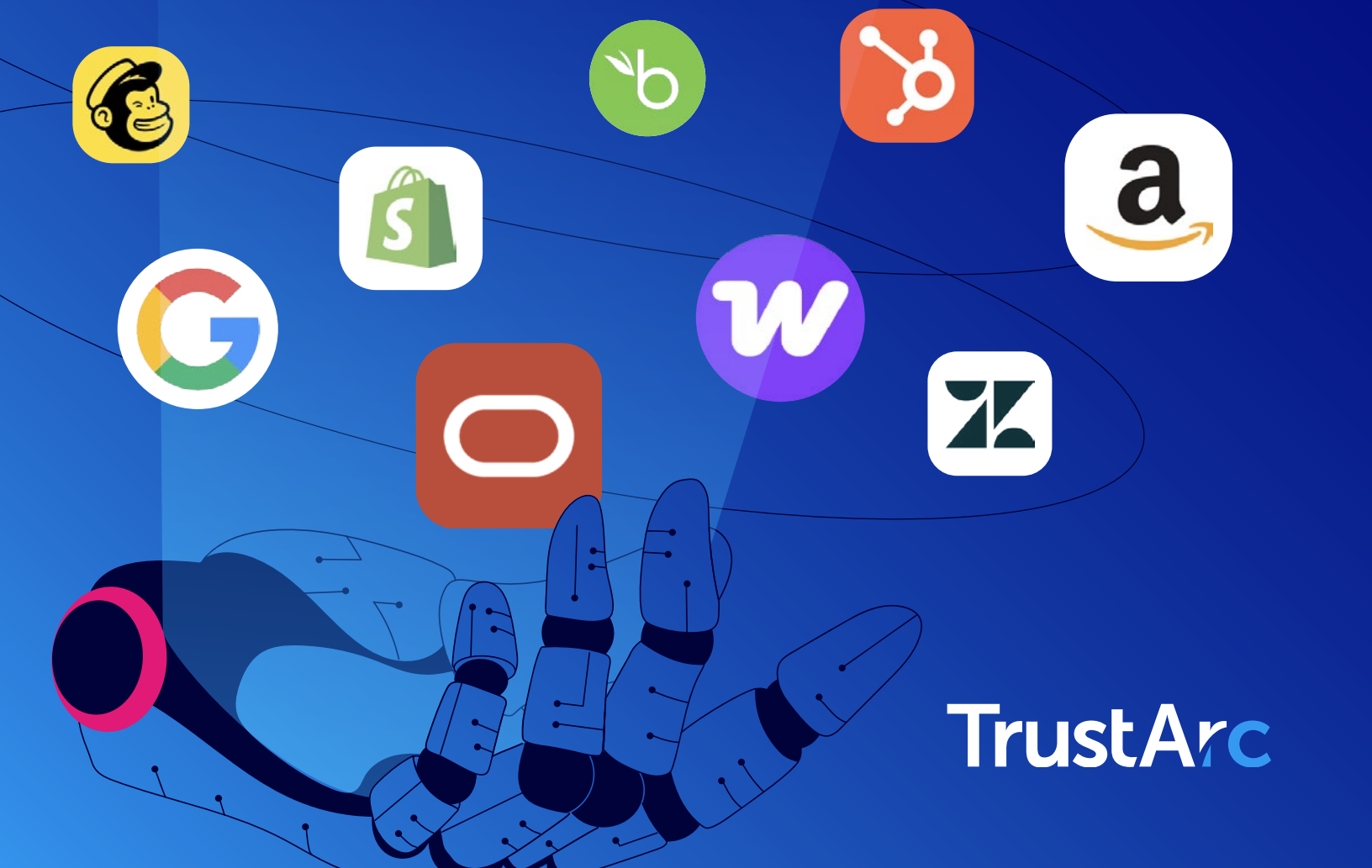
eBook

# The Privacy Automation Cookbook

A hands-on guide to automation logic, task design, and Recipe efficiency for privacy teams.



TrustArc

# Table of contents



TrustArc

# Welcome to Your Cookbook

## You've Set the Table. Now it's Time to Cook.

If you're here, you've done the groundwork:

- You've worked through the Getting Started eBook
- You know what a Recipe is and why it matters
- You understand how TrustArc Integrations connect your systems
- You're ready to move from concepts to real automation

This Cookbook is your next step. Here, you'll move beyond concepts and start learning how to structure flows, measure task usage, and build logic that fits your real privacy jobs.

