How to verify purchase for android app in server side (google play in app billing v3)

Asked 9 years, 1 month ago Modified 2 years, 10 months ago

Viewed 155k times Part of Mobile Development Collective

156

I have a simple app (needs user login with account). I provide some premium features for paid users, like more news content.

I need to record if the user has bought this item in my server database. When I provide data content to user's device, I can then check the user's status, and provide different content for paid user.

I checked the official Trivialdrive sample provided by Google, it does not provide any sample code for server-side verification, here are my questions.

1. I found the sample use my app's public key inside to verify purchase, it looks not good, I think I can just move the verification process to my server combined with user login credentials to see whether the user purchase completed, and then update the database.
2. Also there is purchase API I can use to query, what I need is to pass the user's purchaseToken into server.

I am not sure what method I should take to verify the user's purchase, and mark the user's status in my database, maybe both?

And I am afraid there is a situation, if a user bought this item from google play, but for some reason, just in that time, when my app launched verification to my server, **the network connection is down or my own server is down**, user just paid the money in google play but I did not record the purchase in my server? What should I do, How can I deal with this situation.

MD android in-app-purchase in-app-billing

Share

Improve this question

Follow

edited Jan 31, 2017 at 13:13

Phil

1,069 ● 1 ● 17 ● 15

asked Nov 22, 2015 at 2:07

virsir

15.5k ● 25 ● 77 ● 110

Possible duplicate of Android : inApp purchase receipt validation google play – Jeshurun Oct 5, 2017 at 18:36

8 Answers

Sorted by: Highest score (default)

It sounds what you're looking for is a way to check if the user has premium features enabled on their account, so

228

this is where I would start;

Ensure there is a flag of some sort on your database indicating if the user has premium features and include that in the API response payload when requesting account info. This flag will be your primary authority for "premium features".

+50

When a user makes an in-app purchase, cache the details (token, order id, and product id) locally on the client (i.e the app) then send it to your API.

Your API should then send the `purchaseToken` to the Google Play Developer API for validation.

A few things might happen from here:

1. The receipt is valid, your API responds to the client with a 200 Ok status code
2. The receipt is invalid, your API responds to the client with a 400 Bad Request status code
3. Google Play API is down, your API responds with a 502 Bad Gateway status code

In the case of 1. or 2. (2xx or 4xx status codes) your client clears the cache of purchase details because it doesn't need it anymore because the API has indicated that it has been received.

Upon a successful validation (case 1.), you should set the `premium` flag to true for the user.

In the case of 3. (5xx status code) or a network timeout the client should keep trying until it receives a 2xx or 4xx status code from your API.

Depending on your requirements, you could make it wait a few seconds before sending again or just send the details to your API when ever the app is launched again or comes out of background if the purchase details are present on the app cache.

This approach should take care of network timeouts, servers being unavailable, etc.

There are now a few questions you need to consider:

What should happen immediately after a purchase? Should the app wait until validation is successful before providing premium content or should it tentatively grant access and take it away if the validation fails?

Granting tentative access to premium features smooths the process for a majority of your users, but you will be granting access to a number of fraudulent users too while your API validates the `purchaseToken`.

To put this in another way: Purchase is valid until proven fraudulent or; fraudulent until proven valid?

In order to identify if the user still has a valid subscription when their subscription period comes up for renewal, you will need to schedule a re-validation on the

`purchaseToken` to run at the `expiryTimeMillis` that was returned in the result.

If the `expiryTimeMillis` is in the past, you can set the `premium` flag to false. If it's in the future, re-schedule it again for the new `expiryTimeMillis`.

Lastly, to ensure the user has premium access (or not), your app should query your API for the users details on app launch or when it comes out of background.

