https://www.twilio.com/blog/building-the-twilio-api-for-whatsapp?utm_campaign=&utm_medium=email&utm_source=drip&utm_content=GRTH_NRTR_2018-SEP-12_WhatsApp_EM1A&utm_term=WW

Abstraction and Orchestration — How We Built the Twilio API for WhatsApp

Businesses want to be where their customers are and popular messaging apps are those places today. *Facebook Messenger*, *WeChat*, *LINE*, *Viber* and other popular messaging apps have already rolled out APIs that businesses can use today. But until very recently, there was one notable holdout—*WhatsApp*.

More than 1.5 billion people globally use WhatsApp everyday to talk to friends and family as well as for work and collaboration, and in August this year WhatsApp launched the highly anticipated WhatsApp Business API. Businesses have been clamoring for an officially supported way to communicate with WhatsApp users for *years* now. The excitement is more than justified.

Twilio provides one Messaging API that developers can use to easily integrate with a variety of channels. In addition to SMS and MMS, it supports Facebook Messenger, LINE, RCS and now, WhatsApp.

WhatsApp is a unique messaging platform with end-to-end encryption between clients. Central servers exist purely for orchestration, but otherwise communication is completely peer-to-peer. This means integrating with the WhatsApp Business API presented a set of interesting challenges for the Twilio team building the product. This post provides a behind-the-scenes look at how we built our solution. The Twilio API for WhatsApp abstracts away the complexity of directly integrating with the WhatsApp network. Twilio developers can use the same API they use for Programmable Messaging for WhatsApp, allowing them to focus on their apps and not be bothered by the plumbing.

Abstraction and Normalization in the Messaging API

A critical part of the Programmable Messaging API is a clever integration gateway utilizing multiple abstraction and normalization layers. Rather than requiring one-off services for each new channel, the gateway has evolved into a powerful and flexible integration engine. After transforming the data, messages sent using the Programmable Messaging API are dynamically routed to the messaging channel's APIs that reside externally on central servers.

Requests to Twilio's Programmable Messaging API become outbound tasks in the pipeline, and can include the following items:

- Text
- Structured templates and values
- URLs
- Media (photos, videos, etc.)
- Conversation context
- "From" and "To" identifiers
- Dates
- Segmentation metadata (breaking a single message into a multiple, sequential messages)
- Callback URLs

Since Twilio provides a single API to send messages through any of our channels, each message needs to be translated into the target channel API's expected request. Items that need to be considered include:

- JSON vs XML vs form data (text/template, URLs, media delivery, conversation context, dates, callback selection, and more)
- Authentication (OAuth, JWT, ephemeral vs long-lived tokens, simple API keys, and others)
- Channel-specific identifiers (Facebook Page ID, LINE Group ID, etc.)

The channel API responses and callbacks must also be translated back into Twilio concepts as well. For inbound messages, Programmable Messaging must plug itself into each channel's delivery mechanism (frequently webhooks), parse the payload, and add the message to our inbound pipeline.

Reusable Dynamic Components

Our gateway abstracts most of these needs into dynamic and reusable concepts, primarily driven by structured specifications and configuration—meaning we *don't* have to build one-

off microservices full of boilerplate code for each integration. With this setup, we're generally able to get new channels into developers' hands quickly. For example, completely integrating with LINE (another popular messaging platform) only required a few days work by a single Twilio engineer.

Adding WhatsApp was partially a similar story. The WhatsApp Business API itself is fairly standard and straightforward. Although there were a few bits requiring enhancements to our gateway, those improvements will benefit the Programmable Messaging platform we're building as a whole. For example, WhatsApp requires businesses to use pre-registered templates for outbound messages if more than 24 hours have passed since the user last responded. This concept was new for Programmable Messaging, and we're now planning to offer the feature with other channels as well. The more comprehensive our middleware layer becomes, the more we're able to decrease the time it takes to add more channels in the future.

However, RunOps is a **unique** challenge with the WhatsApp Business integration.

WhatsApp Business API Infrastructure and Orchestration

Unlike most other messaging platforms, WhatsApp was designed as an encrypted, purely peer-to-peer communication network. That end-to-end encryption requires all communication to flow through a "device" with a unique phone number, with each side using keys to encrypt and decrypt the payload. WhatsApp servers are purely there to help route the messages, remaining unable to decrypt or parse them in any way.

Rather than hitting a central API, WhatsApp Business provides a set of Docker images that must be installed in-house (or in AWS) and actively maintained. These containers emulate a "device" for use on the peer-to-peer network. Our gateway can then hit the WhatsApp Business API, exposed as web apps through the containers.

Each phone number's containers can be run using single, high availability, and/or "multiconnect" (load balanced) configuration. A typical enterprise setup can result in 10+ containers per number! Every time you need a new number on WhatsApp –for example, when you expand to a new country– you'll need to set up another set of containers.

Creating and Maintaining WhatsApp Containers

Each container must be registered and verified, which requires several steps:

- Create a Facebook app.
- Request access to WhatsApp Business API and wait for Facebook's approval.
- Register the phone number through the Facebook app dashboard.
- Submit all message templates for approval.
- Download the phone number's encoded WhatsApp certificate.
- Boot all required Docker containers, through Docker Compose or setup shell scripts provided by WhatsApp.
- Use the web container's API or admin dashboard to:
  - Create necessary admin accounts.
  - Generate an API auth token.
  - Register the phone number using the downloaded certificate.
  - Verify each number's registration using an SMS or voice call sent to the number by WhatsApp.
  - Provide all business info, app info, logos, and other information.

