https://verifone.cloud/docs/application-development-kit-version-48/pg_tec_programmers_guide
Updated: 13-Jun-2025

ADK-TEC Programmers Guide

This document is for programmers and developers who want to understand and use the ADK-TEC.

Audience

This guide provides all the information required for application developers to integrate and utilize the functionality of the ADK-TEC.

Organization

This guide is organized as follows:

Introduction: Provides a summary of ADK-TEC.

Use Cases: Presents typical flows.

Getting Started: Presents an introduction in ADK-TEC usage.

Programming: Supplies ADK-TEC programming information.

System Setup and Requirements: Supplies information about required dependencies.

PP1000: Supplies information about pairing and PIN transfer with PP1000.

Troubleshooting: Gives solutions for possible issues in ADK-TEC.

Appendix: Links to related documents.

Introduction

ADK-TEC provides **technology selection** functionality. Supported technologies are

- Magstripe cards
- EMV contact chip cards
- EMV contactless cards and mobile phones
- Contactless NFC cards (Mifare, Felicy, ISO, ...)
- Contactless Value Added Service (VAS) solutions

To make use of ADK-TEC you need the following components:

- tec.h
- libtec.so

Additionally you need

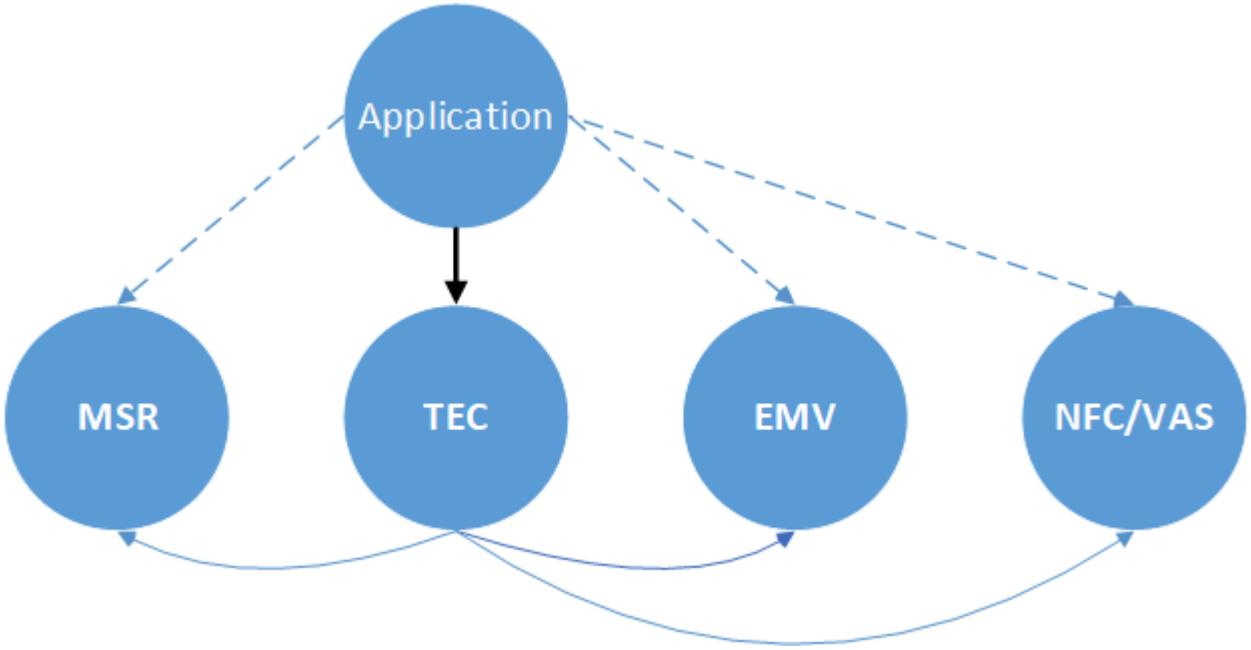
- ADK-MSR, see ADK-MSR Programmers Guide
- ADK-EMV, see
  - ADK-EMV Contact Programmers Guide
  - ADK-EMV Contactless Programmers Guide
- ADK-NFC, see ADK-NFC Programmers Guide

Flow Overview

On **terminal startup** the application needs to setup the components (see below picture, dashed lines):

Package name	Description
`tec-doc-x.x.x-xx.zip`	Documentation

ADK-MSR	ADK-EMV Contact	ADK-EMV Contactless	ADK-NFC
MSR_SetOptions() if desired	EMV_CT_Init_Framework()	EMV_CTLS_Init_Framework()	NFC_Client_Init()
	EMV_CT_SetTermData()	EMV_CTLS_SetTermData()	NFC_Terminal_Config(
	EMV_CT_SetAppliData()	EMV_CTLS_SetAppliDataSchemeSpecific()	NFC_VAS_UpdateConf
	EMV_CT_StoreCAPKey()	EMV_CTLS_StoreCAPKey()	



Transaction flow:

- Application prepares components (dashed lines):
  - ADK-EMV Contactless

        - if not CTS_PURE_CARD_DETECTION: EMV_CTLS_SetupTransaction()
    - ADK-NFC if VAS desired: NFC_VAS_PreLoad()
- Application starts card detection
    - invoke cts_StartSelection() (black line)
    - ADK-TEC starts a thread to poll the involved components (solid blue lines)
- User swipes, inserts, or taps a card (or mobile phone)
    - ADK-TEC gets notification about this event (solid blue lines)
- ADK-TEC signals card detection to the application (black line)
- Application completes the transaction (dashed lines)
    - if a magstripe card was swiped:
        - use ADK-MSR functions to fetch the read data
    - if a card was inserted:
        - use ADK-EMV Contact to make the transaction
    - if a card (or mobile phone) was tapped:
        - if CTLS EMV is signaled:
            - transaction was already done
            - use ADK-EMV Contactless to fetch the results
        - if CTLS NFC or VAS is signaled:
            - use ADK-NFC to realize the desired APDUs
            - if desired (and possible): do EMV transaction by means of ADK-EMV Contactless

Android

In Android the ADK-TEC is hidden inside SDI. So application has to use the "Card Detection (23-01)" from ADK-SDI Programmers Guide.

VOS3

In VOS3 ADK-TEC is accessible through SDI. The ADK-TEC-interface is rebuild in ADK-SDI-Client Programmers Guide.

Two-Piece Solution

Former Two-Piece Solution with Client/Server architecture for MSR, TEC, ... is not supported anymore.

SDI EPP has to be used instead (see ADK-SDI Programmers Guide).

Use Cases

Magstripe and EMV Contact, on Contactless EMV and transparent ISO APDU

Requirement:

All three technologies shall be detected: Magstripe Card Swipe, Contact Chipcard Insertion, Contactless Card (or Mobile Phone) Tap.

On *Contactless interface an EMV payment* shall be done.

Flow (see below diagram):

If not CTS_PURE_CARD_DETECTION: Before EVERY start the application has to set up EMV Contactless (call EMV_CTLS_SetupTransaction()).

To start technology selection the application calls cts_StartSelection() with CTS_CHIP, CTS_MSR, and CTS_CTLS.

TEC component will start a thread to poll for swipe, insert or tap.

Once a card (or mobile phone) is detected the application gets the result by means of cts_WaitSelection().

Parameter `usedTechnology` informs which technology is used:

1. CTS_MSR: A magnetic card was swiped.

   Application has to read the magstripe data (with help of ADK-MSR Programmers Guide) and process transaction.
2. CTS_CHIP: A contact chip card was inserted.

   Chip may already be activated (depending on CTS_NO_POWERON).

   Application has to execute the chip transaction by means of ADK-EMV Contact Programmers Guide.
3. CTS_CTLS: A contactless card or mobile phone was tapped (see ADK-EMV Contactless Programmers Guide).
   - if CTS_PURE_CARD_DETECTION:

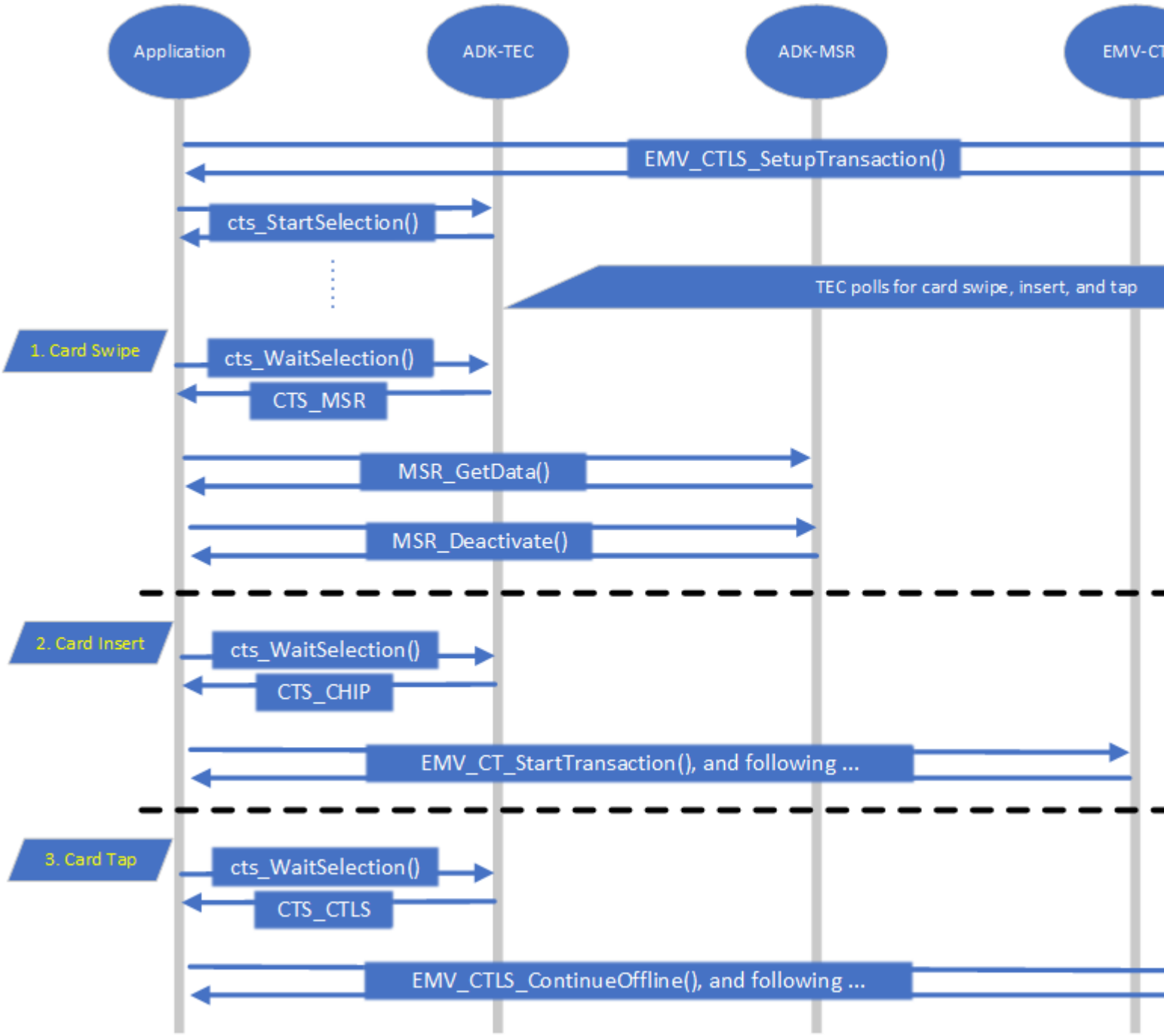
     Chip card was activated.

     Application can communicate with chip by means of EMV transparent commands (EMV_CTLS_SmartISO() and others).

