2026

TAP Connection Interface

CARDCOM LTD.

Tap On Phone API – Summary for Integrators

This API enables third-party vendors to securely perform Tap On Phone (contactless payment) transactions using Cardcom's platform. The API is designed for integration with your mobile or server-side applications, allowing you to register devices, prepare them for transactions, process payments, and retrieve transaction history.

How to Use:

- **Authenticate:** All API requests require your API username and password for authentication.
- **Prepare Device:** Use the PrepareDevice endpoint to validate and prepare a device before processing payments.
- **Process SDK Transaction:** Submit the request and response data of the SDK transaction to the ProcessSDKTransaction endpoint to process a payment.
- **Get Transactions:** Retrieve transaction history for a specific terminal and date range.
- **Get Devices:** List all devices registered to your company.

Notes:

- All requests and responses use **application/json**.
  Detailed request and response structures are provided in the full documentation.
- Date fields are in ISO 8601 format.
- The GetTransactions endpoint limits the date range to 1 year.

Prerequisites for TAP Transaction via SDK

- Only Android platform is supported
- Device must include NFC
- Location services access must be enabled
- "Developer Options" setting must be turned off
- We need the SHA256 of your application.
  If it will be deployed manually – You can get it from the keystore file.
  Please see Extract SHA256 certificate fingerprints from keystore file section below.
  If it will be deployed via the Google Play Store – You should get it from the Google Play Console.

Full API documentation can be found on:

https://secure.cardcom.solutions/Api/v11/Docs#tag/TapTransactions

General Explanation of Steps for TAP Payment

1. When the application launches for the first time:
   a. Call Cardcom API to obtain details and register the end device.
   b. Register the device with the SDK using **registerDevice** method.

2. For each payment transaction:
   a. Perform the TAP transaction with the SDK using **performTransaction** method.
   b. Receive a response from the device and forward it to the Cardcom API.

Note: In Step 1, the device registration has a validity period, so this step needs to be repeated periodically each time the validity peried has expired.

SDK Installation in the Application

1. Ensure the application has the following permissions:

```
<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-feature    android:name="android.hardware.nfc" android:required="false" />
```

2. Add the following code to the build.gradle file of your project:

```
allprojects {
        repositories {
                jcenter()
                flatDir {
                        dirs 'libs'
                }
                maven {
                        url "s3://denovo-android.s3.amazonaws.com"
                        credentials(AwsCredentials) {
                        accessKey "AKIA26XMFQBITA36GRDR"
                        secretKey "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"
                }
        }
}


dependencies {
...
   implementation "com.denovo:topliteapp-PROD:[version]"
   implementation files("../libs/Cardcom-[version]-Release_Prod.aar")
}
```

Note: If the **libs** folder does not exist, you should create it under your project's root folder.

3. Use the given proguard-rules.pro file or add its content to your existing one.
4. After creating new APK, you must perform finalization, please see APK Finalization section below.

APK Finalization

In order to use the Tap On Phone SDK you must perform APK Finalization for your APK file.

Use the given AppFinalizationFolder.zip file content.

1. Put your apk file in the AppFinalizationFolder folder.
2. Put your keystore.jks file in the AppFinalizationFolder folder.
3. Put the given release.fin file in the AppFinalizationFolder folder.
4. Use CMD and run those commands from the AppFinalizationFolder folder:

Finalizing step:

```
finalizer -i APKFILE.apk -f release.fin -o finalizedApkFolder
```

Zip aligning step:

```
zipalign -p -v 4 finalizedApkFolder\APKFILE-unaligned-unsigned-protected.apk finalizedApkFolder\APKFILE_aligned_unsigned.apk
```

APK signing step:

```
apksigner sign --ks keystore.jks --ks-pass "pass:KEYSTOREPASSWORD" --out finalizedApkFolder\APKFILE_aligned_signed.apk
finalizedApkFolder\APKFILE_aligned_unsigned.apk
```

NOTE:

- APKFILE.apk is your apk file, rename the commands with the actual apk file name.
- KEYSTOREPASSWORD is your keystore file password, rename command with the actual password.

5. After the APK finalization is complete, you should have the finalized aligned-signed apk file in the finalizedApkFolder folder, this is the apk file you should use.

Create JKS file from keystore file

You should have a JKS file for performing the APK finalization.

