**Phone Authentication**

Phone authentication allows users to sign in to Firebase using their phone as the authenticator. An SMS message is sent to the user (using the provided phone number) containing a unique code. Once the code has been authorized, the user is able to sign

into Firebase.

Firebase Phone Authentication is not supported in all countries. Please see their [FAQs](https://firebase.google.com/support/faq/#develop) for more information.

**Setup**

follow these steps:

1. Enable Phone as a Sign-In method in the [Firebase console](https://console.firebase.google.com/u/0/project/\_/authentication/providers).

2. If you haven't already set your app's SHA-1 hash in the [Firebase console](https://console.firebase.google.com/), do so.

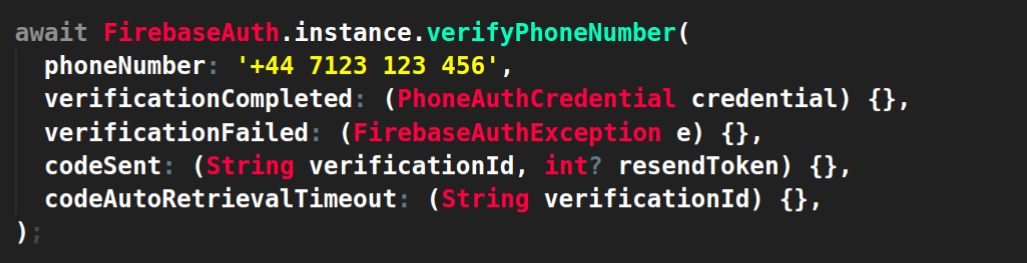
See [Authenticating Your Client](https://developers.google.com/android/guides/client-auth) for information about finding your app's SHA-1 hash.

**Verify Phone Number**

The user's phone number must be first verified and then the user can either sign-in or link their account with a

[`PhoneAuthCredential`](!firebase\_auth\_platform\_interface.PhoneAuthCredential).

First you must prompt the user for their phone number. Once provided, call the ‘verifyPhoneNumber()’ method:

****

There are 4 separate callbacks that you must handle, each will determine how you update the application UI:

1. [verificationCompleted]: Automatic handling of the SMS code on Android devices.

2. [verificationFailed]: Handle failure events such as invalid phone numbers or whether the SMS quota has been exceeded.

3. [codeSent]: Handle when a code has been sent to the device from Firebase, used to prompt users to enter the code.

4. [codeAutoRetrievalTimeout]: Handle a timeout of when automatic SMS code handling fails.

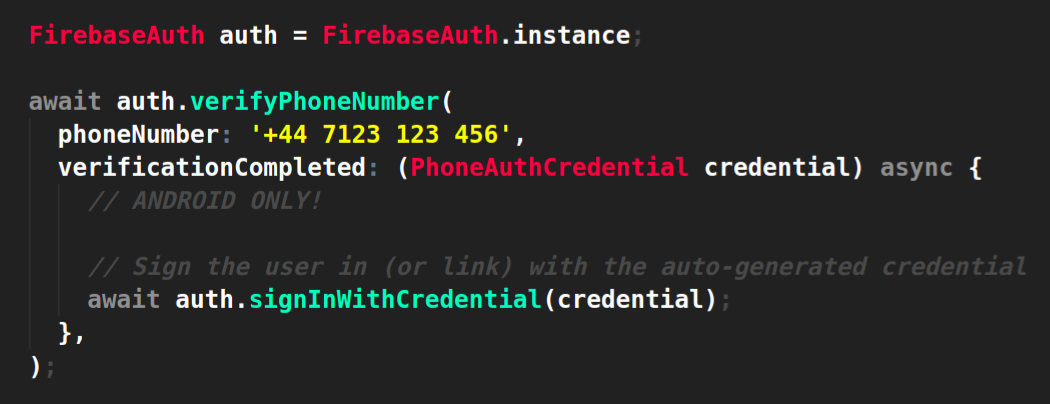
**verificationCompleted**

This handler will only be called on Android devices which support automatic SMS code resolution.

When the SMS code is delivered to the device, Android will automatically verify the SMS code without

requiring the user to manually input the code. If this event occurs, a `PhoneAuthCredential` is automatically provided which can be

Use d to sign-in with or link the user's phone number.