     And may do a EMV transaction starting with EMV_CTLS_SetupTransaction().
   - else:

     The complete EMV transaction was already done.

     The application has to fetch the results with EMV_CTLS_ContinueOffline().



Magstripe and EMV Contact, on Contactless NFC

Requirement:

All three technologies shall be detected: Magstripe Card Swipe, Contact Chipcard Insertion, Contactless Card (or Mobile Phone) Tap.

On *Contactless interface several card types* shall be handled: MiFare, Felica, EMV, and others.

Flow (see below diagram):

To start technology selection the application calls cts_StartSelection() with CTS_CHIP, CTS_MSR, and

CTS_CTLS.

Additionally CTS_NFC_ENABLE and options[12..15] (NFC technologies) have to be set.

TEC component will start a thread to poll for swipe, insert or tap.

Once a card (or mobile phone) is detected the application gets the result by means of cts_WaitSelection().

Parameter `usedTechnology` informs which technology is used:

- CTS_MSR: A magnetic card was swiped.

  Application has to read the magstripe data (with help of ADK-MSR Programmers Guide) and process transaction.
- CTS_CHIP: A contact chip card was inserted.

  Chip may already be activated (depending on CTS_NO_POWERON).

  Application has to execute the chip transaction by means of ADK-EMV Contact Programmers Guide.
- CTS_CTLS + CTS_DATA_TLV: A contactless card or mobile phone was tapped.

  Parameter `dataBuffer` contains NFC related data in TLV format, see TEC result data tags.

  Application shall use ADK-NFC Programmers Guide to realize the desired functionality, e.g.:
  - Felica
  - Mifare
  - APDU exchange

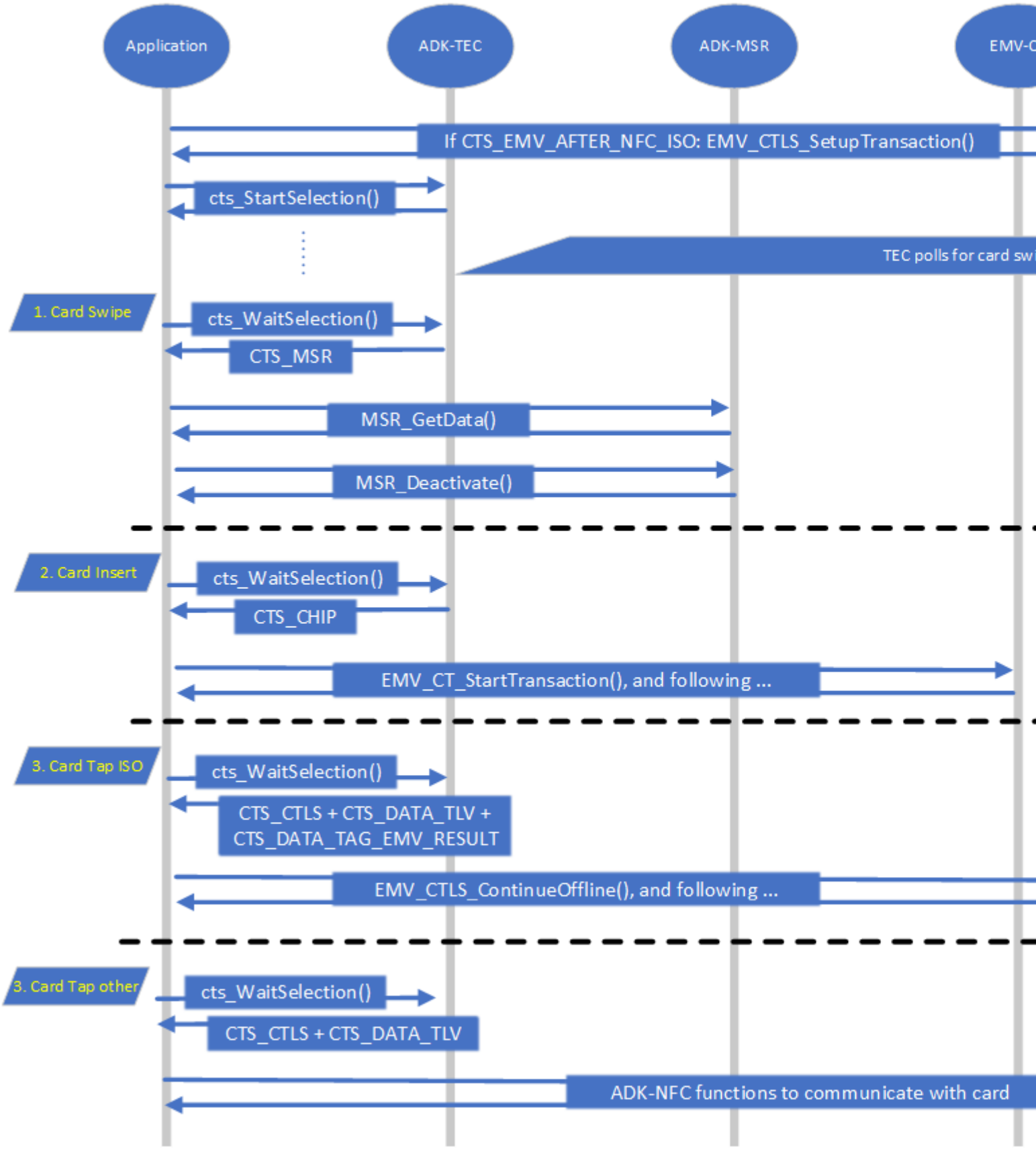
Automatically perform an EMV Contacless transaction:

The application may set CTS_EMV_AFTER_NFC_ISO.

In that case it also needs to call EMV_CTLS_SetupTransaction() before cts_StartSelection().

ADK-TEC will perform an EMV contactless transaction if an ISO14443 card is tapped.

If another card is used the application will get the above mentioned NFC results.



Magstripe and EMV Contact, on Contactless EMV and VAS (Value Added Services, Wallets)

Requirement:

All three technologies shall be detected: Magstripe Card Swipe, Contact Chipcard Insertion, Contactless Card (or Mobile Phone) Tap.

