let's make payment happen

CCV
Android Developer Guidelines & FAQs

Latest update: August 2022

www.ccv.eu

CCV Andorid Developer Guidelines and FAQs

This document will give you an overview about the basic requirements for developing an app for our CCV Android Terminals and *CCV*Store. **Please read this document carefully to prevent that you invest time in a technical realization which CCV Android Terminals don't support.**

In addition to this document there are further sources to support you:

» The CCV mAPI documentation, which can be found here **https://developer.myccv.eu/**
» The document "requirements and test catalogue interoperability", which later on also will be part of the Contract "App Partnership and *CCV*Store"
» For IM30: the "Certification test list CCV IM30"

1. General guidelines

Here you will find our standard guidelines and requirements for an app in order to be included in our *CCV*Store. If your app is in conflict with these rules, please contact CCV to discuss a solution.

a) We recommend developing native apps for our CCV Android terminals, because web apps (=browser-based apps) can limit the performance of the terminals and have to be checked in depth to ensure security standards. However, if you already have a finished browser-based app or if a native app is not possible for your solution, please contact us to discuss the possibilities for a smooth integration before starting the development.

Further explanation: Native apps are programmed specifically for Android and run exclusively on Android devices, ensuring that all interfaces to hardware work uniformly and the resources of the device are used optimally. The native app is uploaded into the *CCV*Store in an .apk format. A web app is a specially programmed HTML5 website that recognizes the device and displays the content in an optimized way.

b) Our operating system "PayDroid" is based on Android. The app cannot access the Google Play Store. CCV operates an own app store – the *CCV*Store.

c) Our CCVStore works with the app format .apk and with V1-signature (the new format .aab, new Android methods won't work with CCVStore). For V1-signature the Android SDK Version 23 or below needs to be used!

d) Be aware that terminals reboot on a daily basis due to PCI requirements.

e) The app may not ask for any PIN (personal identification number). As an example it may not ask “your PIN please” or “please enter your PIN”. This is due to legal and security reasons. Password requests in clear context with the application login are allowed.

f) Be aware that at “idle mode” (mode when the terminal is not used) the app should be optimized regarding resource consumption. No background services or long-running tasks should be performed. The CPU and RAM usage of the app should not be over 50%. When the application is moved to the background by the payment application it should stop all processing.

g) The app may start card-related actions like payments using the mAPI library. The mAPI is a Java library, which is an abstraction layer and should be used for internal communication. mAPI is based on OPI. Documentation of the mAPI as well as an mAPI example app can be found here https://developer.myccv.eu/documentation/pos_integrations/android_sdk/general/demo-application/. For access to the mAPI example app an NDA needs to be singed first. Please contact us if you would like to access it.

To receive email notifications once a new mAPI release is available, please subscribe here.

Although OPI and ZVT (only for SecPOS Evo version 68.xx and higher) can technically also be used for internal communication, we highly recommend to use mAPI, since it provides a range of useful functionalities such as error recovery, logging and autodetect. If you use ZVT or OPI and plan to provide your app for both CCV Payment Engines (SecPOS Evo and acCEPT), CCV cannot guarantee that this is possible with the same .apk. Depending on the way of realization you might need to deliver two .apks to make the app available for both Payment Enginges.

OPI should be used for external communication (M2M).

h) Due to security reasons, there is no browser allowed on CCV Android terminals.
For cash register systems: please also note this with regard to any duties regarding registrations at tax authorities (“Kassensicherungsverordnung” or “Registrierkassenpflicht”, also see point 2.1 h).

i) The Neptune library cannot be used by customer apps or third-party apps (only CCV apps are allowed to use it). The app will not pass the app test when using the Neptune library.

j) The access of the app to card and QR codes:

» Scan QR-codes / bar-codes (simplified way using the mAPI: https://developer.myccv.eu/documentation/payment_terminals/local_-_android_sdk/guide/qr-and-barcode-scanner/)

» Mifare UID using CCV-OPI. Currently not supported by mAPI (available shortly). On the latest version of the payment engine for the A920 (SECpos Evo) there is the reader interface for reading (Mifare) cards implemented.

