

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

This guide is intended to provide an overview to healthcare organizations that are exploring the *Appointment Management with EHR Integration [Quick Deploy App](#)*, of the intent of the application, how it works, and initial installation steps for getting the application deployed as a prototype into a private environment. More granular implementation and customization details can be found in the application's [EHR Integration Guide](#) and README file (which is accessible after deployment).

This app is intended for prototyping and testing purposes only. Not for use in production environments!

Application Overview

The Appointment Management with EHR Integration app packages together the core components of a deployable prototype for basic, two-way SMS communication between patient and provider using appointment information that is shared between the application and an integrated Electronic Health Record (EHR). This app aims to support healthcare provider organizations interested in building their own appointment management solution to understand what is possible using Twilio, and to accelerate the path to success by providing core building blocks and necessary workflows. This application is not intended to be a production-ready app, but rather will allow you to install a functioning prototype into your test environment, establish a sandbox EHR integration, and to explore how different Twilio components and functions can be leveraged to meet your needs.

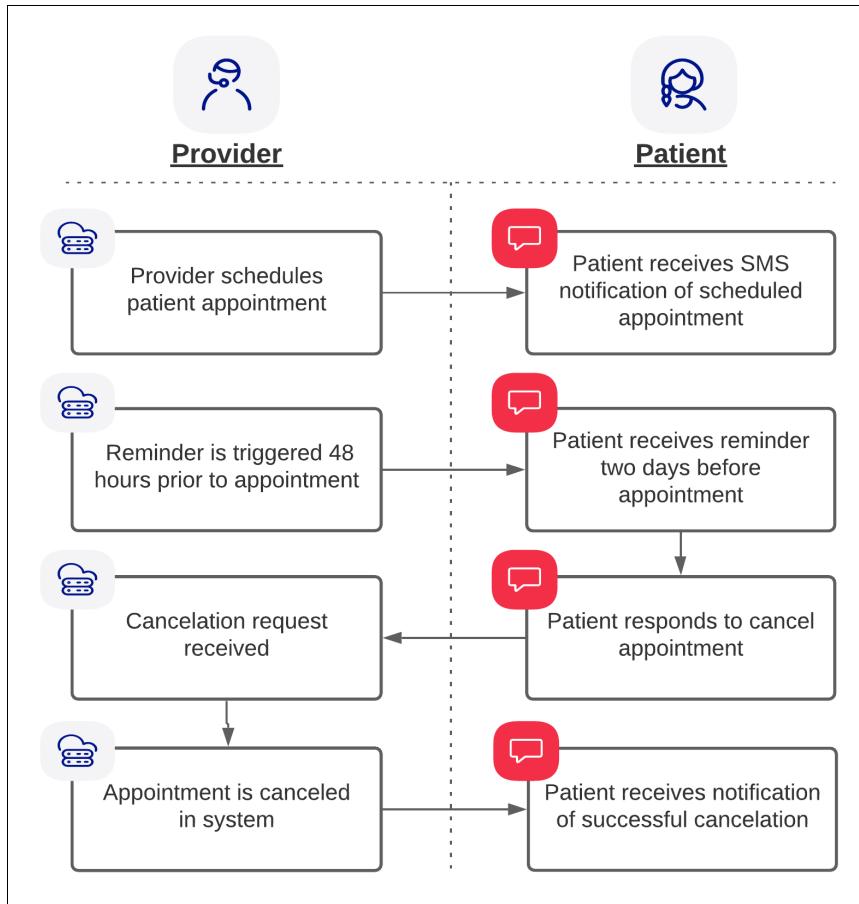
Customers with that are subject to the Health Insurance Portability and Accountability Act (HIPAA) and intend to utilize Twilio's products and services to develop communication workflows containing protected health information (PHI) must execute a Business Associate Addendum (BAA) to [Twilio's Terms of Service](#). Twilio considers HIPAA compliance as a shared responsibility between the customer and Twilio. To learn more about how to build a HIPAA compliant workflow using Twilio's offerings, please refer to our guide on [Architecting for HIPAA on Twilio](#).

The application includes the necessary Twilio components and a cloud-based scheduling service (using Amazon Web Services), all pre-configured for the code-free deployment of a working prototype, ready to integrate directly with your sandbox EHR for demonstration of the possibilities. The separate step of EHR integration is required for the app to work. See the [EHR Integration Guide](#) for more information, including details on appointment events supported by this application.