On Contactless interface it's needed to handle *Value Added Services (VAS) Wallets* and EMV payments.

Flow (see below diagram):

Application has to prepare ADK-NFC by means of NFC_VAS_PreLoad() (see ADK-NFC Programmers Guide).

If the used VAS config includes "Pay" (= EMV) than EMV_CTLS_SetupTransaction() is needed.

To start technology selection the application calls cts_StartSelection() with CTS_CHIP, CTS_MSR, and CTS_CTLS.

Additionally CTS_VAS_ENABLE has to be set.

TEC component will start a thread to poll for swipe, insert or tap.

Once a card (or mobile phone) is detected the application gets the result by means of cts_WaitSelection().

Parameter `usedTechnology` informs which technology is used:

- CTS_MSR: A magnetic card was swiped.

  Application has to read the magstripe data (with help of ADK-MSR Programmers Guide) and process transaction.
- CTS_CHIP: A contact chip card was inserted.

  Chip may already be activated (depending on CTS_NO_POWERON).

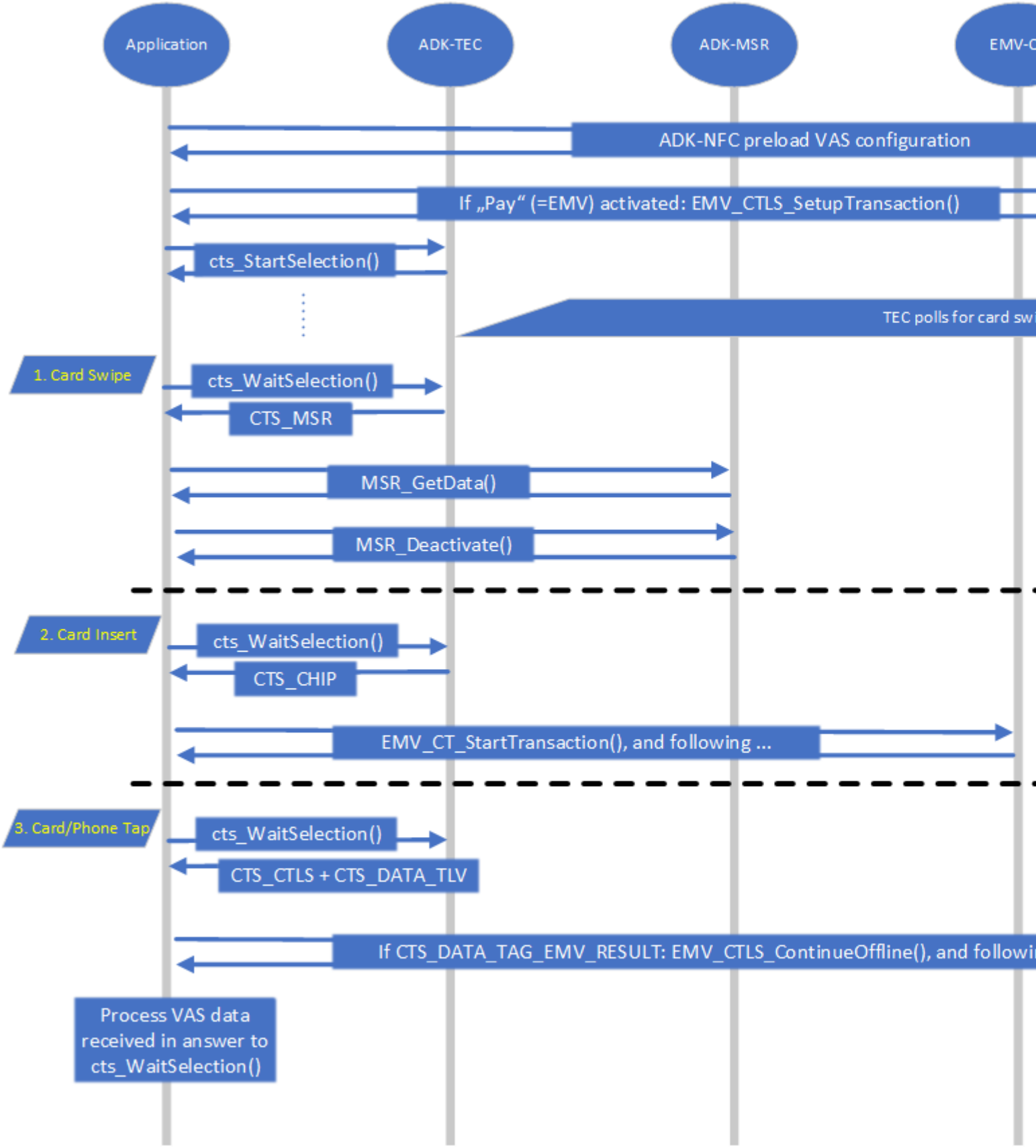
  Application has to execute the chip transaction by means of ADK-EMV Contact Programmers Guide.
- CTS_CTLS + CTS_DATA_TLV: A contactless card or mobile phone was tapped.

  Parameter `dataBuffer` contains VAS related data in TLV format, see TEC result data tags.

  If result data contains CTS_DATA_TAG_EMV_RESULT the TEC processed a complete EMV transaction.

  Application has to use ADK-EMV to collect the results.

  Afterwards it can process the VAS data as needed.