» ICC-cards or contactless cards with AID in the whitelist in the terminal. Currently not supported by mAPI. In order for reading cards for payments there is the Android-OS reader interface. There is only the select command blocked, when the card is not in whitelist. For IM30 which using IM30-OPI-NL these functionalities are not available.

» SAM cards are supported by the Android-OS reader interface.

k) The app may use the hardware of the device which can be accessed using the Android OS. For using the camera for QR-code or barcode reading there is a simplified way of integration via the mAPI (please see here: https://developer.myccv.eu/documentation/payment_terminals/local_-_android_sdk/guide/qr-and-barcode-scanner/). Please be aware magnetic stripe reader are unter control of the CCV payment application.

l) For unattended terminals (e.g. IM30) there is an optional functionality – called "kiosk mode" – which locks the front screen/default screen of the app. In this mode it is not possible for a user to exit the app and see the Android main screen. Due to customer support reasons this functionality (kiosk mode) should be able to be activated and deactivated at the settings in the app.

Since a customer should never be able to access the Android home screen of the IM30, apps will always have to be full screen which is also enforced by the PayDroid operating system. The kiosk mode should not be disabled. For attended terminals (e.g. A920) we would suggest that there should not be a kiosk mode.

m) Before the .apk is uploaded into the *CCV*Store an app test by CCV is compulsory. For the app test please send us a release-type of .apk (not a debug version).

n) Why is a build version number in the manifest essential? Every build needs to have a unique (internal and external) version. Updates in the field are usually done via the store. Therefore, it must be ensured that the build generates a valid manifest with continuously increasing build numbers, which must also be stored in the manifest accordingly.

o) If you need to regularly withdraw logs or if you need to regularly push config files, ideally this should be facilitated via an own backend system which is connected with your app.

To have apps as for example TeamViewer on the terminal is not compliant with PCI regulations, because Pin-entry can be seen remotely. Therefore it is not possible to use them.

For integrators using CCV unattended Android terminals (IM30), which create an app for their own use case, usually we provide access to an own environment within the CCVStore. Within this environment the "Air Viewer App" can be used to see the terminal display and to take over remote control of the terminal.

2. FAQs

2.1 General questions

a) Is there a specific framework to be used for the app?

Our operating system PayDroid is based on Android 7.1, which is why the used frameworks have to be compatible with Android 7. Reader Interface intents are based on the Android SDK. The mAPI is a Java library. For those components the Android SDK has to be used.

Other components can be used in any other framework as long there is an interface with the Android SDK. The Android SDK (for 7.1.1) can be downloaded for free within the Android Studio.

b) What can I do with the debug terminal?

We provide test terminals which are set in a debug mode so developers can transfer the .apk via sideload (usb) to the device and adjust the app on the device.

c) Is there a PAX specific SDK which is required?

No, CCV supports the standard Android SDK (without Playstore features). If there is a need to get access to the hardware then CCV we provide drivers and libraries. More specifications for IM30 please see under 2.3.

d) What is the required API level to upload an app into the CCVStore?

Currently a minimum API level of 25 is necessary so that an app can be uploaded into the CCVStore.

e) If an app is already installed on terminals in the field and there is an update of the app available, how does the update work?

The update is always done in the field with the *CCV*Store. Therefore it must be ensured that the App build generates a valid manifest with constantly increasing build numbers, which must also be stored in the manifest accordingly. After downloading from the *CCV*Store to the device, the Android installer is called by the store client and the update package is installed.

f) If an app shall be available for IM30 (unattended terminal) and for Mobile A920 or A77 (attended terminal), what differences regarding the app have to be taken into account for adjustment?