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Specifically, the Appointment Management with EHR Integration application implements the following capabilities:



(Outbound) SMS notifications sent to patients based on appointment events occurring in the EHR:

- booking/scheduling
- rescheduling (date/time changes)
- modification (location and/or provider)
- confirmation
- cancellation
- noshow

(Inbound) SMS response sent from patient "to the provider" (*technically, to the EHR*)

- confirmation request
- cancellation request

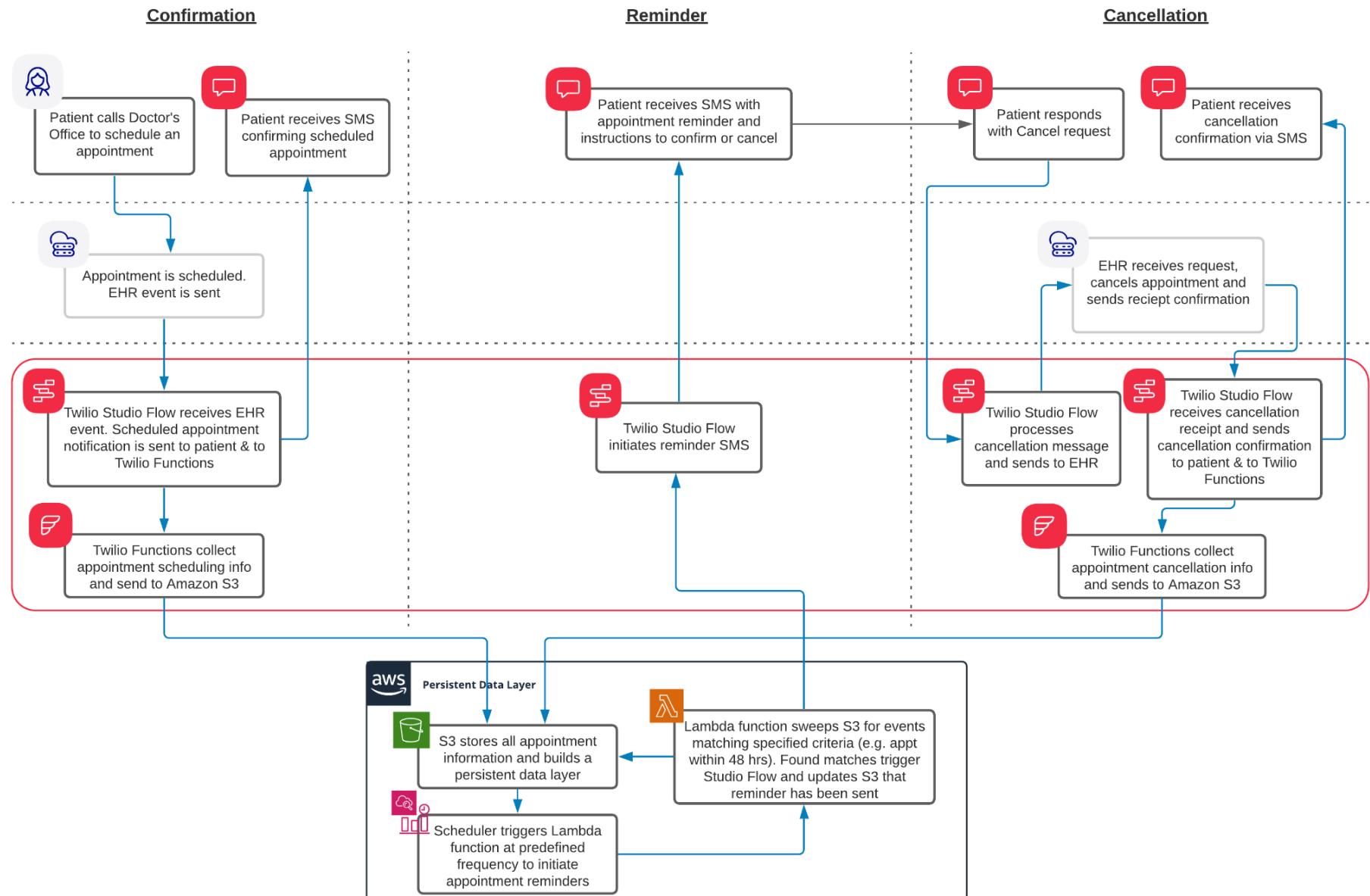
(Outbound) SMS reminders sent to patient based on scheduled appointments (up to 2 reminders per patient per day)

Architecture Highlights

This section provides a high-level overview of the application's architecture, including a discussion of the baked-in application components, the EHR integration that is necessary for the app to function, and a Reference Architecture diagram showing how the pieces fit together.

