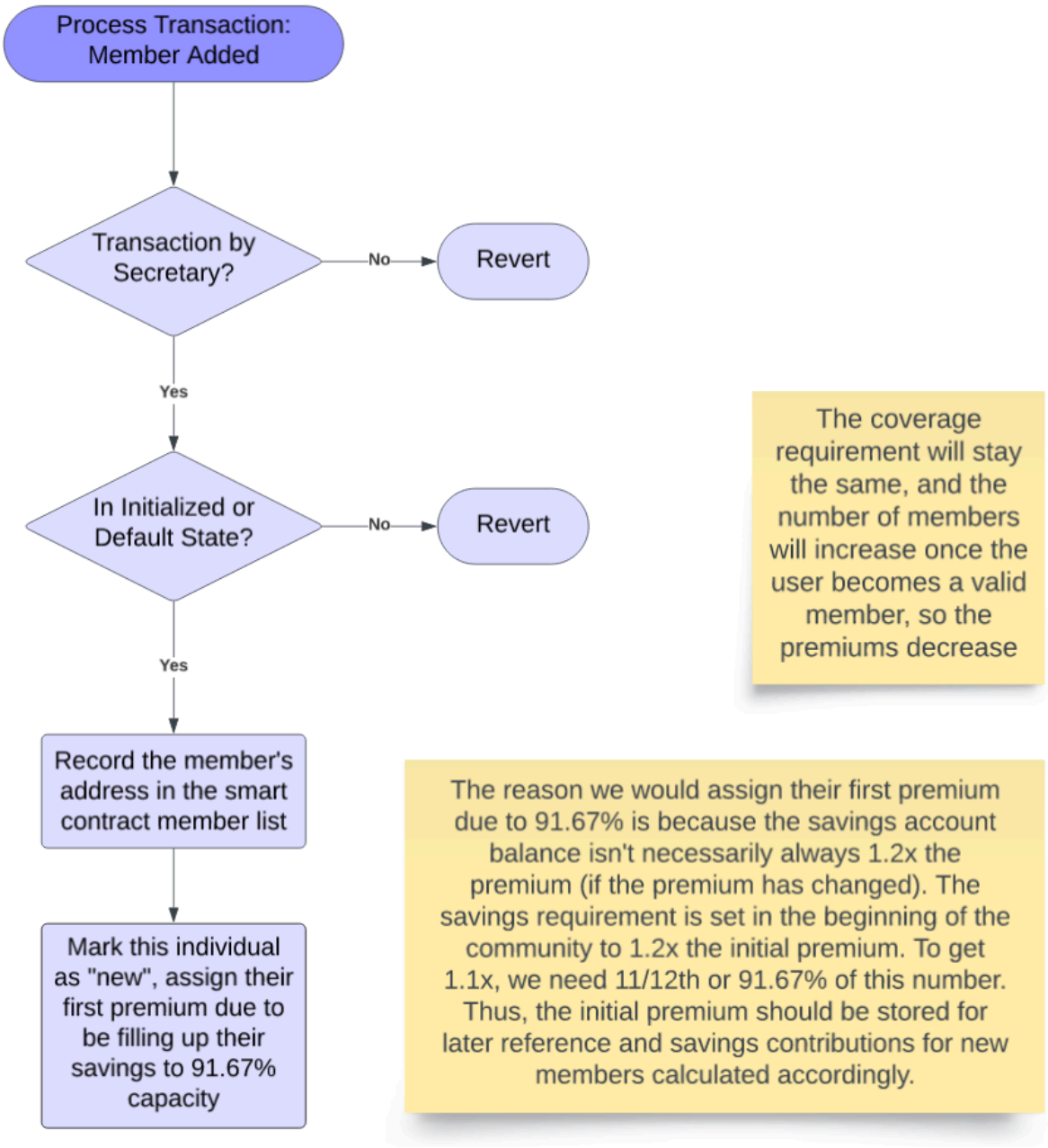
Caveats

There are two caveats when it comes to this transaction, including:

- Only the secretary may send this transaction. If it is sent by any other user, then it will be reverted. This is because the secretary is ultimately in charge of the TandaPay community, and decides who is allowed to join.
- The community must be in either the initialization or default state. Members are not allowed to join when the community has a fractured consensus.

Logic

When the transaction to add a new member to the community is received, first the two caveats must be checked. If either of the caveats are not satisfied, the transaction is reverted. Otherwise, the member's address will be recorded in the smart contract member list, and the individual will be marked as, "new," so that their first payments can be calculated accordingly, and to prevent their savings accounts from being debited during the period which they don't have coverage. Below is a graphical representation of the transaction:



And a pseudocode representation can be seen below:

```
FUNCTION process_transaction_member_added:
INPUT: sender, state, member_to_add
OUTPUT: status

      // Ensure secretary sent tx
      IF sender != secretary:
            RETURN "transaction reverted"

      // Ensure state is initialization or default
```

```
    IF state != initialization OR default:
        RETURN "transaction reverted"

    // Add member to the list and mark them as new
    member_list += member_to_add
    member_to_add.status = "NEW"
    RETURN "success"
```

A member is assigned to a subgroup

Explanation

The secretary has the option to assign a member to a subgroup, which will normally be invoked whenever a new member first joins the community and needs to be assigned to a subgroup, or whenever a member's old subgroup becomes invalid and they need to be reorged. In order for a member to be assigned to a subgroup, three conditions need to be met:

1. The secretary must sign a transaction to assign them to that group
2. They must sign a transaction to approve of their group assignment
3. If they are being reorged, each member of their new subgroup must sign a transaction to approve of their addition.

Given these conditions, this transaction alone will not directly place a member into a subgroup, but it is just one of possibly several transactions that need to be signed in order to actually place the member into the subgroup. This will be described in greater detail during the time-based logic.

In addition, the size of all subgroups within TandaPay must be between 4 and 7 members. This is *not* enforced by the transaction based logic. If the secretary makes an attempt to assign an 8th member to a subgroup, the transaction will not be reverted for that reason. Instead, subgroups with an invalid size are handled during the time-based logic.

At the end of a period, if a subgroup has too many members, the excess members will be kicked in order of newest to oldest. This will be described in greater detail in the time-based logic section of this document.

Caveats

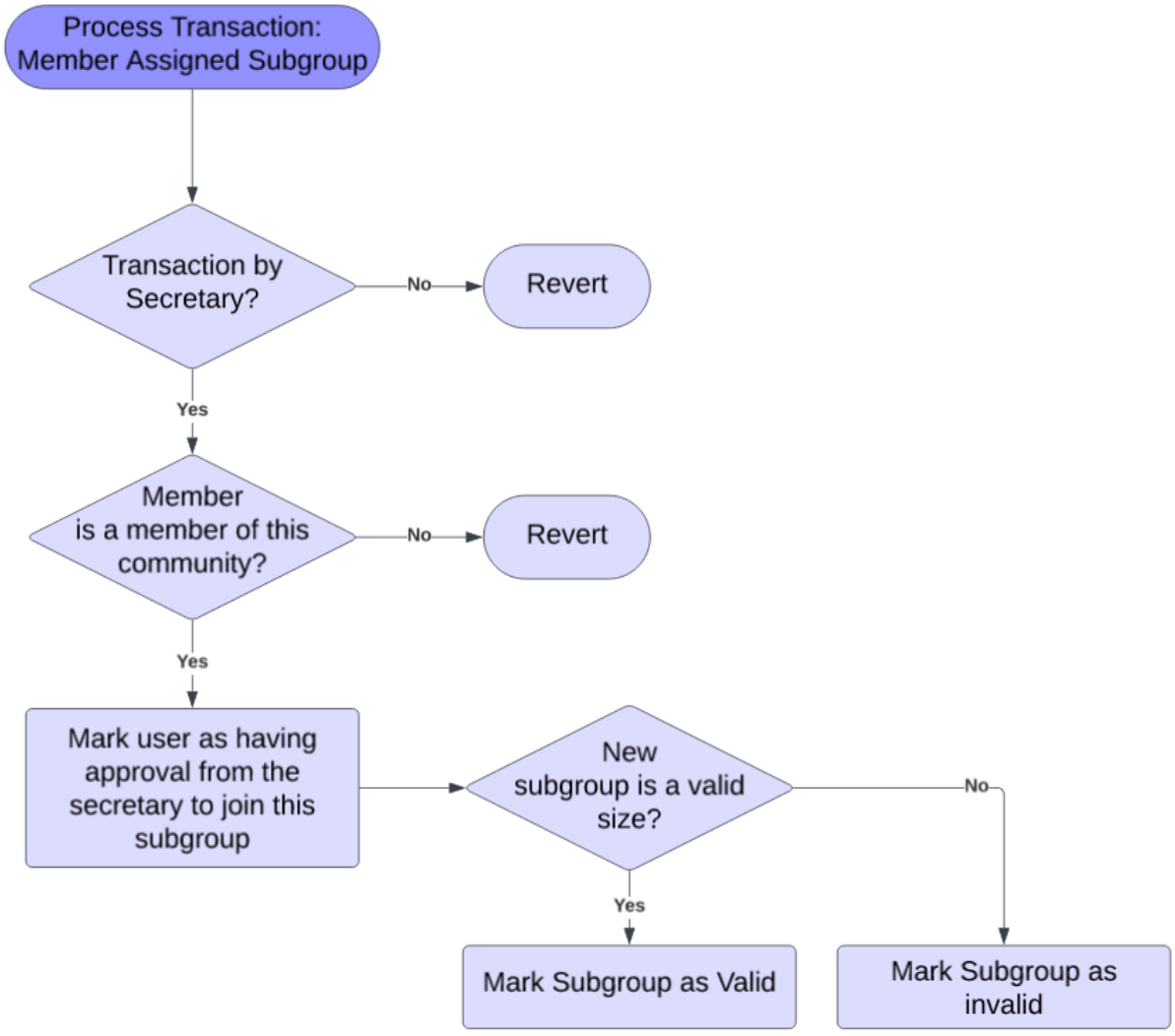
There is a very basic set of caveats for this transaction to be processed, those are:

- Only the secretary may send this transaction. If it is sent by any other user, it will be reverted.
- The individual they are trying to assign to a subgroup must already be a member of the community. If the wallet address of the individual who is being assigned to a subgroup is unrecognized by the smart contract, the transaction will be reverted.

Logic

When the transaction to assign a member to a subgroup is received, first the caveats are checked. If either of the caveats are not satisfied, the transaction is reverted. Otherwise, the user will be added to the subgroup.

Upon adding the user to the subgroup, the size of the subgroup is determined to be either valid or invalid. If the subgroup has less than four members or greater than seven, it will be marked as invalid. Otherwise, if the size is equal to 4, 5, 6, or 7, the subgroup will be marked as valid. A visual representation of the logic is available below as a flowchart diagram:



For additional clarity, some pseudocode is also included below:

```
FUNCTION process_transaction_member_assigned_subgroup:
INPUT: sender, member_to_assign, subgroup
```

```
OUTPUT: status

      // Ensure transaction was sent by secretary
      IF sender != secretary:
            RETURN "transaction reverted"

      // Ensure the individual is a member of the community
      IF member_to_assign NOT IN member_list:
            RETURN "transaction reverted"

      // Mark them with approval from the
      // secretary to join this subgroup
      subgroup.secretary_approves_join(member_to_assign)

      // Mark subgroup as either valid or invalid
      IF 4 <= subgroup.size <= 7:
            subgroup.state = "VALID"
      ELSE:
            subgroup.state = "INVALID"

      // transaction was a success
      RETURN "success"
```

A member approves their subgroup

Explanation

This is a very simple transaction that occurs after a member has been assigned to a subgroup by the secretary. They just need to, “agree,” with this decision to actually be added to the subgroup. If they do not agree, then they will be kicked from the community since they won’t have a subgroup once the period advances, which will be explained in greater detail in the time-based logic section.

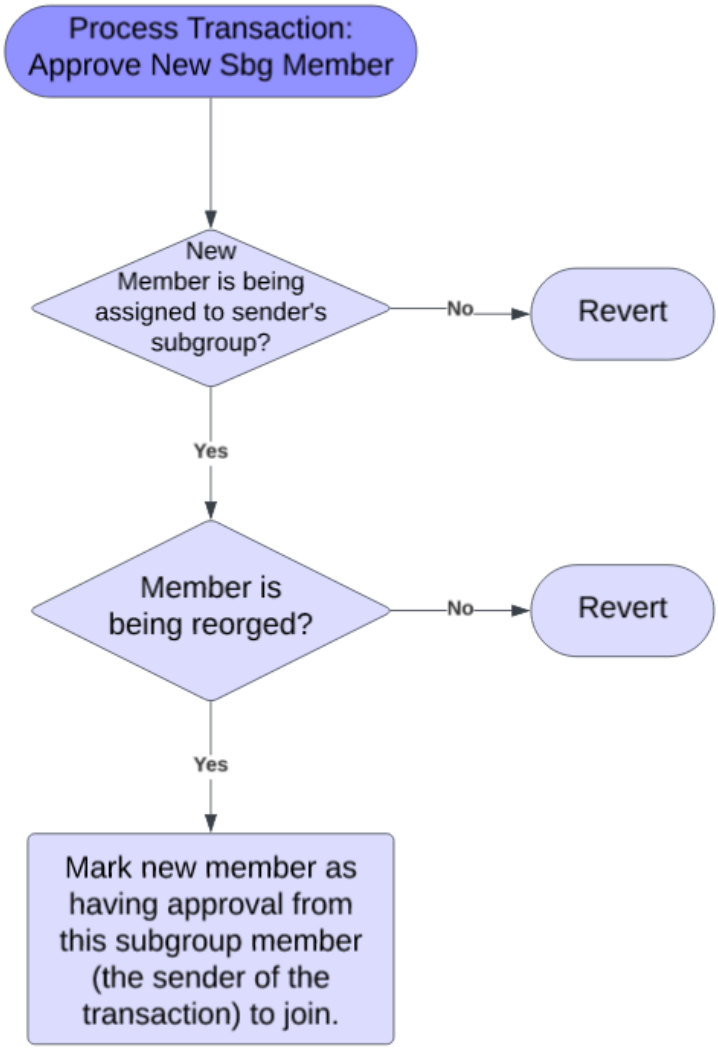
Caveats

There is only one caveat for this transaction, which is:

- The transaction needs to be sent by a member of the community who was assigned to a subgroup by the secretary.

