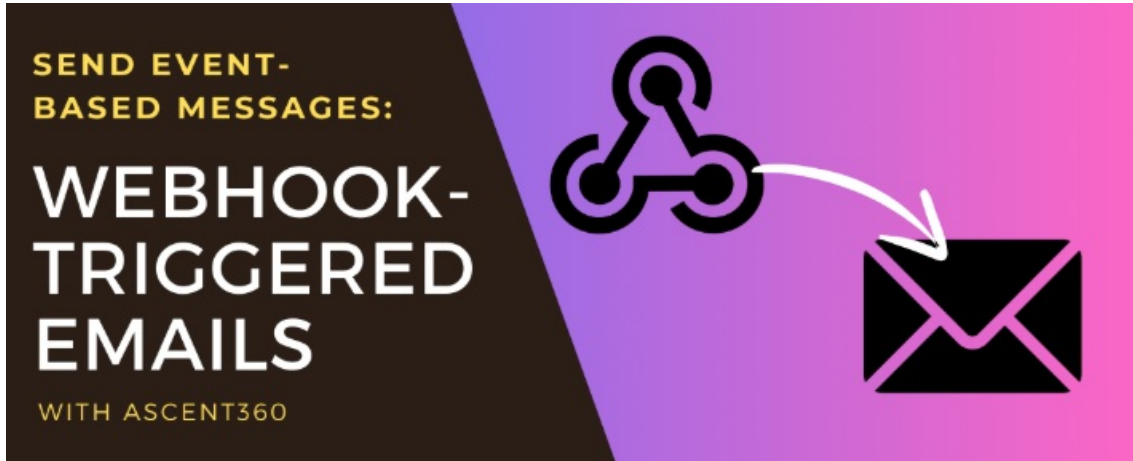


You can now trigger emails through a webhook in Ascent360, allowing you to send **event-based emails** and include dynamic data in the payload. This guide will walk you through the steps to set up a webhook-triggered email.



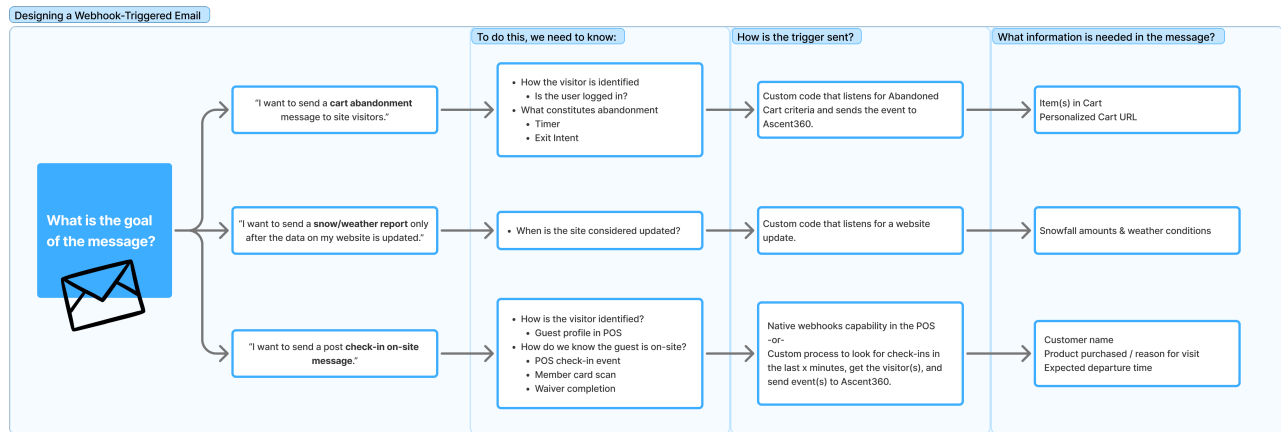
In this article:

- [Why Use a Webhook to Trigger an Email?](#)
- [Prepare your email with merge tags](#)
- [Enable a Webhook Triggered Email](#)
- [Webhook Trigger Examples](#)
- [Advanced Technique: Pass HTML to the Editor](#)

Why Use a Webhook to Trigger an Email?