If you don't have the JKS file, and you have the keystore file, you can get the JKS file using this method:

1. Go to JAVA bin folder (Usually C:\Program Files (x86)\Java\jre-1.8\bin)
2. Copy the [file-name].keystore file to JAVA bin folder
3. Run this command in CMD from JAVA bin folder:
   `keytool -importkeystore -srckeystore [file-name].keystore -destkeystore [file-name].jks -deststoretype jks`
4. Enter password for destination JKS file (choose password and remember it - you will need it for signing the apk)
5. Enter password for source keystore file
6. You will get a new JKS file: [file-name].jks

Extract SHA256 certificate fingerprints from keystore file

1. Go to JAVA bin folder (Usually C:\Program Files (x86)\Java\jre-1.8\bin)
2. Copy the [file-name].keystore file to JAVA bin folder
3. Run this command in CMD from JAVA bin folder:
   `keytool -alias [keystore-file-alias] -keystore [file-name].keystore -list -v`
4. You will get a screen with some info, you can find the SHA256 under 'Certificate fingerprints'

Step 1.a – PrepareDevice

Get device information before performing a Tap On Phone transaction

POST https://secure.cardcom.solutions/api/v11/TapTransactions/PrepareDevice

Request

```
{
  "ApiName": "string",
  "ApiPassword": "string",
  "TerminalNumber": int,
  "DeviceUUID": "string",
  "DeviceManufacturer": "string",
  "DeviceModel": "string"
}
```

Response

200 OK

```
{
  "ResponseCode": int,
  "Description": "string",
  "TPN": "string",
  "MerchantCode": "string",
  "SapakMutavNumber": "string",
  "RavMutavTerminalNumber": int
}
```

400 Bad Request or 401 Unauthorized

```
{
  "ResponseCode": int,
  "Description": "string"
}
```

Step 1.b - Registering (RegisterDevice)

Register the device by invoking the **topService.registerDevice** method

Parameters for the method

Name	Description	Type
TPN	As received in step 1.a	string
MerchantCode	As received in step 1.a	string

Parameters Received in onRegisterSuccess

Name	Description	Type
message	Success message	string
session_key	Store at the application level; will be passed during payment in step 2.a	string
session_date	Session Key created timestamp in milliseconds is used to validate the session key expiry	string
session_expire_time	This will indicate that the session key expiry time in milliseconds.	string

Parameters Received in onRegisterError

Name	Description	Type
error_code	Error code - see list of codes	string
message	Error message description	string

Sample code:

```
ToPService topService = new ToPService(Activity activity);
topService.registerDevice(tpn, merchantCode, new ToPService.onRegistrationListener() {
	@Override
	public void onProcess(String message) {
		//UPDATE PROGRESS DIALOG message HERE
	}

	@Override
	public void onRegisterSuccess(JSONObject successObject) {
		//DISMISS PROGRESS DIALOG HERE
	}

	@Override
	public void onRegisterError(JSONObject errorObject) {
		//DISMISS PROGRESS DIALOG HERE
	}
});
```

Step 2.a - Charging (PerformTransaction)

Charge the credit card by invoking the **topService.performTransaction** method

Parameters for the method:

sessionKey (string) - As received during registration in step 1.b

inputObject (json)

```
{
  "type": "SALE", // either "SALE " or "REFUND"
  "amount": "20.00", // the transaction amount in NIS
  "lang": "iw", // should always be "iw"
  "CustomObject": {
    "CustomerName": "", // not mandatory, you can either pass value or pass an empty string
    "CustomerEmail": "", // not mandatory, you can either pass value or pass an empty string
    "PhoneNumber": "", // not mandatory, you can either pass value or pass an empty string
    "HolderID": "", // not mandatory, you can either pass value or pass an empty string
    "NumberOfPayments": 1, // the number of installments
    "FirstPaymentSum": 0, // 0 for single installment or calculated sum for multiple installments
    "OtherPaymentsSum": 0, // 0 for single installment or calculated sum for multiple installments
    "CreditType": 1, // 1 – single installment / 6 – credit installments / 8 – regular installments
    "SapakMutavNumber": "", // the exact value as received in PrepareDevice response (step 1.a)
    "TransactionUniqueIdForQuery": "", // should always be an empty string
    "AllowFullCardNumberInResponse": true // should always be true
  }
}
```

Note:
FirstPaymentSum and *OtherPaymentsSum* values should be both 0 in case of single installment.
When the customer is paying with more than 1 installment, you should calculate and set the values for the *FirstPaymentSum* and *OtherPaymentsSum* properties.
The *FirstPaymentSum* value is the sum of the first installment, and the *OtherPaymentsSum* value is the sum of each of the remaining installments.

