Rotation logic

Rotation is the process of periodically replacing the contact's email used for mailing with other emails associated with the contact in order to determine which emails are currently valid.

Rotation takes place between two tables: emails_status_active (which stores the primary emails – the active pool) and emails_status (which stores backup candidate emails – the semi-active pool).

Triggers and Specific Conditions of the Rotation Algorithm

1.1. Trigger rotation every night if at least one of the following conditions is met:

a). When the conditions for email removal from the active pool (outlined in Section 2) are met and at least one replacement candidate is found (as per Section 3).

b). When a contact does not have any email in the active pool and at least one replacement candidate is found (as per Section 3).

Additionally , Solr indexing, which is also performed every night, should be executed after the rotation process.

1.2. Trigger unscheduled rotation upon the occurrence of any of the following events:

a). When an email in the active pool is retired and at least one replacement candidate is found (as per Section 3). In this case, the reason for removal from the active pool is retirement of the email, so additional active pool removal conditions from Section 2 do not need to be checked.

b). When a new email is added to the database via DPO upload according to DK-1484, clause 4.2 a22.

- If the contact does not have an email in the active pool, the new email goes directly into the active pool.
- If the contact already has an email in the active pool, the new email replaces the existing one via rotation, without checking the rules described in Sections 2 and 3.

Rationale: An email collected directly from the contact via DPO call is considered more reliable than emails currently in our database.

→ Note: Clarification needed from Vitalik — What exactly happens during a new DPO upload if the contact already exists in the DB and has an email in the active pool, and a different email comes via DPO upload? How is this new email added to the active pool — is it an overwrite? Our intended behavior is: the new email should go into the active pool, and the old email should be moved to the semi-active pool.

1.3. In the following cases, when a new email is added to the database, do NOT perform rotation, but instead place the emails in the semi-active pool (i.e., they become rotation candidates):

a). Upload source (importing module)
b). Create-list (importing module)

1.4. In cases when a new email appears in the database (**1.2b, 1.3a, 1.3b**), its "active pool insert counter" must not be set to 0. Otherwise, it will be constantly rotated into the active pool until it catches up with the other emails in terms of insert count.

Instead, set its insert counter to the minimum of the existing "active pool insert counter" values among all of the contact’s current emails that are in either the active or semi-active pools.

Note: For case **1.3b** , this should be done not at the moment the email appears (as it's placed into the inactive pool), but later — at the moment the email is activated (i.e., moved into the active or semi-active pool).

1.4.1. Also, assign default counter values to the new email:

- Counter_NoEngagements = 0
- Counter_FreezedUp = 0

Conditions for Email Disqualification from the Active Pool (analysis of emails from app_system.emails_status_active)

TERMS:

Introducing a single counter — Counter_NoEngagements , which will be used simultaneously for two cases:

- Case in Section 2.3 (**NOT** matching a CONFIRMED PATTERN)
- Case in Section 2.4 (matching a CONFIRMED PATTERN)

The counter should count the number of mailings sent to the email address (**excluding** DPO TYE (Thank You Email) and NC (Nurturing Campaign)) that result in no engagements of types **P1** or **P2** , starting from the moment the email was placed into the active pool.

Introducing a second counter — Counter_FreezedUp , used for Section 2.5, to count the number of mailings sent to the email address (**again, excluding** DPO TYE and NC) without **P1** or **P2** engagements, starting from the timestamp of the last engagement of type P1 or P2 after being added to the active pool.

Note: Both Counter_NoEngagements and Counter_FreezedUp must exclude mailings of type DPO TYE and NC.