Getting Started

The following two examples show how to use technology selection (ADK-TEC):

Sample1: Application using ADK-TEC without callback

```
#include "tec/tec.h"#include "msr/msr.h"#include "EMV_CT_Interface.h"
#include "EMV_CTLS_Interface.h".... a) MSR, CT, CTLS EMV only
// initialize:EMV_CT_InitFramework(...);EMV_CTLS_InitFramework(...);// setup tra
```

Sample2: Application using ADK-TEC with callback

```
#include <pthread.h>#include "tec/tec.h"#include "msr/msr.h"#include
"EMV_CT_Interface.h"#include "EMV_CTLS_Interface.h"
....static pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;static pthread_cond_
100];static  unsigned short data_len = sizeof(data);static  int
 ret; static void tecselCallback(void *data){  pthread_mutex_lock(&mutex);  ret =
0
)  pthread_cond_signal(&condition);  pthread_mutex_unlock(&mutex);} .... a) MSR,
// initialize:EMV_CT_InitFramework(...);EMV_CTLS_InitFramework(...);// setup tra
```

Link your application with libtec.so and load the shared library libtec.so on your device. If you don't want to periodically call cts_WaitSelection() you can supply a callback function to cts_StartSelection(). This callback function is called exactly once after the technology selection has finished (due to detected technology, timeout, or error). After this callback function has been called (or even withing the callback function) you can obtain the result by cts_WaitSelection() setting its timeout to 0. A callback function is supported for card removal detection as well (see cts_WaitCardRemoval()).

Programming

Programming and API Principles

The API consists of the following functions:

tec.h
cts_Version()
cts_SetTraceCallback()
cts_SetOptions()
cts_StartSelection()
cts_StopSelection()
cts_WaitSelection()
cts_RemoveTechnologies()
cts_AddTechnologies()
cts_WaitCardRemoval()
cts_WaitCardRemoval2()
ped_SetSendRcvCb()
ped_Pairing()

ped_MovePin()
cts_SetNotificationCallback()

Some notes regarding the different technologies

In general only one of CTS_MSR, CTS_CHIP, CTS_CTLS is detected but in special cases (see Detecting MSR and CTLS simultaneously and Special behavior on hybrid readers (UX30x)) two technologies can be detected at once.

- CTS_MSR: If you want to use the magnetic card reader, you do not need to call MSR_Activate() before starting technology selection. ADK-TEC will do this for you. After technology selection finishes and the detected technology is not CTS_MSR, MSR_Deactivate() is internally called as well. So you do not need to do this either. Only if the detected technology is CTS_MSR, ADK-MSR is still activated to allow the application to fetch the magnetic card data with MSR_GetData(). After this you shall call MSR_Deactivate(). If using an UX device MSR_Deactivate() shall be called as well if technology selection detects CTS_CHIP or returns CTS_NO_CHIP (see Special behavior on hybrid readers (UX30x)).
- CTS_CHIP: If you want to use the chip card reader, you should first call EMV_CT_InitFramework() to enable the contact part of ADK-EMV. After technology selection detects a chip card the card is already powered up (except if you set option CTS_NO_POWERON) and the application can call EMV_CT_ContinueOffline().
- CTS_CTLS: (EMV only, see below for NFC)
  - Card detection only (option CTS_PURE_CARD_DETECTION)

    ADK-TEC activates the card by means of EMV_CTLS_SmartReset().

    So calling application can continue to work with the card by EMV_CTLS_SmartISO().

    And finally it shall call EMV_CTLS_SmartPowerOff() to switch off the RF field.

    In case an EMV transaction is desired the EMV_CTLS_SmartPowerOff() has to be called.

    And then EMV_CTLS_SetupTransaction() and EMV_CTLS_ContinueOffline().
  - EMV transaction

    First call EMV_CTLS_InitFramework() and prior to each technology selection you have to call EMV_CTLS_SetupTransaction().

    ADK-TEC will internally call EMV_CTLS_ContinueOffline() to detect the card and perform the transaction.

    If a contactless card is detected, the application can call EMV_CTLS_ContinueOffline() again to obtain the transaction results.

    If no contactless card is detected, ADK-TEC internally calls EMV_CTLS_Break().

    One additional remark regarding EMV_CTLS_SetupTransaction(): If ADK-TEC is used, you must not set parameter ServerPollTimeout because in this case ADK-TEC takes care of polling.

Processing NFC with ADK-TEC

This is the general routine used in ADK-TEC for detecting and processing CTLS cards (pseudocode), it should help you to understand how ADK-TEC behaves depending on the various CTLS options.

```
[0] if both CTS_NFC_ENABLE and CTS_VAS_ENABLE are set:        exit    end[1
] if CTS_NFC_ENABLE is set:        call NFC_PT_Polling()        if
 ISO A/B card found and CTS_EMV_AFTER_NFC_ISO is set:           goto [3
]        end        exit    end[2] if CTS_VAS_ENABLE is set:        call
 NFC_VAS_Activate()        if VAS_DO_PAY is returned:           goto [3
]        end        exit    end[3] if CTS_PURE_CARD_DETECTION is set
:        call EMV_CTLS_SmartReset()    else        call
 EMV_CTLS_ContinueOffline()    end
```

If it is possible that EMV_CTLS_ContinueOffline() is called by ADK-TEC, application has to call EMV_CTLS_SetupTransaction() before starting technology selection. If ADK-TEC detects a card with NFC_PT_Polling() and no subsequent EMV transaction is started, ADK-TEC keeps the RF field on to allow the application to communicate with this card. In this case the application has to call NFC_PT_FieldOff() and NFC_PT_Close() afterwards. Furthermore the first CTLS LED is left on by ADK-TEC in this case. The application generally wants it to shine while communicating with the card or even wants to switch on further LEDs. So as soon as the application has finished the transaction, it needs to switch off the LEDs or restart idle blinking.

Detecting MSR and CTLS simultaneously

After CTS_CTLS has been detected technology selection can wait a certain amount of time for CTS_MSR before returning the result to the application. If a magnetic card is swiped within this period of time technology selection will return CTS_CTLS|CTS_MSR as technology. The timeout can be configured by the options parameter of cts_StartSelection().

Special behavior on hybrid readers (UX30x)

Hybrid readers have a single slot for handling magstripe and contact chip.

ADK-TEC provides special functionality to help the application in making the decision which technology to choose.