For example:
Transaction of 30.00 NIS with 2 installments:
amount is “30.00”
NumberOfPayments is 2
FirstPaymentSum is 15
OtherPaymentsSum is 15

Or:

Transaction of 60.00 NIS with 3 installments:
amount is “60.00”
NumberOfPayments is 3
FirstPaymentSum is 10
OtherPaymentsSum is 25

Parameters Received in onTransactionResponse

Name	Description	Type
response_code	Response code	string
message	Response description	string
status	Transaction status	string
transaction_id	Transaction ID	string
transactionResult	Transaction details	
customMessage		
DE48		
ReturnCode	Response code	string
ReturnMessage	Response description	string
CardNumber	Card number	string
ExpDate_MMYY	Card expiration date	string
CardBrandCode	Card brand code	string
CardIssuerCode	Card issuer code	string
CardFinancerCode	Card financier code	string
Token	Card token	string
ApprovalNumber	Approval number	string
ResultRecord	Encrypted transaction details	string
UID	Transaction identifier	string

Parameters Received in onTransactionError

Name	Description	Type
error_code	Error code - see list of codes	string
message	Error message description	string

Parameters Received in onRegisterNeeded

Name	Description	Type
error_code	Error code - see list of codes	string
message	Error message description	string

Sample code:

```
ToPService topService = new ToPService(Activity activity);
topService.performTransaction(inputObject, sessionKey,  new ToPService.OnTransactionListener} ()
        @Override
        public void onTransactionResponse (JSONObject successObject) }
                //note: response_code:'00' means Successful transaction other than '00' means Failure transaction
        {

        @override
        public void onTransactionError (JSONObject errorObject)}

        {

        @override
        public void onRegisterNeeded (JSONObject errorObject)}

        {
});
```

Step 2.b – ProcessSDKTransaction

Process a Tap On Phone transaction

POST https://secure.cardcom.solutions/api/v11/TapTransactions/ProcessSDKTransaction

Request

```
{
   "ApiName": "string",
   "ApiPassword": "string",
   "TerminalNumber": int,
   "RavMutavTerminalNumber": int, // as received from PrepareDevice in step 1.a
   "DeviceUUID": "string",
   "SdkRequest": "PerformTransactionRequest", // as passed to performTransaction in step 2.a
   "SdkResponse": "PerformTransactionResponse", // as received from performTransaction in step 2.a
}
```

Response

200 OK

```
{
   "ResponseCode": int,
   "Description": "string",
   "DealNumber": long
}
```

400 Bad Request or 401 Unauthorized

```
{
   "ResponseCode": int,
   "Description": "string"
}
```

GetTransactions

Get all TAP transactions by terminal number and date range (up to 1 year)

POST https://secure.cardcom.solutions/api/v11/TapTransactions/GetTransactions

Request


```
{
   "ApiName": "string",
   "ApiPassword": "string",
   "TerminalNumber": int,
   "FromDate": "yyyy-MM-ddTHH:mm:ss",
   "ToDate": "yyyy-MM-ddTHH:mm:ss"
}
```


Response

200 OK


```
{
   "ResponseCode": int,
   "Description": "",
   "Transactions": [{
         "Status": "string",
         "DealNumber": long,
         "TransactionDateTime": "yyyy-MM-ddTHH:mm:ss",
         "CardBrand": "string",
         "CardIssuer": "string",
         "CardFinancer": "string",
         "CardNumber": "string",
         "CardExpirationDate": "string",
         "CardHolderName": "string",
         "CardHolderPhone": "string",
         "CardHolderEmail": "string",
         "CardHolderIdNumber": "string",
         "TransactionType": "SALE|REFUND",
         "Amount": decimal,
         "NumberOfPayments": int,
         "FirstPayment": decimal,
         "ConstPayment": decimal,
         "ApprovalNumber": "string",
         "ExternalTransactionId": "string"
      }
   ]
}
```


400 Bad Request or 401 Unauthorized


```
{
   "ResponseCode": int,
   "Description": "string"
}
```

GetDevices

Get all registered TAP devices for the authenticated company

POST https://secure.cardcom.solutions/api/v11/TapTransactions/GetDevices

Request

```
{
   "ApiName": "string",
   "ApiPassword": "string",
   "TerminalNumber": int
}
```

Response

200 OK

```
{
   "ResponseCode": int,
   "Description": "",
   "Devices": [{
         "UUID": "string",
         "Manufacturer": "string",
         "Model": "string",
         "TPN": "string",
         "MerchantCode": "string",
         "IsActive": true,
         "CreationDate": "yyyy-MM-ddTHH:mm:ss",
         "LastActivityDate": "yyyy-MM-ddTHH:mm:ss"
      }
   ]
}
```

400 Bad Request or 401 Unauthorized

```
{
   "ResponseCode": int,
   "Description": "string"
}
```