

# Use cases

## Wallet SDK Initiated

This section describes use cases which are initiated from Wallet SDK.

### Wallet SDK Setup

Setup of Wallet SDK (both modules UCP SDK and MDC SDK) is main step which needs to be made at the very beginning. MDC SDK should be always setup at first because it is core module. During setup main configuration should be provided. Moreover there is some configuration which is related with HCE payments: MPA should be registered as default application for payment (Tap & Pay) and also should implement HostApuService to emulate an NFC card inside an Android service component. Application has to consider scenario where is woken up by HostApuService. Please find more details in *Wallet SDK API* document.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "MPA" as mpa
participant "Wallet SDK" as sdk
```

```
mpa->sdk: 1. MDC::setup(configuration)
mpa<--sdk: 2. result
mpa->sdk: 3. UCP::setup(configuration)
mpa<--sdk: 4. result
@enduml
```

## Pair Device on Wallet Server

This section describes pairing device process. Device pairing is process which authenticates device in context of given user. During this process device data and keys used in communication are exchanged with Wallet Server. To make possible device pairing, user needs to be already registered on the Wallet Server. Every device is identified by unique identifier. After every pairing device request, Wallet Server gives unique installation identifier. It means that particular installation of the application installed on particular device belongs to given user. Different users can use same device for separate installations. If any active installation on given device already exists during pairing device, Wallet Server will delete and create new installation in context of new user. Only one active installation is possible on particular device. Registration and device pairing is done by SDK during IBAN digitization process.

## Pair Device By Trusted Identity

Pairing is done automatically during IBAN digitization process and Trusted Identity is part of data passed to digitization. Pairing is described separately to show how User authentication mechanism works. Only User who is authenticated on Wallet Server may access to its own data. During pairing device process Wallet Server check whether previously on given device was installation which had device Payment Tokens, during pairing these device Payment Tokens are deleted asynchronously.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
}
```

```

LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "MPA" as mpa
participant "Wallet SDK" as sdk
participant "Wallet Server" as ws
participant "MDES" as mdes
participant "AHI" as ahi
sdk->ws: 1. pairDeviceByTrustedIdentity\r(trustedIdentity, fcmRegistrationToken, deviceInfo)
activate sdk
activate ws
ws->ws: 2. verify trusted identity
alt isActiveInstallationOnGivenDevice
ws->ws: 3. deleteActiveInstallation
ws->mdes: 4. deleteDeviceTokens
activate mdes
deactivate mdes
note left: asynchronous
end
ws->ws: 5. createNewDeviceInstallationRecordForUser
ws-->sdk: 6. response\r (userSessionToken, installationId)
deactivate ws
sdk->sdk: 7. store userSessionToken
sdk-->mpa: 8. result
deactivate mpa
deactivate sdk
@enduml

```

## IBAN Digitization

IBAN can be digitized for two different goals:

1. For contactless payments, in this scenario device Payment Token is created.
2. For e-commerce payments, in this scenario static Payment Token is created.

This section describes process related to IBAN digitization.

## Device Token Digitization

IBAN digitization via Wallet SDK is process which performs digitization of given IBAN which is assigned to given user for contactless payments. Device token is created in INACTIVE state just after digitization. After profile provisioning device Payment Token status is changed to ACTIVE.

Whole process of IBAN Digitization via Wallet SDK consists of some subprocesses, however whole complexity is behind the scene and MPA needs to call only one method to perform all needed steps. If any of the steps will fail, MPA will be notified and whole process can be retried. Since the mobile environment is not secure, Wallet Server expects signed data as JWT which should be passed via Wallet SDK.

It is possible to digitize many IBANs. For each IBAN, SDK digitization method should be called separately. One User (identified by user id) can have many IBANs and one IBAN can belong to many different Users.

In case when user (identified by user id) changes his device, whole process looks the same from MPA perspective.

@startuml

skinparam ParticipantPadding 30

skinparam BoxPadding 30

skinparam noteFontColor #FFFFFF

skinparam noteBackgroundColor #1C1E3F

skinparam noteBorderColor #1C1E3F

skinparam noteBorderThickness 1

skinparam sequence {

ArrowColor #1C1E3F

ArrowFontColor #1C1E3F

ActorBorderColor #1C1E3F

ActorBackgroundColor #FFFFFF

ActorFontStyle bold

ParticipantBorderColor #1C1E3F

ParticipantBackgroundColor #1C1E3F

ParticipantFontColor #FFFFFF

ParticipantFontStyle bold

LifeLineBackgroundColor #1C1E3F

LifeLineBorderColor #1C1E3F

}

actor User

participant "MPA" as MPA

participant "Wallet SDK" as SDK

participant "Wallet Server" as WS

participant MDES as MDES

participant "AHI" as AHIS

User -> MPA: 1. enableNFCForIBAN

activate MPA

MPA -> AHIS: 2. createSignedAccountInfo

activate AHIS

AHIS -> AHIS: 3. signDataAsJwt \r (userId, iban(bank account number),\r  
countryCode\rphone/email)