One app (one .apk) can be used for both attended and unattended CCV Android terminals when using the same Payment Engine. Our "mAPI Demo app" can serve as an example, which can be found on the CCV Developer Portal (we will provide the password after the NDA is singed and upon your request).

For creating an app for attended and unattended CCV Android terminals, please also consider the following hints:

- Regarding the flow, apps for the IM30 have to be created without navigation. Whereas apps for the A920 or A77 should offer navigation which is ensured by Paydroid.

- Apps for unattended terminals (e.g. IM30) will always be in kiosk mode. In kiosk mode the consumer/user of the terminal cannot leave the app, but the payment engine can still bring itself into the foreground. For attended terminals (e.g. A920) there will be no kiosk mode. If the kiosk mode is active or not this is steered by the Android settings. If you plan to use the app for both attended and unattended terminals you have to ensure that the layout still is displayed in the correct way in both modi.

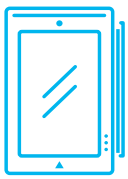
- For the IM30 apps always have to identify themselves as an Idle App or ECR App (=identifier in the code, please see this document under section 2.3 Specifications for the CCV IM30). For A920 or A77 such an identifier is not needed. If an app was initially developed for the A920 does not have these identifiers yet, it is only a small adjustment to include it. If an app was initially developed for the IM30 it has the identifier included. The identifier does not need to be removed if the app also runs on the A920 or A77.

- Ports: in general we highly recommend using the mAPI. mAPI can automatically detect the correct setting (autodetection of the ports) when using the autodetection feature (documentation on the CCV Developer Portal). As an alternative the ports can be addressed directly. For acCEPT the port by default is 4100/4102. For SecPOS Evo depending on the explicit use case different ports have to be used, which are explained in more detail under the chapter 2.2).

- The A920 has an integrated printer (A77 and IM30 don't). CCV provides the "printerservice" to address the printer (more on this see chapter 2.2). If the mAPI from CCV is used for the integration and the "printerservice" for printing, the .apk can be used for both – A920 with SecPOS Evo and acCEPT. If OPI or ZVT is used for the integration and the printer shall be addressed, this means that two .apks have to be delivered – one for Sec POS Evo Terminals and one for acCEPT terminals.

- The card reader interface for an A920 using the SecPosEvo payment engine is currently not available on IM30 but will be added in the future.

- The IM30 works with the Payment Engine acCEPT. The A920 / A77 works with both acCEPT or Sec POS Evo. In order to address a different payment engine, a different IP address must be used. Either the partner can create the app in a way so that the IP address can be adjusted remotely by using a script. If this is not possible, one .apk with the corresponding IP address for acCEPT and one .apk for Sec POS Evo must be delivered - provided the app partner wants the application to be available in connection with both payment engines



g) When is logging compulsory?

If you are using mAPI, then for certification the application must have enabled logging to the console for the mAPI library, using the MPALogger. An explanation is available on mAPI-developer side https://developer.myccv.eu/documentation/payment_terminals/local_-_android_sdk/guide/logging/
We highly advise to have this also active for productive set-up, so that CCV can provide more effective troubleshooting, using the Android-logcat logging (retrievable over the *CCV*Store).
For IM30: if you are creating an Idle Application (or using ZVT, OPI or MBD) logging is not necessarily required by CCV.

h) What has to be considered regarding cash register apps and the registration with the tax authorities? (“KassenSichV”, “Registrierkassenpflicht”)

It is the responsibility of the app provider to activate the TSE on the solution.We generally recommend to use a cloud-TSE. For SD-card solution: It is common to activate the TSE unit within the cash register app/within the solution (due to security reasons no browser is allowed on CCV Payment Terminals).

i) What is WebView and which version to use?