Share Improve this answer

Follow

answered Nov 30, 2015 at 18:01

Marc Greenstock

11.6k ● 4 ● 32 ● 55

For paid app, how I will get the receipt from google? – Merbin Joe Dec 20, 2016 at 11:38

2 Hi! There is no way to access subscriptions history at google? How to prevent loosing of fact that used already purchased cubscription in the case if app crashes at the moment of storing purchaseToken? – scythargon Feb 6, 2017 at 14:38

2 I have a similar problem.. instead of letting the app sending the token to the api wouldnt more reliable to instruct the google developer server to do so with a push notification straight to my api? – Gianluca Ghettini Oct 23, 2017 at 7:26

1 For **subscriptions** that were canceled Google Play Developer API will still return 200 after the cancel, if the same old purchase token is used for validation. – Cezar Cobuz Feb 27, 2020 at 13:01

So for a subscription, you are suggesting that after the first call on the server, we store the purchase token, and product

ID, and schedule an other verification call (re-run the same request) when the expiryTimeMillis happens ? Is that how we are supposed to verify subscription validity ? Are there any guidelines from Android on how to do that ? Apple got a WWDC video about it that explains the good practice of it pretty clearly, but can't find much about Play Store.
– schankam Apr 20, 2020 at 13:24

70

The documentation on this is confusing and weirdly verbose with the things that are almost inconsequential while leaving the actually important documentation almost unlinked and super hard to find. This should work great on most popular server platform that can run the google api client libraries, including Java, Python, .Net, and NodeJS, among others. Note: I've tested only the Python api client as shown below.

Necessary steps:

1. Make an API project, from the API Access link in your Google Play console
2. Make a new service account, **save** the JSON private key that gets generated. You'll need to take this file to your server.
3. Press Done in the Play console's service account section to refresh and then grant access to the service account

4. Go get a google api client library for your server platform from https://developers.google.com/api-client-library
5. Use your particular platform's client library to build a service interface and directly read the result of your purchase verification.

You do *not* need to bother with authorization scopes, making custom requests calls, refreshing access tokens, etc. the api client library takes care of everything. Here's a python library usage example to verify a subscription:

First, install the google api client in your pipenv like this:

```
$ pipenv install google-api-python-client
```

Then you can set up api client credentials using the private key json file for authenticating the service account.

```
credentials =
service_account.Credentials.from_service_account_file(
```

Now you can verify subscription purchases or product purchases using the library, directly.

```
#Build the "service" interface to the API you want
service = googleapiclient.discovery.build("androidpubl
credentials=credentials)

#Use the token your API got from the app to verify the
result =
```

```
service.purchases().subscriptions().get(packageName="y
subscriptionId="sku.name", token="token-from-app").exe
#result is a python object that looks like this ->
# {'kind': 'androidpublisher#subscriptionPurchase', 's
'1534326259450', 'expiryTimeMillis': '1534328356187',
'priceCurrencyCode': 'INR', 'priceAmountMicros': '7000
'IN', 'developerPayload': '', 'cancelReason': 1, 'orde
1234-1234..5', 'purchaseType': 0}
```

The documentation for the platform service interface for the play developer API is not linked in an easy to find way, for some it is downright *hard to find*. Here are the links for the popular platforms that I found:

Python | Java | .NET | PHP | NodeJS (Github TS) | Go (Github JSON)

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Aug 15, 2018 at 11:59

Dhiraj Gupta

10.4k ● 8 ● 51 ● 56

14 Agree, documentation is horrible... Any ideas of how to do this with Firebase (Firestore) and Cloud functions as a backend? – Jeff Padgett Sep 5, 2018 at 22:06

If your Cloud functions are in NodeJS then you could perhaps use the NodeJS link above to get the API Client library working? – Dhiraj Gupta Sep 6, 2018 at 8:20

2 Hard to find since they want to promote Cloud PubSub that incur charges – Amit Khetan Jul 7, 2021 at 1:19

2 this info is GOLD!! – Xao Jan 21, 2022 at 11:41

2 Thanking you lots - I was going to give up on doing server side verification because of the paid for pub sub debacle. – LordWabbit Jul 3, 2023 at 23:26

30

Complete example of using Google API Client Library for PHP:

1. Setup your **Google Project** and access to **Google Play** for your *service account* as described in Marc's answer here https://stackoverflow.com/a/35138885/1046909.
2. Install the library: https://developers.google.com/api-client-library/php/start/installation.
3. Now you are able to verify your receipt the following way:

```
$client = new \Google_Client();
$client->setAuthConfig('/path/to/service/account/c
$client->addScope('https://www.googleapis.com/auth
$service = new \Google_Service_AndroidPublisher($c
$purchase = $service->purchases_subscriptions->get
$token);
```