## What This Cookbook Is

This is a builder's handbook for privacy teams.

**YOU'LL LEARN HOW TO:**

- → Break down automation into clear, manageable steps
- → Understand how every action affects your task usage
- → Design flows that save time, stay reliable, and adapt as your needs grow
- → Map logic visually so every decision makes sense
- → Use checklists and prompts to plan out your own flows with confidence

The focus here is on helping you think like a builder — making smart decisions about how, when, and why each step runs.

If you ever need to revisit the basics, you can always flip back to the Getting Started eBook. But when you're ready for hands-on, tactical advice, this Cookbook will help you build with clarity and purpose.

## How to Use This Cookbook

Every section covers a core concept of Integrations — task counting, logic, structure, and efficiency.

**YOU'LL FIND:**

- → Planning checklists and prompts to help you design your logic
- → Visual breakdowns that show you how a flow works, step by step
- → Practical explanations for how tasks are counted, how conditions work, and how to avoid costly missteps
- → Tips and reminders to help you adapt what you learn to your own systems

You can work through this eBook in order, or jump to the section that matters most right now. Use it as a reference whenever you need more detail on how to plan, measure, and optimize your automations.

# Core Concepts

## The Builder's Mindset

**This section helps you shift from knowing what a Recipe is to thinking like someone who builds them well.**

Great builders don't start with steps.
They start with outcomes.

They ask, **What job am I trying to automate? What change should happen at the end of this flow?**

### HERE ARE THE HABITS THAT MAKE A DIFFERENCE:

→ **Start with the outcome.**
Before you build, ask: What does this need to do for my privacy program? What should happen?

→ **Take the shortest clear path.**
If three steps will do the job, don't use six. Remove anything that doesn't add value.

→ **Review after you run.**
Every Recipe will teach you something. What failed? What took more tasks than expected? What's worth changing?

**This is how you build automations that last. Not just ones that work — but ones you can trust, explain, and improve over time.**

# Getting Ready

**You can't build a solid Recipe without knowing what you're working with.**

The best automations come from good prep — clear inputs, specific goals, and a handle on the systems involved.

## Recipe Checklist

Think of this as your **mise en place**. Use it upfront to help you design your Recipe before building.

- **Systems**: What systems do you want to integrate? Are they in the connector library?
- **Credentials**: Do you have the credentials to the systems to set up the connection? If you need admin access or special roles, coordinate with your IT or system owner before you start.
- **Field Inventory**: Do you know which fields you'll need for the flow — both coming in and going out?
- **Business Rules**: Have you defined what should trigger the Recipe, and any filters or conditions that apply? What actions need to happen in each system?
- **Workbook**: Download or print the TrustArc Workbook. Use it to sketch your field mappings, map business logic, and plan error handling.

# Recipe Design Prompts

Before you drag-and-drop your first action into the builder, take a few minutes to answer these questions. They'll save you from rework and troubleshooting later.

1. **What will trigger this Recipe?**

   Is it real-time, such as a form submission or new record? Or is it scheduled, like checking Salesforce every hour? Nail down the exact event and system that starts your flow.