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App



Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Application Components

The application's architecture consists of 3 main components that interact closely together: Twilio Studio Flow, Twilio Functions and AWS resources.

- **Twilio Studio Flow** implements the SMS interaction between the patient and provider organization (i.e. customizable message text and workflow) by taking configured parameters (from both the EHR events and from your preferred message details configured in the Flow itself) and sending appropriate messages.
- **Twilio Functions** collect appointment events from the EHR and store them in AWS S3 (for appointment reminders).
- **Amazon Web Services (AWS) Resources** are used to build a persistent data layer which drives the appointment reminder messages. The AWS layer serves as the source of truth for scheduling notifications that are not triggered by real-time events from the EHR, and to queue the appointment information for initiating scheduled reminders.

The AWS layer consists of:

- an **S3** bucket for storing up-to-date appointment information
- **Lambda** function to identify triggering events and communicate with Studio Flow, and
- A **scheduler** to trigger the appropriately timed appointment reminders

(Although the baked-in architecture leverages AWS offerings for the persistent data layer, the application can be modified to use other cloud services providers, such as Microsoft Azure and GCP (Google Cloud Platform), that offer similar capabilities)

EHR Integration

This application is intended to sit next to your EHR, and will rely on a near real-time EHR integration interface coupled with the application's components, in order to function. As long as your EHR integration interface can facilitate the real-time data exchange with the EHR, the app can integrate with a variety of integration methods including HL7 v2 messaging, FHIR, native EHR APIs, or available third-party integration APIs. Once scheduling messages are received by Twilio from your EHR, they are converted into JSON to complete the information flow through Twilio and AWS.

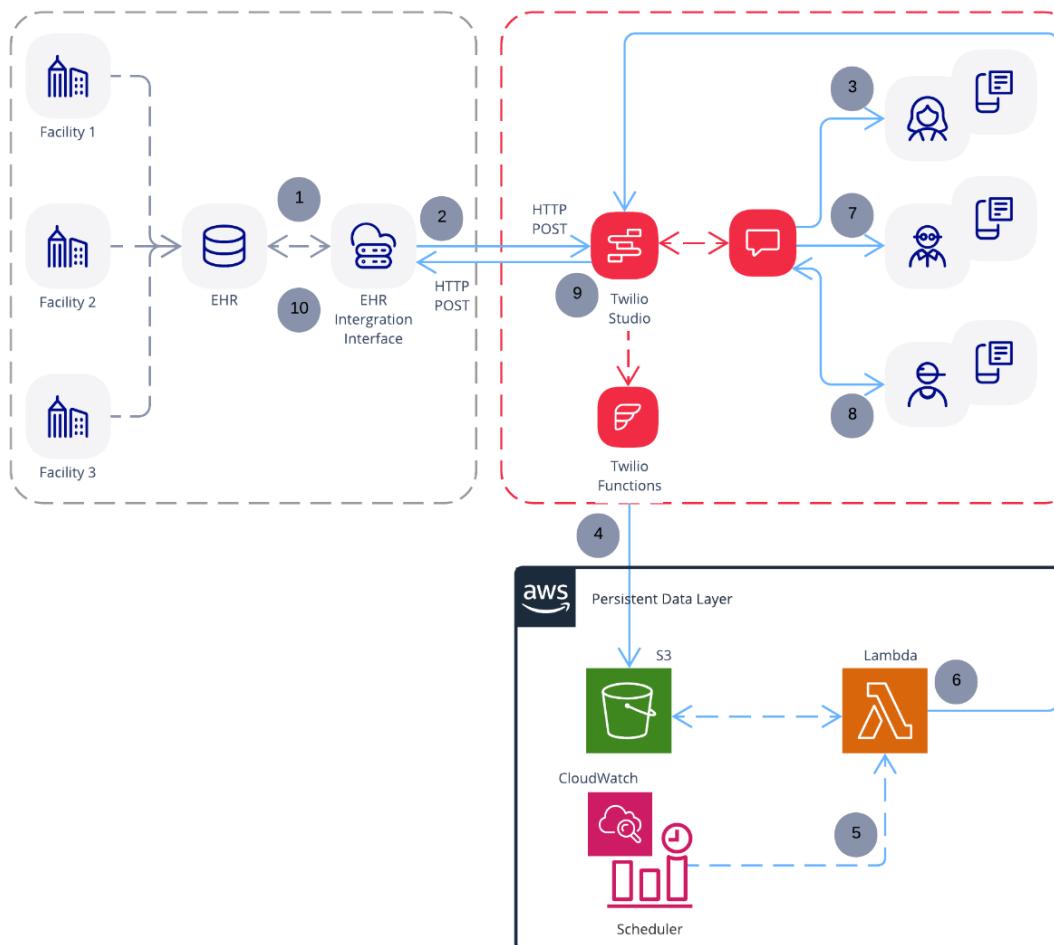
The [**EHR Integration Guide**](#) provides detailed information on how to configure integration with your EHR, and provides details of the appointment events supported by this application.

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Reference Architecture

The Reference Architecture diagram below shows a more detailed description of the process and data flow between the application components and an integrated EHR.



- 1 EHR integration interface serves as a single point of interface between all departments or facilities under an EHR instance. If there are multiple EHR instances, it is possible to route all EHR instances through a single integration engine provider.
- 2 EHR integration interface processes appointment-related events in the EHR and makes an HTTP POST call to a Twilio Studio Flow.
- 3 For real-time notifications, the Twilio Studio Flow takes configured parameters from the HTTP POST call and sends an SMS message to the patient.
- 4 Twilio Functions collects all appointment events and stores them in S3 to build a persistent data layer of all appointment information. This serves as a source of truth for scheduling additional notifications that are not triggered by a real-time event in EHR.
- 5 For scheduled notifications, the scheduler triggers a Lambda function at a pre-defined frequency.
- 6 The Lambda function sweeps through the data in S3 to identify configured triggering events (such as an appointment existing within 48 hours). When a match for the configured criteria is found, it calls the Twilio Studio Flow.
- 7 Twilio Studio Flow takes configured parameters from the Lambda function (patient name, phone number, date/time of appointment, location, etc.) and sends an SMS message to the patient.
- 8 For two-way notifications, patient responses are collected through Studio.
- 9 The response triggers Twilio Studio Flow to a) send a message back to the EHR integration interface via HTTP POST call and b) collect the response via Functions to S3 to populate the appointment database.
- 10 The EHR integration interface takes the patient response and executes the appropriate action in the EHR.

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Installation and Configuration

This section details the requirements for a successful deployment and installation of the prototype application, including the necessary prerequisite steps, the environment variables that are needed to initiate installation, the installation steps themselves, and some information about the separate step of EHR integration.

This app is intended for prototyping and testing purposes only. Not for use in production environments!

This Quick Deploy App is not a generally available product and should not be fully deployed in a production environment. The Quick Deploy App (including all code and related documentation) is provided “AS IS.” Twilio disclaims all express or implied warranties of any kind with respect to the Quick Deploy App, including but not limited to any implied warranties of merchantability or fitness for a particular purpose. Twilio shall have no liability or obligation to you or any other individual or entity for any damages of any kind or nature whatsoever arising out of or relating to the use of or inability to use the Quick Deploy App, including but not limited to any direct, indirect, incidental, consequential, or special damages, even if Twilio has been advised of the possibility of such damages. Twilio has no obligation to support or maintain the Quick Deploy App. Use of the Quick Deploy App is subject to all of the terms and conditions of the applicable [license agreement](#).

Prerequisites

The following prerequisites must be satisfied prior to installing the application.