AHIS --> MPA: 4. response (signedAccountInfo)

deactivate AHIS  
MPA -> MPA: 5. getFcmToken(walletFirebase)  
MPA -> SDK: 6. UCP::digitizeIbanForDevice\r(signedAccountInfo, fcmToken, userLanguageCode)  
activate SDK  
SDK -> WS: 7. addUserWithIban\r (signedAccountInfo)  
activate WS  
WS -> WS: 8. createUserRecordAndIban  
WS --> SDK: 9. response(userId, ibanId\r (sha256(iban(bank account number))))  
SDK -> SDK: 10. isDevicePaired  
alt isDevicePaired=false  
SDK-> WS: 11. pairDeviceByTrustedIdentity\r(signedAccountInfo, fcmRegistrationToken, deviceInfo)  
note over SDK, WS #1C1E3F: See Pair Device process for more details  
else isDevicePaired=true  
SDK -> WS: 12. loginByTrustedIdentity  
note over SDK, WS #1C1E3F: See Login process for more details  
end  
SDK -> SDK: 13. isDeviceRegisteredForPayment  
alt isDeviceRegisteredForPayment=false  
SDK -> WS: 14. getPkCertificate  
WS -> MDES: 15. getPkCertificate  
activate MDES  
MDES --> WS: 16. response(pkCertificate)  
WS --> SDK: 17. response(pkCertificate)  
SDK -> SDK: 18. prepareDataForRegistration(pkCertificate)  
SDK -> WS: 19. registerDeviceForPayment\r (paymentDeviceInfo, userSessionToken)  
WS -> MDES: 20. registerMobilePaymentApplication\r (paymentDeviceInfo)  
MDES --> WS: 21. response(mobileKeys,\r remoteManagementUrl)  
WS --> SDK: 22. response(mobileKeys,\r remoteManagementUrl)  
end  
SDK -> WS: 23. digitizeIbanForDevice\r (ibanId, userSessionToken, userLanguageCode)  
WS -> MDES: 24. checkEligibility(iban(bank account number)\r countryCode, ahild, paymentAppIdForDevice,\r paymentAppInstanceId)  
MDES --> WS: 25. response (eligibilityReceipt)  
WS -> MDES: 26. digitize(eligibilityReceipt)  
MDES --> WS: 27. response  
deactivate MDES  
WS --> SDK: 28. response\r (devicePaymentTokenInfo)  
note over SDK, WS #1C1E3F: See Approved diagrams  
deactivate WS  
SDK --> MPA: 29. result  
deactivate SDK  
MPA --> User: 30. IBAN digitized, please wait for activation  
deactivate MPA  
@enduml

When digitization for DEVICE token is succeed, then profile provisioning takes place (See [Profile Provisioning](#)).

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES as MDES
group DEVICE digitization Approved
MDES --> WS : 1. responseFromDigitization(tokenInfo)
activate MDES
activate WS
WS --> SDK : 2. response
activate SDK
SDK --> MPA : 3. result
activate MPA
MPA --> User : 4. result
MDES -> SDK : 5. provision
deactivate SDK
deactivate MPA
deactivate WS
deactivate MDES
... Profile Provisioning ...
note over MPA, MDES #1C1E3F: See Profile Provisioning diagram
end
```

@enduml

## Static Token Digitization

IBAN digitization for a static token is a process where static Payment Token is created and can be used for e-commerce payments. Static Payment Token is created on MDES side where token PAN and expiration date is stored. Once token is created, MDES notifies Zapp which is responsible for CVC2 generation and storing for this particular token. There could be some circumstances where Zapp will not receive such notification and CVC2 will be not generated. For that reason Wallet Server introduces mechanism to delete such tokens after some period of time(see [Remove Static Tokens Without CVC2](#)). Just after static token digitization token PAN and FPAN are enrolled to ACS(see [Enroll Static Token to ACS](#)).

@startuml