2. **Which systems are involved and who has credentials for them?**

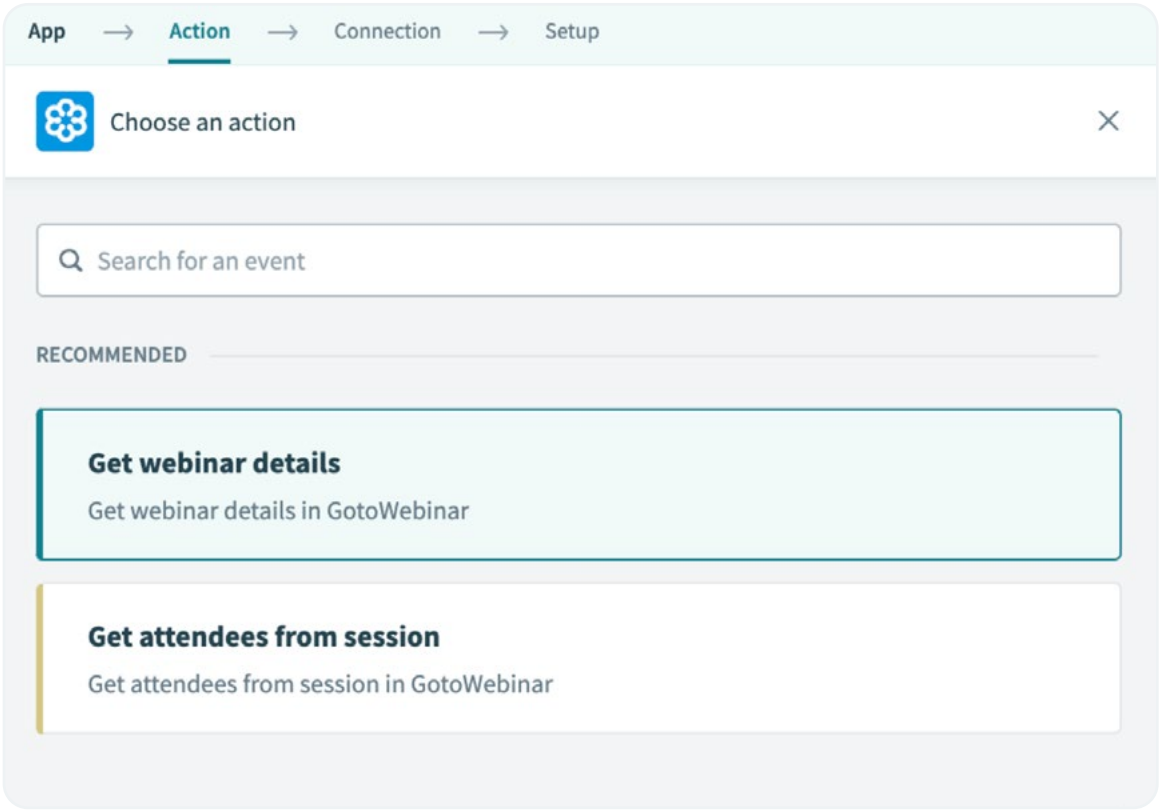
   Figure out permissions and system ownership now, before you start building. It's easier to unblock access early than to pause mid-way through.

3. **How often should this flow run?**

   Does this need to check for changes every 15 minutes? Hourly? Daily? Match your cadence to the real urgency of your workflow. Frequent triggers may cost more in tasks if you're pulling lots of data each time.

4. **What happens if something fails?**

   Do you need to retry? Alert someone? Stop the flow? If your Recipe hits an error and fails silently, you may not notice for days. Plan how to handle failures or system downtime — especially for flows that touch compliance-critical data.



# Task Mastery Guide

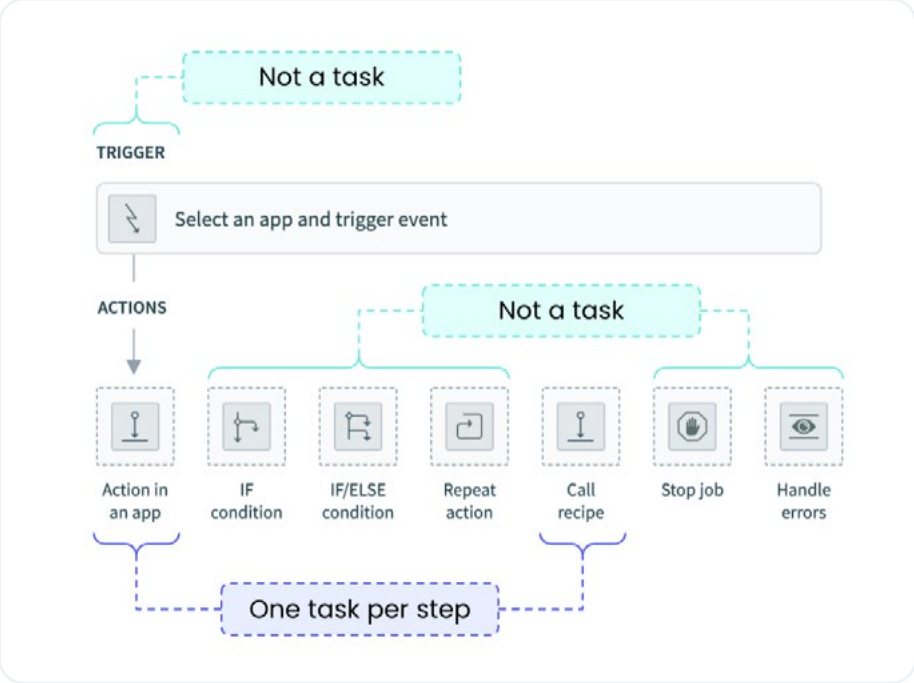
Every time a Recipe runs, it uses up tasks. A task is the core unit TrustArc uses to track usage and bill for Integrations. When your Recipe sends, receives, updates, or deletes data across systems, it consumes tasks.