The WebView component is a separate APK, which has to be installed first. We can push it via the *CCV*Store to your debug terminal. Depending on the terminal, there are minimum requirements regarding the OS and CRP versions: A920: PayDroid_7.1.2_Aquarius_V02.5.06_2019112
IM30: PayDroid_7.1.1_Taurus_V05.3.07_20200804, comes WebView version 52.0.2743.100; A more recent WebView will be released in future, probably 86.0.4240.114

j) Can two 3rd-party apps on Android terminals communicate with each other?

Our Android terminals have the same behaviour as other Android devices. One 3rd-party app can can bring another 3rd-party app to the foreground and exchange data (e.g. via deeplink).

2.2 Questions for the CCV Mobile A920 and A77

a) Are there any specifications/requirements for printing?

For addressing the built-in printer on the A920 (with SecPOS Evo or with acCEPT) please use our "printerservice". You will find the whitepaper/coding examples in the "download section". The password for the download section are sent to you after the NDA was singed and around the time, when you receive your debug-terminal.

For other ways to realize printing, please contact us before you start the development, since there are restrictions and ways of realization which are not supported by CCV A920. E.g. ESC POS (via Bluetooth) cannot be used, because the printer of the A920 then cannot print "Umlaute" which are used in German language. Also for printing please mind, that the **Neptune API is strictly not allowed** by CCV (apps with Neptune API do not pass the app test).

The IM30 and the A77 do not have an in-built printer. Alternatively, CCV e-receipt solution can be used.

b) Will all communication be routed via the local host or are there any specific cases?

Yes, all communication is routed via localhost, if the payment engine (SECpos Evo software) on the A920 and the POS application (cash register system) sends the OPI-request.

c) What needs to be taken into account regarding the foreground-background mechanism (of the App and the Payment Engine) in connection with ports?

1) For SecPOS Evo

		Internal cash register (app directly on terminal)	External cash register
ZVT	SecPOS stays always in the foreground	20007	20007
	SecPOS automatically withdraws into the background after transaction	40007	40007 (for SecPOS Evo version 68.07 and higher)
mAPI (based on OPI)	If mAPI is used, the AutomaticTerminalDiscovery feature is compulsory		
OPI	SecPOS stays always in the foreground	20002/20007	20002/20007
	SecPOS automatically withdraws into the background after transaction	20002/20007	20002/20007

2) acCEPT

We strongly advise that you use the mAPI. The mAPI offers the AutomaticTerminalDiscovery. As alternative the default port is 4100/4102.

2.3 Specifications for the CCV IM30

a) What should I consider regarding the debug device before getting started?

Although it is common practice for development/testing for smart phones, PLEASE DO NOT RESET OR DELETE THE APAS PP APP from the IM30 (not locally and also not via *CCV*Store). This will cut the connection with the acCEPT payment platform and makes the terminal inoperable. This cannot be fixed remotely. If it happens we have to send you a new debug device.

b) In what case and how to register an app as an ECR or idle app?

In external ECR mode (ECR= electronic cash register), payment transactions are initiated by an externally located ECR (e.g. over LAN). If no payment transaction is taking place the terminal payment application either display a static screen or it calls an Idle Screen App to the foreground.

In internal ECR mode, the role of the ECR is fulfilled by an on-device Android app. An ECR app handles the responsibility of starting a payment transaction and fulfilling the role of a vending machine controller (VMC). It communicates with the payment application via an ECR protocol (e.g. OPI) over the loopback IP interface. In internal ECR mode, the terminal displays the ECR app when not processing a payment transaction. ECR apps are given preference over Idle Screen apps. If an ECR app is loaded onto a CCV Android terminal, no Idle Screen app is called.