```
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
    ArrowColor #1C1E3F
    ArrowFontColor #1C1E3F
    ActorBorderColor #1C1E3F
    ActorBackgroundColor #FFFFFF
    ActorFontStyle bold
    ParticipantBorderColor #1C1E3F
    ParticipantBackgroundColor #1C1E3F
    ParticipantFontColor #FFFFFF
    ParticipantFontStyle bold
    LifeLineBackgroundColor #1C1E3F
    LifeLineBorderColor #1C1E3F
}
actor User
participant "MPA" as MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES as MDES
participant "AHIS" as AHIS
User -> MPA: 1. enableECommerceForIBAN
activate MPA
MPA -> AHIS: 2. createSignedAccountInfo
activate AHIS
```

AHIS -> AHIS: 3. signDataAsJwt \r (userId, iban(bank account number),\r  
 countryCode\rphone/email)  
 AHIS --> MPA: 4. response (signedAccountInfo)  
 deactivate AHIS  
 MPA -> MPA: 5. getFcmToken(walletFirebase)  
 MPA -> SDK: 6. UCP::digitizeIbanForStatic\r(signedAccountInfo, fcmToken, userLanguageCode)  
 activate SDK  
 SDK -> WS: 7. addUserWithIban\r (signedAccountInfo)  
 activate WS  
 WS -> WS: 8. createUserRecordAndIban  
 WS --> SDK: 9. response(userId, ibanId\r (sha256(iban(bank account number))))  
 SDK -> SDK: 10. isDevicePaired  
 alt isDevicePaired=false  
 SDK-> WS: 11. pairDeviceByTrustedIdentity\r(signedAccountInfo, fcmRegistrationToken,  
 deviceInfo)  
 note over SDK, WS #1C1E3F: See Pair Device process for more details  
 else isDevicePaired=true  
 SDK -> WS: 12. loginByTrustedIdentity  
 note over SDK, WS #1C1E3F: See Login process for more details  
 end  
 SDK -> SDK: 13. store userSessionToken  
 SDK -> WS: 14. digitizeIbanForStatic\r (ibanId, userSessionToken, userLanguageCode)  
 WS -> MDES: 15. tokenize(iban(bank account number)\r countryCode, ahild)  
 activate MDES  
 MDES --> WS: 16. response  
 deactivate MDES  
 WS --> SDK: 17. response\r (staticPaymentTokenInfo)  
 deactivate WS  
 SDK --> MPA: 18. result  
 deactivate SDK  
 MPA --> User: 19. digitization succeed  
 deactivate MPA  
 @enduml

## Enroll Static Token to ACS

When merchant initiates 3DS authentication for static token, depending on the AHI and market requirement, challenge or step up might be required to be performed for transaction used initiated using static token. For the Step-Up to be initiated, ACS needs to have the following information available prior to the authentication so it can map and initiate the step up:

- FPAN,
- token PAN/DPAN – used for 3DS\_V1,
- User's credential required for the step-up (i.e, phone number/email).



To enable Step-Up all required information needs to be enrolled to ACS to perform the authentication challenge. Enrolment is done automatically and asynchronously after IBAN digitization for static token.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "MPA" as MPA
participant "Wallet Server" as WS
participant MDES as MDES
participant "ACS" as ACS
MDES --> WS: 1. response(staticToken)
activate MDES
activate WS
deactivate MDES
deactivate WS
WS -> MDES: 2. getTokenDetails(tokenUniqueReference)
activate WS
activate MDES
MDES --> WS: 3. response(FPAN, DPAN)
deactivate MDES
WS -> ACS: 4. enroll(FPAN, DPAN, phone/email)
activate ACS
ACS --> WS: 5. response
deactivate ACS
@enduml
```

# Unenroll Static Token from ACS

Static token which is deleted is also unenrolled from ACS automatically.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "MPA" as MPA
participant "Wallet Server" as WS
participant MDES as MDES
participant "ACS" as ACS
MDES --> WS: 1. response(staticTokenDeleted)
activate MDES
activate WS
deactivate MDES
deactivate WS
WS -> MDES: 2. getTokenDetails(tokenUniqueReference)
activate WS
activate MDES
MDES --> WS: 3. response(FPAN, DPAN)
deactivate MDES
WS -> ACS: 4. unEnroll(FPAN, DPAN, phone/email)
activate ACS
ACS --> WS: 5. response
deactivate ACS
@enduml
```

# Handle Message From Server

In whole system there are processes where server needs to send messages to the device. Wallet Server has separate component which is responsible for sending messages to the device. This component uses different channels for message delivery. There are two channels: SSE(Server Sent Events) and RNS(Remote Notification Service). When message is ready for delivery, Wallet Server uses both channels to deliver such message. In first versions of Wallet Server only RNS was used, however sometimes messages were not delivered and to improve delivery new SSE channel was introduced. This channel helps in processes which start from the device and device expects message from the server. Moreover device checks messages which are still not delivered on actions where such messages are expected. Below diagram describes how delivery message process works and how needs to be handled on MPA side.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "MPA" as MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant "RNS" as RNS
opt
SDK -> WS: 1. callActionAfterWhichMessagelsExpected
WS --> SDK: 2. response
SDK -> WS: 3. openSSEConnection
end
WS -> WS: messageReadyForDelivery
opt If device has opened connection
```

