Sending Your Data: a Pipelines Overview

Last Modified on 07/16/2026 6:23 pm EDT

Your campaign displays to the right people. Now it's time to decide what happens after someone hits submit. That's the job of Pipelines.

What is a Pipeline?

A Pipeline is the layer between a campaign submission and your ESP or CRM, such as Klaviyo or Iterable. It receives the raw submission data and routes it to the destination through a series of tasks that run in order.

Flow: Form submission → Pipeline (Digioh) → ESP/CRM.

For example, a simple Pipeline might: add the submitter to a Klaviyo list, and fire off a custom event. Each of those is a task, and they run in the order you arrange them.

Pipelines run automatically the moment a campaign gets a submission. You don't have to trigger them manually once they're set up and published.

How tasks work

A Pipeline is built from tasks, and most Pipelines lean on three types:

- **Map Data:** a row-by-row field mapper. It takes values from the submission (or a hardcoded value) and writes them to the field names your ESP expects, for example, mapping form.email to email. This is where most of the configuration work happens.
- **ESP API task (Klaviyo API, Iterable API, etc.):** makes the actual call to your ESP: subscribe to a list, update a profile, track an event. This task is intentionally lightweight, since Map Data already built the payload.
- **Render Template (advanced):** builds a fully custom JSON payload using a templating language like Liquid. Reserve this for ESP requirements that Map Data can't handle on its own, like conditional fields or nested objects.

Any task can also carry a condition, so it only runs when specific criteria are met, for example only running an SMS step if form.phone isn't empty, or routing to a different list based on a quiz answer.

For a step-by-step on adding a task to an existing Pipeline, see How to add a new task to a Pipeline.

Mapping Data: Fields and Named Custom Fields

Every submission passes a structured payload with a few top-level objects you'll reference by dot notation, like form.email, analytics.country_name, prq.results_url, and so on:

- **form:** the visitor's submitted data. What you'll map most often.
- **analytics:** session and visitor data captured automatically (device, browser, UTMs, geo, page URL).
- **attributes:** submission metadata (submission ID, campaign ID, date created), useful for logging

and dedupe.
- **prq:** quiz-specific data (answers, results, a shareable results URL), present only on quiz submissions.
- **Named custom fields:** on quiz submissions, form.named_custom_fields lets you reference an answer by its actual question name instead of a field number. This protects against a mapping breaking if field order ever changes.

Field mapping mistakes (the wrong field mapped to the wrong list or property in your ESP) are the single most common Pipeline problem we see. Getting this right depends on your specific ESP's field names and structure, which is exactly what the ESP-specific guides below walk through in detail.

For the full field reference, see Available Fields in Digioh Pipelines.

Viewing Pipeline Logs

Your account must have Data Storage enabled before logs will be available for pipelines.

When a sync fails, logs are the first place to look. Every Pipeline execution is logged under **Pipeline → Activity → Logs**, and each log shows the full request sent to the ESP and the full response, which is where the actual error usually surfaces.

From a log's detail view, you can also click **Test Pipeline** to re-run that exact submission (even editing the payload first, like changing the email) without waiting for a new live submission.

If the logs are completely empty, the Pipeline was never triggered. Check that it's published, connected to the campaign, and that the campaign's "override campaign integrations" setting isn't skipping it. See How to view pipeline logs for the full walkthrough.

Logs are only retained for 72 hours. If you're troubleshooting an intermittent issue, don't let it sit too long before pulling the log.

Pick the Guide that Matches your ESP

The Map Data setup above is the same everywhere, but the exact fields, connections, and quirks differ by destination. Pick the guide for your platform:

- Connecting a Klaviyo pipeline
- Connecting an Iterable pipeline
- Connecting other ESPs and CRMs

Need help deciding which one applies to you, or troubleshooting a specific sync? Reach out to support@digioh.com.

Connecting a Klaviyo Pipeline

This article walks through connecting Digioh to Klaviyo and building your first Pipeline: one that subscribes campaign Submissions to a Klaviyo list. If you haven't read Sending your data: a pipelines overview yet, start there for the concepts behind Pipelines.

Connect Digioh to Klaviyo

You only need to do this once. Every Klaviyo Pipeline you build afterward can reuse this connection.

If you've already connected Klaviyo in your Digioh account, you can jump directly to creating a pipeline.

1. In Digioh, hover over **Integrations** in the top navigation and select **Integrations**.
2. Scroll to **All Integrations**, search for **Klaviyo**, and select the Klaviyo card.
3. You'll be redirected to the Klaviyo connections page. Click **Add New Connection**.
4. Give it a name, something identifiable, like "Klaviyo - Production." and click **Install**.
5. You'll be redirected to Klaviyo. On the permissions screen, click **Install**, then **Allow**.

That's it. Digioh and Klaviyo are now connected.

Create a Pipeline to subscribe Submissions to a list

1. Go to your **Integrations** page.
2. Click the Klaviyo integration you just connected.
3. Scroll to the Klaviyo Pipeline templates, and click **Create Pipeline** for the **Klaviyo Subscribe to List** template.
4. Name your Pipeline, something like "Subscribe to Newsletter List."
5. Select the Klaviyo connection you set up above.
6. Select the Klaviyo list you want Submissions sent to.
7. Select the Campaign that should use this Pipeline.