Logic

The logic for this transaction is very simple, the caveat is checked, and if satisfied, the member is marked as having approved their new subgroup assignment. A graphical representation is shown below in the form of a flowchart diagram:



Additional pseudocode is also included:

```
FUNCTION process_transaction_member_approve_sbg_assignment
INPUT: sender, subgroup
OUTPUT: status
      // ensure sender was assigned to this subgroup
      IF NOT subgroup.has_secretary_approval(sender):
            RETURN "transaction reverted"

      // mark member as having approved their new assignment
      subgroup.self_approves_join(sender)
```

Approve new subgroup member

Explanation

This transaction is another simple transaction relating to adding a member to a subgroup. Essentially, when a community member is being reorged into a new subgroup, one of the subgroup members will also have to approve the new member who is joining.

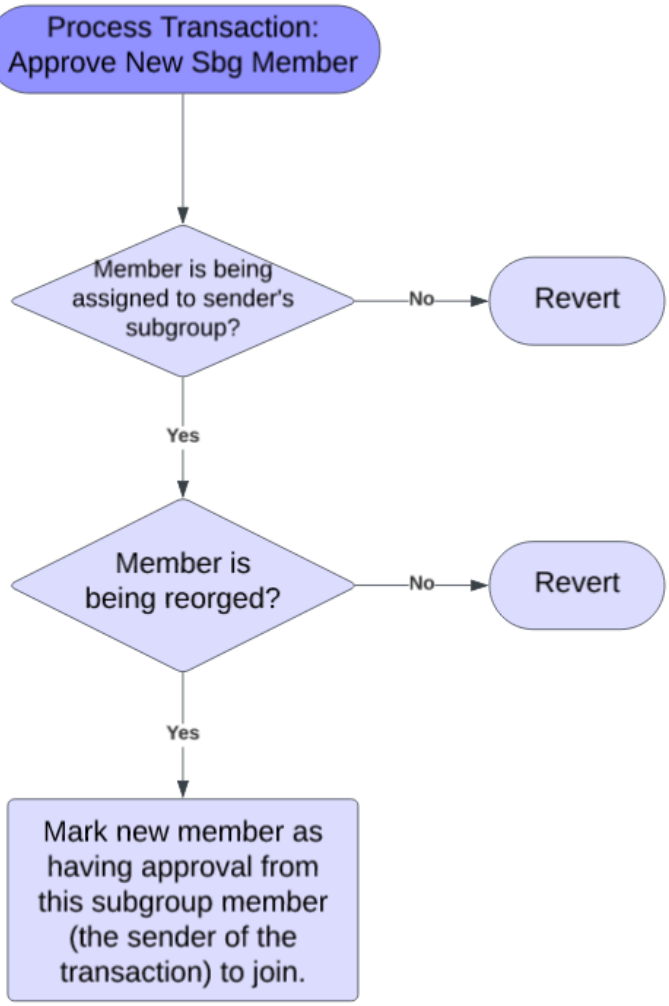
Caveats

There are two simple caveats with this transaction:

- The sender is approving a member who is being assigned to their subgroup by the secretary. If not, the transaction is reverted.
- The member who was assigned to their subgroup is reorging, e.g. not a new member. If they are new, the transaction is reverted.

Logic

The logic for this transaction involves simply checking that the caveats are satisfied, reverting the transaction if they are not, and otherwise marking the member who is joining as having approval by the sender of this transaction. Below is a flow chart representation of this logic:



Additional pseudocode is also included:

```
FUNCTION process_transaction_member_approve_sbg_assignment
INPUT: sender, subgroup, member_joining
OUTPUT: status
      // ensure sender was assigned to this subgroup
      IF NOT subgroup.has_secretary_approval(member_joining):
            RETURN "transaction reverted"

      // ensure member is being reorged
      IF member_joining.status == "PAID-INVALID"
            RETURN "transaction reverted"

      // mark this subgroup member (sender) as having approved the
      // new member who is joining the subgroup.
      subgroup.existing_sbg_member_approves(sender, member_joining)
```

A claim is whitelisted

Explanation

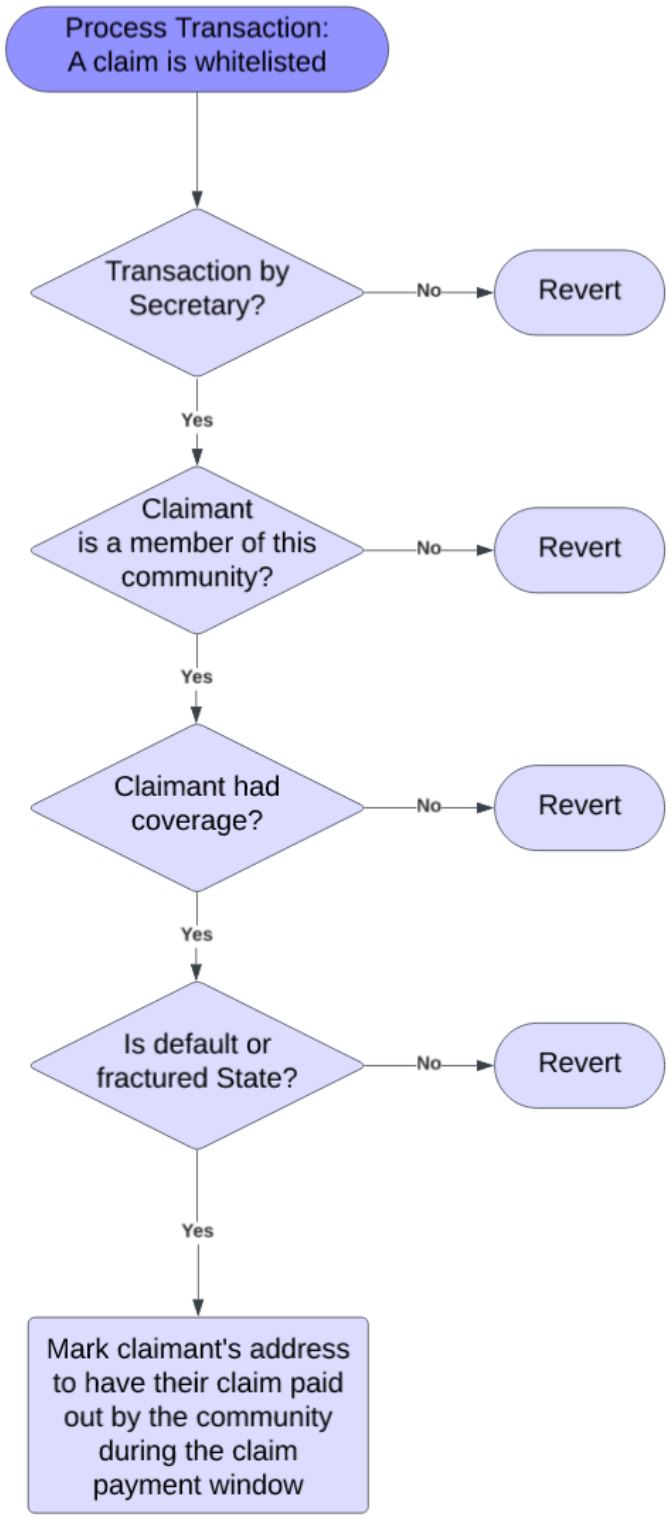
When a claim occurs, the secretary determines whether or not it is eligible and if so, will whitelist it. This transaction involves whitelisting the claimant's address to have the coverage requirement paid out to them. If there are more than one claimants, the coverage requirement is divided evenly among all of them. This transaction can only occur in periods where community members have coverage. In the first period, the community is in the initialization state because everyone is filling up their savings accounts, and thus nobody has coverage, so no claims may happen. When the community collapses, the smart contract enters the collapsed state, becoming inactive and reverting all transactions. This means that claims may only happen when the community is in either the default or fractured state.

Caveats

- This transaction may only be completed between the 1st and 15th day of the period, if it is past the 15th day, the transaction will be reverted and the secretary will need to wait until the next period in order to whitelist the claim.
- This transaction may only be sent by the secretary, if it is received from any other wallet address, then it will simply revert
- The claimant must be a valid member of the community, meaning they are an active member who currently has coverage, otherwise they can not be whitelisted.
- The community must either be in the default state or the fractured state.

Logic

This is a transaction that must be completed within a specific time window during the period, because it will be reverted if it is past the 15th day of the period. All of the caveats must also be satisfied in order for the transaction to be successful, otherwise it will be reverted. Below is a graphical representation in the form of a flowchart diagram; the time window is not shown in this flowchart, and a separate diagram is provided in this document which consolidates all of the logic regarding the acceptable time windows for different transactions:



For the secretary whitelisting claims, all they really need to do is meet these conditions and then it is whitelisted. The claimant needs to be a member of the community who has coverage, and claims may only happen when the smart contract is in the default or fractured state (e.g. not collapsed or initialize)

The logic for handling claim payments, calculating everyone's new premiums, and dealing with shortfalls will be taken care of by the time-based logic rather than the transaction logic.

Additionally, some pseudocode is shown below for increased clarity:

```
FUNCTION process_transaction_whitelist_claim:
INPUT: sender, claimant, current_day_in_period, state
OUTPUT: status
      // ensure the secretary is sending this transaction
      IF sender != secretary:
            RETURN "transaction reverted"

      // ensure that the day is between 1 and 15
      IF current_day_in_period > 15:
            RETURN "transaction reverted"

      // ensure the claimant is a member of the community
      IF claimant NOT IN member_list:
            RETURN "transaction reverted"

      // ensure the claimant is a member who has coverage
      IF claimant.status != "VALID":
            RETURN "transaction reverted"

      // ensure that the state is default or fractured
      IF state != default OR fractured:
            RETURN "transaction reverted"

      // whitelist the claimant's address to be paid out
      whitelist_claim(claimant)
      RETURN "success"
```

A member defects

Explanation

When a member of the community defects, this is a statement that the individual believes a claim whitelisted by the secretary violates the charter. Every individual has the opportunity to defect during the claims payout window, prior to the claim being paid out. Defections do not happen automatically if the user simply doesn't pay their premium. If an individual doesn't pay their premium, they will become unpaid-invalid, and lose coverage for the following period; however, if a claim occurred, the funds they previously paid in will still go towards the claimant unless they actively make the decision to defect from the community.

When an individual defects, they exit the community and receive a full refund of all funds within the system that are rightfully theirs. This includes the balance of their savings account, as well as any funds they have paid into the community escrow which have not already gone to a claimant.

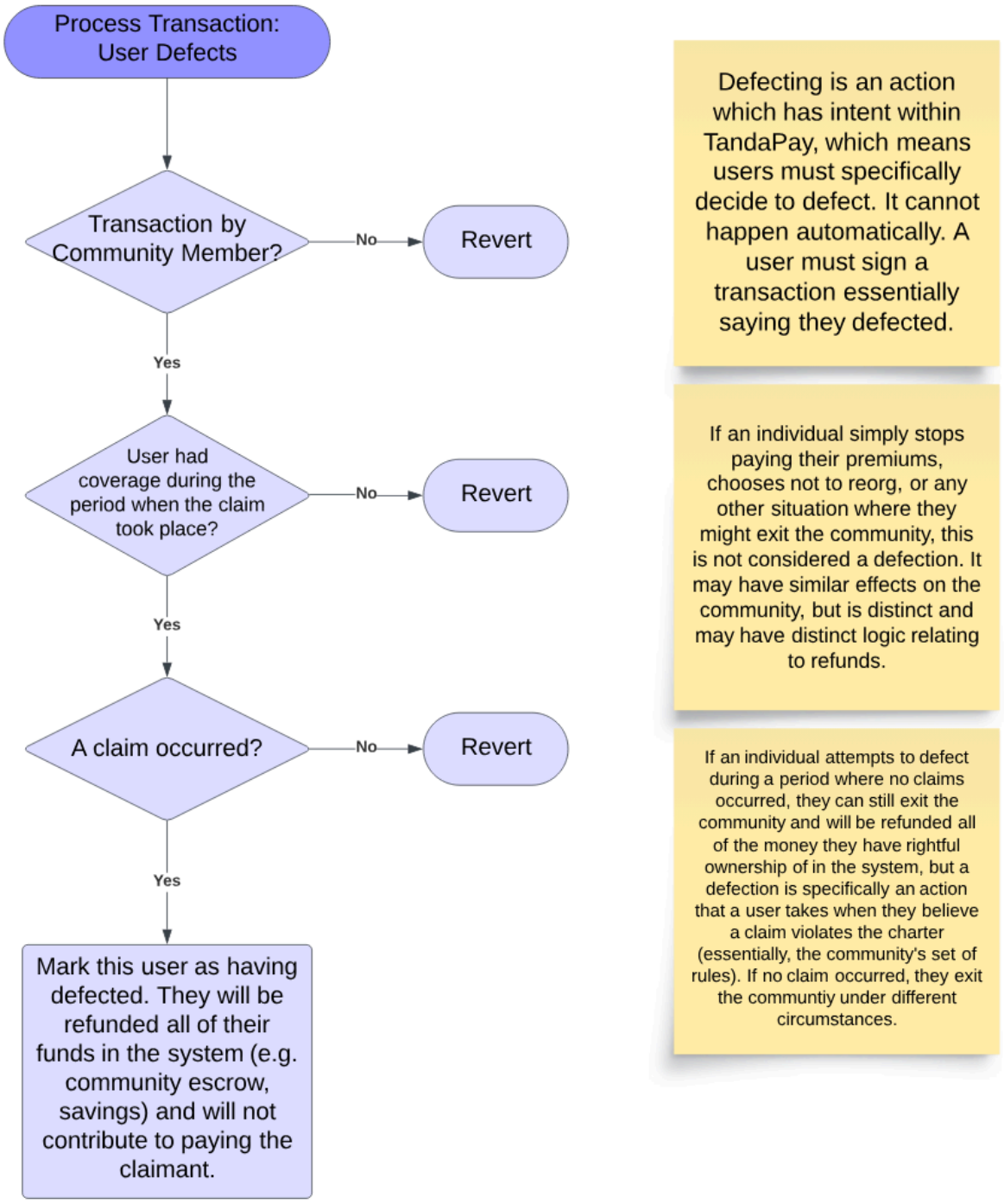
Caveats

There are a few simple caveats to ensure that the transaction makes sense, but otherwise there is nothing that ever necessarily prevents a user from exiting the community with their funds. These caveats include:

- The transaction must have been made by a member of the community (e.g. a wallet address that is on the smart contract's member list)
- The user needs to have had coverage when the claim took place, otherwise they are just considered a quitter. They may still exit the community and receive a refund but it is not referred to as a defection, as we want to reserve the term, "defection," as an action with specific intent.
- A claim needs to have occurred, otherwise it doesn't make sense for the individual to defect, since a defection is a statement about the claim violating the charter.
- The transaction must occur during the claims payment processing window, which is between day 1 and 3 of the period.

