The Journey to SMS Conversations

Author: Kaila Corrington

Claps: 134

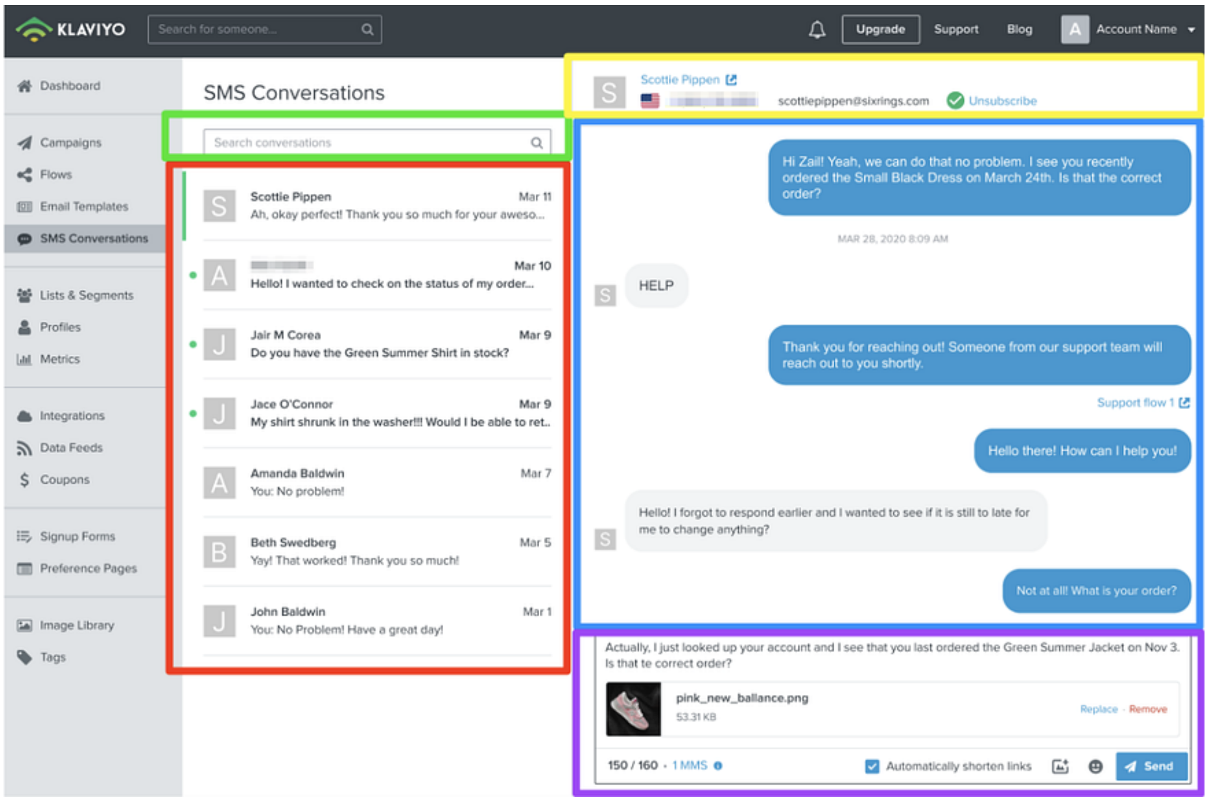
Date: Nov 18, 2021

Klaviyoâ€™s first Product Event went live to the world on March 30, 2021. It was an exciting time for the entire company, but it marked something special for the SMS engineering team because it was the first time Klaviyoâ€™s full SMS product offering would be announced. Part of the feature set being unveiled included SMS Conversations, which unlocks the ability to engage in one-on-one communication with your audience via text messaging. This post will explore how we brought SMS Conversations to life from a technical perspective, creating as seamless a UI/UX experience as possible while also building a solid foundation from which direct messaging can be extended beyond SMS in the future.

Understanding the Problem Space

Klaviyo has long offered the ability to send targeted messaging to audience segments using email, SMS, and push notifications as messaging channels. The key distinction was that all pre-existing sending within the application assumed a one-to-many relationship between messages and recipients. For example, you could create a flow that would send a templatized message to customers prompting them to return to an abandoned cart, or you could create a campaign that would target your VIP list to let them know of an upcoming sale. Advancing from this concept of one-to-many to one-to-one messaging feels like a very natural next step from a product offering perspective, but it was different enough from our existing messaging mechanisms and paradigms that there were a variety of full stack technical challenges under the hood.