**HERE'S WHAT WE'LL COVER:**

- What does and doesn't count as a task
- How filters, loops, and conditions factor in
- Where to watch task usage
- How to estimate your task usage

## What Counts as a Task

Not every step in a Recipe costs you tasks. The only steps that count are the ones that actually move or change data between systems. All actions in an app or call Recipe is a task.



## Always Counts as 1 Task

| ACTION TYPE | EXAMPLES |
|---|---|
| **Calling a connector action** | Create a record in Salesforce, update a contact in Klaviyo, delete a row in Snowflake |
| **Fetching or retrieving data** | "Search accounts," "Get file," "List users"<br><br>Searches that hit an external system (e.g., "Search records in Salesforce") count as 1 task. |
| **Sending or pushing data** | Email, webhook, Slack message, FTP file drop, HTTP post, etc. |
| **Calling another Recipe** | The call itself = 1 task; every step in the called Recipe counts individually |
| **Looping through items** | **Each action inside a loop counts as 1 task per item processed.**<br><br>**The loop step itself is free but every connector action inside it is counted.** |
| **Retries (if successful)** | A retried step counts again if it succeeds on retry |

## Sometimes Counts — It Depends

| CASE | TASK? | NOTES |
|---|---|---|
| **IF/ELSE condition** | No | Evaluating an *IF* does **not** cost a task |
| **Actions inside the IF** | Yes, if run | If the step inside the *IF* executes, it counts as 1 task |

A task is only counted if the condition leads to a connector action.

## Never Counts Toward Task Usage

| STEP TYPE | WHY IT'S FREE |
|---|---|
| **Recipe triggers** | Triggers (webhook, scheduled, polling) **don't consume tasks themselves** |
| **Notes or comments** | Purely for documentation |
| **Variable assignments** | Declaring or updating variables is free (unless it triggers connector actions) |
| **Inline formulas** | Transforming data inside fields doesn't count unless it's inside an external call |
| **Skipped steps** | A skipped step doesn't execute |
| **Polling interval itself** | The act of polling costs nothing **but the following connector action will count as a task** |

# How Tasks Are Calculated

Let's learn by doing. Here are three Recipe examples — each one slightly more complex than the last. You'll see how task usage builds as logic and system involvement grow. You'll also get a feel for what's predictable, what's conditional, and what's optional based on behavior.

Let's start simple.

## Example 1: The Basics

**Scenario**: You want to run a Recipe when a new user is created in TrustArc and log the event to Airtable.

| STEP | TASK COUNT |
|---|---|
| Trigger: New user added | 0 |
| Action: Log to Airtable | 1 |
| **Total** | **1 task** |

**WHY THIS MATTERS:**
The trigger itself doesn't count as a task. It just signals that something happened, like "a new user was added" or "a file was uploaded." That signal starts the Recipe but doesn't do any work yet.

What counts is what happens *after* the trigger.

Here, logging the event to Airtable is the only step that uses a task.

**Tip**: A trigger starts the Recipe, but only connector actions consume tasks. Think: search, create, update, delete.

## Example 2: One IF, Two Outcomes

**Scenario**: You want to update a record in Salesforce only if the user's region is "EU".

| STEP | TASK COUNT |
|---|---|
| Trigger: Record updated | 0 |
| IF: Region = EU | 0 |
| IF TRUE ⟶ Update Salesforce | 1 |
| IF FALSE ⟶ Skip | 0 |
| **Total (if TRUE)** | **1 tasks** |
| **Total (if FALSE)** | **0 tasks** |

**WHY THIS MATTERS:**
IF statements **do not cost a task by themselves**.
Only the action that runs inside the branch is counted.

## Example 3: Multi-System Sync Based on User Opt-in

Let's look at a multi-branch Recipe that updates consent records across different systems based on what the user opted into. This one reflects a confirmed flow from Consent & Preference Manager (CPM) to downstream tools like SFMC and Microsoft Dynamics.

