

Serviceanfragen (Workflows)

Aus einer wiederkehrenden Bitte - ein neuer Laptop, ein Onboarding, eine Firewall-Änderung - wird ein Angebot mit festen Schritten. Jeder kann es in einem Formular anfragen, die richtigen Teams bekommen ihren Schritt in der richtigen Reihenfolge, und Automatisierung wie n8n kann mitmachen.

Überblick

Ein **Serviceangebot** (die Vorlage) beschreibt die Schritte, wer sie erledigt, in welcher Reihenfolge und was der Anfragende angeben muss. Eine **Anfrage** ist ein Durchlauf eines Angebots. Sie besteht aus

- einer **Hauptaufgabe**, die die Angaben des Anfragenden trägt und sich schließt, wenn der letzte Schritt erledigt ist, und
- einer **Schrittaufgabe** je Schritt, im Team des Schritts. Schritte ohne Vorgänger sind sofort aktiv; die anderen werden aktiv, wenn der Schritt davor erledigt ist.

Wer	Macht was
Designer (Rolle Workflows oder RC Admin)	Baut die Angebote, legt fest, wer sie starten darf, richtet Trigger ein und beobachtet das Trigger-Dashboard.
Anfragender	Startet eine Anfrage aus einem Angebot, das er nutzen darf, und verfolgt sie unter Meine Serviceanfragen .
Teammitglieder	Bearbeiten ihren Schritt wie jede Aufgabe. Das Erledigen weckt den nächsten Schritt.

1. Ein Angebot entwerfen

Öffne **Serviceanfragen** → **Übersicht** und klicke **Neues Serviceanfrage-Angebot** (du brauchst die Rolle **Workflows**).

- 1 **Name und Beschreibung** sehen die Anfragenden in der Liste. Tags helfen beim Finden.
- 2 **Wer darf es ausführen:** Wähle die Rollen, die das Angebot starten dürfen, oder lass es für alle offen. Die Liste zeigt ein Angebot nur Personen, die es starten dürfen, und der Server prüft es beim Absenden noch einmal. Ein RC Admin darf immer jedes Angebot starten.
- 3 **Eingabefelder der Anfrage** füllt der Anfragende aus: *Text* (optional durch einen regulären Ausdruck eingeschränkt, z. B. eine IP-Adresse), *Zahl*, *Datum* oder *Auswahlliste*. Alle sind Pflicht und werden auf dem Server geprüft, nicht nur im Browser. Der **Name** eines Feldes ist zugleich eine Variable für Trigger-Adressen (siehe unten).
- 4 **Schritte:** Wähle je Schritt das Team, Titel und Beschreibung, die Schätzung und den Schritt, von dem er **abhängt** (keiner = er startet sofort). Die Schätzung bestimmt auch den Fortschrittsbalken der Anfrage.
- 5 **Schrittfelder** (je Schritt): Werte, die die bearbeitende Person ausfüllen muss, bevor sie den Schritt erledigen kann, zum Beispiel eine Ticketnummer oder Seriennummer. Ein Schritt mit leerem Schrittfeld lässt sich nicht erledigen - weder mit dem Erledigt-Knopf noch über die API.
- 6 **Trigger** (je Schritt, optional): eine Adresse, die aufgerufen wird, wenn der Schritt erledigt ist, siehe Abschnitt 5.

Eine Änderung am Angebot ändert laufende Anfragen nicht: Sie behalten die Schritte, mit denen sie gestartet wurden.

2. Eine Anfrage starten

Klicke ein Angebot unter **Serviceanfragen → Übersicht** an. Gib der Anfrage Titel und Beschreibung, wähle die **Lösung** (den Service, für den sie gilt) und fülle die Anfragefelder aus.

Es gilt ganz oder gar nicht: Ist ein Feld leer oder falsch formatiert, erfährst du den Grund und es wurde nichts angelegt. Sonst entstehen die Hauptaufgabe und jede Schrittaufgabe; die ersten Schritte sind aktiv und erscheinen in den Pools ihrer Teams. Jede Aufgabe bekommt den üblichen Eintrag „Aufgabe erstellt“, und Automatisierung wird informiert (Abschnitt 7).

3. Die Schritte bearbeiten

Ein Schritt ist eine normale Aufgabe in seinem Team: Sie lässt sich aus dem Pool nehmen, zuweisen, kommentieren und buchen. Die Angaben des Anfragenden stehen im ersten Kommentar der Hauptaufgabe.

- Fülle die **Schrittfelder** des Schritts aus und klicke dann **Erledigt**.
- Hat der Schritt einen Trigger der Art *Vor dem Erledigen aufrufen*, wird er jetzt aufgerufen; der Schritt gilt nur als erledigt, wenn die Adresse 200 antwortet (Abschnitt 5).
- Die Schritte, die auf diesen gewartet haben, werden aktiv. Wenn der letzte offene Schritt erledigt ist, schließt sich die Hauptaufgabe und die Anfrage ist abgeschlossen.