After that $purchase is instance of **Google_Service_AndroidPublisher_SubscriptionPurchase**

```
$purchase->getAutoRenewing();
$purchase->getCancelReason();
...
```

Share Improve this answer Follow

edited Oct 5, 2017 at 9:50

answered Oct 5, 2017 at 9:45

MingalevME

1,965 ● 1 ● 23 ● 19

This is not working, I keep getting (401) Login Required and setAuthConfig doesnt accept the service account credentials json – Raulnd Nov 14, 2017 at 20:45

This one worked for me putenv('GOOGLE_APPLICATION_CREDENTIALS=credentials.json'); $client = new Google_Client(); $client->useApplicationDefaultCredentials(); $client->addScope('googleapis.com/auth/androidpublisher'); $service = new Google_Service_AndroidPublisher($client); $purchase = $service->purchases_products->get($packageName, $productId, $token); var_dump($purchase); – Raulnd Nov 14, 2017 at 21:43

This thing is in case of inapp billing. What if I want to get orderId to my database whenever user buys my app from play store instead of inapp ? – Ankesh kumar Jaisansaria Feb 10, 2018 at 6:41

stackoverflow.com/questions/48662787/… Please see this question. I am looking for answer to this question. This has active bounty also – Ankesh kumar Jaisansaria Feb 10, 2018 at 6:42

@MingalevME What if the token format is invalid and PHP gets a fatal error, how do I catch this error? – alexx0186 Jul 2, 2018 at 15:08

12

You can try using Purchases.subscriptions: get server-side. It takes packageName, subscriptionId and token as paramaters and requires authorization.

> Checks whether a user's subscription purchase is valid and returns its expiry time.

If successful, this method returns a Purchases.subscriptions resource in the response body.

Share Improve this answer

Follow

answered Nov 23, 2015 at 7:24

random

10.3k ● 8 ● 59 ● 102

11 I'm having serious problems to get the authorization working. – user5781295 Jan 3, 2017 at 21:08

11 Seriously. For how critical purchases are to some apps, the support and documentation is inane. This is what you need to do on the server: github.com/google/…. More info here: stackoverflow.com/questions/35127086/… – user Sep 13, 2017 at 10:07

2 and we have to pay 30% for this crap. Not to mention that its enforced on digital contents to use IAP... – Xao Jan 21, 2022 at 11:51

1 Authorization is very trouble some processes. The documentation is pretty bad – Satyam Jan 27 at 16:33

I answer to this concern

1

> the network connection is down or my own server is down, user just paid the money in google play but I did not record the purchase in my server? What should I do, How can I deal with this situation.

The situation is:

User purchases 'abc' item using google play service -> return OK -> fail to verify with server for some reasons such as no Internet connection.

Solution is:

On the client side, before showing the 'Google Wallet' button, you check if the 'abc' item is already owned.

- if yes, verify with server again
- if no, show the 'Google Wallet' button.

> Purchase purchase = mInventory.getPurchase('abc');
>
> ```
> if (purchase != null) // Verify with server
>
> else // show Google Wallet button
> ```

https://developer.android.com/google/play/billing/billing_reference.html#getSkuDetails

Share Improve this answer

Follow

answered May 9, 2017 at 11:56

thanhbinh84

18.4k ● 7 ● 65 ● 75

4 I don't get why validating on the server is more secure than validating on the app. At the end of the day it's the app that unlocks the features so it's still possible to remove or invert the code in the app that checks if the server response is "OK" – Gianluca Ghettini Oct 26, 2017 at 14:08

6 @GianlucaGhettini because, sometimes the server is what provides the purchased service not the app besides, the app could be reverse engineered and then with some struggle the verification process could be hacked. – Mohyaddin Alaoddin Jul 23, 2018 at 7:25

0

Marc Greenstock's answer is definitely enlightening, a few things to pay attention though which took me a long time to figure out (at least way more time than I expected):

1. I had to check "Enable G Suite Domain-wide Delegation" on Service Account settings. Without this I kept getting this error: "The current user has insufficient permissions to perform the requested operation" Image with Enable G Suite Domain-wide Delegation option checked
2. For testing purposes you can create a JWT token for your service account here, just don't forget to select RS256 Algorithm.