To enable the functionality of SMS Conversations within Klaviyo, we would be creating a new page that displayed all existing conversation threads and offered users the ability to directly respond, much like iOS Messages or Facebook Messenger.



Scottie Pippen Mar 11
Ah, okay perfect! Thank you so much for your aweso...

Mar 10
Hello! I wanted to check on the status of my order...

Jair M Corea Mar 9
Do you have the Green Summer Shirt in stock?

Jace O'Connor Mar 9
My shirt shrunk in the washer!!! Would I be able to ret...

Amanda Baldwin Mar 7
You: No problem!

Beth Swedberg Mar 5
Yay! That worked! Thank you so much!

John Baldwin Mar 1
You: No Problem! Have a great day!

scottiepippen@sixrings.com

Hi Zail! Yeah, we can do that no problem. I see you recently ordered the Small Black Dress on March 24th. Is that the correct order?

MAR 28, 2020 8:09 AM

HELP

Thank you for reaching out! Someone from our support team will reach out to you shortly.

Hello there! How can I help you!

Hello! I forgot to respond earlier and I wanted to see if it is still to late for me to change anything?

Not at all! What is your order?

Actually, I just looked up your account and I see that you last ordered the Green Summer Jacket on Nov 3. Is that te correct order?

pink_new_ballance.png
53.31 KB

150 / 160 · 1 MMS

Early design mocks with annotations from planning.

Conversations would exist for any customer that had sent the brand an inbound message that did not match a recognized keyword (e.g. `START` or `STOP` for opting into or out of marketing communications). Opening a conversation thread would reveal the history of all SMS messages between a customer and the brand, including any direct conversation communications as well as any messages from flows or campaigns. For legal reasons, accounts would only be able to send to customers that had consented to receive text messages.

The good news: much of this infrastructure and these paradigms were already built into the Klaviyo application. After all, we already had mechanisms for building and sending outbound SMS messages, processing inbound messages and recording them on a customer profile, and preventing sending to customers that had not provided explicit consent.

The challenge: almost all of said infrastructure and paradigms were built assuming that outbound SMS messages were sent within the context of a flow or campaign, which inherently had a one-to-many relationship between message and recipients. Conversations are one-to-one in a way that truly didnâ€™t exist anywhere in the Klaviyo application â€¦ yet. Additionally, we wanted to make sure that any technical decisions we made would not negatively impact the ability to extend conversations beyond just SMS in the future.

Armed with the feature specification document from our Product team and plenty of wireframes from Design, we chose to start our journey by analyzing the boundaries where the frontend and backend intersected: what API interactions would be required to bring this page to life? For creating an initial minimum viable product (MVP), what immediately jumped out to us was:

- Retrieving the most recent conversation threads and displaying data about each associated customer
- Marking conversations as read upon opening a thread
- Loading the most recent messages to display in the message window upon opening a thread
- Sending a message

Designing the Models

After identifying at a high level what data weâ€™d need to send back and forth over the wire, we took a look at how we would actually model and store the data. Much of the information we wanted to display already existed in our system and there was no reason to reinvent the wheel (at least not for the MVP). On a per customer basis, there were existing interfaces for accessing the following data from MySQL tables across several different AWS RDS databases:

- Customer profile details (e.g. full name, profile photo)
- Consent information (e.g. whether the customer had actively consented to receive SMS messages)
- SMS events (e.g. inbound SMS messages from the customer to the brand as well as outbound SMS messages from campaigns and flows).

Weâ€™d need to augment this data with new records pertaining to the conversations themselves. We decided to create two new models:

- `SMSConversation`
- `SMSConversationMessage`

`SMSConversation` stores records indicating that a conversation exists for a particular customer. Along with net new information like a conversationâ€™s unread or blocked status, the overall conversation record also summarizes the conversation by storing data such as the latest message body and timestamp at which the message was received or sent. This summary information is denormalized from other sources to optimize loading and sorting conversation threads without needing to query all of a companyâ€™s SMS events.

`SMSConversationMessage` stores records of outbound conversational messages sent to customers via the SMS Conversations UI. Because we wanted to be able to make use of the sending and event processing pipeline utilities already in place for email and SMS campaigns and flows, we analyzed the existing class hierarchies for messages in our system and made some adjustments to ensure that we were inheriting as much behavior as would be useful but not any more than necessary. We coordinated with other teams to migrate several methods and properties that lived on the existing `BaseMessage` base class into more relevant concrete classes and also abstracted common logic that existed on our flow and campaign SMS models into a `BaseSMSMixin` that could be shared with the new `SMSConversationMessage` type to provide a common means for storing fields such as metadata about media attachments.