WS -> SDK: 4. deliverUsingSSE  
end  
WS -> RNS: 5. deliverMessage  
RNS --> WS: 6. response  
RNS -> MPA: 7. deliverMessage  
MPA -> MPA: 8. checkWalletSenderId  
MPA-> SDK: 9. MDC:CloudMessage#process(pushData)  
SDK -> SDK: 10. deduplicateMessage  
SDK -> WS: 11. acknowledgeMessage  
WS --> SDK: 12. response  
SDK -> SDK: 13. processMessage  
...Obtain pending messages...  
MPA -> SDK: 14. someActionWhereMessageMayBeStillPending  
SDK -> SDK: 15. doAction  
SDK ->> WS: 16. getPendingMessages  
WS --> SDK: 17. response  
SDK -> SDK: do actions from 10 to 13

@enduml

## Update RNS Token

Wallet Server is responsible for sending push notifications to the Wallet SDK. For that reason RNS token is passed to Wallet Server during pairing device or in some cases is obtained by SDK from MPA whenever is needed. However this token can be updated. MPA will be notified when token is being updated and then needs to obtain new RNS token and update via Wallet SDK on Wallet Server. Retrieving push notifications and RNS tokens is responsibility of the MPA.

@startuml

skinparam ParticipantPadding 30  
skinparam BoxPadding 30  
skinparam noteFontColor #FFFFFF  
skinparam noteBackgroundColor #1C1E3F  
skinparam noteBorderColor #1C1E3F  
skinparam noteBorderThickness 1  
skinparam sequence {  
ArrowColor #1C1E3F  
ArrowFontColor #1C1E3F  
ActorBorderColor #1C1E3F  
ActorBackgroundColor #FFFFFF  
ActorFontStyle bold  
ParticipantBorderColor #1C1E3F  
ParticipantBackgroundColor #1C1E3F  
ParticipantFontColor #FFFFFF  
ParticipantFontStyle bold

```

LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant "Remote Notification Service" as RNS
activate RNS
RNS -> MPA: 1. onTokenRefresh
deactivate RNS
activate MPA
MPA -> MPA: 2. obtainNewToken(walletFirebase)
MPA -> SDK: 3. MDC::updateRegistrationToken(newRNSToken)
activate SDK
SDK -> WS: 4. updateRNSToken(deviceInstallationId, newRNSToken)
activate WS
WS -> WS: 5. updateRNSToken
WS --> SDK: 6. response
deactivate WS
SDK --> MPA: 7. result
deactivate SDK
deactivate MPA
deactivate RNS
@enduml

```

## Profile Provisioning

During this process digitized card profile is delivered to the device. This process is triggered automatically after successful digitization where outcome is APPROVED or REQUIRE\_ADDITIONAL\_AUTHENTICATION. It is not possible to retry provisioning itself. To retry provisioning, previous token needs to be deleted and new digitization called hence when SDK reports that provisioning has failed then given token is automatically deleted and User can perform digitization once again. During process there is few point of failures and provisioning can be not finished at all. In this scenario Payment Tokens which are not provisioned for long period of time are delete by Wallet Server (see Removing Not Provisioned Tokens). From User perspective it can be good approach to treat digitization and provisioning as one process and inform User about steps(if User has to wait long time without any information then can treat this as some failure). From MPA perspective can be also good approach to wait for provisioning status as long as User stays on view dedicated to it. If User wants to cancel the process because provisioning status is not available for long period of time it is recommended to delete Payment Token(see Delete Payment Token via SDK) once User click cancel or back. Thanks to deletion, new digitization can be called and User does not have to wait until Payment Token is deleted by Wallet Server due to lack of provisioning.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES as MDES
MDES->WS: 1. sendRemoteNotificationMessage(mdesRemoteMessage)
activate MDES
deactivate MDES
activate WS
WS-> SDK: 2. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
activate SDK
SDK-> MDES: 3. provision
activate MDES
MDES --> SDK: 4. response(cardProfile)
SDK-> MDES: 5. notify provisioning result
MDES --> SDK: 6. response
deactivate MDES
SDK -> SDK: 7. store card profile
SDK -> WS: 8. confirmProvisioningStatus(SUCCESS/FAILURE)
activate WS
alt FAILURE
WS -> MDES: 9. deleteToken
activate MDES
MDES --> WS: 10. response
```