3. The public key is the "private_key_id" from your downloaded JSON file. It also has the following format:

   -----BEGIN PUBLIC KEY----- {private_key_id} ----- END PUBLIC KEY-----

4. The private key is the "private_key" from your downloaded JSON file
5. The required claims for the JWT generation are described here.
6. Confused about what exactly a JWT Token is and how it is assembled? Don't be ashamed, check this link. Odds are you are just like me and took a long time to bother looking for what exactly it is, it is (way) simpler than it looks.

Share Improve this answer

Follow

answered Sep 19, 2020 at 14:56

Leonardo Bilck

155 • 3

0

I had some serious problems using the suggested google API python library, but implementing the communication from scratch is not so hard. First of all you have to create a service account at Google Play Console as described in all answers and get the JSON file containing the private key. Save it to your server. Then use the following code. No need to obtain the google API client library. You only need the following (very common) python libraries Requests and Pycrypto

```
    import requests
    import datetime
    import json
    import base64
    from Crypto.Signature import PKCS1_v1_5 as Signatu
    from Crypto.Hash import SHA256
    from Crypto.PublicKey import RSA

    jwtheader64 = "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ
    #SERVICE_ACCOUNT_FILE: full path to the json key f
    with open(SERVICE_ACCOUNT_FILE) as json_file:
        authinfo = json.load(json_file)

    packageName = #your package name
    product = #your inapp id
    token = #your purchase token

    #create the JWT to use for authentication
    now = datetime.datetime.now()
    now1970 = (now - datetime.datetime(1970,1,1)).tota
    jwtclaim =
{"iss":authinfo["client_email"],"scope":"https://www.g
"https://oauth2.googleapis.com/token","iat":now1970,"e
    jwtclaimstring = json.dumps(jwtclaim).encode(encod
    jwtclaim64 = base64.urlsafe_b64encode(jwtclaimstri
8')
    tosign = (jwtheader64+"."+jwtclaim64).encode(encod

    #sign it with your private key
    private = authinfo["private_key"].encode(encoding=
    signingkey = RSA.importKey(private)
    signer = Signature_pkcs1_v1_5.new(signingkey)
    digest = SHA256.new()
    digest.update(tosign)
    signature = signer.sign(digest)
    res = base64.urlsafe_b64encode(signature).decode(e

    #send it to Google authentication server to obtain
    headers = {'Content-Type': 'mapplication/x-www-for
    payload = "grant_type=urn%3Aietf%3Aparams%3Aoauth%
bearer&assertion="+jwtheader64+"."+jwtclaim64+"."+res
    r =
requests.post("https://oauth2.googleapis.com/token",he
```

```
    if r.status_code == 200:
        authdata = json.loads(r.text)
        accesstoken = authdata['access_token']
        bearerheader = {'Authorization':'Bearer '+auth
        #Now you have at last your authentication toke
make calls. In this example we want to verify a subscr
        url =
"https://androidpublisher.googleapis.com/androidpublis
        subscription = requests.get(url,headers=bearer
```

Share Improve this answer

Follow

answered Apr 16, 2021 at 8:57

befstrat

449 ● 4 ● 7

0

the network connection is down or my own server is down,

You don't have to think like this. Client knows own's consume product. so, client can send all tokens back to the server.

Just re-check token with produce id and transaction id. And Server checks consume product.

1. if you fail check
2. make UI button client can re-send token.
3. server re-check token for items.
4. It's done.

Share Improve this answer

Follow

edited Feb 24, 2022 at 21:44

Yunnosch

26.6k ● 9 ● 44 ● 62

answered Jan 23, 2022 at 12:16

DEV JUNGLE

41 ● 1 ● 6

Your answer could be improved with additional supporting information. Please edit to add further details, such as citations or documentation, so that others can confirm that your answer is correct. You can find more information on how to write good answers in the help center. – Community Bot Jan 24, 2022 at 4:31