Logic

Whenever a user exits the community by signing the defection transaction during the claims payment processing window, they will be considered a defector, provided that the caveats are satisfied, otherwise the transaction will be reverted. If the transaction is successful, they are refunded all of the funds that they have rightful ownership of, from their savings account and what they contributed to the community escrow, on day 4 when refunds are processed. These funds will no longer be used to contribute to paying out the claimant(s), and thus the remaining members of the community will have to make up for the shortfall. This logic is illustrated below in the form of a flowchart diagram:



Pseudocode is also included below:

```
FUNCTION process_transaction_member_defects:
INPUT: sender, claimant_list, current_day_in_period
OUTPUT: status
     // must be in claim payment processing window
```

```
    IF current_day_in_period > 3:
        RETURN "transaction reverted"

    // sender must be a member of the community
    IF sender NOT IN member_list:
        RETURN "transaction reverted"

    // sender must be liable for this claim, e.g. had
    // coverage at the time it was whitelisted
    IF claimant.status != VALID:
        RETURN "transaction reverted"

    // a claim needs to have occurred
    IF claimant_list IS EMPTY:
        RETURN "transaction reverted"

    // mark this individual as a defector, which will
    // refund all of their money on day 4, refunds day
    mark_as_defector(sender)
    RETURN "success"
```

A premium is paid

Explanation

During the last 3 days of each period, premiums are paid for the following period. The base premium is calculated as the coverage requirement divided by the number of valid members in the community. The amount each individual actually has to pay will be their base premium, in addition to the amount required to fill up their savings account. If their savings account is full, then they will simply pay the base premium. The exception to this is when a member first joins the community; during the first period when they do not receive coverage, they will simply fill up 91.67% of their savings account, and then in the following period, they will fill the remainder as well as pay the base premium, as this will be the first period in which they receive coverage.

Caveats

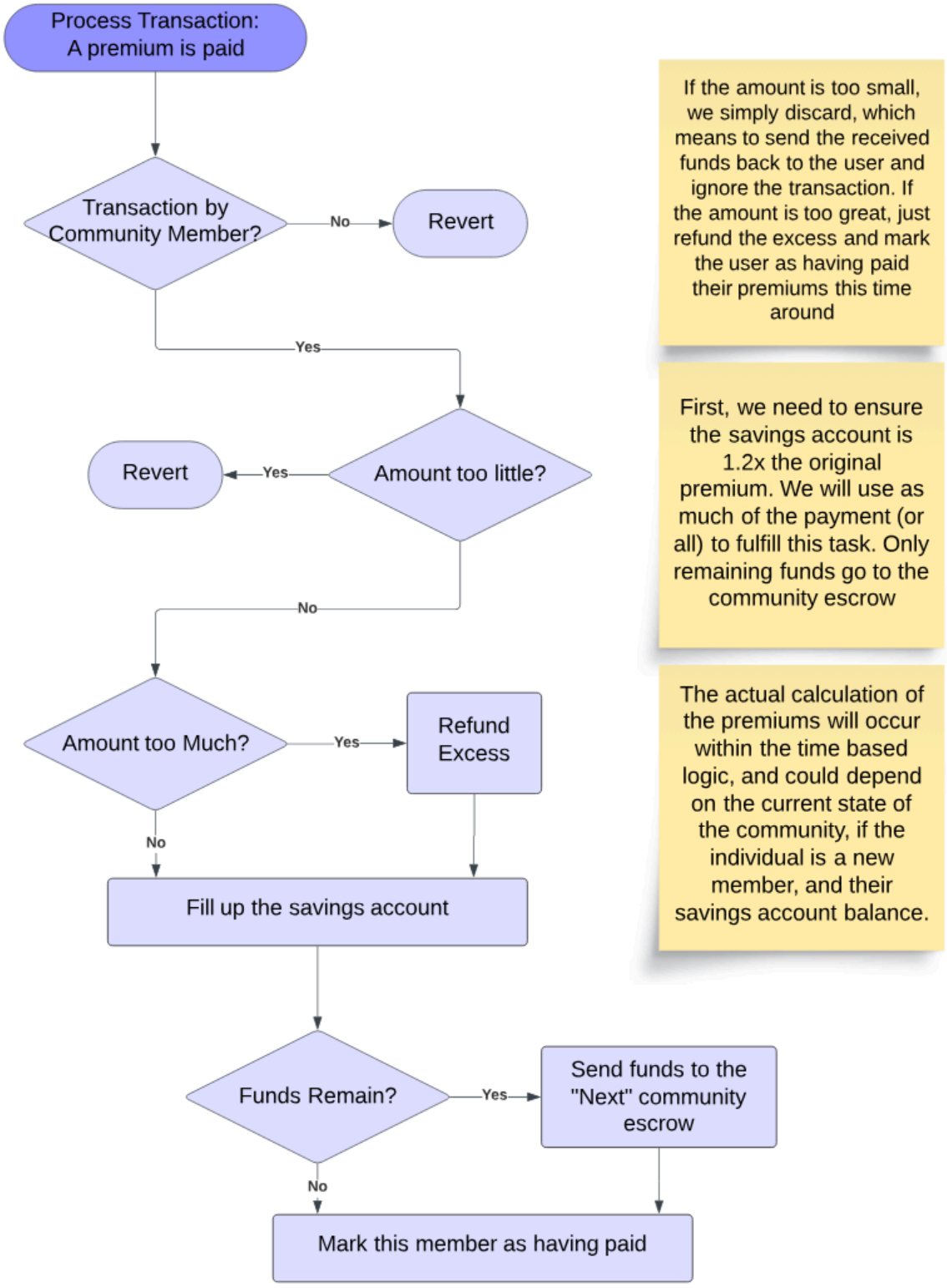
There are several caveats for this transaction, including:

- The transaction must be made by a member of the community, otherwise it will be reverted
- If the amount is too small, then the transaction will be reverted
- If the amount is too large, the excess funds will be refunded to the sender

Logic

When a premium is paid, if the caveats are satisfied, then the transaction will be successful. The funds that are paid in will first go towards filling up the savings account, and whatever is leftover will be contributed to the community escrow for the next period. Finally, the member is marked

as having paid and will retain their coverage for the next period. Below is a graphical representation of this logic, in the form of a flowchart diagram:



In addition, pseudocode is provided below:

```
FUNCTION process_transaction_premium_paid
INPUT: sender, amount, current_day_in_period
OUTPUT: status
      // must be in the premium payment window
      IF current_day_in_period < 28:
            RETURN "transaction reverted"

      // must be a community member
      IF sender NOT IN member_list:
            RETURN "transaction reverted"

      // must not be too small of an amount
      IF amount < sender.total_owed:
            RETURN "transaction reverted"

      // refund excess if too much was sent
      IF sender.total_owed < amount:
            issue_refund(sender, amount - sender.total_owed)

      // process payment by first filling up the
      // savings account, and using whatever is left
      // to contribute to next period's community escrow
      process_payment(sender, amount)
      RETURN "success"
```

The coverage requirement is changed

Explanation

The coverage requirement of a TandaPay community is the total amount of money that must be paid into each period's community escrow to be paid out to claimants in the event of a claim. Every valid member who pays their premium contributes to their share to the coverage requirement. When individuals do not pay their premiums, the shortfall is divided up evenly among the rest of the community's savings accounts, to ensure that the community escrow always meets the coverage requirement.

The secretary may sign a transaction to set the coverage requirement. This is one of the first things that is done when the community is first instantiated, but as the community evolves over time, the secretary may adjust the coverage requirement. Any time the coverage requirement is altered, it also alters the savings account requirement for every member. When the coverage requirement is set, the base premium is initially calculated as the coverage requirement divided by the number of valid members in the community during the premiums collection window. The savings account balance is also set at this time, as 1.2x the base premium. Unlike the base

premium however, the savings account requirement does NOT change after members exit the community, only whenever the coverage requirement is changed.

Within the TandaPay app which will interact with the smart contract, a vote will be conducted and the secretary will be presented with an option to sign the transaction for the agreed upon coverage requirement. This logic is encoded in the app, and is not relevant to the smart contract. For the purposes of the smart contract, the secretary simply has the permission required to set the coverage requirement; however, if they were to somehow circumvent the app and set the coverage requirement without the vote, this would be seen on the blockchain and would immediately cause the community to distrust the secretary, so no voting logic is actually included in the smart contract itself.

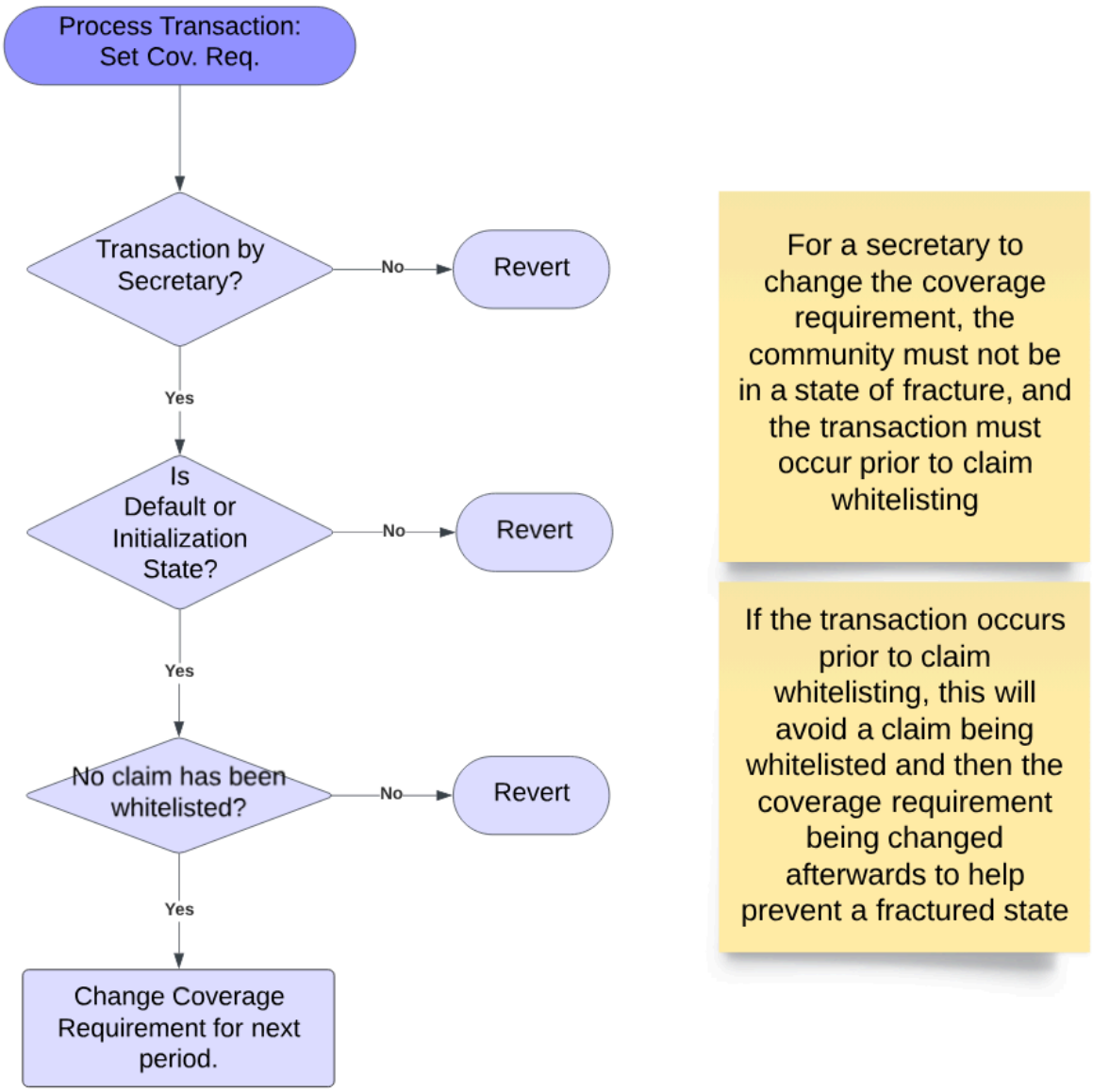
Caveats

There are three main caveats for setting the coverage requirement:

- This transaction may only be signed by the secretary, otherwise, it is reverted.
- The community must be in either the initialization or the default state. If the community has a fractured consensus, the ability to alter the coverage requirement could be abused to give the secretary a mechanism by which to prevent the group collapsing, by simply reducing the coverage requirement.
- No claim has been whitelisted yet during this period

Logic

Provided that the caveats are satisfied, the new coverage requirement will go into effect for the following period, with all of the premiums being recalculated as well as the new savings requirements. If the coverage requirement decreased, individuals who have excess funds in their savings accounts will receive a refund, and their base premiums will decrease as well. If the coverage requirement increases, the base premiums will increase and all members will also be required to pay an additional amount to fill up their savings accounts to the new threshold. Below, this logic is displayed in the form of a flow chart diagram:



Additionally, some pseudocode is provided:

```
FUNCTION process_transaction_set_coverage_requirement:
INPUT: sender, amount, claimant_list
OUTPUT: status
      // transaction must be by secretary
      IF sender IS NOT secretary:
            RETURN "transaction reverted"

      // state must be initialization or default
      IF state != default OR initialization:
            RETURN "transaction reverted"

      // ensure no claim has been whitelisted
```

```
    IF claimant_list IS NOT EMPTY:
        RETURN "transaction reverted"

    // set the new coverage requirement,
    // recalculate savings account requirements
    set_coverage_requirement(amount)
    RETURN "success"
```

Secretary successor(s) defined

Explanation

Each TandaPay community has a line of succession which defines the members who will take over in the event that the secretary needs to be replaced. For communities between 12 and 35 members, at least two successors must be defined. For communities larger than 35 subgroup members, at least six successors should be defined. There are three scenarios in which the secretary may be replaced by a successor:

- **Smooth handoff**: The secretary signs a handoff transaction and specifies a member who is already in the line of succession to replace them. It can be any member in the line of succession, but this member will also have to sign a transaction within a day of the secretary signing the handoff transaction in order to take on the role of secretary.
- **One-sided handoff**: In the event that the secretary signs the handoff transaction and specifies a successor to replace them, but this successor does not reciprocate by signing the handoff acceptance transaction, the first member in the line of succession shall replace the secretary instead.
- **Emergency handoff**: If two successors use their keys within the same day to sign the emergency handoff transaction where they specify a successor to take over the role of secretary, then the current secretary will be removed and the successor will take their place.

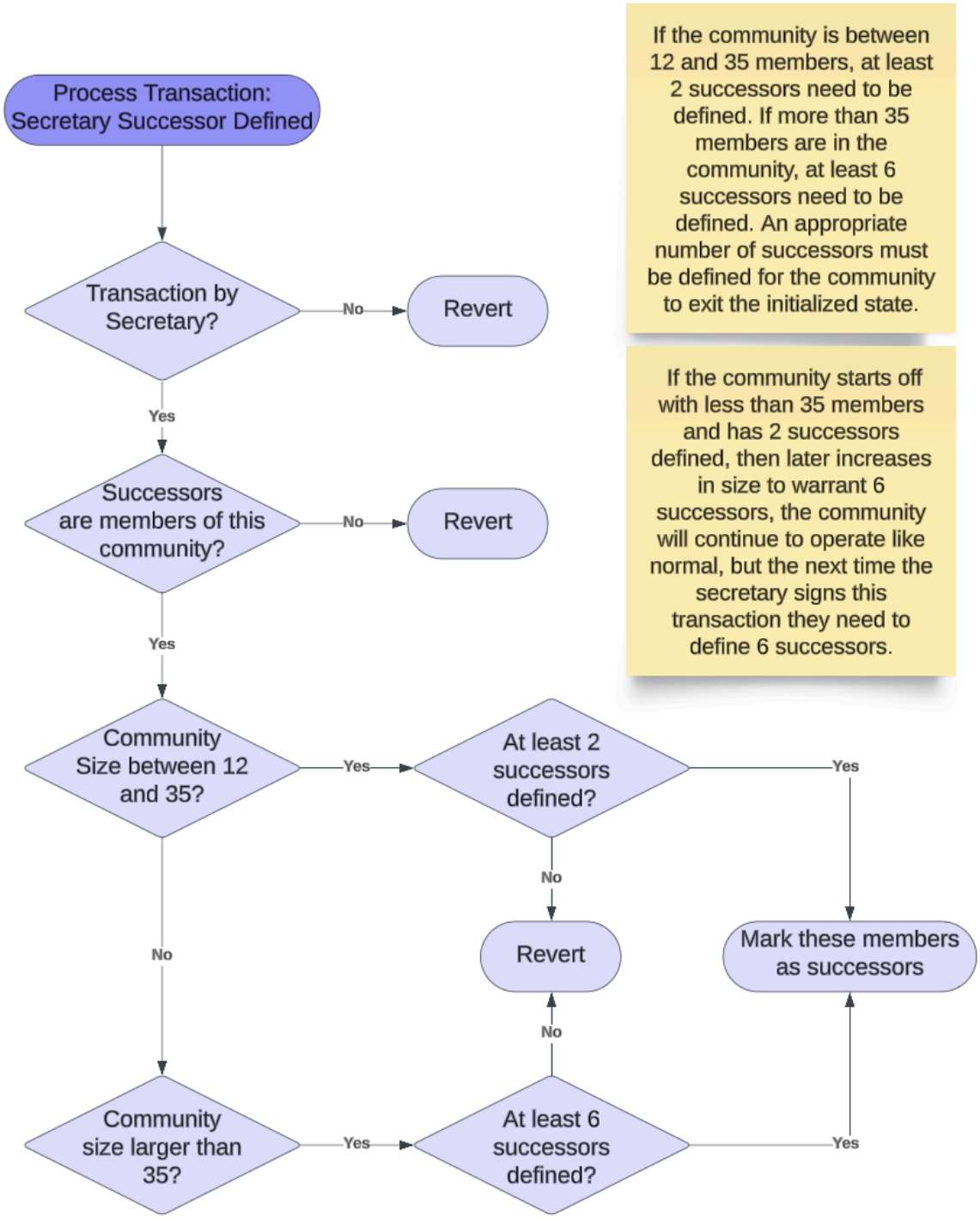
Caveats

This transaction has a few caveats, two of which mandate the amount of successors that must be defined, and one which has an effect on the state of the community as a whole:

- If the community is between 12 and 35 members in size, then for this transaction to succeed, at least two successors must be defined.
- If the community is larger than 35 members in size, then for this transaction to succeed, at least six successors must be defined.
- The community will remain in the initialization state until this transaction has been signed by the secretary to define his or her successors.

Logic

Provided that the caveats are satisfied, this transaction takes effect immediately with no delays. The line of succession is defined in full each time this transaction is signed. If the community starts off with less than 35 members and two successors are defined, but later it grows to be larger than 35 members, the community will continue to function as normal with only two successors; however, if the secretary signs this transaction again it will require six successors.



Additionally, pseudocode is provided:

```
FUNCTION process_transaction_define_secretary_successors:
INPUT: sender, successor_list, community_size, member_list
```

```
OUTPUT: status
      // transaction must be by secretary
      IF sender IS NOT secretary:
            RETURN "transaction reverted"

      // ensure that all successors are in this community
      FOR successor IN successor_list:
            IF successor NOT IN member_list:
                  RETURN "transaction reverted"

      // must have at least two successors defined
      IF successor_list.length < 2:
            RETURN "transaction reverted"

      // must have at least six successors defined if
      // community size is greater than 35 total members
      IF successor_list.length < 6 AND 35 < community_size:
            RETURN "transaction reverted"

      // set the line of succession to these successors,
      // in the order of which they were provided.
      set_successors(successor_list)
      RETURN "success"
```

Secretary Handoff

Explanation

When the secretary chooses to step down and have someone replace them, they will sign this transaction. This transaction can lead to either the smooth handoff scenario or the one-sided handoff scenario, depending on if the successor the secretary asks to replace them reciprocates this by signing the handoff acceptance transaction within 24 hours.

Upon being replaced, the secretary is added to the line of successors, taking the place of the individual who just became the new secretary. This ensures that the line of successors does not decrease in size when this transaction occurs, preventing a new secretary from having to sign a transaction defining a new line of succession every time. This is likely the behavior that most communities would want anyways, and if they choose to make a change, they still have that option.

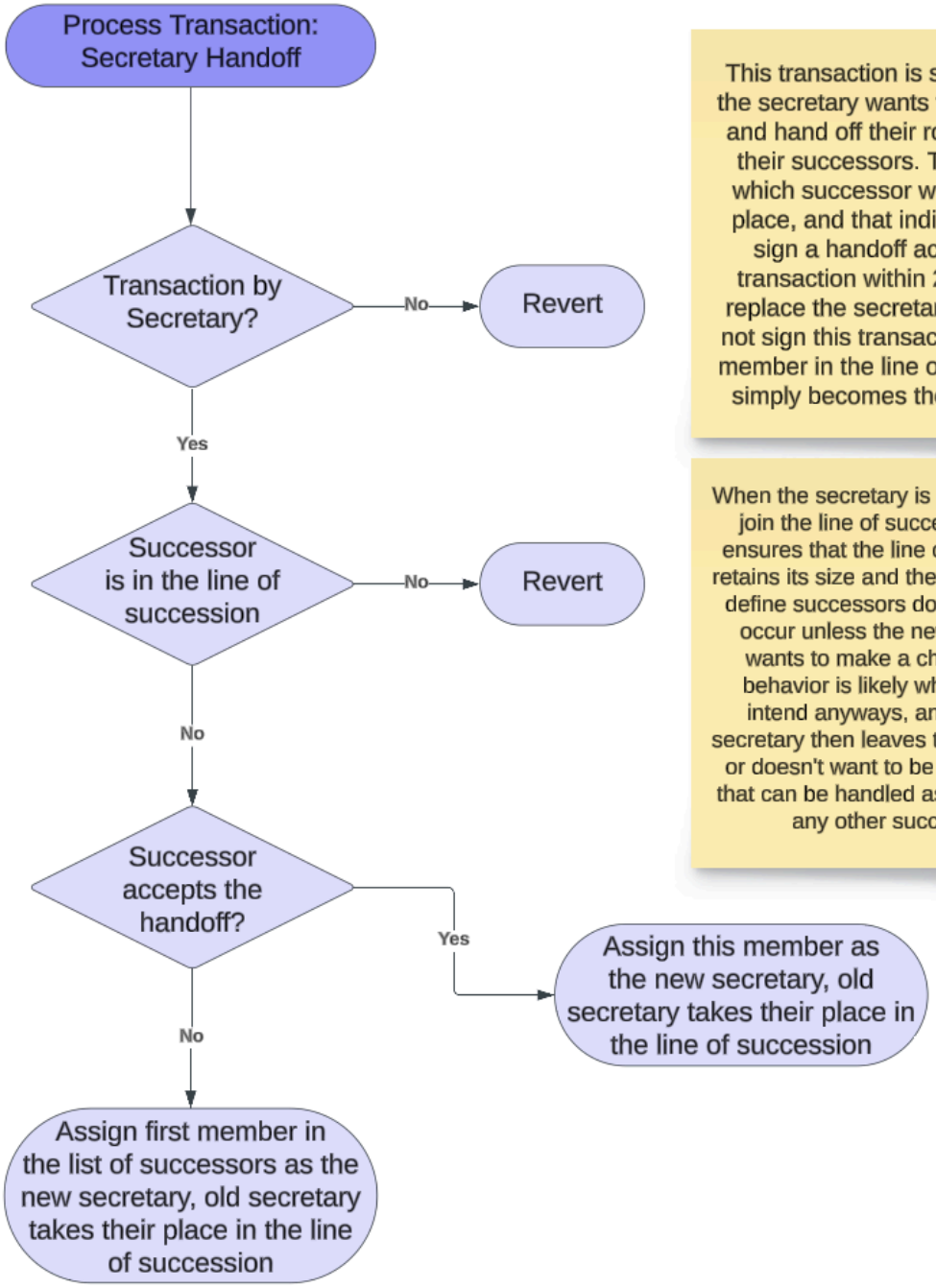
Caveats

This transaction has two main caveats which affect whether it will be reverted or not:

- The transaction must be made by the secretary
- The member the secretary defines to replace them must be in the line of succession

Logic

Provided that the caveats were satisfied, the smart contract will wait up to 24 hours to see if the member the secretary defined signs the handoff transaction. If they do not, a one-sided handoff occurs and the first member in the line of succession will simply replace the secretary. If the successor does sign the handoff acceptance transaction, the smooth-handoff occurs and they will replace the secretary.



This transaction is signed when the secretary wants to step down and hand off their role to one of their successors. They define which successor will take their place, and that individual must sign a handoff acceptance transaction within 24 hours to replace the secretary. If they do not sign this transaction, the first member in the line of succession simply becomes the secretary.

When the secretary is replaced, they join the line of succession. This ensures that the line of succession retains its size and the transaction to define successors doesn't need to occur unless the new secretary wants to make a change. This behavior is likely what they will intend anyways, and if the old secretary then leaves the community or doesn't want to be a successor, that can be handled as it would with any other successor.