> You can only subscribe contacts to a Klaviyo **List**, not a **Segment**. Segments are computed dynamically in Klaviyo and can't receive direct API writes. If you select a Segment ID here, the subscribe call fails. In Klaviyo, confirm the ID you're using under **Lists & Segments** actually belongs to a List.

This same List-vs-Segment distinction matters on the targeting side too. If a Display Rule on your Campaign targets a Klaviyo Segment, that Segment can silently expire. Klaviyo deactivates segments that haven't been used in about 45 days unless you star them. When that happens, the Display Rule breaks and the Campaign can't be published until you fix it. See Conditioning your campaign with display rules for how targeting works, and Fix a Display Rule that references a deleted or deactivated Klaviyo segment if you hit this.

Map your Fields and Check Every Mapping Before you Publish

The Map Data task is where you tell Digioh which Submission fields go to which Klaviyo fields. This is also where almost every Klaviyo Pipeline problem starts, so don't rush it.

1. Open your Pipeline and edit the **Map Profile Fields** (or **Map Data**) step.
2. For each row, set the **Input Field**, the Digioh Submission field, using dot notation like form.email, form.first_name, or form.custom_5 for a custom field.
3. Set the matching **Output Field** in Klaviyo's syntax. Custom profile properties live under attributes.properties.your_field_name, for example attributes.properties.favorite_color.
4. If the Klaviyo field isn't a plain text field (a boolean opt-in, a date, a number), select the matching **Transform** so the value arrives in the format Klaviyo expects. A boolean field needs the BOOL transform, or Klaviyo will create it as plain text instead of a true boolean, which makes it unusable in Flow branching later.
5. To subscribe contacts to a specific list from within a mapping (rather than the list picker above), set that row to **Value** mode instead of **Field**, and enter the list ID mapped to klaviyoListId. Value mode keeps the ID from clearing when you save.
6. Click **Save**, then go back and **Publish** the Pipeline.

Make sure every field your form actually collects has a corresponding mapping. An unmapped field simply never reaches Klaviyo. There's no error, it just silently doesn't arrive.

If you're using a custom field (anything other than form.email) to capture email, turn on email syntax validation for that field in the form builder. A trailing space or typo in that field will make Klaviyo reject the Submission, and the error in the Pipeline log won't obviously point back to the real cause.

Once you're done, run a live test Submission and check Pipeline › Activity to confirm the request and response look right before you consider this Pipeline done.

What's next

That's it. Submissions are flowing into Klaviyo. If you use another ESP or CRM alongside or instead of Klaviyo, see Connecting other ESPs and CRMs. Ready to see how your Campaign is performing? Head to Measuring your results. Need a hand? Reach out to support@digioh.com.

Connecting an Iterable Pipeline

This article walks through connecting Digioh to Iterable and building your first Pipeline: one that sends campaign Submissions into Iterable as a list subscription, profile update, or tracked event. If you haven't read Sending your Data: A Pipelines Overview yet, start there for the concepts behind Pipelines.

If you've already connected Iterable in your Digioh account, you can jump directly to creating a pipeline.

Get a Server-Side API key from Iterable

1. In your Iterable project, go to **Integrations → API Keys**.
2. Click **New API Key**.
3. Choose **Server-Side** as the key type and give it a clear, recognizable name (for example, "Digioh").
4. Click **Create API Key**.

Copy the key immediately. Once you close the window in Iterable, you can't view the full key again. You'd need to generate a new one.

Connect Digioh to Iterable

1. In Digioh, hover over **Integrations** in the top navigation and select **Integrations**.
2. Scroll to **All Integrations**, search for Iterable, and select the Iterable card.
3. You'll be redirected to the Iterable connections page. **Click Add New Connection**.
4. Name the connection and paste in the Server-Side API key from Iterable. Treat this key like a password. Don't share or expose it.
5. Select the identifier your Iterable project uses: email, userId, or hybrid.
6. Click **Save**.

By default, Digioh restricts which Iterable profile fields it can retrieve to email, userId, and the IDs of subscribed lists, message types, and channels. If you need more fields available for targeting or personalization, you can add them in the connection's Include/Exclude

Profile Fields settings.

Create a Pipeline: List, Profile, or Event

An Iterable Pipeline can do one or more of the following: subscribe a user to a static list, add or update their Iterable profile, update their subscriptions, or trigger a custom tracking event.

1. Go to your **Integrations** page and open the Iterable integration you just connected.
2. Scroll to the Iterable Pipeline Templates and choose the template that matches your goal: Subscribe to List, Profile Update, or Track Event.
3. Name your Pipeline.
4. Select the Iterable connection you set up above.
5. If you're using the Track Event template, enter the event name you want sent to Iterable (for example, newUserSignup or Quiz Taken).
6. Select the Campaign that should use this Pipeline.