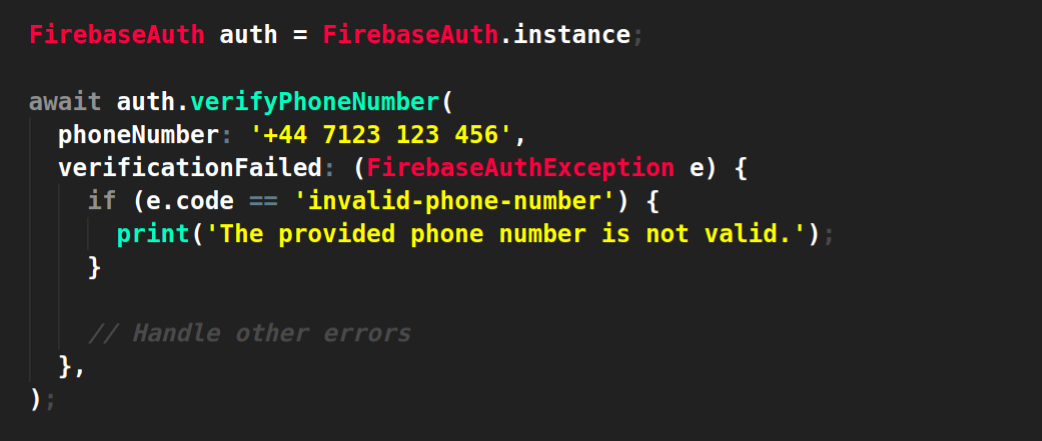


**verificationFailed**

If Firebase returns an error, for example for an incorrect phone number or if the SMS quota for the project has exceeded,

a `FirebaseAuthException` will be sent to this handler. In this case, you would prompt your user something went wrong depending on the error

code.



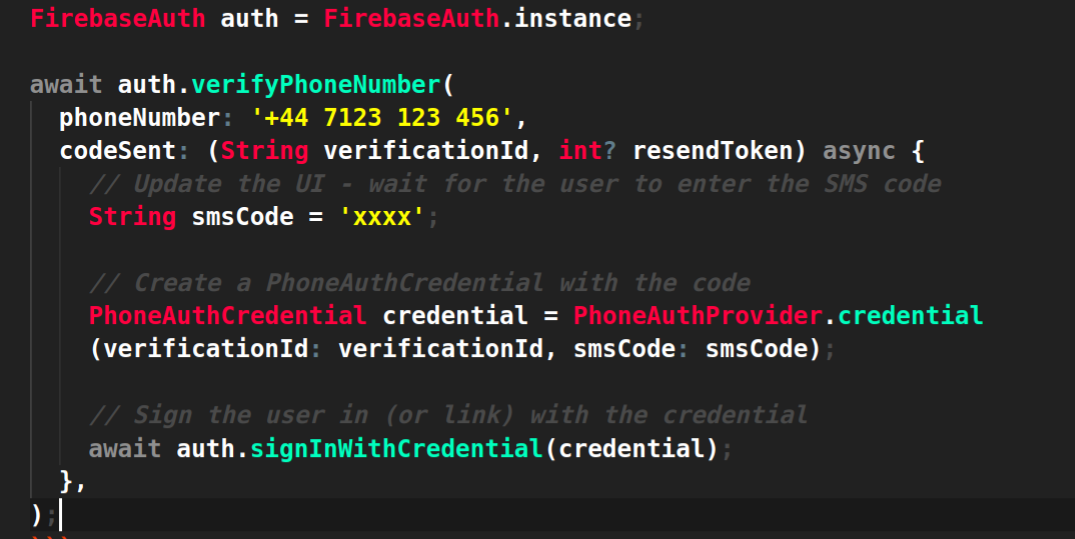
**codeSent**

When Firebase sends an SMS code to the device, this handler is triggered with a `verificationId` and `resendToken` (A `resendToken`

is only supported on Android devices, iOS devices will *\_always\_* return a `null` value).

Once triggered, it would be a good time to update your application UI to prompt the user to enter the SMS code they're expecting.

Once the SMS code has been entered, you can combine the verification ID with the SMS code to create a new `PhoneAuthCredential`:



By default, Firebase will not re-send a new SMS message if it has been recently sent. You can however override this behavior

by re-calling the `verifyPhoneNumber` method with the resend token to the `forceResendingToken` argument.

If successful, the SMS message will be resent.

**codeAutoRetrievalTimeout**

On Android devices which support automatic SMS code resolution, this handler will be called if the device has not automatically

resolved an SMS message within a certain timeframe. Once the timeframe has passed, the device will no longer attempt to resolve

any incoming messages.

By default, the device waits for 30 seconds however this can be customized with the `timeout` argument:

