TrustArc Integrations Workbook

TrustArc

How to Use This Workbook

This workbook helps you think through how to architect and design a TrustArc Integration Recipe:

- What outcome you're trying to achieve
- How the data should flow between systems
- When that flow should be triggered
- What steps are needed (and which are wasteful)
- What might break, and how you'd catch it
- What the long-term cost and upkeep might look like

HOW TO USE IT:

- Print it out or fill it in digitally
- Use one full pass per Recipe or use case

TrustArc

Map the Job You're Solving

Let's start by getting clear on your use case. What systems you want to integrate, what outcomes you want to happen, and why.

What's the Job to Be Done?

Fill this in clearly. Be specific.

When [trigger event happens],

I want [specific action/outcome to occur],

So that [privacy/compliance/business goal is achieved].

EXAMPLE:

When *a new vendor record is created in ServiceNow,*

I want *that vendor and its core details automatically synced to TrustArc,*

So that *our privacy inventory always reflects the latest third-party relationships without using spreadsheets or sending emails manually.*

Tip: This exact Recipe comes prebuilt in TrustArc. You can use it out of the box and adapt it to fit your own systems or business processes. Just swap ServiceNow for another vendor source, or tailor the fields you want to sync.

What Systems Are Involved?

List every system that touches this flow.

SYSTEM NAME	ROLE IN THE RECIPE (Trigger or Action)	SPECIFY WHAT HAPPENS IN THE SYSTEM

What Kind of Data Is Moving?

This helps you understand what data properties to use.

DATA TYPE (e.g., record, status, tag, ID)	FROM SYSTEM	TO SYSTEM	NOTES

When Should This Run?

Choose the best match:

- ☐ Every time a record changes (event-triggered)
- ☐ At scheduled intervals (polling/scheduled)
- ☐ Only when a human takes action (manual trigger)
- ☐ Other: ______________________________

If you're unsure how polling vs event triggers affect task usage, revisit the Privacy Automation Cookbook.

What Should Not Happen?

This helps prevent scope creep and wasted tasks.

→ **Example**: "Don't process records with status = draft"

→ **Example**: "Exclude any entry not linked to a consent profile"

Write your own guardrails:

Where do guardrails belong? Set your "don't process" rules as early as you can — ideally in the trigger or first filter step. This keeps unwanted records out from the start. If you need more specific checks, add IF or skip logic later in the flow. Good Recipes filter before they act.

Design the Recipe Flow

"Structure reveals logic."

This section helps you outline your Recipe — step by step — before you build it.

Think of this like building a skeleton. Each action is a joint. Each condition is a hinge. Each trigger is the first push that sets a series of actions.

Your Recipe Skeleton

Start with this structure. Fill it in clearly. See the following page for more information.

STEP #	**WHAT SHOULD HAPPEN HERE?**	**WHAT SYSTEM DOES THIS HAPPEN IN?**	**WHEN SHOULD THIS STEP RUN?** (List any conditions or filters)
1 (Trigger)			
2 (Action)			
3 (Action)			
4 (Action)			
5			
6			
7			
8 (Optional)			

About the "When should this step run?" column:

Not every step will apply to every record. Sometimes a step only makes sense when the record meets your rules. This is where you write those rules down.

For example:

- → Only continue if vendor status is "approved"
- → If the data subject request type is "erasure," notify IT; if the type is "access," notify compliance

Use this column to spell out any "if" or "when" conditions. These keep your Recipe focused and prevent unwanted steps. If you find yourself writing a long or complicated answer, it might be a sign you need to split this step or let a different system handle part of the logic.

Tip: If you find yourself writing long or messy answers:

- → You may need to split the steps.
- → Or rethink what system should handle it.

Add Repetition (Loops or Batches)

Will any step need to run for multiple records at once?

- ☐ No, every record is handled one at a time
- ☐ Yes, and I want to process them one by one (loop)
- ☐ Yes, and I want to process them all at once (bulk)

Tip: Loops cost more tasks but allow precision. Bulk saves on tasks but has less flexibility.

Describe the repeated step:

Optional: Add a Stop Condition

Sometimes, a Recipe should not continue if certain conditions are met.

→ **Example**: If a record has no email, stop here.

→ **Example**: If status = "archived," skip

Write your stop rule here:

Quick Logic Check

Before you move on:

→ Does each step do only one thing?

→ Are you avoiding steps that try to cover two unrelated outcomes?