Introduce rule-2.1:
Upon receiving a P1 or P2 engagement:

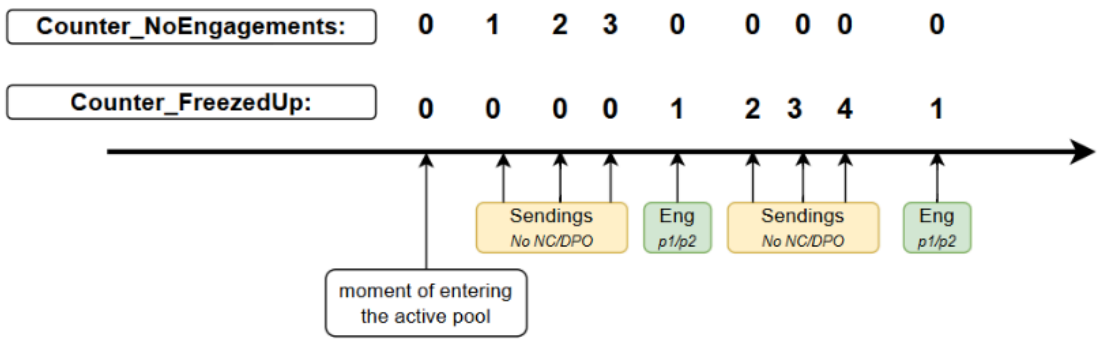
- Set Counter_NoEngagements = 0
- Set Counter_FreezedUp = 1

Introduce rule-2.2:
At the moment an email is disqualified from the active pool:

- Reset Counter_NoEngagements = 0
- Reset Counter_FreezedUp = 0

Introduce rule-2.3:
If an email has Counter_FreezedUp > 0 while it is in the active pool:

- For subsequent mailings without engagements, do not increase Counter_NoEngagements (i.e., keep it at 0) — only increment Counter_FreezedUp (Counter_FreezedUp = Counter_FreezedUp + 1)
- For mailings that result in a P1 or P2 engagement, apply rule-2.1



Introduce rule-2.4:
Emails that are part of special lists marked as **isDPO** or **isNC** must remain associated with those lists regardless of any rotation events.
This means mailings of type **DPO TYE** and **NC** must continue to be sent to the same emails assigned to those **isDPO** and **isNC** lists, regardless of whether those emails are rotated out of the active pool.

This likely does not require implementation-level changes but needs to be verified to ensure existing behavior aligns with this rule.

DATA PROCESSING:
Introducing the disqualification conditions for an email to be removed from the active pool:

2.1. OR The email is considered **not valid** ,
(regardless of whether it was invalid before entering the active pool or became invalid afterward — such emails should not be added to the active pool from the semi-active pool anyway under other rotation rules).

Validation check based on values in the following table:

- app_system.emails_status_active.valid NOT IN (**1, 2, 3, 7, 8**)

2.2. OR The email is considered **dangerous** ,
(similarly, regardless of the timing — before or after adding to the active pool — such emails should not reach the active pool under rotation logic).

Danger status check is:

- app_system.emails_status_active.dangerous > 0

2.3. OR The following condition is met:

- The email **does NOT** match a CONFIRMED PATTERN,
- AND Counter_NoEngagements ≥ (int)(k[i] / 10.0),
- AND Counter_FreezedUp == 0,
- AND (CurrentDate – LastSendingDate > 3 days)

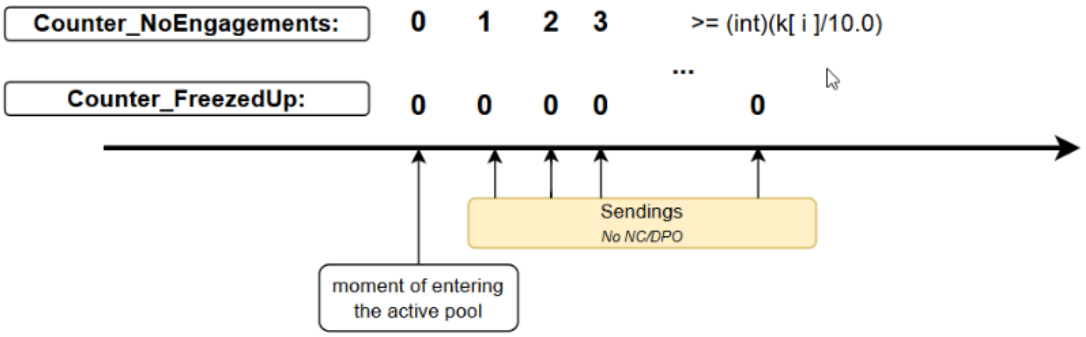
This 3-day delay is introduced to give the email a fair chance to respond and become engaged.

