Hold Until: smart pauses in journeys

The **Hold Until** step adds a deliberate pause in a journey, waiting for specific user actions or a predefined time limit before progressing. This lets you create highly personalized experiences by responding to user behavior (or the lack thereof) at the right moment.

Because the Hold Until step introduces a checkpoint in your journey where the next action depends on user behavior, it creates opportunities for:

- Personalization, by tailoring user interactions based on their actions.
- Efficiency, helping you avoid sending irrelevant messages by waiting for meaningful triggers.

How Hold Until works

When a journey reaches a Hold Until step:

1. pauses and waits for one of the configured events to occur.
2. the event occurs, the journey moves down the corresponding branch immediately.

3. If no event occurs within the specified time, the journey moves down the default maximum hold duration branch.

Configurable parameters

The following table explains the parameters you can configure for the Hold Until step:

PARAMETER	DETAILS
Branches	Configure up to 4 event branches, each tied to a specific event and optional event property filters. Events must share a unique identifier with the entry event if the journey allows re-entry. Branches must be mutually exclusive to avoid validation errors.
Filters	Event properties refine the triggering conditions for a branch.
Maximum hold duration	The fallback branch activates after the hold period, ranging from 5 minutes to 182 days (about 6 months)

Additional features

The Hold Until step includes optional settings that let you customize how Segment stores and processes events in your journey. These features give you more control over event timing, data inclusion, and journey logic.

Send profiles back to the beginning of this step

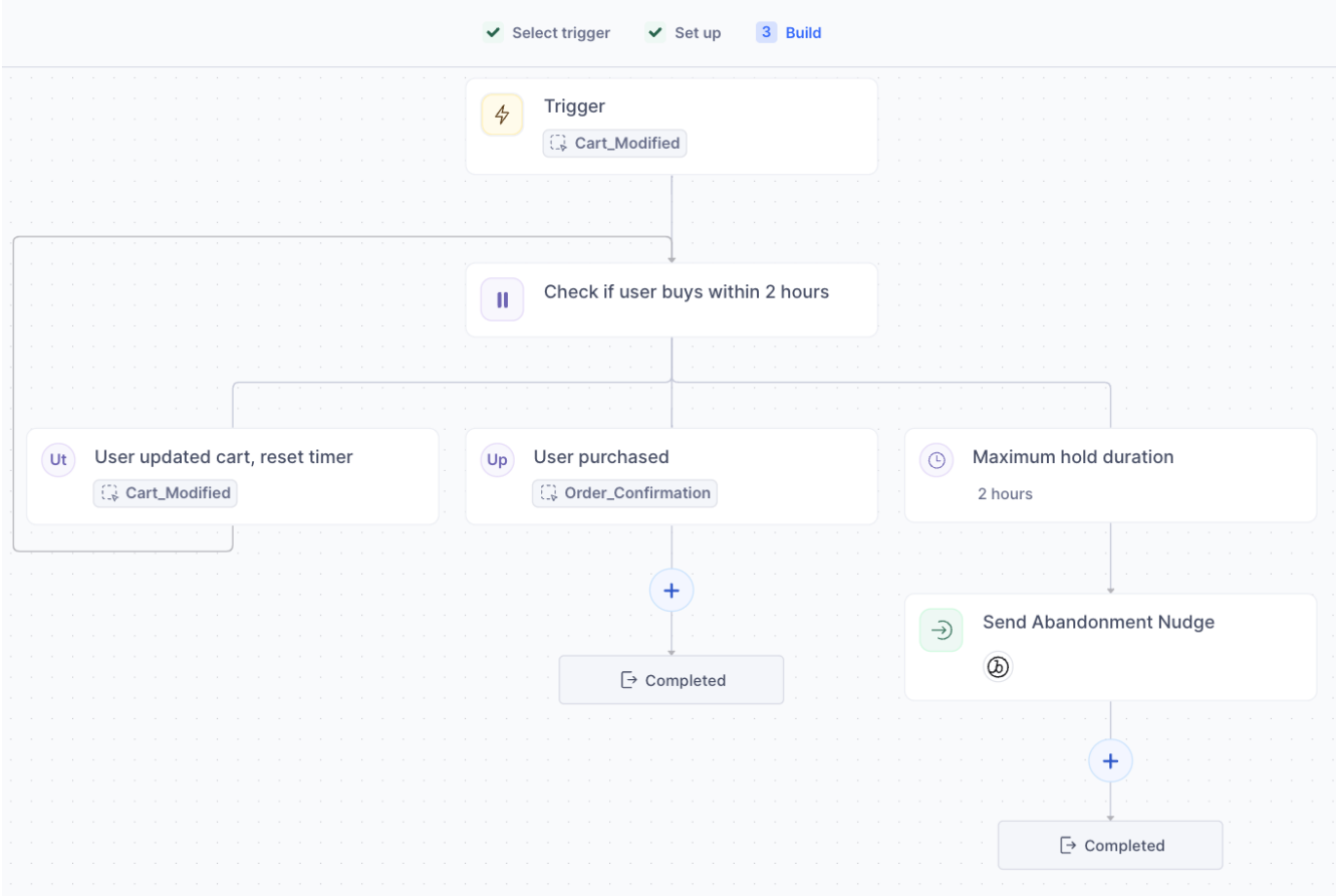
The Hold Until step can restart when a specified event reoccurs. This resets the hold duration and updates the journey context with the most recent event data.