```

deactivate MDES
WS --> SDK: 11. response
SDK -> SDK: 12. deleteToken
SDK -> MPA: 13. onProvisioningFailure
activate MPA
MPA -> User: 14. please try again
deactivate MPA
else SUCCESS
WS --> SDK: 15. response
deactivate WS
SDK -> MPA: 16. onProvisioningSuccess(paymentInstrument)
deactivate SDK
activate MPA
MPA -> User: 17. card digitized successfully
deactivate MPA
end
@enduml

```

# Transaction Credentials Replenishment

Transaction Credentials are unique per transactions keys that are used to calculate cryptograms in transactions. Each set of credentials is linked with a unique Application Transaction Counter (ATC). Each set of credentials can only be used for one transaction. There is a limit (set on MDES onboarding) of transaction credentials stored on device. There are several types of replenishment:

- Initial
- Automatic
- Manual

## Transaction Credentials - Automatic Replenishment

After every transaction Wallet SDK checks if number of transaction credentials is below, preconfigured during SDK setup, threshold. If yes then SDK will call replenish. During replenish process transaction credentials are being delivered to mobile application.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F

```

```

ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES as MDES
SDK->SDK: 1. detectTransactionCredentialsRemainingBelowThreshold
alt Request session if required
activate SDK
SDK-> MDES: 2. requestSession
activate MDES
MDES--> SDK: 3. response
MDES-> WS: 4. sendRemoteNotificationMessage(mdesRemoteMessage)
deactivate MDES
activate WS
WS -> SDK: 5. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
end
SDK-> MDES: 6. replenish
activate MDES
MDES-> MDES: 7. checkIfPaymentTokenIsActive
MDES-->SDK: 8. response(transactionCredentials)
deactivate MDES
SDK->MPA: 9. onReplenishSuccess(paymentInstrument)
deactivate SDK
@enduml

```

## Transaction Credentials - Initial Replenishment

There are scenarios when automatic replenishment is not possible (lack of internet connection) and number of transaction credentials decrease to 0. In such case MPA should handle NO\_TRANSACTION\_CREDENTIALS error from transaction listener, show user proper alert and call replenish method manually. MPA can also check number of transaction credentials at any other time and do manual replenishment. It is not recommended call manual replenishment if number of transaction credentials is above threshold since in such case SDK will manage that.



Initial replenishment is process which starts directly after successful token activation. Wallet SDK is notified by Wallet Server using push notification or refreshing payment instruments. No action is needed by MPA.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES as MDES
WS -> SDK: 1. notifyTokenUpdated(Active)
activate WS
deactivate WS
activate SDK
alt Request session if required
SDK-> MDES: 2. requestSession
activate MDES
MDES--> SDK: 3. response
MDES-> WS: 4. sendRemoteNotificationMessage(mdesRemoteMessage)
deactivate MDES
activate WS
WS-> SDK: 5. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
end
SDK-> MDES: 6. replenish
activate MDES
```

```
MDES-> MDES: 7. checkIfPaymentTokenIsActive
MDES-->SDK: 8. response(transactionCredentials)
deactivate MDES
SDK -> MPA: 9. onReplenishSuccess(paymentInstrument)
deactivate SDK
@enduml
```

## Transaction Credentials - Manual Replenishment

There are scenarios when automatic replenish is not possible (user is not able to connect with Internet) and after some number of transactions, transaction credentials number will decrease to 0. In such case MPA should handle NO\_TRANSACTION\_CREDENTIALS error from transaction listener, show user proper alert and call replenish method manually. MPA can also check number of transaction credentials at any other time.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES as MDES
MPA -> SDK: 1. UCP::replenishCredentials(paymentInstrumentId)
activate MPA
deactivate MPA
activate SDK
alt Request session if required
SDK-> MDES: 2. requestSession
```

```
activate MDES
MDES--> SDK: 3. response
MDES-> WS: 4. sendRemoteNotificationMessage
deactivate MDES
activate WS
WS-> SDK: 5. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
end
SDK-> MDES: 6. replenish
activate MDES
MDES-> MDES: 7. checkIfPaymentTokenIsActive
MDES--> SDK: 8. response(transactionCredentials)
deactivate MDES
SDK -> MPA: 9. onReplenishSuccess(paymentInstrument)
deactivate SDK
@enduml
```

# Transacting

Wallet SDK provides functionalities to make contactless payments (using HCE) and e-commerce using static Payment Token data.