Pseudocode is also provided:

```
FUNCTION process_transaction_secretary_handoff:
INPUT: sender, successor_list, successor
OUTPUT: status
      // transaction must be by secretary
      IF sender IS NOT secretary:
            RETURN "transaction reverted"

      // ensure the successor is in the successor list
      IF successor NOT IN successor_list:
            RETURN "transaction reverted"

      // If the successor the secretary provided accepts the
handoff,
      // then they will become the secretary. Otherwise, the member
      // who is at the top of the successors list
      IF listen("successor accepts handoff"):
            set_secretary(successor)
      ELSE:
            set_secretary(successor_list.pop())

      // add the old secretary to the successor list
      successor_list.push(sender)

      // set the line of succession to these successors,
      // in the order of which they were provided.
      set_successors(successor_list)
      RETURN "success"
```

Successor Accepts Handoff

Explanation

Whenever a TandaPay community's secretary chooses to resign and have one of their successors replace them, they will sign the secretary handoff transaction and specify which successor shall take their place. Whenever this happens, the successor they specified has 24 hours to sign this transaction to accept the handoff. If the successor signs this transaction and accepts the handoff, they will take the role of secretary. If they do not sign this transaction within the allotted time window, then the first successor in the line of succession becomes the new secretary instead.

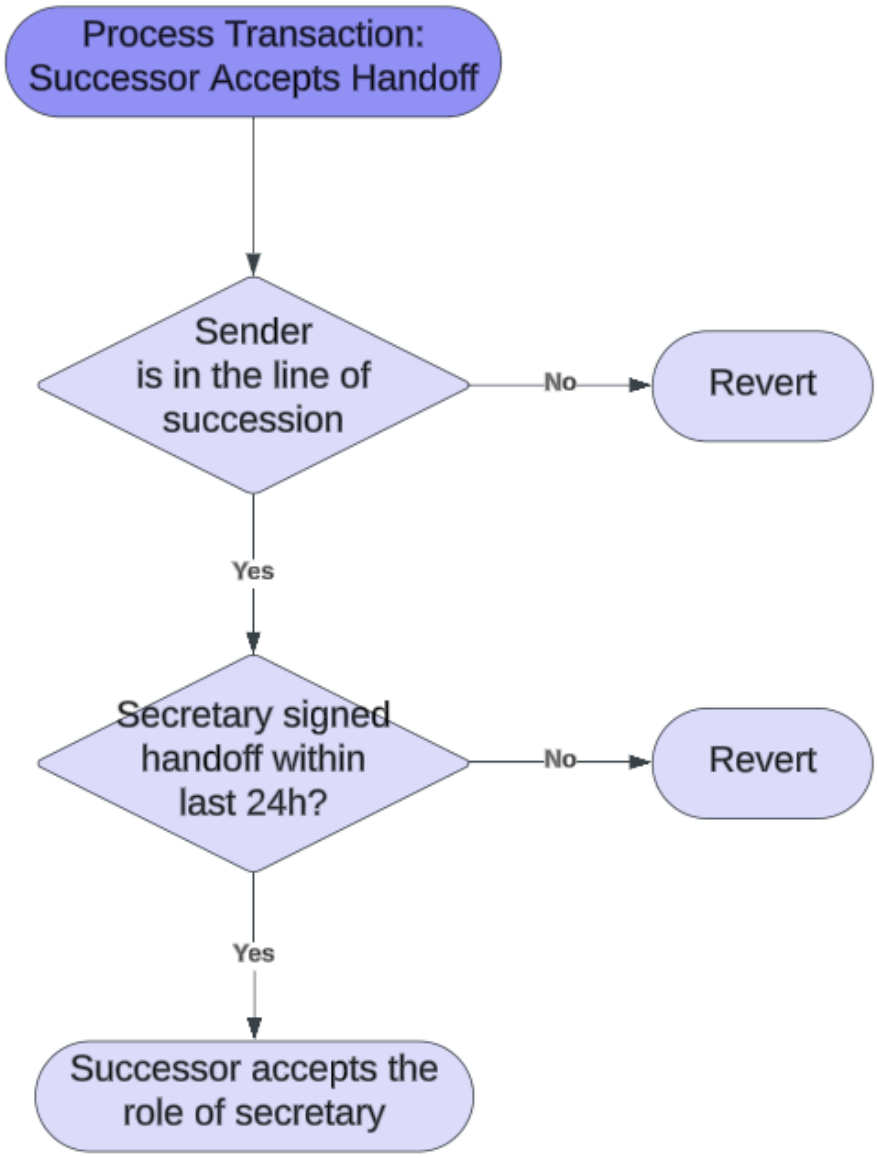
Caveats

There are two caveats when it comes to this transaction, and both of them are straightforward. These include:

- The sender must be in the line of succession to the secretary
- The secretary needs to have signed the handoff transaction within the last 24 hours

Logic

Assuming that the caveats are satisfied and the transaction is not reverted, the smooth handoff scenario will occur, with the sender of this transaction becoming the new secretary, and the old secretary joining the line of succession. This logic is mostly defined in the other transactions, the logic pertaining to this transaction is mostly just ensuring that the caveats are met, e.g. it makes sense for the sender to sign this transaction.



Pseudocode is not provided for this transaction since it is so straightforward.

Emergency Handoff Transaction

Explanation

When two members in the secretary's line of succession sign the emergency handoff transaction and specify the same successor to take the secretary's place, the old secretary is

removed and the successor takes their place as the new secretary. This transaction is meant to be utilized whenever the secretary is incapacitated (becomes unresponsive to the community or is unable to perform their duties for any reason).

The way it works is by having two of the secretary's successors sign the emergency handoff transaction, specifying a new secretary to take the old secretary's place. These transactions must be done within a short window, which would require coordination between the two successors. In order to avoid a situation where multiple pairs of successors sign this transaction and the secretary is replaced multiple times in rapid succession, this action can only go through once per period.

Caveats

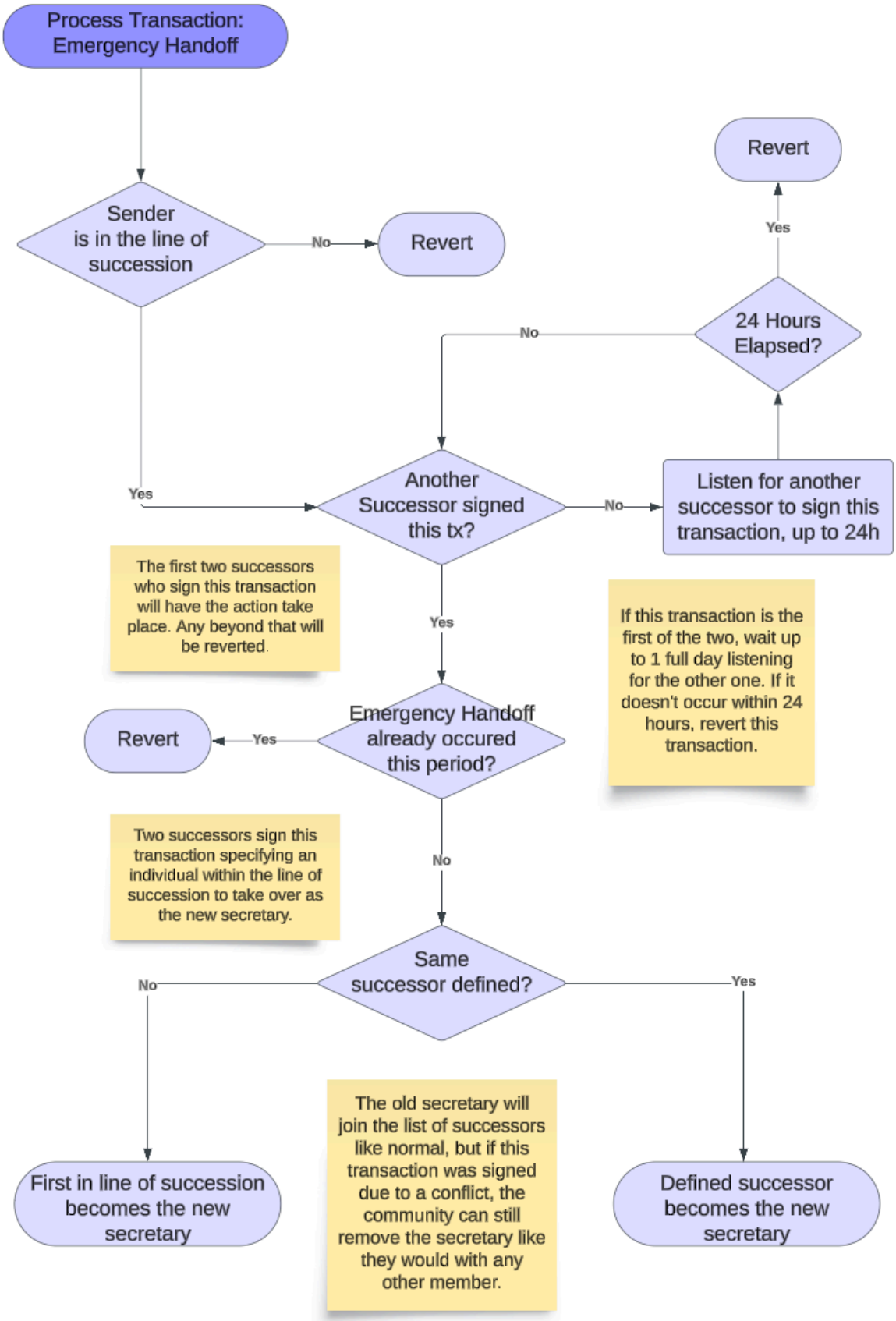
There are a few caveats with this transaction, in order to ensure that it makes sense and serves its purpose:

- It must be signed by an individual who is in the secretary's line of succession. By necessity, this also means that the individual would be a member of the community, as an individual who is not a member of the community would not be able to be in the secretary's line of succession.
- Two successors must both sign this transaction within a 24 hour period, specifying who they want to take over for the secretary. If there is a conflict and the two individuals who signed this transaction specify a different successor to take over, the first in line simply takes over.
- This emergency handoff action may only occur once per period. The first pair of successors who perform the action will result in the secretary being replaced, with following pairs of transactions doing nothing.

Logic

The caveats must first be satisfied, otherwise the transaction will be reverted. The individual who signed the transaction is checked to ensure they are in the line of succession. After that, if this is the first in the pair of transactions required to execute the emergency handoff action, the protocol will listen for the second transaction for up to 24 hours, reverting the transaction if a second one is not signed. If the emergency handoff already occurred within this period, then the transaction will be reverted because the action may only occur once per period. Finally, if the two successors who signed the emergency handoff transaction defined the same successor to take the place of the secretary, then this individual will replace the secretary, otherwise the individual at the top of the list to take over for the secretary will take over. Keep in mind that the individual designated to take over in the emergency handoff transaction must be in the line of succession; if one of the two individuals defines an invalid person to take over or they both do, then the successor at the top of the list shall take over.

This logic is outlined in the graphical flowchart representation below:


Process Transaction: Emergency Handoff
Sender is in the line of succession
No
Revert
Yes
Revert
24 Hours Elapsed?
No
Yes
Another Successor signed this tx?
No
Listen for another successor to sign this transaction, up to 24h
The first two successors who sign this transaction will have the action take place. Any beyond that will be reverted.
Yes
If this transaction is the first of the two, wait up to 1 full day listening for the other one. If it doesn't occur within 24 hours, revert this transaction.
Revert
Yes
Emergency Handoff already occured this period?
Two successors sign this transaction specifying an individual within the line of succession to take over as the new secretary.
No
Same successor defined?
No
Yes
First in line of succession becomes the new secretary
The old secretary will join the list of successors like normal, but if this transaction was signed due to a conflict, the community can still remove the secretary like they would with any other member.
Defined successor becomes the new secretary

Additionally, pseudocode is provided below:

```
FUNCTION process_transaction_emergency_handoff
INPUT: sender, successor_list, successor, cur_secretary
OUTPUT: status
      // transaction must be by a successor
      IF sender NOT IN successor_list:
            RETURN "transaction reverted"

      // check for the second transaction, and get the
      // successor that this individual defined
      other_tx_successor = NULL
      IF NOT tx_occurred("Emergency Handoff"):
            other_tx_successor = listen_24h("Emergency Handoff")
            // If the other transaction did not occur,
            // then revert this transaction.
            IF other_tx_successor IS NULL:
                  RETURN "transaction reverted"

      // ensure that the emergency handoff
      // occurred only once this period
      IF emergency_handoff_occurred:
            RETURN "transaction reverted"

      // If the successor defined by the two transactions does not
      // match, simply make the first in line the new secretary
      IF successor != other_tx_successor:
            set_secretary(successor_list.pop())
      // If the successor defined by the two transactions is not in
      // the successors list, first in line becomes new secretary
      ELSE IF successor NOT IN successor_list:
            set_secretary(successor_list.pop())
      // finally, if both transactions specified the same successor,
      // and this successor is in the successor list, they become
      // the new secretary.
      ELSE:
            set_secretary(successor)

      // add the old secretary to the successor list
      successor_list.push(cur_secretary)

      RETURN "success"
```

Secretary Injects Funds

Explanation

Whenever a shortfall occurs in the community escrow, and it cannot be made up through the usual rules for deducing from savings accounts, the secretary has two options. They can either inject their own funds to make up for the shortfall, or they can invoke a special transaction that divides the remaining shortfall up among all remaining savings account balances evenly. The inject funds transaction is the former option, essentially the secretary is simply adding their own personal funds to the community to keep it solvent and prevent a collapse.

This is a rare circumstance that normally only occurs when there are repeated claims and a serious fracture in consensus, and it does technically give the secretary the ability to prevent the community from officially collapsing; however, the idea is that by injecting funds into the community to keep it alive, the community and the secretary will lose credibility, as this transaction can be viewed by anyone on the blockchain.

