Best practices for authenticating users via WhatsApp | Developer Documentation

Best practices for authenticating users via WhatsApp

Updated: Nov 21, 2025

Security

To register with WhatsApp, users must register using their phone number. During this sign-up process, WhatsApp verifies the user has ownership of this phone number by sending a 6-digit registration code via SMS or phone call.

For many WhatsApp users, their phone number will continue to be the same as the number they have registered with WhatsApp. However, WhatsApp does not enforce ownership of the phone number past initial registration, so there is no guarantee that a phone number and the WhatsApp account tied to that phone number are owned by the same individual. In particular, since phone numbers are recycled by mobile providers, it is possible that if your user currently owns the phone number and does not use WhatsApp, the previous owner of that phone number still has access to the WhatsApp account tied to that phone number.

As such, for sensitive authentication use cases such as account recovery (where a code sent via WhatsApp may be the only authentication factor), a phone number and the WhatsApp account tied to that phone number should not be treated interchangeably. In these cases, some best practices may apply:

Explicitly verifying that your user owns the WhatsApp account as you would any other new authentication channel, by sending e.g. an initial OTP and having the user enter it in your app

during registration or while they are logged in.Showing an additional challenge to verify the user, on top of the code sent via WhatsApp.

With the first method, you can take advantage of our identity change check systems on Cloud API to “bind” the WhatsApp account to your user’s account, making sure that future messages only reach the WhatsApp user who received that initial OTP. For example, on Cloud API you should store the identity hash received after sending the initial OTP and (assuming the verification was successful) include it in all subsequent message send requests. This setup would improve security over SMS as message delivery would fail if the phone number is recycled and the new owner registers on WhatsApp (OTP codes will not be accidentally sent to an unintended recipient).

To combat phishing, WhatsApp disables forwarding of authentication messages. Messages travel end-to-end encrypted between Cloud API and the user.

WhatsApp does not support and cannot validate the security practices of unofficial apps. There is no guarantee that authentication via WhatsApp is secure for users who use these apps.

UX

Collect opt-in

Per the WhatsApp Business Messaging Policy, you must get opt-in before send you can send a message to a WhatsApp user. A common implementation is to offer users a choice of authentication channels (WhatsApp, e-mail, SMS, etc.), as shown in our sample application.

Resolving message delivery issues

If you are seeing issues where users are selecting WhatsApp but messages end up being undeliverable, it is possible users are accidentally selecting WhatsApp when they in fact are not registered on WhatsApp. To mitigate this, on Android, you can check if WhatsApp is installed and only show WhatsApp in this case.

```
fun isWhatsAppAvailable(context:Context){return isAppAvailable(context,"com.whatsapp")||
          isAppAvailable(context,"com.whatsapp.w4b")}


fun isAppAvailable(
   context:Context,
   packageName:String):Boolean{
 val intent =Intent()
 intent.setPackage(packageName)
 intent.action ="com.whatsapp.otp.OTP_REQUESTED"
 val packageManager = context.packageManager
 val listActivities = packageManager.queryBroadcastReceivers(intent,0)return listActivities.
```

```
isNotEmpty()}
```

If you are still seeing issues where users are selecting WhatsApp but messages end up being undeliverable, it is also possible that the WhatsApp phone number is not correct. This could be through a user typo error, or that an app is incorrectly assuming the initial registration phone number is the same as the WhatsApp phone number. Users may have a phone number used for SMS and a different phone number used for WhatsApp, in the case where they might have multiple SIM cards for traveling. You should ensure that if WhatsApp is chosen as the authentication channel that the user has a chance to confirm their WhatsApp phone number.
If messages are being delivered but you are seeing lower than expected conversion rates in your authentication flows, consider adopting our more seamless “one-tap autofill” functionality, available for Android.

Support all apps

Your users may be ready to receive messages through WhatsApp or the WhatsApp Business App (or both). If following our Android client implementation guide, your “one-tap autofill” messages should work with any combination of installs, but we recommend testing one-tap across both the consumer app and the business app.

Be ready to receive the code from WhatsApp

If integrating with “one-tap autofill” functionality, you should be able to handle the code arriving as soon as you have sent the handshake. WhatsApp will send the code when received regardless of what screen is currently shown on your app. For example, in situations with poor network connectivity, you may receive the code before you are able to load the code entry screen in your app. To handle these cases, one option is to store the received code such that the app can retrieve it when the next screen is fully loaded. This way, the code can be automatically filled by your app as soon as the code is received.

Business accounts and phone numbers

Every business must have its own WhatsApp Business Account and be sending authentication templates through its own phone number, as opposed to sharing WABAs and phone numbers with separate business entities. Sharing a WABA across multiple businesses is against policy as it conflicts with the WhatsApp Business Terms of Service and WhatsApp Business Messaging Policy, in addition to creating poor user and business experiences on WhatsApp.

Checking if WhatsApp is installed on Android and iOS

Checking on Android

You can check WhatsApp installation before offering WhatsApp as an option if you expect both WhatsApp and your app to be on the same device.
First, you need to add the following to your `AndroidManifest.xml` file:

```
<queries>
<packageandroid:name="com.whatsapp"/>
<packageandroid:name="com.whatsapp.w4b"/>
</queries>
```

Using the SDK (Preferred)

Instantiate the `WhatsAppOtpHandler` object:

```
WhatsAppOtpHandler whatsAppOtpHandler =newWhatsAppOtpHandler();
```

Check if the WhatsApp client is installed by passing the `isWhatsAppInstalled` method as the clause in an `If` statement:

```
If(whatsAppOtpHandler.isWhatsAppInstalled(context)){// ... do something}
```

Without the SDK

```
if(this.isWhatsAppInstalled(context)){// ... do something}publicboolean isWhatsAppInstalled(
final@NonNullContext context){return isWhatsAppInstalled(context,"com.whatsapp")||
           isWhatsAppInstalled(context,"com.whatsapp.w4b");}publicboolean isWhatsAppInstalled(
final@NonNullContext context,final@NonNullString type){finalIntent intent =newIntent();
    intent.setPackage(type);
    intent.setAction("com.whatsapp.otp.OTP_REQUESTED");PackageManager packageManager = context
.getPackageManager();List<ResolveInfo> receivers = packageManager.queryBroadcastReceivers(
intent,0);return!receivers.isEmpty();}}
```

Checking on iOS

Use the following code in your iOS application to check if WhatsApp is installed

```
let schemeURL = URL(string:"whatsapp://otp")!let isWhatsAppInstalled =UIApplication.shared.
canOpenURL(schemeURL)
```

Revision #3
Created 2026-04-01 14:28:47 UTC by New Admin
Updated 2026-04-06 17:49:39 UTC by New Admin