 The "event triggered" pattern is a popular way to kick off an Iterable Journey, for example by firing a Quiz Taken event instead of just adding someone to a list. All of the Submission's data is included in the event payload, so you can reference it in the Journey's email content, not just use it as a trigger condition.

Map your Fields and Check Every Mapping Before you Publish

1. Open your Pipeline and edit the **Map Profile Fields** step (and, for a Track Event pipeline, the **Map Event Data** step as well).
2. Map the **top-level email** (or userId) field first. This is separate from anything you put inside dataFields, and it's how Iterable identifies the contact. Use form.email, or the correct custom field if your form captures email differently.
3. For each additional field, set the **Input Field** using dot notation (form.first_name, form.custom_5 for a numbered custom field, or form.named_custom_fields.[Question Name] for quiz submissions) and the **Output Field** using Iterable's syntax: custom fields live under dataFields.your_field_name.
4. Apply a **Transform** wherever the Iterable field isn't a plain string. Iterable is strict about types. A date field needs a real date/timestamp, and a boolean field needs the BOOL transform, since form.opt_in otherwise arrives as the plain string "true" rather than a real boolean.
5. **Save** each step, then go back to the Pipeline and **Publish** your changes.

Tips to Keep in Mind:

- A blank top-level email is the single most common Iterable failure. If email only lives inside dataFields and the top-level identifier is empty, Iterable rejects every Submission with "successCount": 0. Confirm the top-level email mapping before you publish, every time.
- Fields that seem to be "missing" in Iterable are almost always a mapping problem, not a bug. Either the field isn't mapped, it's mapped to the wrong output name, or Remove Unmapped Fields stripped it. Double-check the mapping in Digioh before assuming something's broken on Iterable's side.
- If your Iterable project is userId based, you will need to provide a userId mapping, Iterable does not automatically generate userIds – You can use analytics.submission_guid for a unique value.

Once you're done, submit a live test and check **Pipeline › Activity** to confirm the request and response look right.

What's next

That's it. Submissions are flowing into Iterable. Ready to see how your Campaign is performing? Head to Measuring your results. Need a hand? Reach out to support@digioh.com.

Connecting other ESPs and CRMs to Digioh

Connecting a Klaviyo pipeline and Connecting an Iterable pipeline walk through those two platforms step by step. If you use Salesforce, HubSpot, Marketo, or something else entirely, the underlying pattern is identical. Only the field names and API details change.

This article covers that general pattern, then points you to the dedicated setup guide for a few common platforms.

The General Pattern

Every ESP and CRM integration in Digioh follows the same five moves, whether you're sending data to a Pipeline task or a legacy integration:

1. **Connect and authenticate.** Add the platform under **Integrations** and enter its credentials: an API key, OAuth connection, or private app token, depending on the platform.
2. **Create a Pipeline task.** Add a task to your Pipeline for the platform (for example, an HTTP Request task or the platform's dedicated API task) and point it at the right connection.
3. **Map your fields.** Use a Map Data task to send each submission field to the correct field name on the receiving end, for example mapping form.email to the ESP's email property, or a hardcoded list or object ID for the destination list.
4. **Test with a real submission.** Submit your Campaign yourself and confirm the record actually appears on the receiving platform, not just that Digioh shows the Pipeline as connected.
5. **Check the Pipeline logs.** Review **Pipeline → Activity** to see the exact payload Digioh sent and the

response the platform sent back.

> If a Pipeline shows as "connected" but nothing shows up on the receiving end, the cause is almost always step 3, not step 1 or 2. A misconfigured field mapping, or the wrong target list ID or object ID, will let the Pipeline run successfully and log a response, while the record silently lands in the wrong place or gets rejected by the ESP. Before assuming the connection itself is broken, open the Pipeline log for that submission and double-check the mapping and the list or object ID against what's on your ESP's dashboard.

Salesforce

Salesforce connects to Digioh through a Salesforce Connected App, authenticated via OAuth. Once authenticated, you can query and update Salesforce records from a Pipeline. See How to Connect Salesforce CRM to Digioh for the full Connected App setup and authentication steps.

HubSpot

HubSpot integrates with Digioh either through Zapier (Digioh's recommended path for most teams) or through a direct API integration using a HubSpot private app token. See How to Integrate with HubSpot for the direct setup, or the contact upsert variant if you need create-or-update behavior against HubSpot's legacy contacts API.

Marketo

Marketo connects through a Pipeline built from HTTP Request tasks: one for OAuth authentication, one to create or update the lead, and optionally a delay-plus-list task if you need the lead added to a specific Marketo list. See How to Create a Marketo Pipeline for the exact task sequence.

Something else?

Most other ESPs and CRMs (Braze, Yotpo, Attentive, Postscript, and generic webhook destinations) follow the same pattern: connect, map, test, check logs. If a platform isn't listed here or doesn't have a dedicated guide, start from Sending your data: pipelines overview and build the Pipeline using a Map Data task plus an HTTP Request or Render Template task pointed at that platform's API.

Once your data is landing correctly, move on to Measuring your results to see how your Campaign is performing. Stuck on a mapping or an unfamiliar API? Reach out to support@digioh.com.