Caveats

There are a few caveats in order for this transaction to be accepted:

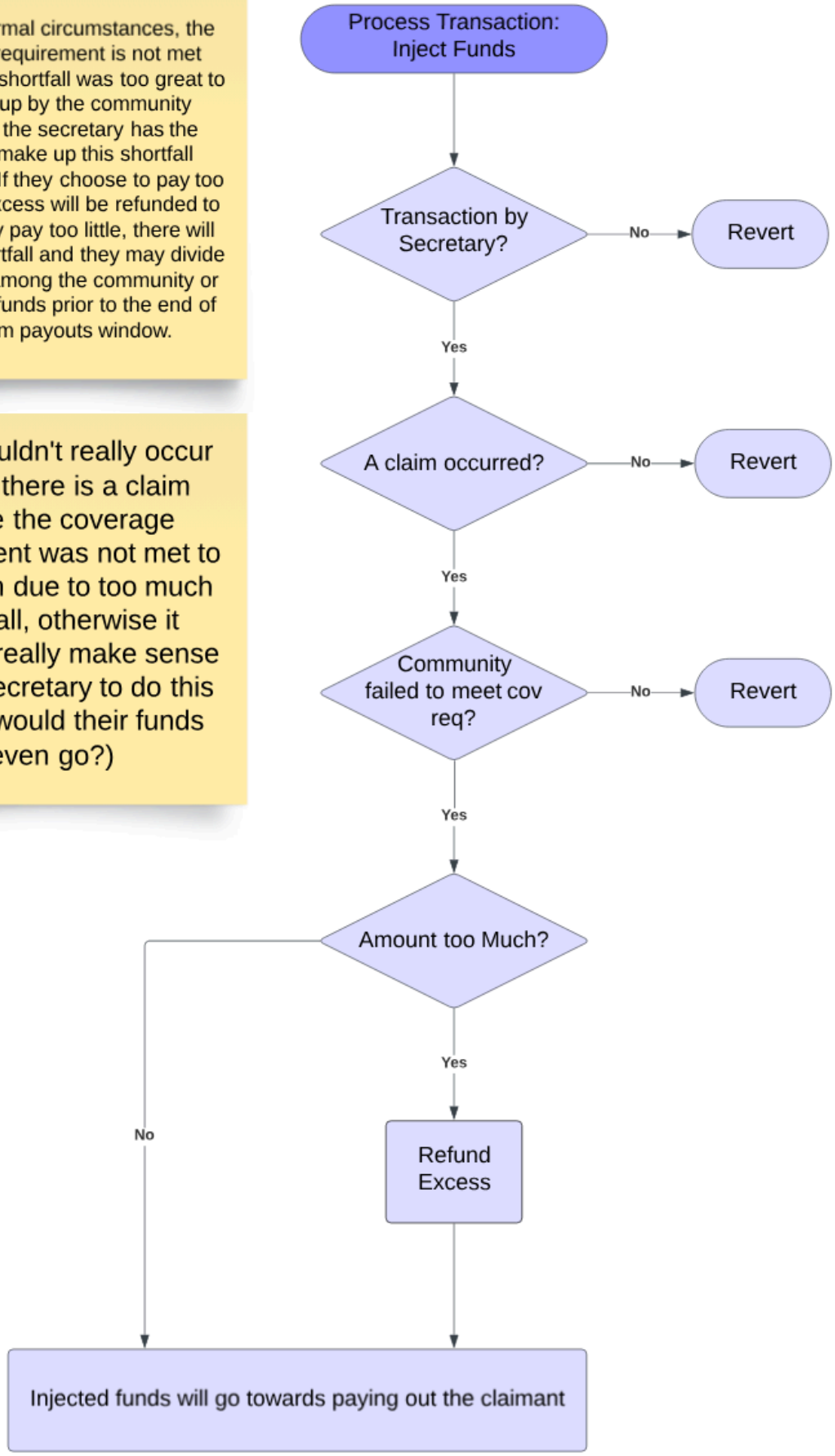
- This transaction may only be signed by the secretary
- This transaction must happen during the claims payment window, between days 1 and 3
- This transaction may only occur if a claim happened which needs to be paid out
- This transaction may only occur if the community failed to meet the coverage requirement via normal circumstances.

Logic

When a secretary injects funds, if any of the caveats are not satisfied, the transaction is reversed. Provided that all caveats were satisfied, any excess funds not required to make up the shortfall will be refunded to the secretary, and the proceeds will go towards paying out the claimant. This logic can be seen represented visually in the form of a flowchart diagram below:

If, under normal circumstances, the coverage requirement is not met because the shortfall was too great to be made up by the community members, the secretary has the option to make up this shortfall themselves. If they choose to pay too much, the excess will be refunded to them. If they pay too little, there will still be a shortfall and they may divide the rest up among the community or inject more funds prior to the end of the claim payouts window.

This shouldn't really occur unless there is a claim where the coverage requirement was not met to pay them due to too much shortfall, otherwise it wouldn't really make sense for the secretary to do this (where would their funds even go?)



Additionally, pseudocode is provided below for clarity:

```
FUNCTION process_transaction_inject_funds:
INPUT: sender, amount, claimant_list, community_escrow_balance,
coverage_requirement, current_day_in_period
OUTPUT: status
      // transaction must be by secretary
      IF sender IS NOT secretary:
            RETURN "transaction reverted"

      // a claim needs to have occurred
      IF claimant_list IS EMPTY:
            RETURN "transaction reverted"

      // the community needs to have failed to meet the coverage requirement
      IF community_escrow_balance < coverage_requirement:
            RETURN "transaction reverted"

      // must be during the claims payment window
      IF current_day_in_period > 3:
            RETURN "transaction reverted"

      // refund excess funds if there are any
      IF coverage_requirement < (community_escrow_balance + amount):
            issue_refund(sender, (community_escrow_balance + amount -
coverage_requirement))

      // use injected funds to meet coverage requirement and pay claimant
      add_to_claim_payout(amount)
      RETURN "success"
```

Secretary Divides Shortfall

Explanation

Whenever a shortfall occurs in the community escrow, and it cannot be made up through the usual rules for deducing from savings accounts, the secretary may also divide this shortfall up among the remaining valid members' savings accounts evenly. The secretary may combine this with injecting funds, injecting an amount that isn't sufficient to make up for the shortfall, but then dividing the remainder up among the valid members' savings accounts. This may cause members to leave, as this transaction must be made within the claims processing window, and if an individual chooses to leave prior to the end of this window, they will receive a full refund like normal rather than having their savings accounts debited.

Caveats

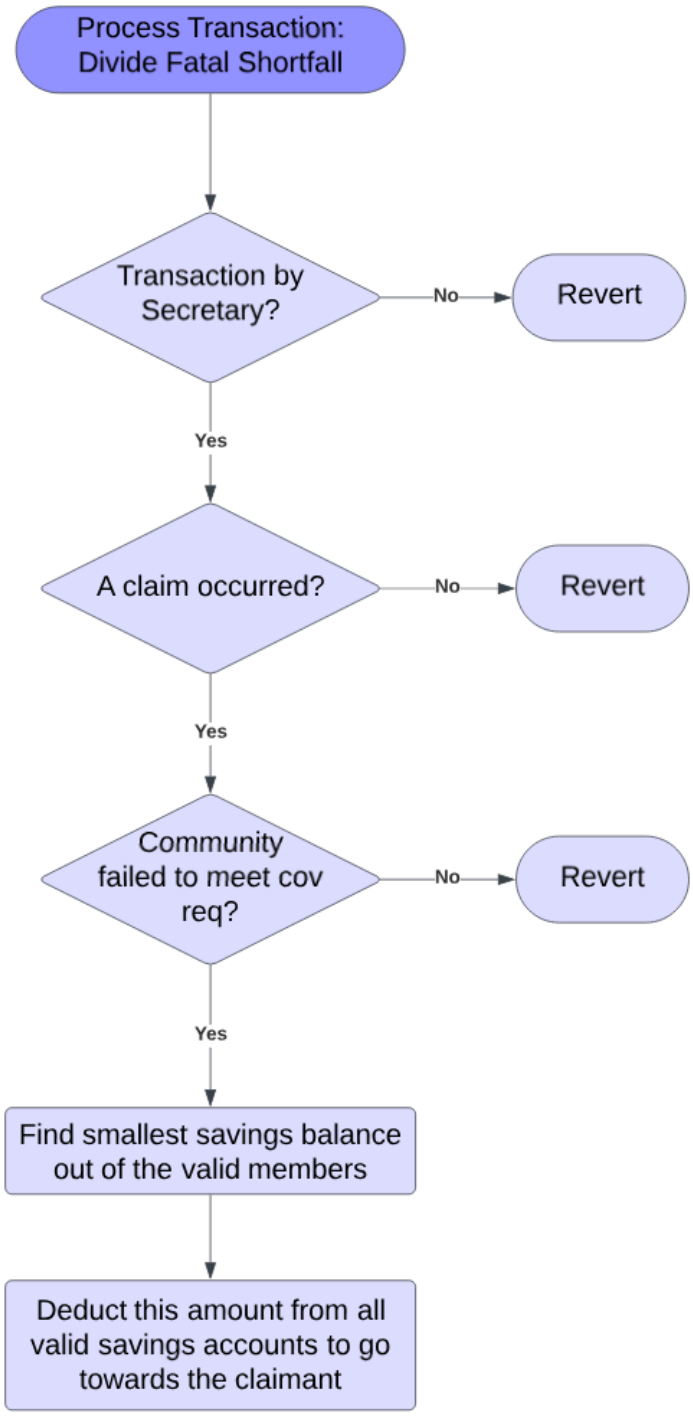
The caveats for this transaction are similar to the inject funds transaction:

- This transaction may only be signed by the secretary
- This transaction must happen during the claims payment window, between days 1 and 3
- This transaction may only occur if a claim happened which needs to be paid out

- This transaction may only occur if the community failed to meet the coverage requirement via normal circumstances.

Logic

When the secretary chooses to divide the remaining shortfall up among the valid members' savings accounts, the savings account with the lowest balance is the limit for how much can be deducted from each account, because the shortfall must be divided evenly among the remaining members. If members choose to defect or the shortfall is too large for the savings accounts to cover equitably, then this transaction may not necessarily resolve the shortfall, in which case the secretary would have to inject their personal funds to prevent the group from collapsing. This logic is shown below in the form of a flowchart diagram:



This is another option that the secretary has in the event that the coverage requirement is not met. The remaining valid members have already authorized their funds to go towards the claimant, which is why this doesn't violate the principle of self custody

The secretary may EVENLY divide the shortfall among all remaining savings account balances of valid members. This means the smallest savings account will be the limiting factor, as they all have to be deducted the same amount

If the amount deducted from the savings accounts are still not enough to meet the coverage requirement, the secretary will have to make up the rest. If the secretary does not make up the rest, then the community will collapse and these users will simply receive their funds back, and the claimant won't be paid

Additional pseudocode is provided for clarity:

```
FUNCTION process_transaction_divide_shortfall:
INPUT: sender, claimant_list, community_escrow_balance, coverage_requirement,
current_day_in_period
OUTPUT: status
      // transaction must be by secretary
      IF sender IS NOT secretary:
            RETURN "transaction reverted"

      // a claim needs to have occurred
      IF claimant_list IS EMPTY:
            RETURN "transaction reverted"

      // the community needs to have failed to meet the coverage requirement
      IF community_escrow_balance < coverage_requirement:
            RETURN "transaction reverted"

      // must be during the claims payment window
      IF current_day_in_period > 3:
            RETURN "transaction reverted"

      // Get lowest savings account balance, and count
      // number of valid members
      min_s = INF
      count = 0
      FOR member IN member_list:
            IF member.status == "VALID" AND member.savings_balance < min_s:
                  min_s = member.savings_balance
                  count += 1

      // calculate shortfall
      shortfall = coverage_requirement - community_escrow_balance

      // determine the minimum amount to be deducted from savings accounts
      // to make up for the shortfall
      amt_to_deduct = min(min_s, shortfall / count)

      // deduct it from remaining valid members' savings at the end
      // of the claims processing window.
      deduct_from_valid_savings(amt_to_deduct)

      // use injected funds to meet coverage requirement and pay claimant
      add_to_claim_payout(amount)
      RETURN "success"
```

Secretary Adds 31st Day

Explanation

One unique transaction within the TandaPay protocol is the ability of the secretary to add an additional day to the period. This will result in a 31st day being added, or alternatively a 24 hour delay at the end of the period before the next period begins. Nothing occurs on the 31st day, and this is simply a mechanism to assist the secretary with keeping the premiums payment window aligned with the beginning/end of the calendar months if they choose to do so.

Caveats

The only caveat for this transaction is that it must be signed by the secretary. Otherwise, it doesn't affect the function of the TandaPay community and is completely straightforward

Logic

Rather than advancing the period immediately at the end, an additional 24 hour delay will occur before the period advances. The logic is very straightforward, and thus a graph or pseudocode would be redundant.

Secretary Triggers Manual Collapse

Explanation

The secretary has the ability to shift the community into a manual collapse. This is an easy way to terminate a TandaPay community for a variety of reasons, including the community no longer serving its purpose, or the secretary simply wants to form a new community perhaps using a later version of this smart contract. It has a delay built in so that this feature cannot be abused as easily by the secretary, and to ensure that it doesn't violate any of the principles of the community.

Caveats

The caveats with this transaction include:

- The secretary must be the one to sign this transaction
- If this transaction is signed, and then the transaction to cancel a manual collapse is signed, this transaction may not be signed again until after the collapse would have occurred (they still must wait out the full delay)

Logic

The actual collapse, if not canceled, will occur at the end of the claims payout window. This is when TandaPay communities collapse under normal circumstances, and so this transaction will remain consistent with that logic. This also ensures that any claims are paid out and that the community is not manually collapsed prior to that. The TandaPay protocol is best designed to

have collapses occur at this specific time. If this transaction is canceled, it cannot be signed again (will be reverted) until after the collapse would have occurred.