Definition:

- LastSendingDate is the date of the most recent email sent which triggered the threshold:
  Counter_NoEngagements ≥ (int)(k[i] / 10.0)

Note:

- The variable k[i] represents a parameter associated with the email—likely contact-specific or configurable—which must be defined elsewhere.



To determine whether an email matches a CONFIRMED PATTERN, use the following table and condition:

- Table: `app_system.email_patterns_domains_categories`
- Condition: `category_id != 4` (a value of 4 indicates a non-confirmed pattern)

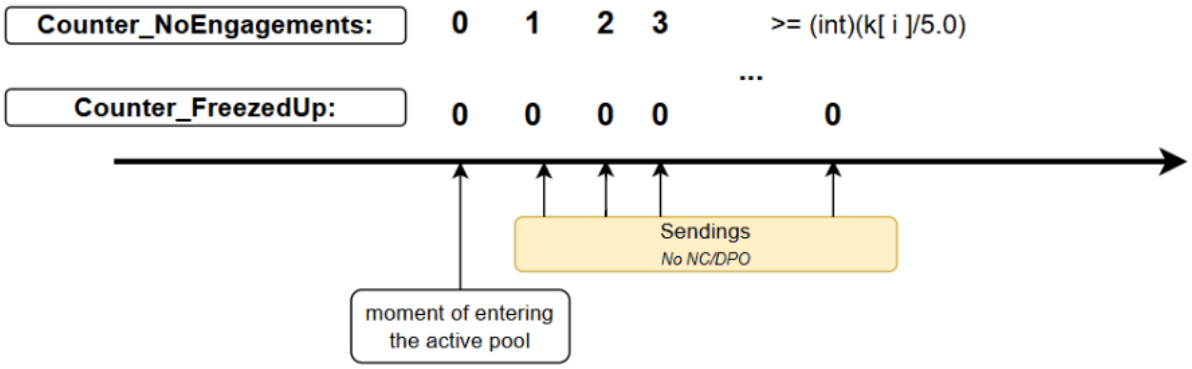
2.4. OR The following condition is met:

- The email **matches a CONFIRMED PATTERN** ,
- AND `Counter_NoEngagements >= int(k[i] / 5.0)` ,
- AND `Counter_FreezedUp == 0` ,
- AND `(CurrentDate - LastSendingDate > 3 days)`

This 3-day delay provides an opportunity for the email to respond and register an engagement before being disqualified from the active pool.

Definition:

- **LastSendingDate** is the date of the most recent mailing that resulted in the condition being met: `Counter_NoEngagements >= sent int(k[i] / 5.0)`



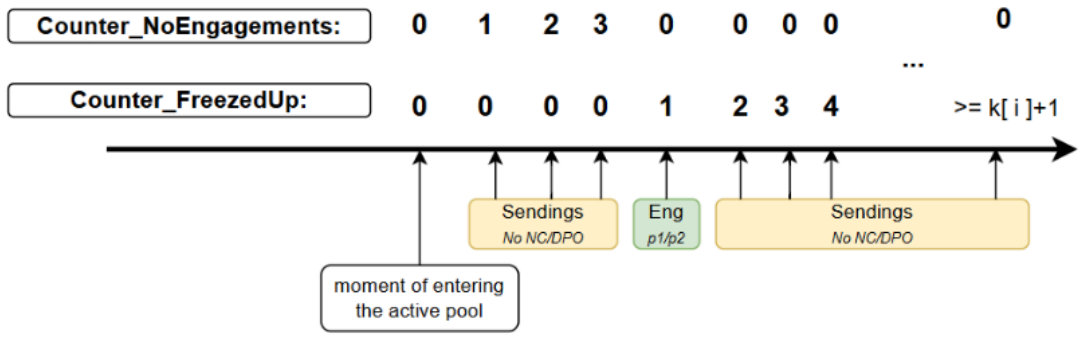
Used tables and values:
app_system.email_patterns_domains_categories.category_id = **4**

2.5. OR the following condition is met:

- Counter_FreezedUp > 0
- AND Counter_FreezedUp ≥ k[i] + 1
- AND (CurrentDate – LastSendingDate > **3 days**)
  (a delay is given so the email has time to respond to the mailing and become engaged)

Where LastSendingDate is the date of the last mailing during which the counter value reached:
Counter_FreezedUp ≥ k[i] + 1

Explanation:
This condition applies in cases where the email was engaged (P1, P2) after being placed in the active pool.



