Webhook mapping: how to translate true/false, 0/1 and Y/N

Deployteq - Suzanne Martens - 2026-09-24 - Comments (0) - Data, Webhooks & API's

What is a boolean?

In the context of a webhook call, a boolean refers to a data type that can have one of two possible values: true or false. It represents a **binary** state or condition.

When a boolean is used in a webhook call, it typically serves to convey information about a certain condition or status.

For example, it might indicate:

- whether a certain action was successful (true) or unsuccessful (false);
- whether a certain condition is met (true) or not met (false);
- or whether a specific setting is enabled (true) or disabled (false).

The issue with booleans in a webhook call, particularly when transmitted as strings (with quotes) instead of actual boolean values (, without quotes, so plain true or false), is related to how Deployteq interprets and processes the data.

Without quotes (actual boolean values):

- When booleans are transmitted without quotes as actual boolean values (true or false), Deployteq can easily interpret and process them.
- This ensures that Deployteq correctly recognizes the boolean values and can perform logical operations or condition checks based on them.

With quotes (string representation):

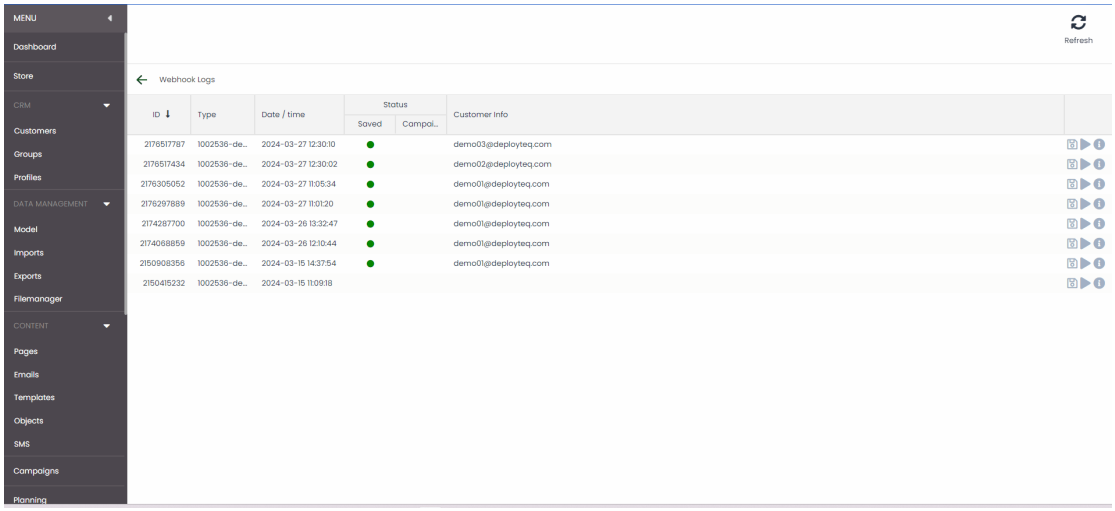
- When booleans are transmitted with quotes as strings ("true" or "false"), they are interpreted as string data rather than boolean values.
- This can lead to issues when Deployteq expects boolean values but receives string representations instead.
- In this case, you will need to add a translation via the translation plugin for these values.
- If this is not done correctly, it can result in unexpected behavior or errors in the processing of the webhook data.

In summary, transmitting booleans as strings (with quotes) can lead to compatibility and processing issues, especially if Deployteq expects actual boolean values.

Note

It's generally recommended to transmit booleans as their actual boolean values (true or false) to ensure proper interpretation and processing by Deployteq.

In the webhook mapping interface, it is **not** possible to see whether the value in a call is a string or a boolean. This can only be seen in the raw data of the webhook call. How you can get there is shown in the video below.



We see in the call information the value **true** for emailOptin. To make sure this is a boolean or a string, we check the raw date of the call. In the raw data, we see that it is an actual boolean value (no quotes).

Translating 0/1 to Y/N

The opt-in fields on customer level expect the values Y (Yes) or N (No). If you do not map an opt-in value, the default value is Y (Yes). This means in this example you will still need to add a translation from 'false' to N and 'true' to Y.

As mentioned , a boolean only has two values, true or false. In computer language, this translates to 1 and 0. Even though the values of 1 and 0 do not show in the raw data, you will need to add them in your translation plugin in the 'from' field.

From' refers to the original values in the webhook data and 'to' refers to the values you want to assign in Deployteq

Please note that in most cases, adding a translation will not be necessary. Deployteq will most likely match the value of true to Y and false to N by itself.

But, there may be other values in your webhook call that Deployteq will not be able to match so easily.

Note

If your mapping does not give the desired outcome during testing, it might be worthwile to check the raw data of your call and see if the problem can be found there!

Warning

Booleans and the Apply Condition Object

When a webhook sends a boolean value (true or false) and that value is stored in a campaign environment, it is internally stored as a numeric value:

- true is stored as 1
- false is stored as 0

For this reason, we recommend always comparing boolean values in an Apply Condition campaign object against 0 or 1, rather than the text values "true" or "false".

What Happens If I Compare Against a String?

Comparing a stored boolean value with an alphanumeric string can lead to unexpected results. In some cases, a comparison against a string may return a match even though the stored value is actually false (0).

To avoid unexpected behavior, always treat webhook booleans as numeric values and compare them using 0 or 1 in your conditions.