**Scenario**: A user submits a consent form through CPM. Based on their selected communication channel (Email or SMS), the Recipe routes data to different systems. If they opt into Email, it updates SFMC. If they opt into SMS, it updates Microsoft Dynamics. If both are selected, both systems are updated. Finally, the sync is logged in Airtable for reporting.

| STEP | WHAT IT DOES | NOTES | TASK COUNT |
|---|---|---|---|
| 0 | **Trigger: New consent submitted via TrustArc CPM form** | Trigger is webhook-based. Doesn't count as a task. | 0 |
| 1 | **IF Email is selected** | Control logic. No connector action here. | 0 |
| 2 | **IF SMS is selected** | Another control check. No task here either. | 0 |
| 3 | **IF Email = TRUE ⟶ Update SFMC with opt-in** | This is a connector action to SFMC. Runs only if user opted in to Email. | 1 if TRUE |
| 4 | **IF SMS = TRUE ⟶ Update Microsoft Dynamics with opt-in** | This is a connector action to Dynamics. Runs only if user opted in to SMS. | 1 if TRUE |
| 5 | **Record consent channel and timestamp in Airtable** | Always runs, even if no opt-ins are selected. Connector action. | 1 |
| 6 | **Total (if both Email and SMS are selected)** | SFMC, Dynamics, and Airtable all run. Three connector actions total. | 3 |
| 7 | **Total (if only one channel is selected)** | Either SFMC or Dynamics runs (not both), plus Airtable log. Two connector actions total. | 2 |
| 8 | **Total (if neither is selected, but form is valid)** | Only Airtable log runs. One connector action. | 1 |

# Where to Monitor Task Usage

Once your Recipes are running, task costs start accumulating. So how do you actually **monitor usage**, catch **spikes**, and stay below your plan's limit?

Let's walk through where to check your task consumption, how to catch unusual patterns, and what to do when something looks off.

## Where to Look: Dashboard and Usage Views

In the **Integrations** section of the TrustArc platform, you'll find the **Usage** tab.

Here's what you'll see right away:

- → Number of **Active Recipes**
- → Total **Jobs Run**
- → Total **Tasks Used**
- → Number of **Failed Jobs**

On the right side of the screen, you'll see your current **task usage total** and how close you are to your allotted task pool.

You're on a **task-based plan** — that means every action counts.

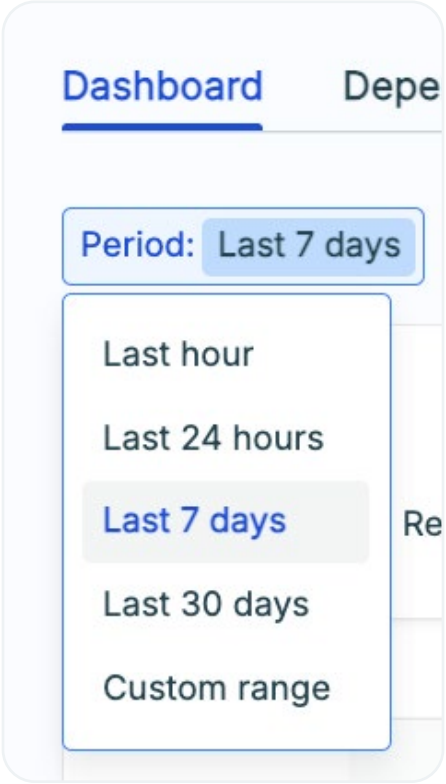
Monitoring task usage is like tracking how many ingredients you're using. The sooner you notice overuse, the easier it is to fix.