Secretary Cancels Manual Collapse

Explanation

The secretary has the ability to cancel a manual collapse signal if it has occurred, but the community has not collapsed yet. This transaction can be signed any time right up until the community is shifted into the collapsed state, and will stop the collapse from happening.

Caveats

The caveats with this transaction include:

- It must be signed by the secretary
- The trigger manual collapse transaction must have been signed

Logic

This transaction is relatively straightforward, it simply cancels the trigger manual collapse transaction. It is signed by the secretary in the event that they either change their mind about terminating the community, or if the secretary is replaced before the community collapses and the new secretary does not wish to have the community manually collapsed.

Time Based Logic

Time is important to the TandaPay smart contract because some actions occur automatically depending on how much time has elapsed. The entire system is designed around payment periods, which encompass a certain length of time, and these payment periods are subdivided further into time windows in which certain actions occur.

Broadly speaking, a TandaPay payment period lasts 30 days, and contains three time windows in which significant actions are always performed. These include:

1. **Claims Payment Window**: Lasts for the first three days of any given period. If a claim occurred, individuals have the opportunity to defect rather than have their funds be used to pay the claimant, and at the end of this time window, claimants are paid out. This is the only time that a TandaPay community may collapse under normal circumstances, as the community shifts into the collapsed state if there are insufficient funds to pay the claimant at the end of this window.
2. **Previous Period Refunds Window**: Refunds for the previous period are issued on the fourth day of any given period. If no claims occurred during the previous period, then members will be refunded their contributions to the community escrows. Other refunds may also occur, such as to members who became paid-invalid for this current period.

3. **Premiums Payment Window**: Occurs within the final three days of any given period. This is when premiums are collected for the next period, and stored in that period's escrow. Shortfalls may occur during this period when individuals do not pay their premiums by the end of it, which are deducted from remaining valid members' savings accounts. Otherwise, the premiums are calculated and collected during this period and should be sufficient to fill the community escrow and refill all member's savings accounts that were deducted in the previous period.

At the end of a TandaPay payment period, we advance to the next period, and this whole process repeats. These are the three primary windows in which automatic actions happen within the TandaPay smart contract, although there are other miscellaneous time-based actions that occur under special circumstances, such as those related to transactions signed by members.

Claims Payment Window

At the beginning of any given TandaPay period where a claim occurred in the preceding period, claim payments are processed. The claims payment window encompasses the first three days of the period, with claimants being paid out at the end. This time window is when defections may occur, and is also when a TandaPay community may collapse under normal circumstances.

During the three day window, members who believe the claim violated the community's charter may opt to sign the defection transaction, which gives them a refund of all money they are entitled to from the system during the next refunds window, and they will exit the community after that. This results in a deficit in the community escrow that is paying out the claim, which is divided evenly among the defector's remaining valid *subgroup members'* savings accounts.

Shortfalls may also arise from individuals not paying their premiums or from individuals becoming paid-invalid, and these shortfalls can be settled by deducting funds from the community's remaining valid members' savings accounts evenly.

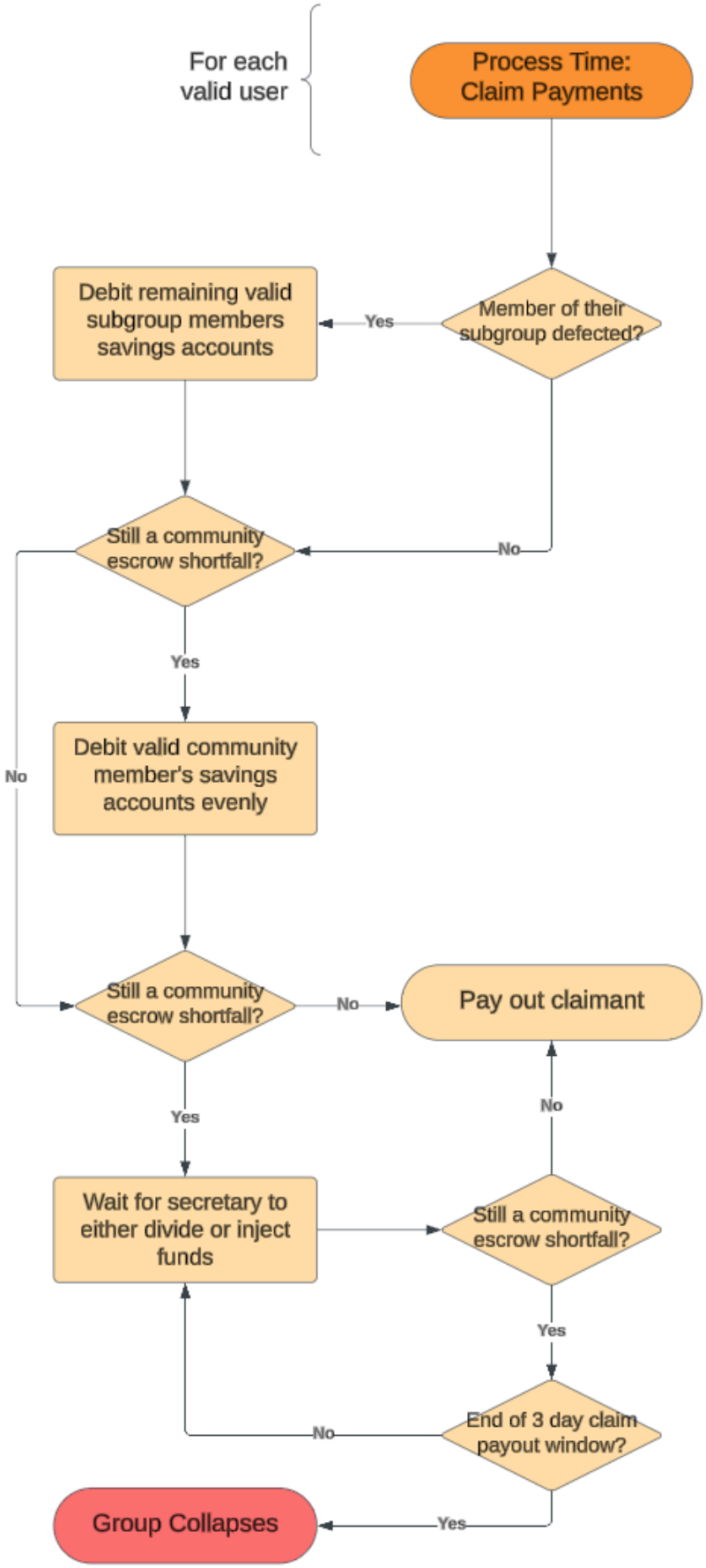
During this three day window, if the secretary believes that the coverage requirement will not be met resulting in the group to collapse, they may either inject funds personally or divide this impending shortfall up evenly among the remaining valid members of the community. These outcomes are covered more in depth in the transactions logic section of this specification. This specifically might occur when there are repeated claims or a high number of individuals leaving the community, resulting in savings account balances not being adequate to cover shortfalls.

When there are multiple claimants in a single period, the coverage requirement is divided evenly among the claimants. If this coverage requirement is not met by the end of the claims payment window, then the TandaPay community will enter the collapsed state.

This logic is also outlined in the following diagram:

When members of a subgroup defect, the shortfall caused by this is made up by all of the remaining members, being evenly divided among their savings accounts

A community escrow shortfall refers to the community escrow not having enough money due to members being paid-invalid or unpaid-invalid, and to make up for this, every valid member in the community has their savings account debited evenly.



Additionally, pseudocode is provided below:

```
FUNCTION process_time_claim_payments
INPUT: claimant_list, member_list, coverage_requirement, community_escrow
OUTPUT: status

      // First, for each member who had defectors in their subgroup, deduct
      // the appropriate shortfall from the remaining valid members' savings
      // accounts to make up for the defectors in their subgroup
      FOR member IN member_list IF member IS valid:
            shortfall = member.subgroup.defector_shortfall /
member.subgroup.valid_remaining
            member.debit_from_savings(shortfall)

      // Next, if there is still a shortfall (e.g. due to members not paying
      // their premium), deduct this evenly among all the valid members
      IF community_escrow < coverage_requirement:
            shortfall = coverage_requirement - community_escrow
            FOR member IN member_list IF member IS valid:
                  member.debit_from_savings(shortfall / member_list.valid_cnt)

      // Finally, while we are still in the claims payout window, continuously
      // check to see if the community escrow meets the coverage requirement.
      // When it does, divide it among the claimants. This is to allow for time
      // for the secretary to inject funds or divide the shortfall
      WHILE day IN [1, 3]:
            IF community_escrow == coverage_requirement:
                  claim_payout = community_escrow / claimant_list.length
                  FOR claimant IN claimant_list:
                        claimant.payout(claim_payout)

                  RETURN "success"


      // Finally, if the claimants were not paid out by the end of day 3,
      // the TandaPay community collapses
      set_state("COLLAPSED")
      RETURN "Community collapsed"
```

Previous Period Refunds Window

The refunds window occurs on the fourth day of any given period. On this day, refunds are issued to members of the TandaPay community who contributed to the previous period's community escrow if those funds were not used to pay out a claimant.

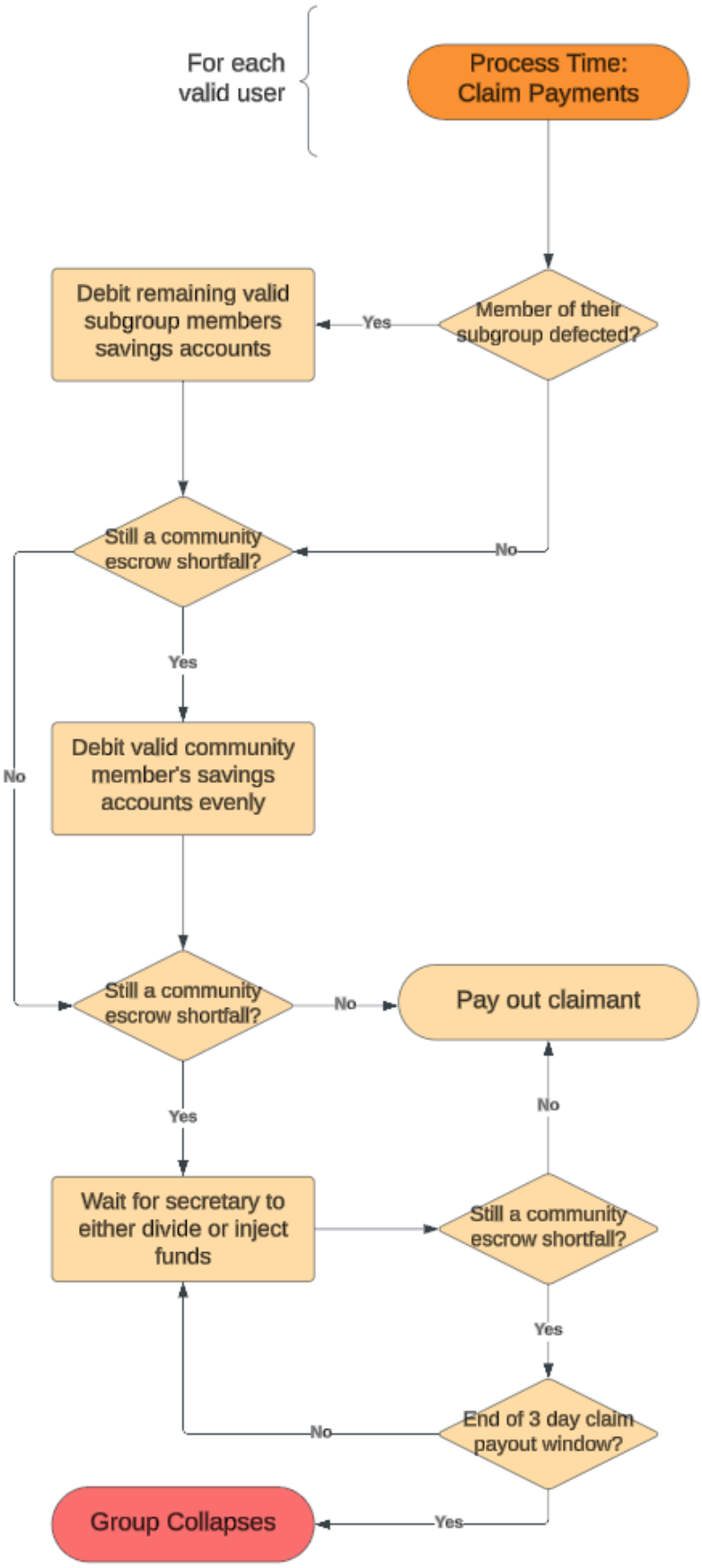
Normally, a member would pay their premium for the upcoming period, then that period would elapse with no claims occurring, and then in the following period during the refunds window, they would be reimbursed their contribution to the community escrow. However, refunds are also issued to members who became paid-invalid on this day, meaning they will be reimbursed their contribution for the current period rather than the previous one.

After this window has elapsed, the previous period's community escrow will be empty, whether it was because the funds were paid out to a claimant in the previous days or issued as a refund during this window. At that point, the previous community escrow ceases to exist and the current period's community escrow becomes the current community escrow in use, for paying out any claims that are whitelisted this period.