ADK-MSR Configuration

When using an UX30x device it is strongly recommended to activate the MSR UX enhancements (see ADK-MSR Programmers Guide):

```
unsigned char options[] = { MSR_UX_ENHANCEMENTS };MSR_SetOptions(options, sizeof
```

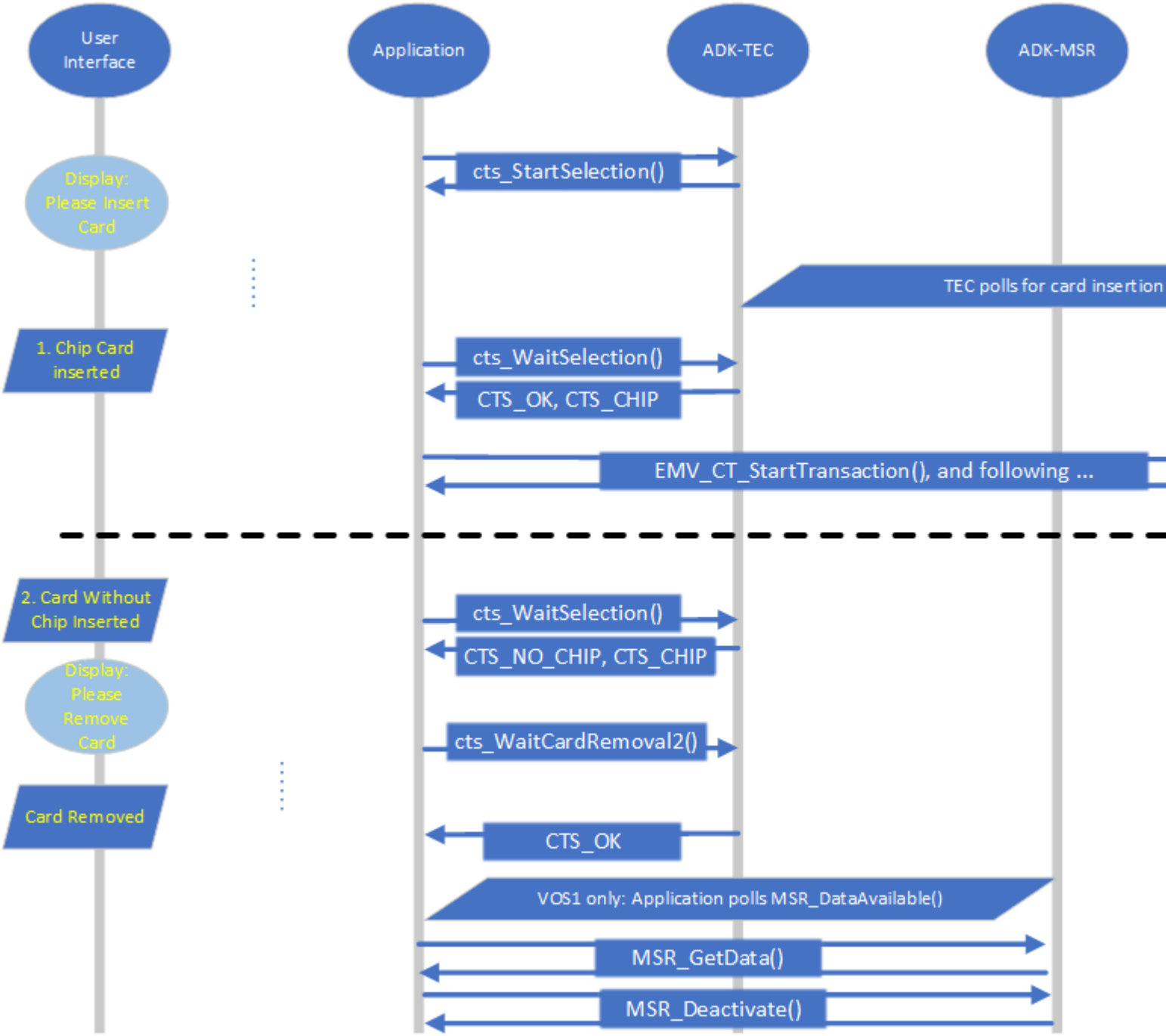
This has to be done only once, before the first call of cts_StartSelection(). These enhancements will prevent MSR from reading the magnetic card on insertion.

If the MSR UX enhancements are not activated, the following remarks are valid as well. Additionally it is possible that technology selection detects both CTS_MSR and CTS_CHIP in parallel.

Use Case: Technologies Contact Chip and Magstripe are supported

cts_StartSelection() is called with requesting technology CTS_CHIP (and CTS_MSR).

CTS_CTLS may or may not be activated.