2.6. OR the following condition is met:

- the email is marked as **DB Extra** (it doesn't matter whether this status was assigned before or after entering the active pool),
- AND Counter_NoEngagements > 0
  (i.e., the email is given a chance — 1 mailing — to get an engagement; if it does, it stays in the active pool according to other rules),
- AND (CurrentDate – LastSendingDate > **3 days**)
  (a delay is given so the email has time to respond to the mailing and become engaged)

Used tables and values:
app_system.emails_status_active.valid = **2**

2.7. OR the email is marked as **Soft Bounced in suspension list**
(it doesn't matter whether this status was assigned before or after entering the active pool, though by other rotation rules such emails should not be moved to the active pool from the semi-active pool).

Do not implement this rule for now, as the suspension list has not yet been implemented.

Used tables and values:
app_system.emails_status_active.valid = **7**

2.8. OR the email is **from a banned domain**
(it doesn't matter whether this status was assigned before or after entering the active pool, though by other rotation rules such emails should not be moved to the active pool from the semi-active pool).

Used tables and values:
app_system.domains_status.blacklisted > **0**
OR
app_system.domains_status.website_status IN (**2, 3**)

2.9. OR the email is categorized as **general**
(it doesn't matter whether this status was assigned before or after entering the active pool, though by other rotation rules such emails should not be moved to the active pool from the semi-active pool).

Used tables and values:
app_system.domains_status.general > **0**

2.10. OR the email is **from a domain with “mx = no”**
(it doesn't matter whether this status was assigned before or after entering the active pool, though by other rotation rules such emails should not be moved to the active pool from the semi-active pool).

Used tables and values:
app_system.domains_status.mx = **"no"**

Determining a Candidate Email for Moving to the Active Pool from the Semi-Active Pool (analysis of emails from app_system.emails_status)

3.1. Selection of eligible candidate emails from the semi-active pool. Candidates must meet the following conditions:

3.1.1. AND the email is **valid**
Used tables and values:
app_system.emails_status.valid IN (**1, 2, 3, 7, 8**)

3.1.2. AND the email is **not dangerous**
Used tables and values:
app_system.emails_status.dangerous = **0**

3.1.4. AND the email is **not from a banned domain**
Used tables and values:
app_system.domains_status.blacklisted = **0**

3.1.5. AND the email is **not from a general domain**
Used tables and values:
app_system.domains_status.general = **0**

3.1.6. AND the email is **not from a domain with “mx = no”**
Used tables and values:
app_system.domains_status.mx != **"no"**

3.2. Selection of the primary email to be moved to the active pool from the list of eligible candidate emails in the semi-active pool:

Candidate emails should be sorted as follows:

3.2.1. Primary sorting: from min to max by the **active pool insert counter**

3.2.2. Nested sorting within 3.2.1: by email pattern category in the following order:
[confirmed, known, unknown, not recognizable]

Used tables and values:
app_system.email_patterns_domains_categories.category_id = **[4 → 1]**

3.2.3. Nested sorting within 3.2.2: by validity type priority:

- Valid / Non-catchall — app_system.emails_status.valid = **1**
- DB Extra Verified — app_system.emails_status.valid = **3**
- Unverified — app_system.emails_status.valid = **8**
- DB Extra — app_system.emails_status.valid = **2**
- Soft Bounce — app_system.emails_status.valid = **7**

3.2.4. Nested sorting within 3.2.3: by random order or in the natural order of results returned from the database (randomization is considered unnecessary here).
The developer should clarify how this final sorting step was implemented – either explicitly stated or relying on the default DB order.

The **primary email** to be moved to the active pool is the first one in the sorted list described above — for example, it would have:

- The lowest insert counter,
- A confirmed email pattern,
- Valid / Non-catchall status.