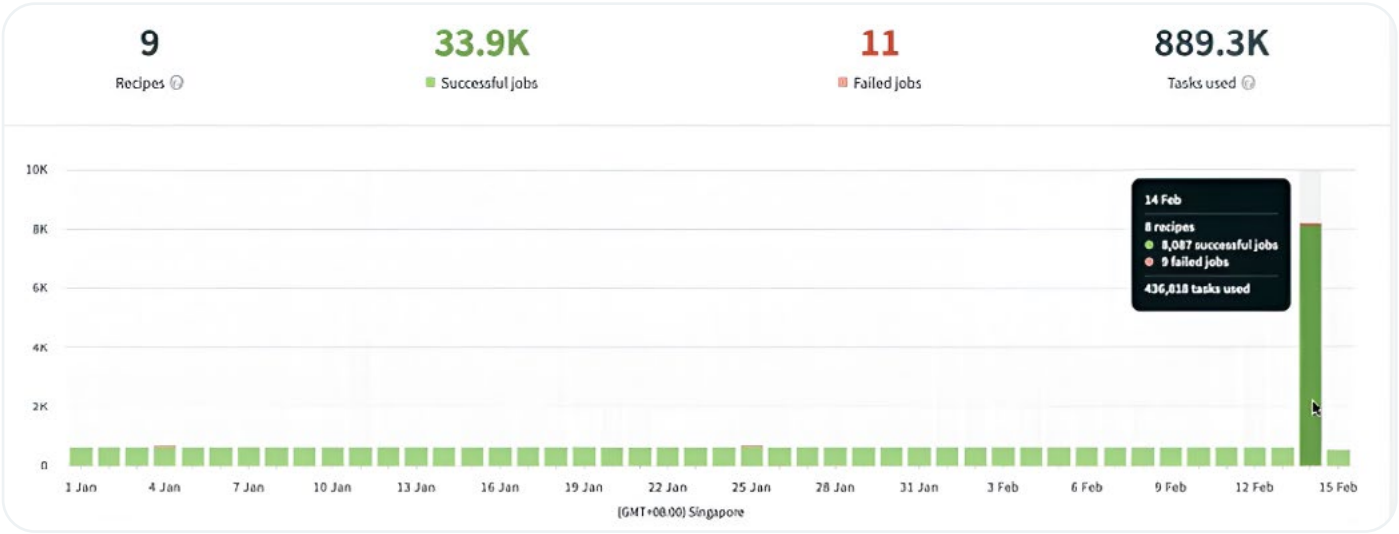
## Set a Time Range That Makes Sense

You can filter the Dashboard by:

- → **Today**
- → **This week**
- → **Last 30 days**
- → Or set a **custom date range** — from the start of your contract to now

This helps you see how usage changes over time and spot any patterns (like spikes that happen at month-end or after a Recipe change).






## What a Spike Looks Like — and How to Investigate It

Let's say you check your usage and see this:

→ On February 14: **415,000 tasks**

→ On February 15: **1,900 tasks**

That's a spike. Now what?

### HERE'S HOW TO INVESTIGATE:

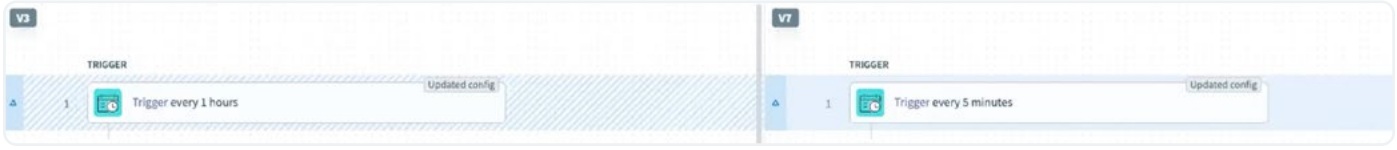
1. In the **Usage Dashboard**, find the **Recipe** responsible for the spike
2. Click into that Recipe
3. You'll see the **Recipe Editor** — but that won't tell you much about what changed
4. Instead, click the **Versions** tab
5. Select the **spike version** and the **previous version**


6. Use the **Compare to Current Version** option

The screen will show you **what changed between versions**.

### FOR EXAMPLE:

You'll see that the Recipe trigger was changed from "Every 1 hour" to "Every 5 minutes." That one change could have caused the spike.



By checking the **Versions tab**, you can see:

- → Exactly what changed
- → When it changed
- → And what impact it had

**Tip**: **Every time you update a Recipe, leave a comment in the Versions tab**. That way, if something spikes later, you won't have to guess why.

## Advanced Monitoring: Use a Recipe to Watch Your Recipes

If you're running many automations — say 500+ Recipes across brands or regions — you need a way to **monitor them all** at once.

### HERE'S HOW WE DO IT:

Yes — we use a **Recipe to monitor other Recipes**.

This monitoring Recipe pulls usage stats across your workspace and:

- Flags any Recipe with a task usage spike
- Compares task usage to the previous version
- Sends alerts if usage rises beyond a defined threshold (e.g. 100%)

You can configure this to:

- → **Pause** high-usage Recipes temporarily
- → **Notify** your internal admin
- → Prevent going over your total task allotment

## Final Thought Before We Move On

Your Recipe is a living thing. Even if it's efficient today, a single change can make it expensive tomorrow.

That's why task monitoring is like checking your stove while you're cooking.

