Send Form Data from Push Planet to Another Platform

Last Modified on 06/24/2026 4:35 pm EDT

Push Planet landing pages and preference centers can send form submission data to external platforms like Iterable or external APIs.

This guide explains how to use **Push Planet merge tags** to map form field values into a JSON payload and send that data to a webhook or API integration.

You'll learn:

- How Push Planet merge tags work
- The different types of merge tags available
- How to structure JSON payloads using merge tags
- How to test your payload after configuration

Quick Rules for Using Merge Tags

Before building your payload, keep these rules in mind.

Merge tags must be uppercase and wrapped in brackets

[EMAIL][FIRST_NAME][FIELD3]

Merge tag value types follow JSON rules

How you quote the merge tag determines the value type sent.

"weeklyAdOptIn": "[FIELD5]" → sends "true" or "false" (string)

"weeklyAdOptIn": [FIELD5] → sends true or false (boolean)

Field number tags reference fields by position

[FIELD1] through [FIELD20]

If fields are reordered in the form editor, the field number may change.

Geo tags return detected location data

These tags use the visitor's IP address and **do not capture values typed into form fields** .

How Merge Tags Work

When a form is submitted, Push Planet replaces merge tags in your configured JSON payload with the values submitted by the user.

For example:

[EMAIL][FIRST_NAME][FIELD3]

During submission, these tags are replaced with the actual values collected from the form.

JSON Type Behavior

Merge tags follow normal JSON behavior.

If the tag is wrapped in quotes, the value is sent as a **string**.

If the tag is not wrapped in quotes, the value is sent as its **native JSON type**.

Example:

"weeklyAdOptIn": "[FIELD5]" → string value

"weeklyAdOptIn": [FIELD5] → boolean value

Types of Merge Tags

Push Planet supports several types of merge tags depending on the data you want to send.

Named Merge Tags

Named merge tags correspond to standard form field key names.

For these tags to resolve correctly, the form field must use the exact key name shown below.

Field key name	Merge tag
email	[EMAIL]
first_name	[FIRST_NAME]
last_name	[LAST_NAME]
name	[NAME]

If the field key name does not match exactly, the tag will not resolve correctly. In that case, use a **field number tag** instead.

Field Number Tags

Field number tags reference fields based on their position in the form.

[FIELD1] through [FIELD20]

These tags work regardless of the field key name and are recommended for custom fields.

To find a field's number, open the form in the Push Planet editor and locate the number displayed next to each field.

Tip: Field numbers are based on the field's position in the form. If fields are reordered or new fields are added above them, the field number may change. Verify field numbers after making form edits.

Geo-Detected Tags

Geo-detected tags are automatically populated using the visitor's IP-based location data.

These tags **do not capture user-entered form values** .

Tag	Description
[POSTAL_CODE]	Detected postal code
[CITY]	Detected city
[REGION]	Detected region or state
[COUNTRY]	Detected country name
[COUNTRY_CODE]	Detected country code

Important

If the user enters their location manually in a form field, use a field tag such as:

[FIELD3]

instead of a geo-detected tag.

Analytics Tags

Analytics tags are automatically populated by Push Planet and do not require a form field.

Available analytics tags include:

- `[WEB_SOURCE]`
- `[SEARCH_ENGINE]`
- `[DEVICE]`
- `[OPERATING_SYSTEM]`
- `[BROWSER_TYPE]`
- `[BROWSER_VERSION]`
- `[PAGE_VISITS]`
- `[PAST_VISITS]`
- `[REFERRING_URL]`
- `[LANDING_URL]`

- [SUBMIT_URL]
- [SUBMIT_DATE]
- [USER_ID]
- [PAGE_ID]
- [PAGE_NAME]
- [CLIENT_ID]
- [SUBMISSION_ID]

These tags provide contextual information about the visitor, device, or submission source.

Building a Payload

When configuring a webhook or API integration, you can include merge tags directly in your JSON payload.

Push Planet replaces each merge tag with the appropriate value when the form is submitted.

Basic Payload Example

```
{
"email": "[EMAIL]",
"dataFields": {
"firstName": "[FIRST_NAME]",
"lastName": "[LAST_NAME]",
"zipCode": "[FIELD3]",
"homeStore": "[FIELD4]"
}
}
```

Checkbox Example

How you quote the merge tag determines the value type sent to your platform.

Send a string value

Wrap the tag in quotes.

```
{
"email": "[EMAIL]",
"dataFields": {
"weeklyAdOptIn": "[FIELD5]"
}
}
```

Result:

"weeklyAdOptIn": "true"

or

"weeklyAdOptIn": "false"

Send a boolean value

Do **not** wrap the tag in quotes.

```
{
"email": "[EMAIL]",
"dataFields": {
"weeklyAdOptIn": "[FIELD5]"
}
}
```

Result:

"weeklyAdOptIn": true

or

"weeklyAdOptIn": false

Use this format if your destination platform requires a true boolean type.

Nested Object Example

You can structure nested objects inside dataFields if required by your destination API.

```
{
"email": "[EMAIL]",
"dataFields": {
"firstName": "[FIRST_NAME]",
"lastName": "[LAST_NAME]",
"preferences": {
"consent": "[FIELD5]",
"source": "pushPlanet_signup"
}
}
}
```

Static values such as "source": "pushPlanet_signup" can be hardcoded directly.

Hardcoded Values

Values that remain constant across submissions can be written directly in the payload.

Example:

```
{
"email": "[EMAIL]",
"dataFields": {
"firstName": "[FIRST_NAME]",
"source": "pushPlanet_weeklyAd_opt_in"
}
}
```

Quick Reference

Situation	Use
Email field	[EMAIL]
Standard name fields	[FIRST_NAME], [LAST_NAME]
Custom name fields	[FIELD_N]
Any custom form field	[FIELD_N]
User-entered zip code	[FIELD_N]
IP-detected zip code	[POSTAL_CODE]
Checkbox as string	"[FIELD_N]"
Checkbox as a boolean	[FIELD_N]
Static value	Hardcode directly