Triggering an email from a webhook allows you to go beyond the standard email schedule options and gives you granular control over *when* to send the email, *whom* to send it to, and *what* to include in the message. Below are some use cases that may suit your needs for sending a webhook-triggered email through Ascent360.

- Send an email whenever an event happens in an upstream system.
 - Website Update (snow reports)
 - Bookings, transactions, or reservations: creation, cancellation, or updates
 - Form submission (for non-Ascent360 forms)
- Send an email and dynamically replace content in the email with data in the event payload.
 - Recent snowfall and upcoming forecast
 - Work Order updates: Technician name, comments, and estimated completion time
 - Shipping updates: Tracking ID and URL
 - Booking Creation: Confirmation ID, arrival date & time, special instructions



Prepare your email with merge tags

Using a Webhook, you can add personalization to your email and pass-in data in a payload that wouldn't normally be available in the CDP. During the sending process, you'll have an opportunity to map these merge tags to your payload if needed.

Merge tags can be inserted via three methods:

- Use the text formatting toolbar within the email to click **Merge Tags**, then select your CDP Field from the list and designate a default value.
- Use the text formatting toolbar to click **Merge Tags**, then select **Custom** and name your merge tag and provide a default value (if applicable).
- Manually declare a merge tag by typing the merge tag syntax. Examples below:

```
{{[myMergeTag]}}
```

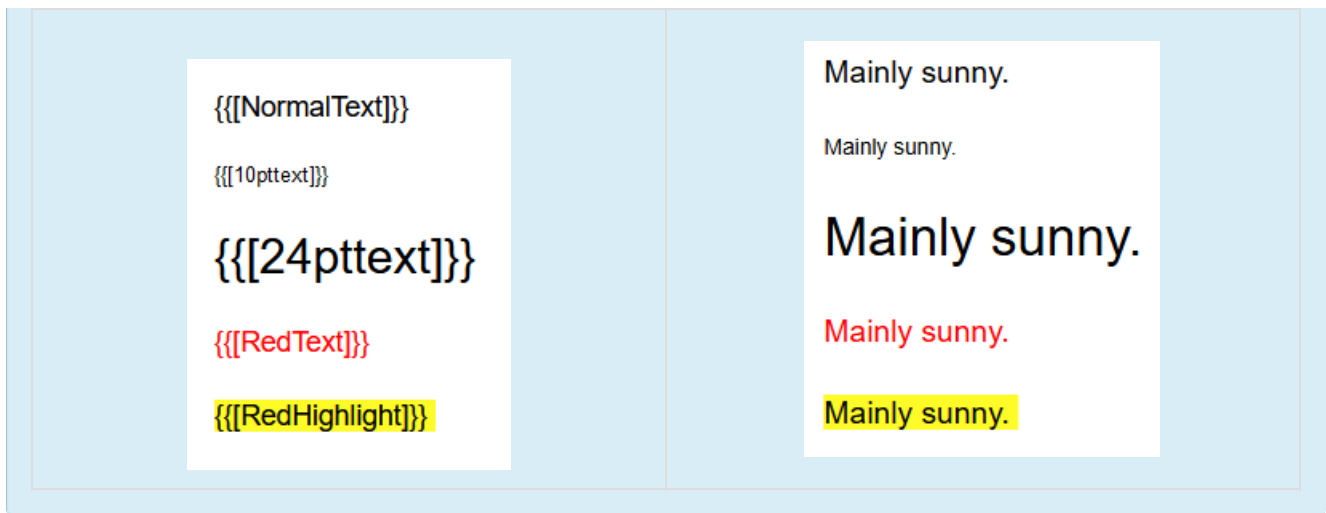
```
{{[myMergeTag|"defaultValue"]}}
```

Custom or manually-declared merge tags should have unique names that are not shared with other fields in your CDP. Your merge tag name should be alphanumeric and contain no spaces.

The Email Send will account for the text styling changes made in the editor, including font style, size, color, highlight, etc.