You don't need to stare at it — but if it's glowing red, don't ignore it.

Next: Let's design Recipes that avoid spikes in the first place.

# Designing Efficient Recipes

By now, you've seen what a task is. You've learned how fast they can pile up. And you know where to watch for spikes.

So the next question is:

**How do you build Recipes that are smart from the start — without wasting steps, tasks, or time?**

This section will walk you through that. We'll go slow, explain the variables and logic where needed, and help you build a mental model that lasts.

Let's begin.

## First, think like this:

A good Recipe isn't just about *doing things*.

It's about doing the **minimum necessary steps** to get the job done correctly, clearly, and once.

- → It shouldn't fetch everything just because it can
- → It shouldn't check every row if only 10 matter
- → It shouldn't write something that's already there

### 1. Filter as Early as Possible (Don't Bring in Noise)

The most common task mistake is bringing in too much data, only to throw most of it away later.

Let's say you're pulling records from Salesforce or from TrustArc Consent & Preference Manager.

**Bad approach:**

> *Pulling all records without filters, like SELECT * or unfiltered list actions*

Then using a bunch of *IF* statements later to discard 95% of them.

**Better approach:**

> *Use filters like updated_at > last_run_time and consent_status = opted_in inside the connector step.*

Here's what each part means:

- → *updated_at > last_run_time*: only brings in records that changed since the last run
- → *consent_status = 'opted_in'*: only brings in the records you need to act on

You're slicing the data ***before*** it enters the Recipe. This avoids triggering actions on rows you don't need.

**WHY THIS MATTERS:**
Every unnecessary record you bring in is one more chance for wasted steps, unnecessary loops, or task-heavy outcomes.

## 2. Choose the Right Trigger for the Job

When your Recipe starts, it does so because something changed. But not every job needs to run the second that happens. Some do. Some don't. So you need to ask: "If this runs every 5 minutes instead of instantly, would it still be useful?"

### REAL-TIME EVENT TRIGGERS

An event trigger is a mechanism that initiates an action or a set of actions based on a specific event that happens in one of your systems. Use these when the event:

- → Needs to update other systems *immediately*
- → Is user-facing (like someone opting out or submitting a DSAR)
- → Comes from an external system with real-time support (e.g. Salesforce or TrustArc CPM)

These use more tasks over time because they fire more often — but that's the cost of being instant.

### POLLING TRIGGERS

These check for changes every X minutes, hours, or days.

Use them when:

- → It's okay to be a little late (e.g. syncing vendor deletions)
- → You want to save task volume
- → You're processing large files that are dropped on a schedule

**Tip**: Triggers don't count as tasks. But every time a polling trigger runs, it will lead to downstream tasks.

### DECISION FRAME:

Will a 15–30 minute delay break something downstream?

- → If no, poll
- → If yes, use a real-time trigger

## 3. Avoid Nested Loops (Unless You Really Have To)

### WHAT IS A LOOP?

A loop (called *Repeat* in the builder) means:

For each record in this list → do these steps

### WHAT IS A NESTED LOOP?

A loop *inside* a loop.

Let's say:

- → You're looping over 200 customer records
- → And inside that, you're looping over 10 subscriptions per customer

That's:

*200 customers × 10 subscriptions = 2,000 iterations*

Now if you're doing even **2 actions per iteration** (like logging and updating), that's:

*2,000 × 2 = 4,000 tasks*

It gets expensive fast.

Only use nested loops if:

- → There's no way to flatten or group the data
- → You've filtered the list before entering the loop

### 4. Only Write When Something Actually Changed

This is one of the simplest but most ignored tips. Let's say you pull 1,000 records, but only 100 were updated. Don't write all 1,000 back into the system. Instead, compare timestamps:

*IF updated_at > last_synced_at*

*→ THEN update record*

*ELSE skip*

You can also compare values:

*IF region has changed*

*→ THEN update Salesforce*

Each skipped action = 1 task saved.

### 5. Log with Intention, Not as a Default

Logging is important. But not every action needs a record.

**TASK REALITY CHECK:**
Every Airtable insert, Slack message, or Google Sheet row = 1 task. That means:

- → Logging 1,000 looped rows = 1,000 tasks
- → Logging only errors = maybe 2–3 tasks

Log:

- → When something fails
- → When you complete a batch
- → When someone needs an alert

Skip logging:

- → For every successful loop iteration
- → If the data is already written elsewhere

### 6. Comment Your Recipe Steps Like You'll Forget Them Tomorrow

You're not writing comments for today. You're writing for:

- → Yourself in 6 months
- → A teammate who just joined

Write:

"Using batch action to reduce task load. Trigger every 15 mins."

Or:

"Only syncing records where marketing_opt_in = TRUE."

**FINAL GUT CHECK:**
Before you turn on a Recipe — ask yourself:

- → Did I filter the data entered?
- → Did I flatten or simplify loops?
- → Am I writing data only when needed?

## Last Thing Before You Go

Smart Recipes don't just run — they improve. The best builders look at what happened, where tasks were spent, and what could go better next time.

That's why every optimization tip here is something you can act on and measure. You'll see it in your logs. You'll feel it in how quickly things run. And you'll notice when your task usage stays flat even as your automation load grows.

From here, the rest is practice. Use your Workbook to map your next Recipe.

# Resources & Appendices

As you continue to build and adapt your privacy automations, use these resources to stay sharp, efficient, and in control.

## Task Cost Cheat Sheet

Quickly estimate the task impact of common actions:

| ACTION/EVENT | TASK COST |
|---|---|
| Create, update, or delete a record in any system | 1 task per record |
| Fetch data (search, list, get) | 1 task per call |
| Bulk/batch update (if supported) | 1 task per batch action |
| Conditional logic (IF/ELSE check) | 0 (unless action triggered) |
| Polling or scheduled trigger | 0 (the jobs triggered count tasks) |
| Variable assignments, notes, comments | 0 |

## Glossary of Integration Terms

Use this list when something unfamiliar shows up in your Recipe, task log, or setup screen.

### RECIPE AUTOMATION BASICS

**Recipe:** Series of connectors, actions and triggers made possible by connections.
**Trigger:** When the recipe starts. Can be real-time, scheduled, or based on system events.
**Action:** Do a thing. Move data, call a system, update a record.
**Task:** Unit of doing a thing. Only successful ones count to your billing usage.
**Job:** One full run of a recipe, from start to finish.
**Connection:** Authentication that lets a recipe access a specific system.
**Recipe Folder:** Place to organize and group saved recipes inside your workspace.

### TRIGGER TYPES

**Real-Time Trigger:** Starts instantly when another system sends a webhook. No delay or schedule needed.
**Batch Trigger:** Pulls multiple records at once before starting the recipe. Often used with polling or pagination.
**Bulk Trigger:** Starts a recipe based on a large input like a CSV file or full dataset. Good for high-volume imports.
**Scheduled Trigger:** Starts the recipe at a set time or interval, even if no new data is found.
**Polling:** Scheduled check that looks for new records. Runs the recipe only when data is found.
**Webhook:** The method used to send real-time triggers from another system.

### LOGIC & CONTROL

**Condition (IF/ELSE):** A decision rule that lets your Recipe take different paths depending on the data.

**Filter:** A rule that limits which records enter a Recipe. Helps avoid unnecessary actions or task usage.

**Loop (Repeat):** A set of steps that repeat for each item in a list, or while a condition holds true. Useful for processing many records.

**Lookup**: A one-time data search used to enrich or match a record inside a Recipe.

**Variable**: A saved value (like a date or ID) that your Recipe can reuse across steps.

**Batch Action**: A step that processes multiple records in one call. Reduces task usage. Not all connectors support it.

### CONNECTORS

**Connector:** A third-party system your recipe connects to, like Salesforce or Snowflake.
**Featured Connectors:** Common in privacy workflows. Work with all recipe features. Often include prebuilt recipes.
**Other Connectors:** Have triggers and actions, but may not have as many tested use cases.

### TEMPLATES & SAVED RECIPES

**Saved Recipe:** A recipe you built and saved in your account.
**Recipe Template:** A working recipe you can copy and customize for real privacy jobs.

### MONITORING

**Error Monitor:** Alerts you when a recipe fails or runs into an issue.

## About TrustArc

As the leader in data privacy, TrustArc automates and simplifies the creation of end-to-end privacy management programs for global organizations. TrustArc is the only company to deliver the depth of privacy intelligence, coupled with the complete platform automation, that is essential for the growing number of privacy regulations in an ever-changing digital world. Headquartered in San Francisco, and backed by a global team across the Americas, Europe, and Asia, TrustArc helps customers worldwide demonstrate compliance, minimize risk, and build trust. For additional information visit **TrustArc.com**.

TrustArc