4. Anfragen verfolgen

Serviceanfragen → **Meine Serviceanfragen** zeigt **eine Zeile je Anfrage** in zwei Tabellen: offen und abgeschlossen. Eine Zeile hat

- Titel, Angebot, Kunde und Anfragenummer,
- einen **Fortschrittsbalken**: den Anteil des **geschätzten Aufwands**, der erledigt ist, nicht der Schritte. Ist der erste von zwei Schritten mit 2 Minuten und der zweite mit 20 Stunden geschätzt, sind nach dem ersten etwa 0 % erreicht. 100 % erscheint erst, wenn alles erledigt ist. Ohne jede Schätzung zählt der Balken Schritte,
- einen Chip je Schritt: erledigt (durchgestrichen), in Arbeit (hervorgehoben, mit der Person) oder wartet auf den Schritt davor. Ein Schritt mit Warnzeichen ⚠ hat einen fehlgeschlagenen Trigger; der Tooltip nennt Grund und Zeit.

5. Schritt-Trigger

Ein Trigger lässt ein anderes System reagieren, wenn ein Schritt erledigt ist: einen Bereitstellungsjob starten, ein Konto anlegen, einen Chat-Kanal informieren. Je Schritt wählst du, **wie** er gesendet wird:

	Vor dem Erledigen aufrufen	Als Webhook senden
--	----------------------------	--------------------

Gesendet als	GET , genau beim Klick auf Erledigt	POST mit JSON-Body, durch den Scheduler (innerhalb etwa einer Minute)
Der Schritt ist erledigt	nur wenn die Adresse innerhalb von 10 Sekunden 200 antwortet; sonst bleibt er offen und die Person sieht den Grund	sofort, egal was der Empfänger tut
Bei Fehlern	keine Wiederholung; erneut auf Erledigt klicken	Wiederholung nach 1 Minute, 5 Minuten, 30 Minuten, 2 Stunden und 12 Stunden (sechs Versuche); danach <i>aufgegeben</i> und die Aufgabe bekommt eine Notiz
Signiert	nein	ja (X-RC-Signature)
Wann nutzen	der nächste Schritt darf nur beginnen, wenn das andere System den Auftrag angenommen hat	das andere System soll nur Bescheid bekommen und darf eine Weile ausfallen

Jeder Schritt, der vor dieser Option existierte, verhält sich wie *Vor dem Erledigen aufrufen*.

Die Adresse und ihre Variablen

Schreibe die Adresse mit den Feldnamen in spitzen Klammern, zum Beispiel

`https://automation.example.com/hook/laptop?size=<size>&ticket=<ticket>`. Beim Auslösen wird jedes `<name>` durch den URL-kodierten Wert dieses Feldes ersetzt: die Anfragefelder des Anfragenden und die bisher ausgefüllten Schrittfelder (der eigene Wert des Schritts gewinnt gegen ein Anfragefeld gleichen Namens).

Die Adresse muss `http` oder `https` sein, ohne Benutzername und Passwort, und - außer die Instanz erlaubt private Adressen (Einstellung `webhook_allow_private` oder Umgebungsvariable `RC_WEBHOOK_ALLOW_PRIVATE=1`, für ein n8n im Intranet) - eine öffentliche. Die Prüfung wird bei der Zustellung wiederholt, und die Verbindung geht an die geprüfte Adresse. Im Modus *Vor dem Erledigen aufrufen* werden bis zu drei Weiterleitungen verfolgt und jeweils geprüft; der Webhook-Modus folgt keiner.

Alles Geheime gehört in die Signatur, nicht in die Adresse: Die Adresse sehen Designer, und ihr Query-String wird nie in Logs oder den Aufgabenverlauf geschrieben (nur Schema, Host und Pfad).

Was der Webhook enthält

```
POST /hook/laptop HTTP/1.1
Content-Type: application/json
X-RC-Event: workflow.step_trigger
X-RC-Delivery: 5f0c...      (bei jeder Wiederholung gleich)
X-RC-Attempt: 1
X-RC-Signature: t=1760437200,v1=9b1f...

{ "id": "5f0c...", "event": "workflow.step_trigger", "created": "2026-10-14T10:20:00+00:00",
  "instance": "https://rc.example.com",
  "data": {
    "id": 13541, "title": "Laptop bestellen", "type": "Service Request", "team_id": 7,
    "workflow": { "request_id": 13538, "template_id": 4, "template": "Neuer Laptop",
                  "kind": "step", "step": 2, "steps": 3,
                  "fields": { "size": "M", "ticket": "INC-7" } },
    "trigger": { "request_id": 13538, "step": 2 } } }
```