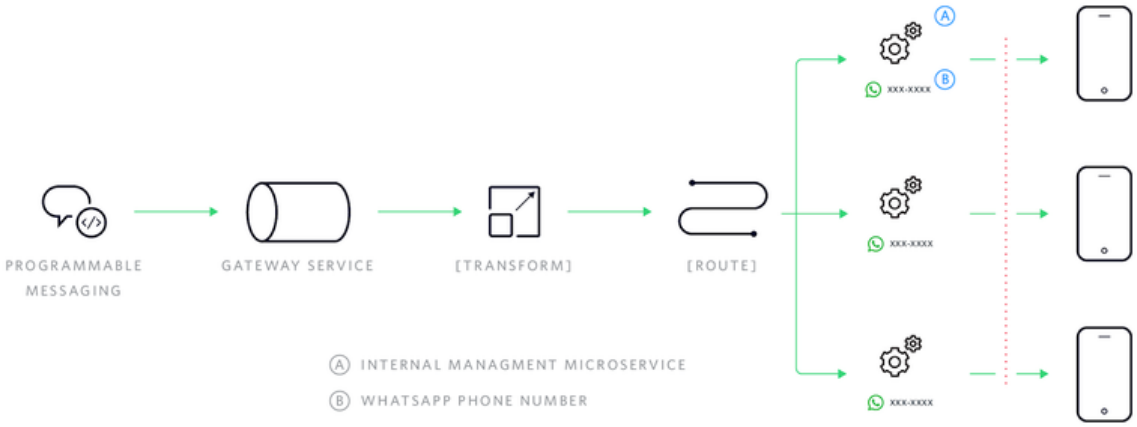
Further compounding the challenge, each container requires active, ongoing maintenance:

- WhatsApp utilizes an aggressive API version deprecation and removal schedule.
- Frequent upgrades are necessary, as is in-depth regression testing.
- Auth tokens are not intended to be ephemeral, so they must be generated through the API and securely stored.

- Those auth tokens expire every 7 days, so they must be regenerated and re-persisted regularly.

Creating the infrastructure and orchestration necessary for running a fleet of containers can be a lot of upfront and ongoing work. The registration process is a long list of manual steps, which must be completed for each phone number. For busy teams and critical apps, this is yet another scope of responsibilities with risks and issues to worry about.

WhatsApp as a Service



The Twilio API for WhatsApp abstracts away the need to build and run the infrastructure for managing these containers at scale. It also gives you access to Facebook Messenger, LINE, RCS, and Programmable Messaging APIs so that you can add more channels to your application by simply changing two lines of code.

You get to reuse all the features within the Twilio platform that you would otherwise need to implement yourself:

- **Message templates and value interpolation:** Outbound messages to customers must utilize templates registered and approved by WhatsApp if more than 24 hours have elapsed since the customer's last inbound message.
- **Inbound webhooks and outbound callbacks:** Asynchronous integration with an external platform typically requires you to build and host a web API that can receive a variety of callback messages. For WhatsApp, that includes consuming incoming messages, as well as asynchronous status callbacks to track delivery of outbound messages.
- **Durable queueing, retry, and transactional logic:** When issues occur while communicating with the WhatsApp API (immediately, mid-stream, during status callbacks, etc.), message delivery still needs to be guaranteed. Developing the necessary resiliency is both vital and tricky.

- **Rate limiting:** WhatsApp currently requires applications to manage their own rate limits, otherwise the API will start returning 429 error responses. Limits are currently 20 requests per second with a 60 request per second burst.
- Logging, tracking, and monitoring of message events, responses, callbacks, errors, and more.

And finally, you get WhatsApp Business Messaging as a fully-managed service:

- Twilio automatically boots, provisions, monitors, and upgrades the appropriate containers for each WhatsApp phone number. This utilizes existing infrastructure tooling that we already use to manage our large fleet of microservices for all Twilio products. You can rely on a hardened RunOps infrastructure to handle all facets of the container management.
- Each number and its set of containers are scaled independently and dynamically.
- Automatic auth token regeneration and persistence is included. As discussed above, WhatsApp's non-ephemerial tokens expire each week. We ensure tokens are automatically regenerated before they expire to prevent your integration from going hard-down.
- An engineering team is on deck for monitoring WhatsApp Business API versions, digging into changelogs, managing upgrades, handling necessary changes, and overseeing the fleet of containers for your WhatsApp numbers.
- The Twilio Sandbox for WhatsApp provides you with a set of pre-provisioned phone numbers and templates that can be used to start testing immediately while you wait for a dedicated WhatsApp Business Profile.

What's Next for Twilio and WhatsApp

We're thrilled to be partnering with WhatsApp to bring the WhatsApp Business API into the hands of developers and businesses all around the world. We're working closely with the team at WhatsApp to iron out kinks and iterate on the architecture as the API continues to evolve rapidly. The container-based "device" architecture is unlikely to go away any time soon due to the unique encryption environment. We're investing in improving the reliability and scalability of our own container orchestration services, streamlining the developer experience, and enabling support for new API features like sending and receiving multimedia messages.

In terms of reach and ubiquity, WhatsApp is currently only surpassed by SMS. At Twilio, we've seen developers build novel experiences and solve difficult real-world problems when given the tools to effortlessly embed SMS into their software. Now that WhatsApp is finally programmable, you have a more powerful and versatile tool available to tackle these problems. We can't wait to see what you build next!