Provision Twilio Assets

You will need the following Twilio resources ready prior to installation:

- **Twilio account**
 - Create a Twilio account by signing up [here](#)
 - (*You will use your Twilio login information to get started with the Quick Deploy installation on the app's CodeExchange page*)
 - If you have multiple Twilio Projects under your account, make sure that you are logged into the Project that you want the app to be deployed to
- **Twilio phone number**
 - After provisioning your Twilio account, you will need to [purchase a phone number](#) to use in the application
 - Make sure the phone number is SMS enabled
 - (*This will be the phone number patients receive texts from*)

**Note: authentication is required in order to complete deployment via the application page, which will generate a nominal SMS charge to your connected Twilio account. Each authentication SMS sent will cost \$0.0075, plus an additional \$0.05 per successful authentication (multi-factor authentication is leveraging [Twilio Verify](#)). If you leverage the application SMS testing option, SMS charges will also apply (this is an optional capability after successful deployment). See [Twilio SMS pricing](#) and [Twilio Verify pricing](#) for more information.*

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Prepare AWS Assets

You will need the following AWS assets ready prior to installation:

- **Create AWS Account** - you will need to create a dedicated AWS account for this application deployment
 - As admin-level privilege will be required to create various AWS resources, we **strongly** recommend that you create a dedicated AWS account separate from other AWS accounts that your organization owns
 - You may place the new AWS account [within your AWS Organizations](#) for consolidated billing, if desired (not required)
- **Create AWS deployer user and role** - in order to deploy the application's AWS components, you will need a deployer user and role
 - Use this [link](#) to create your AWS deployer user and role through CloudFormation Quick Create
 - **Note:** this runs a CloudFormation template file to create the CloudFormation stack that creates the deployer user and role. If desired, you can inspect this file [here](#)
 - Select “I acknowledge that AWS CloudFormation might create IAM resources with custom names” and then select “Create Stack”
 - Once the stack is created successfully, you will be taken to the CloudFormation console page that shows the newly created ‘twilio-patient-appointment-management-deployer’ stack
 - **Note:** This [link](#) can be used to check whether the deployer user and role have been created successfully
- **Take note of AWS Deployer Key ID and Access Key** - once the deployer user and role have been created, a key ID and Access Key will be available, which are required for app deployment from the Twilio Code Exchange page
 - Select this [link](#) to be navigated to your Secrets Manager
 - Scroll down to “Secret value” section and select “Retrieve Secret Value”
 - Take note of the Key ID and Access Key. These variables will be required for application deployment

Ensure unique application name

In order to deploy correctly, it is important that you **do not** have an existing Twilio Studio Flow or Function Service called “patient-appointment-management”. If you do, you will need to delete (or appropriately update the existing name of) those application components to ensure a conflict doesn’t occur during installation.

Environment Variables

The following environment variables are required for proper deployment of this application (you will use these environment variables in the app's initial CodeExchange page prior to deployment). After the application is deployed, you can find these environment variables in your Twilio function console's ‘Environment Variable’ section.

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Variable	Description	Required
TWILIO_PHONE_NUMBER	The Twilio phone number you want to use for sending and receiving SMS through the app	Yes
APPLICATION_PASSWORD	The Password used to restrict access to sensitive data (this password will be required to access and manipulate the application after deployment)	Yes
ADMINISTRATOR_PHONE	The phone number that will be used to receive the multi-factor authentication SMS during application authentication	Yes
CUSTOMER_NAME	The organization name which can be configured to appear in the SMS to the patient	Yes
CUSTOMER_CODE	The organization short name, which will be suffixed to AWS resources; must be all lowercase with no spaces and no special characters	Yes
REMINDER_OUTREACH_START	<p>The <i>start time</i> of the outreach window in which SMS can be sent to patients for appointment reminders (the outreach window can be used, for example, to ensure SMS are not sent to patients at unreasonable times of day)</p> <ul style="list-style-type: none"> Configure as HHMM This variable is <i>inclusive</i> (meaning the outreach window will be later than <i>or equal to</i> the HHMM you configure here) Default is 0000 (i.e. midnight) The app will honor the patient's local timezone 	Yes
REMINDER_OUTREACH_FINISH	<p>The <i>end time</i> of the outreach window in which SMS can be sent to patients for appointment reminders (the outreach window can be used, for example, to ensure SMS are not sent to patients at unreasonable times of day)</p> <ul style="list-style-type: none"> Configure as HHMM This variable is <i>exclusive</i> (meaning the outreach window will be before, <i>but not equal to</i>, the HHMM you configure here) Default is 2400 (i.e. midnight) The app will honor the patient's local timezone 	Yes
REMINDER_FIRST_TIMING	<p>The hours/minutes prior to appointment time in which the <i>first</i> reminder should be sent to the patient</p> <ul style="list-style-type: none"> Configure as HHMM Default is 4800 	Yes