Prüfe die Signatur, bevor du dem Body traust. Es ist dasselbe Verfahren wie bei jedem Webhook der Instanz: `HMAC-SHA256` über `t + "." + body` mit dem Trigger-Geheimnis der Instanz, hexadezimal. Weise einen Zeitstempel zurück, der mehr als ein paar Minuten alt ist.

```
# Python
import hmac, hashlib, time
def valid(secret, header, body, tolerance=300):
    parts = dict(p.split("=", 1) for p in header.split(","))
    if abs(time.time() - int(parts["t"])) > tolerance:
        return False
    expected = hmac.new(secret.encode(), (parts["t"] + ".").encode() + body, hashlib.sha256)
    return hmac.compare_digest(expected, parts["v1"])
```

```
// Node
const crypto = require('crypto');
function valid(secret, header, body, tolerance = 300) {
    const p = Object.fromEntries(header.split(',').map(x => x.split('=')));
    if (Math.abs(Date.now() / 1000 - Number(p.t)) > tolerance) return false;
    const expected = crypto.createHmac('sha256', secret).update(p.t + '.' + body).digest('hex');
    return crypto.timingSafeEqual(Buffer.from(expected), Buffer.from(p.v1));
}
```

Das **Geheimnis** steht im Trigger-Dashboard (nächster Abschnitt). Ein RC Admin kann ein neues erzeugen; jeder Empfänger muss es dann erhalten.

Bestätige mit einem beliebigen `2xx`-Status. Jede andere Antwort, keine Antwort innerhalb von 10 Sekunden oder keine Verbindung zählt als fehlgeschlagener Versuch; `410 Gone` gibt sofort auf.

6. Das Trigger-Dashboard

Serviceanfragen → Workflow-Trigger (Rolle **Workflows** oder RC Admin) zeigt, wie die Trigger aller Angebote gelaufen sind.

- **Kacheln** für den gewählten Zeitraum (7, 30 oder 90 Tage, optional ein Angebot, optional nur Fehler): Aufrufe, erfolgreich, fehlgeschlagen, Erfolgsquote, durchschnittliche Dauer und die Zahl der hängenden Schritte.
- **Schritte, die an einem fehlgeschlagenen Trigger hängen**: offene Schritte der Art *Vor dem Erledigen aufrufen*, deren letzter Aufruf fehlgeschlagen ist. **Erneut ausführen** ruft die Adresse auf, ohne den Schritt zu erledigen, so kannst du die Korrektur testen; wenn es klappt, erledige den Schritt.
- **Webhook-Trigger, die noch nicht zugestellt sind**: Zustellungen, die auf den ersten Versuch oder eine Wiederholung warten oder aufgegeben wurden. **Erneut senden** startet eine frische Versuchsreihe.
- **Nach Angebot** und **letzte Aufrufe**: was aufgerufen wurde (ohne Query-String), das Ergebnis, die Zeit, wer erneut ausgeführt hat.
- Das **Signatur-Geheimnis** (verborgen, bis du es anzeigst) und für RC Admins der Knopf für ein neues.

Fehler nennen den **Grund**:

Grund	Bedeutung
die Adresse antwortete 500	Sie wurde erreicht, antwortete aber nicht 200 (Aufruf-Modus) bzw. 2xx (Webhook). Schau im empfangenden System nach.
die Adresse ist nicht erlaubt	Kein http/https, mit Benutzername oder eine private Adresse, obwohl die Instanz solche nicht erlaubt.

keine Antwort innerhalb von 10 Sekunden	Der Empfänger ist zu langsam. Lass ihn zuerst antworten und danach arbeiten.
keine Verbindung möglich	Falscher Host, Port oder falsches Zertifikat, oder das System ist aus.
zu viele Weiterleitungen	Mehr als drei; nimm die endgültige Adresse.

Jeder Aufruf hinterlässt außerdem eine kurze Notiz im Verlauf der Anfrage, und bei einem fehlgeschlagenen *Vor dem Erledigen aufrufen* sieht die Person den Grund auf der Seite, auf der sie Erledigt geklickt hat.

7. Webhook-Ereignisse und n8n

Neben dem Schritt-Trigger tragen auch die normalen Webhooks (**Globale Einstellungen → Externe Dienste → Webhooks**, oder je Team) den Workflow:

Ereignis	Wann
<code>workflow.started</code>	eine Anfrage wurde gestartet (Daten: die Hauptaufgabe)
<code>workflow.step_ready</code>	ein Schritt kann bearbeitet werden: die ersten Schritte beim Start, spätere, wenn der Schritt davor erledigt ist
<code>workflow.completed</code>	der letzte Schritt wurde erledigt und die Hauptaufgabe geschlossen
<code>task.created</code> , <code>task.updated</code> , <code>task.done</code>	für die Hauptaufgabe und jeden Schritt, wie bei jeder Aufgabe

Jedes Aufgaben-Ereignis einer Workflow-Aufgabe trägt den oben gezeigten `workflow`-Block, mit Anfrage- und Schrittfeldern in `fields`. Private Aufgaben verlassen das System nie.