When the same event occurs again, the hold timer resets, and Segment updates the journey context with the latest event data. However, Segment only includes events in the journey context if the profile follows the branch where the event was processed.

For example, in an abandoned cart journey, if a user modifies their cart during the hold period, the cart contents are updated and the two-hour timer resets. This prevents premature follow-ups and keeps the data up-to-date.

Enable this feature by selecting **Send profiles back to the beginning of this step each time this branch event occurs** in the step configuration. For more details about how journey context handles triggering events, see Destination event payload schema.

Segment recommends putting branches for recurring events at the top of the list to improve readability.



In this example, users enter the journey when they modify their cart and wait for either a purchase or two hours to pass. If the user modifies their cart again during those two hours, the cart contents are updated, and the two-hour timer resets. As a result, follow-ups reflect the latest information.

Event name aliases

Event name aliases let you reuse the same event in multiple branches or steps without losing track of data. This approach encourages data clarity and integrity by preserving event-specific context for each branch or step where the alias is applied.

By default, when the same event is triggered multiple times, the most recent event data overwrites earlier occurrences. When you use aliases, though, each branch or step can maintain its own version of the event for more granular control. This is especially useful in journeys that involve repeated events or complex branching logic.

For example, an onboarding journey with a `Signup Completed` event could trigger multiple actions:

- In one branch, the event leads to an email sequence welcoming the user.
- In another branch, the same event triggers a survey request.

As another example, consider the `Cart_Modified` event in an abandoned journey:

1. A user enters the journey by modifying their cart, which triggers the `Cart_Modified` event.
2. During the Hold Until step, the user modifies their cart four more times.

The destination payload after the Hold Until step would look like this:

```
{
  "properties": {
    "journey_context": {
      "Cart_Modified": {
        "organization": "Duff Brewery",
        "compression_ratio": 5.2,
        "output_code": "not_hotdog"
      },
      "Cart_Modified - user updates cart": {
        "organization": "Acme Corp",
        "user_name": "Homer Simpson",
        "output_code": "always_blue"
      }
    }
  }
}
```

In this example:

- `Cart_Modified` captures the properties of the first event that initiated the journey.
- `Cart_Modified - user updates cart` captures the most recent modification within the Hold Until branch.

Segment generates aliases for each instance of an event by concatenating the event name and branch name (for example, `Cart_Modified - user updates cart`, like in the previous payload example). This approach allows both branches to retain the specific event context needed for their respective actions.

Segment creates these aliases automatically during setup, and they show up in the journey context and downstream payloads. While you can't customize alias names, using clear and meaningful branch names helps maintain clarity and precise tracking.

Managing Hold Until steps

Deleting a Hold Until step can impact downstream steps that rely on it. When you delete a configured step, Segment displays a modal that summarizes the potential impact on related branches and steps. Review all dependencies carefully to avoid unintentionally disrupting the journey.

Send to Destination

The **Send to Destination** step lets you send journey data to one of your configured Engage destinations, enabling real-time integration with tools like marketing platforms, analytics systems, or custom endpoints.

This step supports Actions Destinations (excluding list destinations) and destination functions. It doesn't support storage destinations or classic (non-Actions) destinations.

How Send to Destination works

When a journey reaches the Send to Destination step, the journey packages the relevant data and sends it to your chosen destination. This could be a third-party platform, like a marketing tool, or a custom destination built using Destination Functions. The data that Segment sends includes key attributes from the journey context, profile traits, and any mapped fields you've configured.

Configure the Send to Destination step

> **Set a destination up first**
>
> Before you add configure this step, make sure you've already set up the destination(s) in Engage.

Here's how to configure this step within a journey:

1. Select and name the step:
   - Choose the destination for the data.

- (Optional:) Assign a unique name for clarity on the journey canvas.

2. Choose the action:
   - Define the change to trigger in the destination, like updating a record.
   - For Destination Functions, the behavior is defined in the function code, so no action selection is needed.
3. Configure and map the event:
   - Name the event sent to the destination.
   - Add profile traits to include in the payload.
   - View a payload preview to map journey context attributes to destination fields.
   - Test the payload to ensure proper delivery and validation.

Before activating the journey, **send a test event to verify that the payload matches your expectations** and that it reaches the destination successfully.

Destination event payload schema

The events that Segment sends to destinations from Event-Triggered Journeys include an object called `journey_context` within the event's properties. The `journey_context` object contains:

- The event that triggered the journey, unless it was replaced by a new event in a Hold Until step.
- Events received during a Hold Until step, but only if the profile followed the branch where the event happened.
- The properties associated with these events.

You can also optionally include profile traits to provide richer context for the destination.

Here's a detailed example of a payload structure, highlighting the journey context and how Segment enriches event data:

```
{
  "event": "<<YOUR CUSTOM EVENT NAME>>",
  "type": "track",
  "userId": "test-user-67",
  "timestamp": "2025-01-15T02:02:15.908Z",
  "receivedAt": "2025-01-15T02:02:15.908Z",
  "originalTimestamp": "2025-01-15T02:02:15.908Z",
  "context": {
    "personas": {
      "computation_class": "journey_step",
      "computation_id": "journey_name__step_name_8943l",
      "computation_key": "journey_name__step_name_8943l",
      "event_emitter_id": "event_tester_lekqCASsZX",
      "namespace": "spa_w5akhv1XwnGj5j2HVT6NWX",
      "space_id": "spa_w5akhv1XwnGj5j2HVT6NWX"
    }
  },
  "properties": {
    "journey_context": {
      "triggering_event": {
        "organization": "Pied Piper",
        "compression_ratio": 5.2,
        "output_code": "not_hotdog"
      },
      "event_from_hold_until_step": {
        "organization": "Tres Commas",
        "user_name": "Russ Hanneman",
        "output_code": "always_blue"
      }
    },
    "journey_metadata": {
      "journey_id": "2GKsjADZkD",
      "epoch_id": "yiC2qPZNIS"
    },
    "user_name": "Richard Hendricks",
    "coding_style": "tabs_only",
    "pivot_count": 12
  },
  "messageId": "personas_up0crko4htawmo2c9ziyq"
}
```


This example shows how data is structured and enriched with contextual details so that destinations receive the information they need to act effectively.

Managing activations

Activations control the configuration for sending data to destinations, including the destination type, selected action, and mapped attributes. Managing activations allow you to adjust how data flows to a destination without altering the overall journey logic.

Editing activations

You can make updates to an existing activation to align mapped attributes with changes in the downstream schema and add or remove profile traits included in the payload.

To edit or delete an activation, click the destination name in the journey canvas and select the **More** menu. Changes apply only to new journey entries after saving your updates.

Deleting activations

If you delete an activation, future instances of the journey step will fail to send data to that destination. To avoid disruptions, make sure you've configured alternative logic or destinations before removing an activation.

Handling missing attributes

There may be cases where events sent to Segment are missing specific properties or when profile traits are unavailable. How Segment handles these scenarios depends on whether the attribute is explicitly mapped.

If values are not mapped

- When an event property is configured but it's not present in the incoming Track event, that property gets excluded from the payload sent to the destination.

- Similarly, if a trait is configured but isn't present on the profile, the trait gets excluded from the payload.

If values are mapped

- If an event property is mapped but is missing in the Track event, Segment still includes the mapped key in the payload but with a value of `undefined`.
- Similarly, if a mapped trait is missing on the profile, the key is included in the payload with a value of `undefined`.

Carefully configuring mappings and handling missing attributes can help you maintain data integrity and avoid errors in downstream systems.

This page was last modified: 23 Jan 2025

Need support?

Questions? Problems? Need more info? Contact Segment Support for assistance!

Visit our Support page

Help improve these docs!

Edit this page

Request docs change

Was this page helpful?

Yes

No

Get started with Segment

Segment is the easiest way to integrate your websites & mobile apps data to over 300 analytics and growth tools.

Your work e-mail

Request Demo

or

Create free account