## Contactless Transaction

Contactless transaction uses Android HCE. On MPA side HostApuService should be implemented. Depending on chosen CDCVM and on how transaction is started, user experience is different and MPA should interact with Wallet SDK in different way. The final decision about transaction processing belongs to MPA. Wallet SDK provides transaction information and based on that and User authentication, MPA can advise to proceed, decline or require authentication(if User should be authenticated but was not). For contactless transaction Wallet SDK provides result of transaction. This result is only from the communication between MPA and Terminal. Transaction Processing with Payment Network is done separately (see Transaction Processing). Also after every contactless transaction, Transaction Credentials Replenishment is performed automatically by SDK if needed(see Transaction Credentials - Automatic Replenishment).

NOTE: The way of authentication depends on MPA. For transaction User may also choose specific card. If no card is chosen, SDK will use the one which is set as default for contactless payments. Whenever user is authenticated or chose card for payment MPA should pass this information when *onContactlessPaymentStarted* is called.

As was described above, the final decision(PROCEED, DECLINE, AUTHENTICATION\_REQUIRED) for given transaction is taken on MPA side based on transaction information and User authentication. Because of that reason there could be different scenarios which may occur and transaction

experience will be single or double tap.

Sample scenarios:

- User can be already authenticated and if MPA will not decline transaction then will be processed as single tap,
- velocity check counters can be applied and even if User was not authenticated MPA can decide to proceed transaction without authentication, taking decision based on transaction information,
- User was not authenticated but MPA recognised transaction as authentication needed. MPA returns AUTHENTICATION\_REQUIRED decision and SDK informs MPA that authentication is needed.

@startuml

skinparam ParticipantPadding 30

skinparam BoxPadding 30

skinparam noteFontColor #FFFFFF

skinparam noteBackgroundColor #1C1E3F

skinparam noteBorderColor #1C1E3F

skinparam noteBorderThickness 1

skinparam sequence {

ArrowColor #1C1E3F

ArrowFontColor #1C1E3F

ActorBorderColor #1C1E3F

ActorBackgroundColor #FFFFFF

ActorFontStyle bold

ParticipantBorderColor #1C1E3F

ParticipantBackgroundColor #1C1E3F

ParticipantFontColor #FFFFFF

ParticipantFontStyle bold

LifeLineBackgroundColor #1C1E3F

LifeLineBorderColor #1C1E3F

}

actor User

participant MPA

participant "HostApduService" as HAS

participant "Wallet SDK" as SDK

participant Terminal as TER

opt User selects particular card for payment

MPA -> User: 1. showCards

activate MPA

User -> MPA: 2. selectCard

deactivate MPA

end

User -> MPA: 3. pay

User -> TER : 4. contactless 1st tap

```
activate TER
loop
TER -> HAS: 5. processCommandApdu(commandApdu, extras)
activate HAS
HAS -> SDK: 6. UCP::Pay#processHceApduCommand(apdu, extras)
activate SDK
group Select PPSE
SDK -> MPA: 7. onContactlessPaymentStarted()
activate MPA
MPA -> MPA: 8. checkIsUserAuthenticated
alt isAuthenticated=true
MPA -> SDK: 9. UCP::Pay#setUserAuthenticatedForPayment(paymentInstrumentId, pin?)
end
alt selectedCard == null
SDK -> SDK: 10. useDefaultPaymentInstrumentForContactless
else selectedCard != null
MPA -> SDK: 11. UCP::Pay#selectForPayment(selectedPaymentInstrumentId)
deactivate MPA
end
end
group Generate AC command
SDK -> MPA: 12. getFinalDecisionForTransaction(isUserAuthenticated,recommendedAdvice, trxInfo)
activate MPA
MPA -> MPA: 13. checkTrxAndAuthentication
MPA --> SDK: 14. result(advice)
deactivate MPA
end
SDK --> HAS: 15. responseApdu
HAS --> TER: 16. responseApdu
deactivate HAS
end
alt advice=AUTHENTICATION_REQUIRED
SDK -> MPA: 17. onAuthRequiredForContactless(paymentInstrument, trxInfo)
activate MPA
MPA -> User: 18. show authentication view with trx info
User -> MPA: 19. authenticate
User -> TER: 20. contactless 2nd tap
loop
TER -> HAS: 21. processCommandApdu(commandApdu, extras)
activate HAS
HAS -> SDK: 22. UCP::Pay#processHceApduCommand(apdu, extras)
group Select PPSE
SDK -> MPA: 23. onContactlessPaymentStarted()
MPA -> MPA: 24. checkIsUserAuthenticated
alt isAuthenticated=true
MPA -> SDK: 25. UCP::Pay#setUserAuthenticatedForPayment(paymentInstrumentId, pin?)
```

```

end
alt selectedCard == null
SDK -> SDK: 26. useDefaultPaymentInstrumentForContactless
else selectedCard != null
MPA -> SDK: 27. UCP::Pay#selectForPayment(selectedPaymentInstrumentId)
end
end
group Generate AC command
SDK -> MPA: 28. getFinalDecisionForTransaction(isUserAuthenticated=true,recommendedAdvice,
trxInfo)
MPA -> MPA: 29. checkTrxAndAuthentication
MPA --> SDK: 30. result(PROCEED)
end
SDK --> HAS: 31. responseApu
HAS --> TER: 32. responseApu
deactivate HAS
deactivate TER
end
end
SDK -> MPA: 33. onContactlessPaymentCompleted(paymentInstrument, trxInfo, trxResult)
deactivate SDK
MPA -> User: 34. show trx info view
deactivate MPA
...Transaction Processing ...
note over HAS #1C1E3F: See Transaction Processing diagram
...Transaction Credentials Automatic Replenishment ...
note over HAS #1C1E3F: See Transaction Credentials Automatic Replenishment diagram
@enduml