→ Do you know when the Recipe runs and why?

→ Are you sure you're not processing every record by default?

Plan for Errors & Edge Cases

"What will break? And what should happen when it does?"

You've scoped your logic. You've estimated your task usage. But no automation runs perfectly forever. This section helps you design for when things don't go to plan — without adding unnecessary complexity.

We're not talking about advanced error handling or retry logic here. Those tools exist inside the Recipe builder and platform. This section is about **thinking like a resilient builder**.

Before you start building, answer:

→ Where can this Recipe fail?

→ What should happen when it does?

→ Who needs to know, and when?

This keeps your flows clean, recoverable, and easier to monitor.

Design Your Alert Logic

You don't need to be paged for everything. But you do need visibility.

WHEN DO YOU WANT TO BE NOTIFIED?

- ☐ On any failure
- ☐ Only on repeated failures
- ☐ Only when a critical system is involved
- ☐ When task usage spikes suddenly
- ☐ When the Recipe fails silently

HOW DO YOU WANT TO BE NOTIFIED?

- ☐ Email
- ☐ Slack
- ☐ Log entry (e.g., Airtable or Sheet)
- ☐ Just mark in the dashboard for later review
- ☐ Other: ____________________

Write your logic below:

→ "If Step ____ fails, send Slack message to privacy-ops channel"

→ "If Recipe fails more than 3 times in 1 hour, alert a team member"

→ "If task usage jumps more than 200% in a day, log alert to admin sheet"

Monitor & Maintain Over Time

What to Monitor

FROM THE INTEGRATIONS DASHBOARD, TRACK:

⟶ Total tasks used per Recipe (monthly trend)

⟶ Number of jobs triggered (volume baseline)

⟶ Failed jobs (and whether they repeat)

⟶ Recipe version changes (who edited what, and when)

Version History as Your Audit Trail

Every time you save a Recipe, a version is created.

In the Version tab, you can:

⟶ Compare two versions to see what changed

⟶ Spot task increases from edits

⟶ Leave comments explaining why the change was made

BEFORE YOU DEPLOY AN UPDATE, WRITE DOWN:

⟶ "What exactly did I change?"

⟶ "What's the risk of this version?"

⟶ "How would I know if this broke something?"

ADD A COMMENT LIKE:

⟶ "Switched from loop to bulk insert to reduce task cost. Should save ~90% per run."

That single line could save hours of debugging later.

Tip: Commenting within the Builder
You can add comments almost anywhere in the Recipe builder. If you want to explain a single step, use a comment right there, it helps anyone reading understand your logic or choice. For big-picture notes, like why you changed a Recipe or what should be watched after an update, comment at the Recipe level. Good comments make life easier for your team and for your future self, whether you're reviewing one action or the whole flow.

How Monitoring Works at Scale

Whether you're running one Recipe or fifty, **TrustArc monitors task usage and behavior across your account**. If your Recipes begin consuming more tasks than expected, you'll receive automated email alerts. These thresholds are built into the platform — so even if you're not actively checking the dashboard, you'll still be notified when something needs attention.

How to Sunset a Recipe (Without Breaking Anything)

Sometimes, a Recipe outlives its usefulness. But turning it off too quickly can break downstream systems.

BEFORE YOU DISABLE OR DELETE A RECIPE:

- ☐ Confirm no other system depends on its output
- ☐ Notify any downstream owners (IT, marketing, security)
- ☐ Archive any logs or data that may be needed for audits
- ☐ Comment in the version history with the reason for sunset

Final Review Checklist

Final Recipe Design Confirmation

Use this table to confirm your flow is clear, your logic is sound, and your setup is efficient.

AREA	NOTES
Job to Be Done Do I have a clear statement of what should happen and why? (e.g., "When X occurs, do Y in system Z so that outcome A happens")	
Trigger Clarity Have I chosen the right trigger type and timing? (Polling, webhook, scheduled)	
Steps and Logic Do I know what each step does and when it runs? Are filters, loops, and conditions defined?	
Task Estimate Do I know the rough task cost per run and per month? Have I checked for bulk/loop efficiency?	
Edge Cases Have I identified what could fail, and what should happen when it does?	
Alerts or Logging Do I know where alerts will go, and what will be logged (if anything)?	
Dependencies Are any systems, fields, or teams needed for this to work correctly? Have those been handled?	
Version Notes Have I written down anything a teammate should know when reviewing the Recipe?	