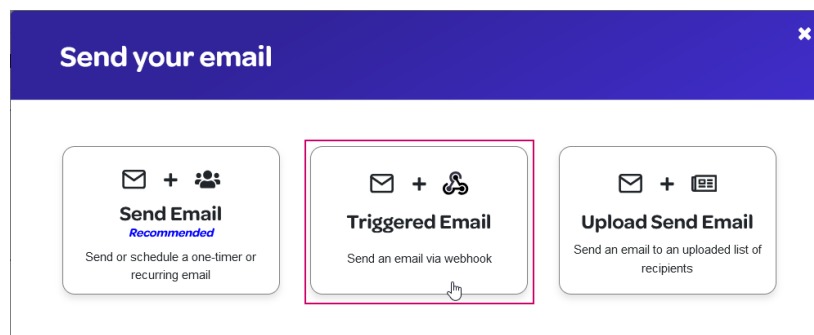
In the Editor:

Live Send:



Enable a Webhook Triggered Email

Once your design is finalized, select **Triggered Email** from the Email Send dialogue.



Send To Section

In the "Send To" section, select whether this email will go to an audience or whether recipients will be provided in the webhook payload.

- **Option 1 - Send to an Audience:** Select a saved audience from the drop-down. Select this option if email recipients will not be provided from the system that is triggering the webhook.
- **Option 2 - Custom Recipient:** Provide a recipient in the webhook payload. Select this option if the email address for the recipient will be provided from the system that is triggering the webhook. **This option supports 1 recipient per webhook trigger.**

IMPORTANT:

- If you send to an audience, Ascent360 will evaluate email permissions before sending the email.
- If you provide the recipient in the payload, Ascent360 will not evaluate CDP email permissions before sending the email. It is your responsibility to ensure that email addresses you are providing are eligible for

email communication through this method.

From Section

Select your From Name, From Address, and Reply To Address for this triggered email.

Properties Section

Configure your email properties, including your Subject, pre-header, and Tracking Parameters

Trigger Section

The Trigger section displays options to view the suggested payload format or provide a payload and mapping.

If you don't already have a predetermined payload that you plan on using, please use our **suggested** format. The suggested format lists the JSON keys & values that Ascent360 will expect when evaluating for recipients and merge tags. Not every key needs to have a value (See Section on Merge Tag Precedence below), but each key in your payload needs to match exactly what is shown (spelling/spacing/capitalization) on this page. Additional keys included in the payload will be ignored or may cause the trigger to fail.

The payload keys/values only apply to merge tags within the email body itself; changing email properties (subject, from name, from email address) via the payload is not supported.

SUGGESTED PAYLOAD

PROVIDE A PAYLOAD

You may use the payload schema below to pass-in merge tags.

```
{
  "Ascent360 ID": "some value",
  "FamName1_": "some value",
  "First Name": "some value",
  "ActivityName": "some value",
  "BookingTime": "some value",
  "ActivityDate": "some value"
}
```

If you aren't able to conform your payload to the suggested format, you can provide a payload and tell Ascent360 how to locate the data that is relevant to the email.

JSON is the only accepted payload format.

NOTE: If you selected the option to send to an audience, sending a payload with the webhook is entirely **optional**. If you have merge tags that you want to dynamically swap, you'll want to provide that data in the payload.

Click **Provide a Payload**, then paste your JSON data. Once pasted, click **Validate & Map** to enter the mapping interface.

In the mapping interface, you'll see a list of your merge tags on the right side (anything that follows the `{{[mergeTag]}}` syntax).

- Within the payload, expand the nodes of the JSON tree to locate the relevant data.
- **Drag** the data from the left side to the corresponding input for that field on the right side. Ascent360 will input the JSON path that will be used to locate the data in subsequent triggers. In order to be dragged, your sample data field needs to have data in it (the interface will not detect nulls).
- Repeat this process for each of the merge tags you need to map.
- You may decide to skip the mapping for CDP Field Merge Tags (See section on **Merge Tag Precedence** below)
- Other data in the payload that is not identified in the mapping will be ignored for future triggers.
- You may drag into already-filled inputs to overwrite the input or click the green check mark next to the data to clear the input.