```

## Transaction Processing

Transaction Processing starts after contactless communication between terminal and MPA. Then transaction authorization is performed. During this authorization ARQC is validated. After authorization MDES notifies Wallet Server about the result of the authorization and sends transaction information. Transaction information is sent to MPA using Remote Notification Service.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F

```

```
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant Terminal as TER
participant "Payment Network" as PN
participant MDES
participant Zapp
participant AHI
TER -> PN: 1. authorizeTransaction(tokenPAN, cryptogram)
activate PN
activate TER
PN -> MDES: 2. detokenize
activate MDES
MDES -> MDES: 3. lookup token mapping
MDES --> PN: 4. response(PAN)
deactivate MDES
PN -> Zapp: 5. authorize(PAN)
activate Zapp
Zapp -> AHI: 6. authorize
activate AHI
AHI --> Zapp: 7. response
deactivate AHI
Zapp --> PN: 8. response
deactivate Zapp
PN --> TER: 9. response
deactivate TER
PN -> MDES: 10. storeTransactionDetails
deactivate PN
activate MDES
MDES -> WS: 11. pushTransactionDetails
deactivate MDES
activate WS
alt store transaction enabled
WS -> WS: 12. storeTransaction
end
```

```

WS-> SDK: 13. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
activate SDK
SDK -> MPA: 14. onNewTransaction(trxDetails)
deactivate SDK
activate MPA
MPA -> User: 15. showSystemNotification(trxDetails)
deactivate MPA
@enduml

```

## Web/E-com Transaction

In this process Wallet Server provides static token payment data for given IBAN id and user id, required for e-commerce transaction. Payment data contains static data like: token PAN, expiration date, CVC2. There might be scenario where even after successful IBAN digitization for static token, Zapp did not receive from MDES notification and static token payment date are not available. In such case User needs to be asked to wait additional time and Wallet Server responds with STATIC\_TOKEN\_PAYMENT\_DATA\_NOT\_AVAILABLE error.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
box "Consumer mobile android device" #white
participant "Merchant App" as MA
participant "MPA" as MPA
participant "Wallet SDK" as SDK
end box

```



participant "Wallet Server" as WS  
participant "AHI" as AHIS  
participant Zapp  
participant "Acquirer Gateway" as AG  
User -> MA: 1. checkOut  
activate MA  
MA -> MPA: 2. openMPA  
activate MPA  
MPA --> User: 3. requestAuthentication  
User -> MPA: 4. presentCredentials  
MPA -> AHIS: 5. authenticate  
activate AHIS  
AHIS --> MPA: 6. response  
MPA -> MPA: 7. prepareOrderUI  
MPA --> User: 8. requestToConfirmOrderAndAuthorizePayment  
User -> MPA: 9. confirmPayment  
MPA -> AHIS: 10. createSignedIbanInfo  
AHIS -> AHIS: 11. signDataAsJwt\r (userId, ibanId(sha256(bank account number)))  
AHIS --> MPA: 12. response(signedIbanInfo)  
deactivate AHIS  
MPA -> SDK: 13. UCP::getStaticTokenPaymentData(signedIbanInfo)  
activate SDK  
SDK -> WS: 14. getStaticTokenPaymentData(signedIbanInfo)  
activate WS  
WS -> WS: 15. getStaticTokenUniqueReferenceForMdes(userId, ibanId)  
WS -> Zapp: 16. searchToken(tokenUniqueReference)  
activate Zapp  
Zapp --> WS: 17. response(tokenPAN, expDate, CVC2)  
deactivate Zapp  
WS --> SDK: 18. response(tokenPAN, expDate, CVC2)  
deactivate WS  
SDK --> MPA: 19. result  
deactivate SDK  
MPA-->MA: 20. result  
deactivate MPA  
MA -> AG: 21. submitAuthTransaction(tokenPAN, expDate, CVC2)  
activate AG  
AG-> AG: 22. bauAuthTransactionFlow  
AG--> MA: 23. confirmAuthStatus  
deactivate AG  
MA--> User: 24. confirmOrder  
deactivate MA  
deactivate Zapp  
@enduml