Idle Screen and ECR apps are always launched by the payment application by calling the Intent. When called, the app shall launch an Activity in the foreground. The Idle Screen app continues to run until the payment application moves itself back to the foreground (e.g. at the start of a new payment transaction). Idle Screen apps should therefore stop running when Android calls **onPause()** on its Activity. ECR apps are also moved to the background but should continue performing their business logic. Idle Screen apps register themselves on CCV Android payment terminals using the following declaration in their app manifest:

<intent-filter> <action android:name="eu.ccv.service.IdleScreen" /> <category android:name="android.intent.category.DEFAULT" /> </intent-filter>

ECR apps register themselves on CCV Android payment terminals using the following declaration in their app manifest:

<intent-filter> <action android:name="eu.ccv.service.ECR" /> <category android:name="android.intent.category.DEFAULT" /> </intent-filter>

c) What is displayed if no transaction id processed?

For APAS PP versions 24.06.01, 23.23.01, 25.01.01 starting form end of November 2021:

As a default setting, APAS PP prevents the Android device to fall into sleep mode (=black screen). If no transaction is being processed and an idle mode app (own app or CCV Media App) is on the terminal, then the defined content of the idle app is shown. If an ECR-App is on the terminal, the ECR app is shown when no transaction is conducted. If no additional app is used, but there is an external ECR, a static screen is shown.

If you wish that the terminal does fall into sleep mode and shows a black screen after some minutes or after the transaction (e.g. because you are working with a second screen and don't want that people get distracted/should focus on this second screen), please actively inform your dedicated Project Engineer. We have to configure that the terminal does fall into sleep mode in APAS PP and have to push it to your terminals.
Before November 2021 and for all other APAS PP versions mentioned above:

For a terminal integration via MDB or for non-idle-mode applications the terminal will go into low power mode (black screen) after one minute if the terminal is not used. If the app is an idle-mode application and the terminal is not used,the terminal first will display the defined content of the idle state. Also here the terminal display will turn black after a couple of minutes. For all three cases: if you wish to prevent that the terminal display turns black, please use the awakelock. Information about the awakelock can be found here https://developer.android.com/training/scheduling/wakelock#wakeful

d) How can I realize a payment integration (apasPP / acCEPT)?

Please contact us with your detailed questions. We are happy to support.

e) How can the hardware of the IM30 be accessed?

- GPIO: by using UPTApi. The GPIO/UptAPI is only allowed if not influencing the power management of the device. Further information see classes pax.util.piio.Gpio and pax.util.pio.PeripheralManager. We can also provide example code, which is stored on a separate file share. Please contact us for access.
- Camera: via standard Android SDK
- USB: via standard Android SDK
- AUX (Audio): standard Android SDK
- Card Readers: Reader Interface not implemented in ApasPP yet
- SAM Reader: Reader Interface, not implemented in ApasPP yet
- For access to the serial port / RS232: as a reference/example we provide a Portmanager package via a separate file share.

f) How to implement a MDB slave device?

If the app needs access the the mdbdispatcher to implement a MDB slave device we will provide it, once you start working on it.

g) What are the ISO standards of NFC cards that the terminal can read?

ISO 18092, ISO 14443 A&B and ISO 15693. For all three standards the terminal needs to have a software handling these cards as well. The PayDroid OS or even the payment software can't handle such cards.

h) Why is it mandatory to include automatic Payment Recovery?

It's possible that the connection with the payment terminal is lost and that the terminal doesn't send a reply back during an electronic payment. When this happens, the payment executor doesn't know whether the transaction was successful or failed and the executer needs to recover the real status. Information to perform a payment recovery to determine the status of the last performed transaction can be found here: https://developer.myccv.eu/documentation/payment_terminals/local_-_android_sdk/guide/payment-recovery/

i) In what format must images and information be designed?

Android supports image formats like JPEG, GIF, PNG und BMP. It also depends on the application to be developed. There are apps which can implement a TIFF Viewer and then you are also able to show TIFF formats in the app.

j) If images and information about products (name, price etc.) are shown in the app, how can these be uploaded?

The business application has to function as an internal ECR (electronic cash register) application.

k) Who needs to feed the device with the new information when the assortment changes in the machine?