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Variable	Description	Required
REMINDER_SECOND_TIMING	The hours/minutes prior to appointment time in which the <i>second</i> reminder should be sent to the patient <ul style="list-style-type: none"> Configure as HHMM Default is 2400 Second reminder can be turned off by setting this variable to 0000 	Yes
DEPLOYER_AWS_ACCESS_KEY_ID	The AWS_ACCESS_KEY_ID of the deployer user created in pre-requisite steps	Yes
DEPLOYER_AWS_SECRET_ACCESS_KEY	The AWS_SECRET_ACCESS_KEY of the deployer user created in pre-requisite steps	Yes

Installation Steps

(Installation of this application is supported on the latest versions of Chrome and Firefox. Installation via Internet Explorer has not been officially tested and although issues are not expected, unforeseen problems may occur)

- Ensure completed prerequisites** - ensure that you have completed all prerequisite steps listed above and have your Environment Variables on hand.
- Input Quick Deploy information** - navigate to the application's [CodeExchange page](#) and fill in all information required for steps 1 - 3 in the 'Quick Deploy to Twilio' section, using your Environment Variables in step 2.
 - you can change these variables after deployment thru Twilio Studio Functions
- Deploy the application** - select 'Deploy my application' in Step 3 to deploy the app to Twilio Functions.

Step 3: Deploy your application with a click

When you click "Deploy my application", we will automatically use your custom details above to deploy your app to Twilio Functions. You can view the application in the browser and edit it to fit your needs using the Functions UI.

[Deploy my application](#)

 This might take up to a minute

- Navigate to the application page** - once the initial deployment is complete, Step 4 will appear. Select 'Go to live application' to be taken to the application page for additional app setup and configuration.

Step 4: View and edit your application

[Go to live application](#)

 Once deployed, any changes to the fields above will not be applied to your app. To make changes, here are two options

 Refresh the page, edit customizations and re-deploy the application

 Edit your customizations directly in code

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

5. **Access the application** - From the application page, you will be prompted to enter your application password prior to further setup and validation of successful deployment. This is the APPLICATION_PASSWORD Environment Variable which you established in a previous step. When you enter the correct password, a verification message with a code will be sent to the number configured as the ADMINISTRATOR_PHONE Environment Variable in order to ensure authorized access to the application (the app's multi-factor authentication uses [Twilio Verify](#)). Enter the code into the prompt and you will be authenticated successfully. If you refresh the application page at any point in the following steps you will need to re-authenticate your application password in order to proceed.

Your password was accepted. For additional security, please enter the security code we sent to your phone.

VERIFY SECURITY CODE

✓ Authenticated successfully

After successful authentication, you should see 'Validated Environment Variables' if all environment variables were configured appropriately.

✓ Validated Environment Variables

If you are instead prompted to correct any environment variables, selecting the 'Edit this application' button will open your Twilio Console, where you can edit the environment variables. Once you correct your issue and save, refresh your application page to ensure the environment variables have been validated

6. **Deploy Studio Flow** - follow the prompt to deploy the app's Studio Flow. You will know this step was successful when you refresh the page and see a checkmark next to the step. From here you can check out the pre-configured Studio Flow in your Twilio Console by selecting 'Open Studio Flow'.

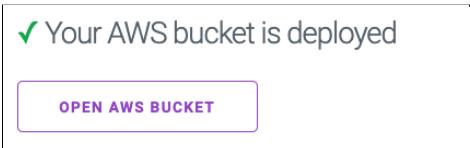
✓ Your Studio Flow is deployed

OPEN STUDIO FLOW

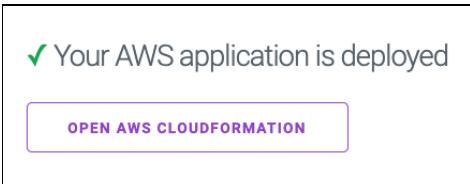
Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

7. **Deploy AWS Bucket** - follow the prompt to deploy your AWS bucket. You will know this step was successful when you refresh the page and see a checkmark next to the step.