Map Provided Payload

For each merge tag, drag from the payload to the merge tag to map the value. Ascent360 will reference this path for each trigger.

Payload:

```
bookingId: 107571
bookingSaleDateTime: 2024-12-20 10:44:25
bookingCloseDateTime: 2024-12-23 23:59:59
  bookingRevenue: {} 6 keys
  customers: {} 1 key
  activities: {} 1 key
    activity: {} 24 keys
      activityId: 35
      activityAccountingCode: {} 0 keys
      activityType: Ticket
      packageId: 0
      activityName: Tubing Tickets - Afternoon Session
      activityIsRental: false
      activityIsMultidayRental: false
```

Merge Tags:

- FamName1_ [Select...]
- First Name [Select...]
- ActivityName [Select...]
- BookingTime [Select...]
- ActivityDate [Select...]

Use the Webhook

Once you have decided to follow the suggested schema or provided your own payload example with mapping, close the **Trigger** section and click the **Continue** button in the top right. After confirming, Ascent360 will produce a webhook to be used for this email. This webhook is live and can be used immediately.

Whenever you wish to send this email, send an HTTP POST to the webhook provided along with a payload, if necessary. This can be done manually (using an HTTP request platform like Postman) or the webhook can be called from an automated process in an external system.

Ascent360 has a size limit of 50MB on the JSON payload. If your payload is properly structured JSON and under the size limit, Ascent360 will respond with a 200 OK and the webhook will remain in a "Ready to Process" state for up to 1 minute and then attempt the send.

NOTE: Providing data through the webhook payload is a great way to leverage data that is either 1) not yet in the CDP (due to overnight timing) or 2) won't be added to the CDP (because it isn't named in the

integration/necessary in the CDP).

Data that is passed-in and used in an email through a payload will **not** be formally added to the CDP as part of this process.

For example, if you are sending a booking confirmation from a PMS or Activities system, Ascent360 still needs to be integrated directly with that system to import the data through our standard nightly process. If all you need to do is send the email and you don't need to see data about the booking on the customer record or in Audience Builder, a direct integration is not needed.

If you wish to leverage a webhook-triggered email for confirmations, you should also double-check that you don't have active recurring sends based on the data flowing in overnight.

Test Sends

All webhook-triggered emails support a key called "TestRecipient" that you can use to override the recipient configuration on the email. Whenever "TestRecipient" is provided in the payload, the email will *only* be sent to the email address named in TestRecipient, even if the email is configured to send to an audience or another key/value pair is mapped as the Recipient Email Address. This is especially helpful for configurations that are set to deliver to an audience; by using "TestRecipient", you can leverage the existing configuration without altering or duplicating the setup.

Example where the email recipient is named in the payload:

```
{
  "TestRecipient": "example@gmail.com", -- Email will be sent to ONLY this email address
  "Emailrecipient": "guest@example.com", -- This email will be ignored in favor of the TestRecipient key/value
  "confirmationnumber": "123XYZ", -- This merge tag will evaluate if mapped
  "arrivaldate": "2026-01-12" -- This merge tag will evaluate if mapped
}
```

Example where the email is pointing at an audience:

```
{
  "TestRecipient": "example@gmail.com", -- Email will be sent to ONLY this email address; audience will be ignored.
  "confirmationnumber": "123XYZ", -- This merge tag will evaluate if mapped
  "arrivaldate": "2026-01-12" -- This merge tag will evaluate if mapped
}
```

Merge Tag Precedence

When receiving a POST to the webhook, Ascent360 will insert dynamic data into the email send based on the order of precedence outlined below:

If your triggered email is sending to an audience:

1. Ascent360 will look for mapped data in the payload first and insert into the email. This applies to both custom

declared merge tags and CDP Field Merge tags.

2. For non-custom CDP Fields that are not found in the payload, Ascent360 will try to fill-in the data from the individual's record in the CDP.
3. For merge tags that are not found in the payload or in the CDP, Ascent360 will use the Default Value for that merge tag.
4. If there is no payload data, CDP data, or default value, the merge tag will evaluate to a null and appear as a blank space in the email.