That needs to be updated in the business application and the developer should determine if this change is updated locally by a mechanic or remotely in a cloud environment. We suggest the later.

l) What appears on the display when a product is selected on the machine and on the keypad?

That depends on the type of application that runs on the terminal, if this is an external ECR app then a price.

m) When does the video/picture (content of the idle app) pause? Exclusively during the payment process or can the user also activate certain functions by tapping the display?

This depends on the way the idle app is being developed. Is the vending machine switching to a menu selection with the proximity sensor or does it require a tap on the screen by the customer. In general, the payment app is leading and it will always overrule the idle app.

n) Idle app: What video formats can be used?

Videos can be displayed through an external idle app. You can connect it with a backend and manage and configure which videos should be shown in idle mode. All formats which are supported by Android 7.1 can be displayed (list of supported formats).

Because of the orientation of the device and display, portrait movies are preferred to make optimal use of the resolution of the screen (720 x 1280 pixels).

o) Idle app: How many different videos can be uploaded to the individual devices or what is the allowed maximum length of a video?

This is not limited to an amount of movies, but to the limitations of the memory of the terminal. The IM30 has a 8GB Flash memory which is partly already used by the PayDroid system. This means 5-6GB are still available. Regarding the length of a video there are no restrictions, but keep in mind the end customers' attention span.

p) What data connection is used?

LAN or 4G.

q) Can an app use Bluetooth and/or Bluetooth Low Energy (BLE)/Beacon technology?

Yes you can use those technologies. The interface should always be controlled via Android.

r) Is there a risk that the screen burning in if constantly a bright logo is displayed?

Generally every TFT has the burn in or memory risk. We recommend to use our CCV Media Display App or to implement/program a screen saver functionality in a way that the logo or picture is moving on the display as long as the terminal is not used.

s) What if the app 3rd-party app crashes?

- On a live terminal the "CCV launcher app" makes in default configuration all apps invisible when the Android home screen is displayed. All apps are managed/pushed/launched via *CCV*Store. So even if the 3rd party app crashes and the Android screen is visible for some seconds, a consumer cannot see and access apps. Debug devices (= special test and development devices for developers) don't have the launcher app on it, since the app makes development and testing more complicated. If a 3rd party app crashes and the Android home screen can be seen, here the installed apps might be visible.
- If a 3rd-party app crashes then Apas PinPad comes to the foreground. Apas PinPad recognized when no transaction takes place and starts the 3rd-party app again automatically (without manual initialization).
- In general a remote manual restart of the terminal can be done via *CCV*Store.
- In the *CCV*Store you can see if a terminal is online or offline and which apps are on the terminal. *CCV*Store cannot show in what state a 3rd-party app is. For this purpose please check if you can incorporate a watchdog principle within your app.

t) Can multiple 3rd-party apps be used on the IM30?

In contrary to attended terminals, for unattended live terminals in the field the user/consumer cannot go to the Android home screen and start another app by simply clicking on it. For unattended terminals either the payment engine or one specific app is always shown full screen (forced by Paydroid and kiosk mode). Reason for this is to ensure the same good user experience for every user.

The payment engine (APAS PP) is the master of the terminal and can only bring one specific app into the foreground. APAS PP cannot steer multiple apps regarding foreground-background management.

If more than one app besides APAS PP shall be displayed on the IM30, the apps have to communicate with each other (when which app has to call the other app to the foreground). The flow is important here. Alternatively, the additional functionality of the second app has to be implemented into the first external app (e.g. as plug-in). If your customer is interested in using multiple apps on the IM30 please always contact your dedicated CCV Projetct Engineer to get clarity on the specific apps and use case in order to provide a solution/advice what is possible and what not.

u) Why is there no microphone integrated in the IM30?

As soon as it comes to voice interaction, the environment is having an essential impact on the hardware used. Therefore, the IM30 provides the AUX port with the normal headset connector. This way integrators can select the microphone which suits best for the specific location and can also chose the ideal position at the machine.