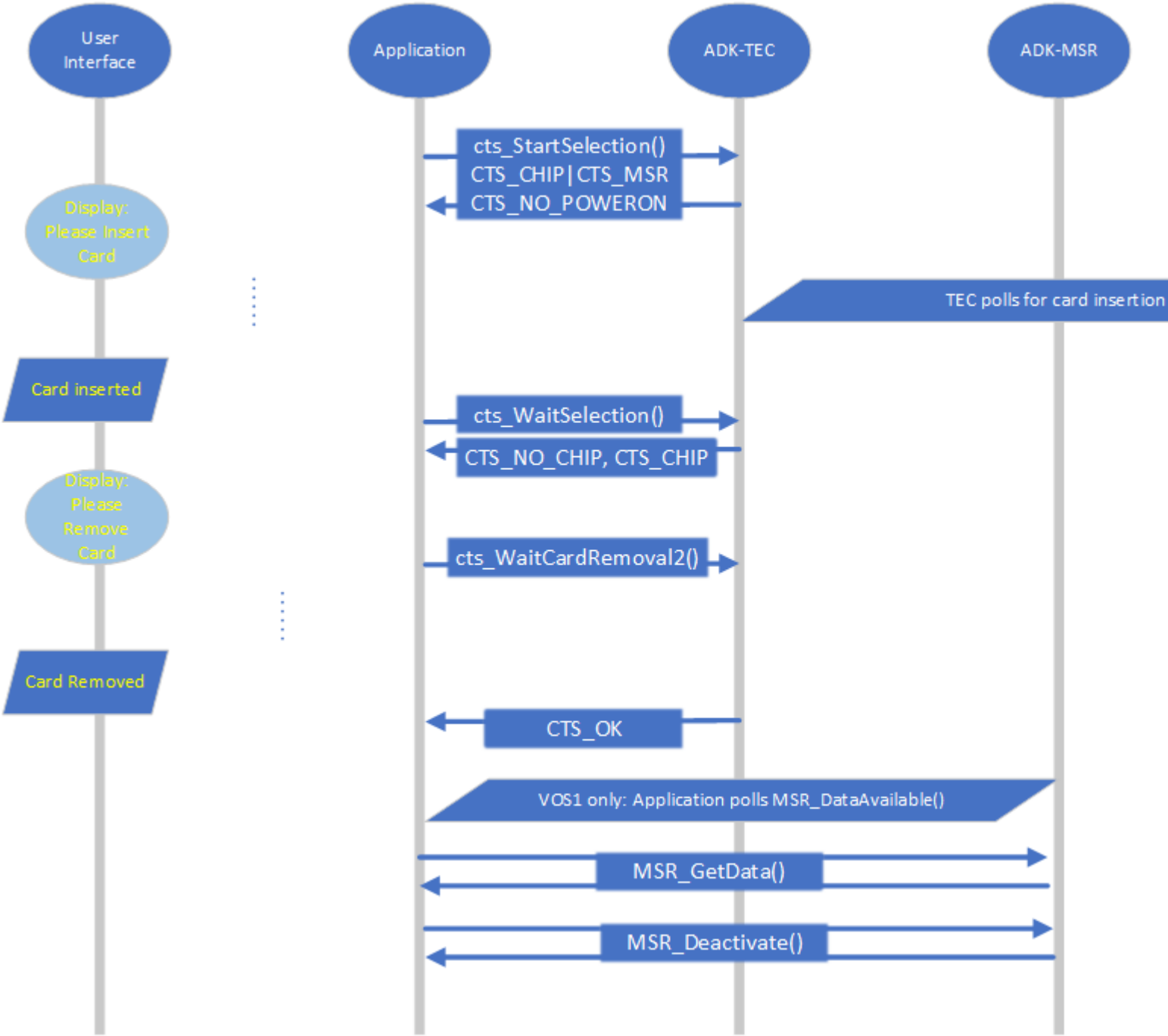
Use Case: Technology Magstripe supported but Contact Chip NOT

cts_StartSelection() is called with requesting technology CTS_CHIP (and CTS_MSR).

CTS_CHIP is necessary to be able to detect the card insertion.

CTS_NO_POWERON shall be set to avoid activation of chip card.

CTS_CTLS may or may not be activated.



ADK-TEC legacy timeout handling

Above shown diagrams are recommended handling.

For downward compatibility the following functionality is still supported.

- **Let ADK-TEC wait for MSR data read on card removal**

This scenario assumes that technologies CTS_CHIP and CTS_MSR are supported.

The application may set a timeout to cts_StartSelection() (options[8..9]).

Technology selection waits for this amount of time for MSR data from card removal.

This functionality is not possible when using CTS_NO_POWERON (so not for SDI and vos3)

- If a **chip card** is inserted it goes the same way as usual:

  cts_WaitSelection() returns CTS_OK and usedTechnology CTS_CHIP

  The application shall use EMV_CT_StartTransaction() etc. to process an EMV contact transaction.
- If a **card without chip** is inserted

  ADK-TEC waits for the above mentioned timeout for card removal (and reading magstripe data)
  - If there is MSR data

    cts_WaitSelection() returns CTS_OK and usedTechnology CTS_MSR

    Application shall
    - call MSR_GetData() to obtain the MSR data
    - call MSR_Deactivate()
  - if not

    cts_WaitSelection() returns CTS_NO_CHIP and usedTechnology 0.

    In that case application
    - may
      - check if card is still inserted, if yes:
      - ask the cardholder to remove the card
      - wait for MSR data by polling MSR_DataAvailable()
      - call MSR_GetData() to obtain the MSR data
    - must (in any case)
      - call MSR_Deactivate()

- **Let ADK-TEC notify on card insertion**

  Detect card insertion, inform application about this event and wait for MSR data read on card removal.

  cts_StartSelection() is called WITHOUT requesting technology CTS_CHIP

  CTS_CTLS may or may not be activated.

  - **Card insertion notification callback**

    cts_SetNotificationCallback() can be used to register the callback

CTS_NOTIFICATION_CBK_TYPE_UX_CARD_INSERTED

That's invoked in case a card is inserted.

This functionality may be used to realize a cardholder display "please remove card".

It's only available in case one of the below mentioned timeouts is set.

- **Card Insertion Timeout**

  The application may set a **timeout** to cts_StartSelection() (options[8..9]).

  After card insertion the ADK-TEC waits for this amount of time for MSR data (from card removal).
  - If there is MSR data

    cts_WaitSelection() returns CTS_OK and usedTechnology CTS_MSR

    Application shall
    - call MSR_GetData() to obtain the MSR data
    - call MSR_Deactivate()
  - if not

    cts_WaitSelection() returns CTS_NO_CHIP and usedTechnology 0.