This logic is outlined in the flowchart diagram below:

When members of a subgroup defect, the shortfall caused by this is made up by all of the remaining members, being evenly divided among their savings accounts

A community escrow shortfall refers to the community escrow not having enough money due to members being paid-invalid or unpaid-invalid, and to make up for this, every valid member in the community has their savings account debited evenly.



```
FUNCTION process_time_refunds
INPUT: member_list, claim_occurred, last_base_premium, current_base_premium, state
OUTPUT: status

      // Iterate through each member to determine their refunds
      FOR member IN member_list:
            // If they didn't pay their last premium and this is the second period
            // in a row where they are unpaid-invalid, refund all funds because
            // they will be kicked from the community
            IF NOT member.paid_last_premium:
                  IF member IS unpaid_invalid:
                        refund(member, member.all_funds)
                        CONTINUE

            // If a claim occurred and a member is paid invalid, they'll still
            // receive their most recent premiums payment (the one they made
            // just days ago) as a refund.
            ELSE IF claim_occurred:
                  IF member IS paid_invalid:
                        refund(member, current_base_premium)
                        CONTINUE

            // If we're in the fractured state, refunds are delayed by queueing,
            // which means we'll issue the next refund in the queue (could be 0)
            IF state == "FRACTURED":
                  refund(member, queued_refunds.pop())
                  CONTINUE

            // Finally, if all of those checks failed and no claim occurred,
            // simply refund their premium
            IF NOT claim_occurred:
                  refund(member, last_base_premium)


      RETURN "Success"
```

Premiums Payment Window

In the last three days of any given period, premiums are collected to fill members' savings accounts as well as the community escrow for the upcoming period. Generally, premiums are calculated at the beginning of this window by dividing the coverage requirement among all of the members in the community, and adding the amount necessary to replenish their savings accounts to full, with the only exceptions being during the initialization state or for new members first joining the community.

Calculating Savings Account Requirements

The savings account requirement is set each time the *coverage requirement* is set by the secretary. This must happen before the community collects its first premiums and exits the

initialization state, but may also happen later on as the community evolves if the secretary signs the transaction to set the coverage requirement.

The savings account requirement is defined as 120% of the coverage requirement, divided by the number of members at the time the coverage requirement is set. It does not change dynamically as individuals exit the community, meaning over time, as people leave, the sum of all savings balances may no longer equal 120% of the coverage requirement.

For instance, if the coverage requirement is set to $10,000 during a period in which the community has 100 members, this would mean that each member would be required to pay a premium of $100 to contribute to the community escrow, and their savings account requirement would be $120.

Each period, members must pay the amount required to contribute to the community escrow, as well as the amount required to completely fill their savings account up to the savings account requirement, with one exception being new members joining. In the initialization state, every member is new, meaning no claims can occur and everyone will abide by the new member logic.

New members must pay a premium that is equivalent to 11/12ths of their savings account requirement. In the scenario outlined earlier, that would be $110. They will not receive coverage for the first period they are in the community, because they didn't contribute to the community escrow, but only contributed to their savings accounts. The reason this is done is to avoid new members having to pay a large upfront premium to join the community. The following period, they will have to abide by the normal rules, paying a premium to contribute to the community escrow and to fill the last 1/12th of their savings account.

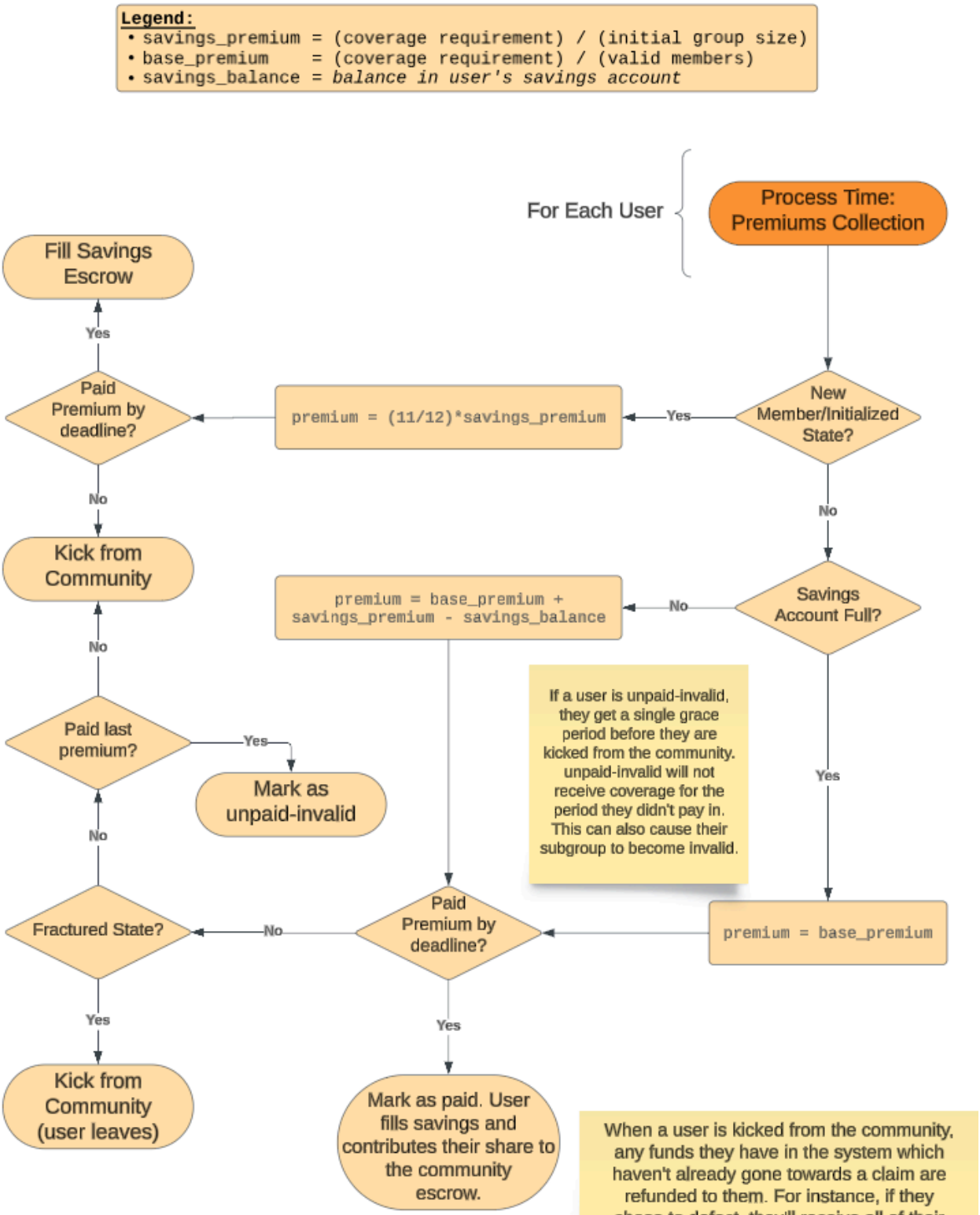
Calculating Base Premiums

Base premiums are simply calculated as the coverage requirement divided by the number of members in the community. This includes any members who might have coverage during the period which the premiums are being collected for, e.g. individuals who are valid, unpaid-invalid, or who reorged. Base premiums go towards the community escrow and are what allow the member to receive coverage, a member who does not contribute to the community escrow for any given period may not receive coverage or file a claim.

Total Premiums Due

Other than the circumstance outlined for new members, the premium due for any given member is calculated as the sum of the base premium and the amount that is required to bring their savings account balance up to the savings account requirement.

This logic is outlined below with a graphical flowchart representation:


Legend:
• savings_premium = (coverage requirement) / (initial group size)
• base_premium = (coverage requirement) / (valid members)
• savings_balance = balance in user's savings account
For Each User
Process Time: Premiums Collection
New Member/Initialized State?
Yes
premium = (11/12)*savings_premium
Paid Premium by deadline?
Yes
Fill Savings Escrow
No
Kick from Community
No
Savings Account Full?
No
premium = base_premium + savings_premium - savings_balance
If a user is unpaid-invalid, they get a single grace period before they are kicked from the community. unpaid-invalid will not receive coverage for the period they didn't pay in. This can also cause their subgroup to become invalid.
Yes
premium = base_premium
Paid Premium by deadline?
No
Paid last premium?
Yes
Mark as unpaid-invalid
No
Fractured State?
No
Yes
Kick from Community (user leaves)
Yes
Mark as paid. User fills savings and contributes their share to the community escrow.
When a user is kicked from the community, any funds they have in the system which haven't already gone towards a claim are refunded to them. For instance, if they chose to defect, they'll receive all of their funds back. If they just stopped paying, funds they paid in will go towards claims if they happened, and anything leftover will be refunded to them once they are kicked.

```
FUNCTION process_time_premium_payments
INPUT: member_list, coverage_requirement, savings_requirement, state
OUTPUT: status

      base_premium = coverage_requirement / member_list.num_valid

      // Iterate through each member to determine their premiums
      FOR member IN member_list:
            // If the member is new or we're in the initial state (where all
            // members would be new), their first premium due will be 11/12 of
            // the savings requirement. If they don't pay it, they are kicked
            // and it is treated as though they never joined
            IF (member IS new) OR (state == "INITIAL"):
                  member.premium_due = savings_requirement * (11/12)
                  IF NOT paid_by_deadline(member):
                        kick_from_community(member)

            // otherwise, the premium due will be the base premium, plus any
            // shortfall in their savings account that needs to be supplemented
            premium = base_premium
            IF member.savings_balance < savings_requirement:
                  premium += (savings_requirement - member.savings_balance)

            member.premium_due = premium

            // If they do not pay  this by the deadline, they will be kicked in the
            // fractured state (they leave), otherwise, if they paid their last
            // premium, they will become unpaid-invalid. If they didn't pay two
            // premiums in a row, they will be kicked regardless of the state
            IF NOT paid_by_deadline(member):
                  IF state == "Fractured":
                        kick_from_community(member)
                  ELSE:
                        IF member.paid_last_premium:
                              member.status = "UNPAID_INVALID"
                        ELSE:
                              kick_from_community(member)


      RETURN "Success"
```

Advancing the Period

Finally, after a full period has elapsed, it is time to advance to the next period. Other miscellaneous time-based logic may be completed as a part of this process. This includes changing the state of the community based on what has occurred in the previous period, dealing with subgroups that have an invalid size, or handling the day offset that the secretary may issue to ensure that the payment dates don't become too unaligned with the month.

Updating the State

Depending on what occurred in the previous period, the state may be updated when the period is advanced. Here are the conditions by which the state may be changed:

- If the previous state was the initialization state, update to the default state
- If the previous state was the default state, and more than 12% of the community defected, update to the fractured state
- If the previous state was the fractured state, and we have experienced three consecutive periods of stability, update to the default state
- The TandaPay community shifts into the collapsed state during the claims payment window, and cannot recover from this state.

Handling Invalid Subgroups

If a subgroup size is too large, the additional members will become paid-invalid or unpaid-invalid, depending on whether or not they paid their premiums, and will not receive coverage for the following period. The secretary will need to reorganize these users into new subgroups before they can receive coverage. If a subgroup size is too small, the additional members will receive a refund and will be kicked from the community, counted as individuals who left.

Handling the Day Offset

The secretary may sign the transaction to add an additional day to the length of any given TandaPay period. This will result in a 31st day, and is primarily used to keep the payment window aligned with the month change. It still may become off balance from the month, but this ability of the secretary can be used to add an additional delay when they see fit. On the 31st day, nothing happens, it simply delays the start of the next period by an extra day.

Q&A

Changes to subgroup dynamics

When members leave a subgroup, they will be treated just like paid-invalid members. They will lose coverage for a full period before they are assigned to their new subgroup. So if you have a subgroup of 7 and this subgroup has 3 members who want to break off and create a new subgroup with a new member who is joining, these 3 members would essentially be treated as paid-invalid and lose coverage for a period, and then after that the 3 + the new 1 would reorg into a subgroup of 4. The 4 who remained in the original subgroup would remain valid.

Additionally, a requirement for the TandaPay community to exit the initialized state will be that there are at least *3 subgroups*, not just 12 members.

This means it will be more difficult for new members to join. People will be unwilling to go paid-invalid for a full period. This will cause strife within the community because people's premiums will go up due to the members who are reorganizing and becoming paid-invalid. This is intentional, and it is the goal of the community to kill groups who aren't willing to make sacrifices and cooperate with eachother.

If the community can't figure out how to add a new member to a subgroup (e.g. subgroups don't want to add a new member, people are unwilling to split off and create a new subgroup for them, etc.) then the new member won't be able to join (they will be kicked). In that situation, the ideal way for the new member to join would be to come with 3 other members so they can immediately form their own, separate, brand-new subgroup upon joining.

To transition from the initial state to the default state, here's what needs to occur:

1. At least 12 members need to be added to the community and organized into subgroups by the secretary. When a new member first joins the community, they must pay to fill up their savings account to 11/12ths (as you have seen before). **The members must be divided into at least 3 subgroups**.

2. In order to actually transition the state from initialized to default, originally my thought was that after all of the conditions were met, 30 days would elapse before the first actual premiums payment would occur. Keep in mind, the new members should fill up their savings accounts to 11/12ths of this amount before they can be true members of the community, this is a prerequisite for them to join. So by waiting 30 days until after the last individual of the 12 join, we can ensure that they have some time to get the funds together, rather than having to pay a large lump sum up front or two payments back to back.

Now, alternatively, we could **have a transaction that allows the secretary to shift from the initialized to default state manually**. This might be more simple to implement, and would allow the community to make a decision for themselves when they want to shift. That way, if the secretary meets the conditions but still needs more time to add additional members, they can wait before shifting into the default state manually. As I think of this now, this is probably the better approach, so I'm going to recommend adding a transaction so that the secretary may do this manually, so long as the conditions are all met.

Is it allowed for a new member to be joined in the premium pay window? Because coverage amount is divided into the number of members.

Yes, earlier I mentioned transitioning from the initial state to the default state that there would be a full period wait. All new members could be treated this way. The way I would implement this is by having them fill up their savings account to 11/12ths immediately upon joining, but not treating them as active valid members of the community (and this not effecting the premiums right away). If newly added members wait one full period, this allows for time for the secretary to figure out their subgroup situation and prevents them from having to fill up their savings accounts to 11/12ths then immediately after pay another premium to receive coverage.