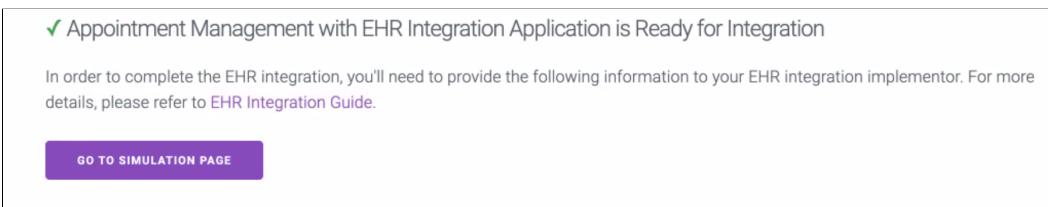


8. **Deploy AWS Application** - follow the prompt to deploy your AWS application. You will know the step was successful when you refresh the page and see a checkmark next to the step



Application Testing

Congratulations! At this point you have successfully completed the installation of the Appointment Management with EHR Integration Quick Deploy App. In order to test that everything is working appropriately and to see the app in action prior to integration with your EHR, you can use the Simulation Page to simulate EHR events.



Although the app supports additional event types as well, the BOOKED and REMIND simulations will allow you to test that messages are properly flowing between the primary architecture components - Twilio, AWS, outbound "to the EHR". In the simulations, hard-coded event messages will be used in lieu of live messages that will ultimately come from your EHR. Every other step of the simulation will use the built-in app functionality.

Simulation Prep

You will need to provide the Name and Number of "the patient" to whom you would like simulation messages to be sent. All other message variables used in the simulation come from the Environment Variables you configured during application deployment, are automatically generated, or are hard-coded for testing purposes. Appointment details that are typically shared from the EHR (such as appointment date, time, facility and provider) will be pre-populated for the simulation and are not editable. You can see more details about these variables in the Event Simulation section of the app page.

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

Messages sent to ⓘ Name: <input type="text" value="First Name"/> ⓘ Number: <input type="text" value="+14085551234"/>	Messages sent from ⓘ Name: <input type="text" value="Lark Health6"/> ⓘ Number: <input type="text" value="+13109452973"/>	Other Variables ⓘ Appointment Date/Time: <input type="text" value="Mon Jan 19 1970 12:16:31 PM"/> ⓘ Provider: <input type="text" value="Dr.Francis Diaz"/> ⓘ Location: <input type="text" value="Pacific Primary Care"/>
---	---	--

Simulation Steps

- To simulate an **appointment booking** message to "the patient", click the SIMULATE APPOINTMENT BOOKING button. This will create an appointment for 24 hours in the future, trigger an SMS notification, and store the appointment information in AWS (simulating steps 2-4 in the [Reference Architecture](#)). The application is configured out-of-the-box to have a 2 minute wait time before appointment reminders can be sent, so you need to let the countdown expire before the reminder option is available.

SIMULATE APPOINTMENT BOOKING

Your appointment request has been sent. Please wait 119 seconds to simulate a reminder.

- To simulate an **appointment reminder** message to "the patient", click the SIMULATE APPOINTMENT REMINDER button. This will trigger the AWS scheduler to check for appointments that are within the next 24 hours and send a reminder SMS (simulating steps 5-7 in the [Reference Architecture](#)).

SIMULATE APPOINTMENT BOOKING

SIMULATE APPOINTMENT REMINDER

*Note: The event simulation does not currently include the ability to test bi-directional messaging (from patient to provider), therefore responding to either the appointment booking or reminder message will not be fruitful.

EHR Integration and Application Customization

Your next task is to integrate the prototype application with your sandbox EHR environment. Detailed information about the EHR integration process can be found in our [EHR Integration Guide](#). Information about further configuration options can be found in the application's README file.

✓ Patient Appointment Management Application is Ready for Integration

In order to complete the EHR integration, you'll need to provide the following information to your EHR integration implementor.

Implementation Guide

Appointment Management with EHR Integration Quick Deploy App

CHANGE LOG

11/02/2021	Added page numbers, updated Architecture Highlights diagram
9/16/2021	Updated details in Installation and Configuration (regarding newly added use of Twilio Verify)
8/10/2021	Updated details in Installation and Configuration; addition of Application Testing section
7/22/2021	First Release: EHR Appointment Management_Implementation Guide



Twilio powers the future of business communications, enabling phones, VoIP, and messaging to be embedded into web, desktop, and mobile software. We take care of the messy telecom hardware and expose a globally available cloud API that developers can interact with to build intelligent and complex communications systems.