If your triggered email is providing the recipient in the payload:

1. Ascent360 will look for mapped data in the payload first and insert into the email.
2. Sending an email with the recipient in the payload bypasses the CDP (permissions and other data); therefore, all relevant merge tag data for the email should be mapped and provided in the payload.
3. For merge tags that are not found in the payload, Ascent360 will use the Default Value for that merge tag.
4. If there is no payload data or default value, the merge tag will evaluate to a null and appear as a blank space in the email.

View Webhook History and Resend

You may click the 3 Dots on the Triggered email card in the Design Library to view recent request history, view the webhook for the email, and update the mapping.

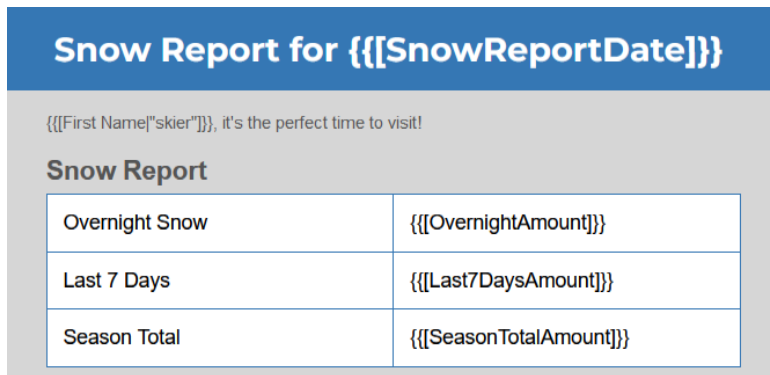
This is also where you may wish to attempt the send again via the **Resend** action. After clicking **Resend**, the webhook will enter the queue to process again using the data originally received. **Ascent360 processes queued webhooks every 60 seconds.**

Mapping updating functionality will be added in a future release. For the time-being please duplicate your email if you need to restructure the mapping.

Webhook Trigger Examples

Send to an Audience Example

Below is an example of sending a Webhook-triggered snow report email to an audience.



- This email will be sent to an audience of individuals who have signed up for Snow Reports. Because we're sending to an audience, the merge tag for First Name does not need to come from the payload data and won't be mapped.

The mapping looks like this (Note First Name is not mapped):

Map Provided Payload

For each merge tag, drag from the payload to the merge tag to map the value. Ascent360 will reference this path for each trigger.

▼ snowreport: {} 5 keys

▼ overnight: {} 4 keys

- title: Overnight
- imperial: 2.8 in
- metric: 7
- display: true

▼ past7Days: {} 4 keys

- title: 7 Days
- imperial: 3.1 in
- metric: 8
- display: true

▼ seasonTotal: {} 4 keys

- title: Cumulative
- imperial: 6 ft 2 in
- metric: 187
- display: true

▼ base: {} 3 keys

First Name

✓ Select...