Ein n8n-Flow

- 1 Füge in n8n einen **Webhook**-Knoten hinzu (Methode `POST` für den Webhook-Modus, `GET` für *Vor dem Erledigen aufrufen*), kopiere seine Produktions-URL und trage sie als Trigger-Adresse des Schritts ein. Hänge `?size=<size>` an, wenn du einen Wert in der Query willst.

- 2 Füge einen **Code**-Knoten hinzu, der `X-RC-Signature` mit dem Geheimnis prüft (Ausschnitt oben; `docs/webhooks/n8n-task-done.json` zeigt das Muster für `task.done`).
- 3 Erledige die Arbeit (Konto anlegen, Gerät bestellen, Ticket in einem anderen System öffnen).
- 4 Optional zurückmelden: Trage das Ergebnis mit `PUT /tasks/{id}/workflow-fields` in die Felder des Schritts ein und erledige den Schritt mit `PUT /tasks/{id} ( {"done": 1} )` - Abschnitt 8. Im Webhook-Modus ist der Schritt schon erledigt; nutze das daher für den *nächsten* Schritt, von dem dir ein `workflow.step_ready` -Abonnement erzählt.

8. REST-API und MCP

Aufruf	Was er tut	MCP-Werkzeug
<code>GET /workflows</code>	die Angebote, die der Aufrufer starten darf, mit Anfragefeldern, Schritten und Schrittfeldern (nie eine Trigger-Adresse)	<code>list_workflows</code>
<code>GET /workflows/{id}</code>	ein Angebot	<code>get_workflow</code>
<code>POST /workflows/{id}/start</code>	eine Anfrage starten	<code>start_workflow</code>
<code>GET /tasks/{id}/workflow</code>	Anfrage, Schritt und Felder einer Aufgabe	<code>get_task_workflow</code>
<code>PUT /tasks/{id}/workflow-fields</code>	die Felder eines Schritts füllen	<code>set_task_workflow_fields</code>
<code>PUT /tasks/{id}</code> mit <code>done: 1</code>	einen Schritt erledigen: prüft die Schrittfelder, führt einen Trigger <i>Vor dem Erledigen aufrufen</i> aus, weckt die nächsten Schritte, schließt die Anfrage nach dem letzten	<code>update_task</code>

```
curl -u USER:TOKEN -X POST https://DEINE-INSTANZ/api/v1/workflows/4/start \
-H 'Content-Type: application/json' \
-d '{"title": "Laptop für Max", "solution_id": 3, "fields": {"size": "M"}}'
# 201 {"request_id": 13538, "step_ids": [13539, 13540, 13541], "url": "..."}

curl -u USER:TOKEN -X PUT https://DEINE-INSTANZ/api/v1/tasks/13540/workflow-fields \
-H 'Content-Type: application/json' -d '{"fields": {"ticket": "INC-7"}}'

curl -u USER:TOKEN -X PUT https://DEINE-INSTANZ/api/v1/tasks/13540 \
-H 'Content-Type: application/json' -d '{"done": 1}'
```

Der Token braucht den Umfang **Lesen und Schreiben** für alles außer den beiden **GET** -Aufrufen. Das Erledigen eines Schritts wird mit **400** und dem Grund abgelehnt, solange ein Schrittfeld leer ist oder ein Trigger *Vor dem Erledigen aufrufen* fehlschlägt.

Regeln und Grenzen

- Wer ein Angebot starten darf, bestimmt die Rollenliste des Designers; keine Liste oder „alle“ heißt alle. RC Admins dürfen immer.
- Das Trigger-Dashboard und seine Knöpfe sind für RC Admins und die Rolle Workflows. Nur ein RC Admin kann ein neues Signatur-Geheimnis erzeugen.
- Die Trigger-Adresse eines Schritts zeigt die API nie, und in Logs steht nur Schema, Host und Pfad; der Query-String enthält ausgefüllte Werte.
- Zeilen des Trigger-Logs bleiben 180 Tage, abgeschlossene Zustellungen 30 Tage.
- Webhook-Trigger stellt derselbe Scheduler-Job zu wie die anderen Webhooks (jede Minute); ohne laufenden Scheduler wird nichts gesendet.
- Ein Trigger *Vor dem Erledigen aufrufen* hält den Klick der Person bis zu 10 Sekunden auf. Ist der Empfänger langsam oder unzuverlässig, nutze den Webhook-Modus.
- Aufgaben von Serviceanfragen sind gewöhnliche Aufgaben: private Aufgaben, Berechtigungen und das Änderungsprotokoll gelten wie überall.