    In that case application
    - may
      - check if card is still inserted, if yes:
      - ask the cardholder to remove the card
      - wait for MSR data by polling MSR_DataAvailable()
      - call MSR_GetData() to obtain the MSR data
    - must (in any case)
      - call MSR_Deactivate()

- **Card Removal Timeout**

  Avoid long delay after card removal in case used card does not have a magstripe.
  - By means of cts_SetOptions() the CTS_OPTION_TAG_UX_MSR_TIMEOUT can be set.
  - It's only effective in case the above mentioned timeout (cts_StartSelection() options[8..9]) is deactivated.
  - Timeout is started after card removal.
  - If MSR data is read (before timeout expiry)

    cts_WaitSelection() returns CTS_OK and usedTechnology CTS_MSR

    Application shall
    - call MSR_GetData() to obtain the MSR data

        - call MSR_Deactivate()
    - If not (timeout expiration)

      cts_WaitSelection() returns CTS_UX_MSRDATA_NOT_AVAILABLE_TIMEOUT and usedTechnology 0

      In that case application
        - may
            - check if card is still inserted, if yes:
            - ask the cardholder to remove the card
            - wait for MSR data by polling MSR_DataAvailable()
            - call MSR_GetData() to obtain the MSR data
        - must (in any case)
            - call MSR_Deactivate()

System Setup and Requirements

Compiler and Linker Settings

include tec.h and link libtec.so

libtec defines it's dependencies to other libs in needed section (you can show that by `"objdump -p libtec | grep NEEDED"`)

Hardware

ADK-TEC is hardware platform agnostic and supports installation on V/OS and VOS2 terminals.

Software

ADK-TEC is designed to be platform agnostic and will be supported on V/OS and VOS2 terminal operating systems.

Deliverables and Deployment

Packages delivered (x - version number digit):

Package name	Description
`tec-doc-x.x.x-xx.zip`	Documentation
`tec-vos-dev-x.x.x-xx.zip`	VOS **development** package, to be installed in PC build environment
`tec-vos2-dev-x.x.x-xx.zip`	VOS2 **development** package, to be installed in PC build environment

PP1000

Pairing and PIN transfer with PP1000

ADK-TEC is capable of performing pairing a countertop device (CTP) with a PP1000 device and transferring the PIN entered at the PP1000 into the vault of the countertop device. On the PP1000 you only need to install the current AQUILA version, no ADK-TEC component is running on the PP1000. The application running on the CTP needs to include the header file ped.h which is provided by ADK-TEC. Within this file the three functions ped_SetSendRcvCb(), ped_Pairing(), and ped_MovePin() are declared. If you call one of these functions you have to additionally install the library libPP1000.so on the CTP. This library is shipped together with ADK-TEC. ped_Pairing() pairs the two devices. The actual pairing is only to be done once. However, if one of the devices is paired with a third device in between, the devices must be repaired. ped_Pairing() first checks if the two devices are successfully paired and performs the pairing only if this is necessary.

If the pairing is successful, a PIN can be transferred from PP1000 to CTP. The function ped_MovePin() does not collect the PIN on the PP1000, so the PIN entry must be triggered by the application. It can directly send the commands to the PP1000 or use the function pp1000_acceptPin() which is provided by libPP1000. After the PIN has been entered, the PIN can be transferred into the vault of the CTP by calling ped_MovePin(). If this is successfully done, the application can proceed as usual, e.g. call EMV_CT_Send_PIN_Offline() if this is an offline PIN.

The communication between PP1000 and CTP has to be handled on application level. Both ADK-TEC and PP1000 lib are platform independent and do not have communication built in. The application has to call either ped_SetSendRcvCb() (provided by ADK-TEC) or pp1000_registerComs() (provided by PP1000 lib) to set functions that send and receive data to/from the PP1000. So the application can freely decide which communication method it wants to use, e.g. you may use ADK-COM or directly call OS functions.

Troubleshooting

Frequently Asked Questions

Q: cts_WaitSelection->timeout_msec: What is the purpose of this timeout if compare with cts_StartSelection->timeout_sec? Provide use cases.

A: cts_StartSelection->timeout_sec is the timeout for the whole technology selection process, e.g. 30 seconds might be reasonable value. cts_WaitSelection->timeout_msec is the timeout for the cts_WaitSelection() function. It blocks and returns only if the timeout expires (in this case CTS_IN_PROGRESS is returned) or a result is available (something != CTS_IN_PROGRESS is returned). The timeout value to use here depends on your application design. If you have set a callback function to cts_StartSelection(), this callback is invoked as soon as a result is available. So you have to call cts_WaitSelection() exactly once after the callback is invoked, set timeout=0 (waiting makes no sense because you know that a result is available) If you do not want to use callback function you can call cts_WaitSelection() with different timeout values. If you have set timeout in cts_StartSelection() to 30 seconds, the easiest thing to do is set cts_WaitSelection->timeout_msec to 35000 ms (maybe even longer if you set options[8..9] because this may prolongate the technology selection). Then you have to call cts_WaitSelection() only once, it blocks and returns as soon as a result is available. This works of course only if cts_StartSelection->timeout_sec does not exceed ~60 seconds. If you set cts_WaitSelection->timeout_msec to smaller values you have to call the function in a loop until a result (something != CTS_IN_PROGRESS) is returned. This makes sense if you want to do other things in the same thread while waiting for result of technology selection, e.g. you may want to call cts_StopSelection() if abort request arrived from GUI or ECR. So in this case the timeout depends on the frequency with that you want to do the other things, e.g. a timeout of 0 is possible but will lead to high system load wheras a timeout of 100ms seems reasonable.

Q: Some time ago, upon reviewing our test logs, you pointed out that we should not call the API MSR_Activate() if next we start the selection with the API cts_StartSelection() because the latter activates the reader by itself. And what about the scenario when we want to establish the MSR callback and then use the selection? Here, MSR_Activate() is the only way to establish such a callback. Is this a legal use case to use simultaneously the MSR callback and the selection which, in turn, may have its own callback?

A: No, this is not a legal use case. You should not establish the MSR callback if you use technology selection. This is confusing and not necessary anyway. If MSR data is available, technology selection will finish, so you get the information from TEC, no need to set MSR callback. If you even call MSR_GetData() upon receiving MSR callback, TEC would most likely not be able to detect that MSR data is available and continue waiting for technology (TEC calls MSR_DataAvailable() and as soon as MSR_GetData() is called, the former will return 'no data available'). So please do not do anything like this.

Logging

You have two options to enable logging, choose one of them (if you think this is helpful, you could actually use both at once):

- Register a trace callback function with cts_SetTraceCallback().
- Use ADK-LOG: Configure logging channel "TEC" by means of log control panel.

Appendix

Appendix is empty.