SnowReportDate

✓ \$.snowreport.timestamp

OvernightAmount

✓ \$.snowreport.overnight.imperial

Last7DaysAmount

✓ \$.snowreport.past7Days.imperial

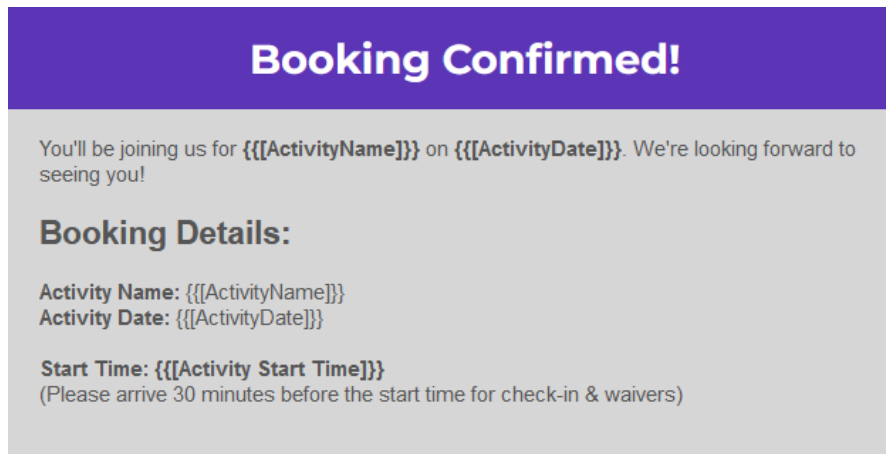
SeasonTotalAmount

✓ \$.snowreport.seasonTotal.imperial

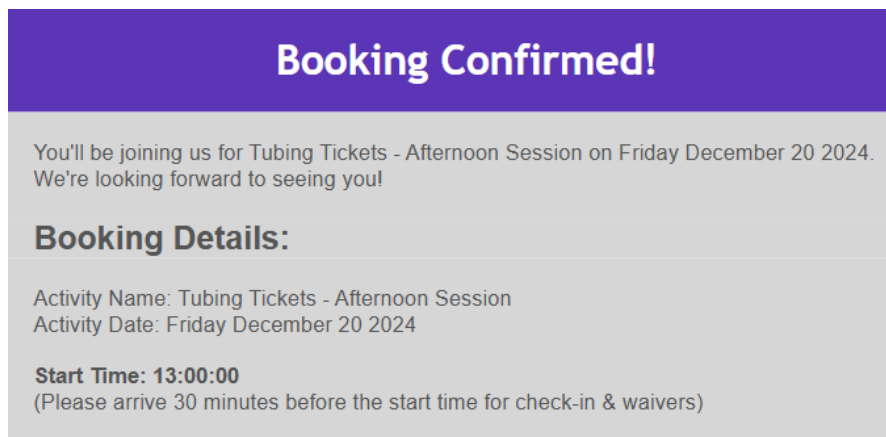
[Finish](#)

Send to a Recipient Example

Below is an example of sending a Webhook-triggered email as a confirmation email for an Activity booking.



After mapping, the resulting email looks like this:



If you have any questions, please reach out to our support team. We'd be happy to help!

Advanced Technique: Pass HTML to the Editor

When receiving the webhook and inserting merge tags, Ascent360 is adding to the existing HTML document for the email. If you want fine-tune control over certain elements of the email (or even the entirety of the email), you may choose to pass in HTML in the JSON. Consider these use cases:

- Include a button that links to a URL that is specific to an individual, like an abandoned cart or a waiver.
- Include an itemized list where the list length varies between recipients (cart contents or receipts).
- Replace the entire contents of the email dynamically.

When you pass in HTML, take note of the following:

- The HTML in the JSON should not include the <HTML> or <Head> tags. Your provided HTML will be wrapped

in these tags.

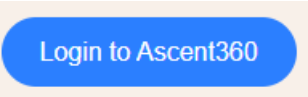
- Use inline styling to achieve element customization.
- Image references should be publicly available.
- The double quotes within the HTML need to be escaped using a \ character before the double quote to be acceptable in JSON format (see examples below)

Example 1: Pass in a button with a dynamic link

Include HTML for button in place of the "button" merge tag.

```
{
  "recipient": "emailaddress@test.com",
  "button": "<a href='\"https://clients.ascent360.com.com/\"' style='\"display: inline-block;\n      padding: 12px 24px;\n      font-size: 16px;\n      color: #ffffff;\n      background-color: #2b7fff;\n      text-decoration: none;\n      border-radius: 25px;\n      font-family: Arial, sans-serif;\">\n    Login to Ascent360\n  </a>\n"
```

Renders in the email as:



Example 2: Itemized list via a table

For consistent formatting of itemized lists, you may wish to pass in an HTML Table.

```
{
  "recipient": "emailaddress@test.com",
  "receiptData": "<table width='\"100%\"' cellpadding='\"10\"' cellspacing='\"0\"' border='\"0\"' style='\"border-collapse: collapse; font-family: Arial, sans-serif; max-width: 600px; margin: auto; background-color: #ffffff; border: 1px solid #dddddd;\">\n    <tr style='\"background-color: #2b7fff; color: #ffffff;\">\n      <th align='\"left\"' style='\"padding: 12px;\">Item</th>\n      <th align='\"center\"' style='\"padding: 12px;\">Quantity</th>\n      <th align='\"right\"' style='\"padding: 12px;\">Price</th>\n    </tr>\n    <tr>\n      <td style='\"border-top: 1px solid #dddddd;\">Single Tube Rental</td>\n      <td align='\"center\"' style='\"border-top: 1px solid #dddddd;\">2</td>\n      <td align='\"right\"' style='\"border-top: 1px solid #dddddd;\">$30.00</td>\n    </tr>\n    <tr>\n      <td style='\"border-top: 1px solid #dddddd;\">Cooler Tube Rental</td>\n      <td align='\"center\"' style='\"border-top: 1px solid #dddddd;\">1</td>\n      <td align='\"right\"' style='\"border-top: 1px solid #dddddd;\">$15.00</td>\n    </tr>\n    <tr>\n      <td style='\"border-top: 1px solid #dddddd;\">Life Jacket Rental</td>\n      <td align='\"center\"' style='\"border-top: 1px solid #dddddd;\">2</td>\n      <td align='\"right\"' style='\"border-top: 1px solid #dddddd;\">$10.00</td>\n    </tr>\n    <tr style='\"font-weight: bold;\">\n      <td colspan='\"2\"' align='\"left\"' style='\"border-top: 2px solid #2b7fff;\">Total</td>\n      <td align='\"right\"' style='\"border-top: 2px solid #007BFF;\">$55.00</td>\n    </tr>\n  </table>"
```

Renders in the email as:

Item	Quantity	Price
Single Tube Rental	2	\$30.00
Cooler Tube Rental	1	\$15.00
Life Jacket Rental	2	\$10.00
Total		\$55.00

Example 3: Replace the entire content of the email dynamically:

Rather than swapping out individual merges tags in the email, you may choose to replace the entirety of the email on-the-fly. To do this, you would put one merge tag in the email design for the rest of the elements to be passed in with each call.

Preview

{{html}}

```
{
  "recipient": "emailaddress@test.com",
  "receiptData": "<body style=\"margin: 0; padding: 0; font-family: Arial, sans-serif; background-color: #f4f4f4;\">\n  <table width=\"100%\" cellpadding=\"0\" cellspacing=\"0\" border=\"0\" style=\"background-color: #f4f4f4; background-image: url('https://d1oco4z2z1fhwp.cloudfront.net/templates/default/9811/15_Background_Palms.png')\">\n    <tr>\n      <td align=\"center\">\n        <table width=\"600\" cellpadding=\"20\" cellspacing=\"0\" border=\"0\" style=\"background-color: #ffffff; margin-top: 20px; border-radius: 8px;\">\n          <tr>\n            <td align=\"center\" style=\"background-color: #007BFF; color: #ffffff; font-size: 24px; font-weight: bold; border-top-left-radius: 8px; border-top-right-radius: 8px;\">\n              How did we do?\n            </td>\n          </tr>\n          <tr>\n            <td style=\"color: #333333; font-size: 16px; line-height: 1.5;\">\n              <p style=\"margin: 0 0 5px 0;\">John, </p>\n              <p style=\"margin: 0 0 10px 0;\">How was your visit? We would love to hear from you on how we can improve.\n            </p>\n          </td>\n          <td align=\"center\" style=\"padding: 20px;\">\n            <a href=\"https://example.com\" style=\"background-color: #f27c85; color: #ffffff; padding: 12px 24px; text-decoration: none; border-radius: 25px; display: inline-block;\">Start Survey</a>\n          </td>\n        </tr>\n      </table>\n    </td>\n  </tr>\n  <tr>\n    <td align=\"center\">\n      <p style=\"margin: 0; line-height: 1.5;\">Cheers,<br><strong>Clear Creek Rentals</strong>\n    </p>\n  </td>\n  </tr>\n</table>\n</body>"
}
```

Renders in the email as:



How did we do?

John,

How was your visit? We would love to hear from you on how we can improve.

[Start Survey](#)

Cheers,
Clear Creek Rentals