# Setting Defaults for Payment

SDK manages default Payment Instrument for contactless payments. After digitization, if there is no default Payment Instrument, SDK sets digitized Payment Instrument after activation as default. In case where there are more than one active Payment Instruments and current default Payment Instrument is deleted or suspended, the SDK will set first active Payment Instrument as default. Default Payment Instrument can be changed at any time. Only active Payment Instrument can be set as default.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
MPA->>User: 1. payment instrument list
activate MPA
User->>MPA: 2. choose default for contactless payment)
MPA->>SDK: 3. UCP::setDefaultForContactless(paymentInstrumentId)
activate SDK
SDK->>SDK: 4. storeDefault
SDK-->>MPA: 5. result
deactivate MPA
deactivate SDK
@enduml
```

## Login on Wallet Server

User data are protected by User session token which is issued by Wallet Server after providing authentication factor. Authentication factor is provided first in pairing device and then session is created. Since session has limited period of validity, it needs to be refreshed using login on Wallet Server methods.

## Login on Wallet Server using Trusted Identity

In the Integrated implementation model User authentication doesn't occur directly on Wallet Server. Wallet Server will require User authentication when some user data will be requested. If User session token is no longer valid, SDK will return USER\_UNAUTHORIZED error. In such case Trusted Identity needs to be prepared on AHI server and sent via Wallet SDK in loginByTrustedIdentity method. MPA can decide whether ask User to provide authentication data or not. The latter case regards situation when user is already authenticated and Trusted Identity can be immediately returned from AHI based on already valid session on AHI side. During login process Wallet Server checks if given device still exists, if not then responds with CANT\_FIND\_DEVICE status which is interpreted on SDK side as given device is deleted and all local data stored on SDK side are cleared.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant "AHI" as Issuer
MPA -> SDK: 1. invoke API
activate MPA
activate SDK
```

```

SDK -> WS: 2. invoke API
activate WS
WS --> SDK: 3. response(USER_UNATHORIZED)
deactivate WS
SDK --> MPA: 4. result(USER_UNATHORIZED)
MPA->User: 5. show authenticate view
User->MPA: 6. put authentication data
MPA->Issuer: 7. authenticate
activate Issuer
Issuer -> Issuer: 8. generateTrustedIdentity - signed user id
Issuer --> MPA: 9. response(trustedIdentity)
deactivate Issuer
MPA-> SDK: 10. MDC::loginByTrustedIdentity(trustedIdentity)
SDK-> WS: 11. loginByTrustedIdentity(trustedIdentity)
activate WS
WS -> WS: 12. check if device exists
alt device exists
WS-> WS: 13. verify trusted identity
WS --> SDK: 14. response(userSessionToken)
SDK -> SDK: 15. store(userSessionToken)
SDK-->MPA: 16. result
MPA -> SDK: 17. invoke API
else device not exists
WS --> SDK: 18. response(CANT_FIND_DEVICE)
deactivate WS
SDK -> SDK: 19. clearAllLocalData
SDK --> MPA: 20. result(CANT_FIND_DEVICE)
deactivate MPA
deactivate SDK
... Pair device ...
note over MPA, WS #1C1E3F: See Pairing Device diagram
MPA -> SDK: 21. invoke API
end
@enduml

```

## Getting Payment Instruments

After digitization process, payment instrument is stored in UCP SDK module of Wallet SDK. Payment instrument in context of UCP SDK is digitized IBAN and contains information like:

- id (specified id which helps MPA to identify payment instrument which was digitized from MPA),

- status,
- transaction credentials count,
- paymentTokenId,
- staticTokenInfo.

MPA can get information about all Payment Instruments from the Wallet SDK at any time. Payment instruments will be retrieved only from local storage that is part of SDK. Payment Tokens for Payment Instruments can be refreshed/pulled from Wallet Server on demand. This scenario should be considered only when User e.g. makes swipe to refresh.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
MPA->SDK: 1. UCP::getAllPaymentInstruments(refresh)
activate MPA
alt refresh = true
activate SDK
SDK -> WS: 2. getAllPaymentTokens(userSessionToken)
activate WS
WS -> SDK: 3. response(deviceAndStaticPaymentTokens)
deactivate WS
SDK -> SDK: 4. updateLocalStorage
end
SDK --> MPA: 5. result(paymentInstrumentList)
deactivate SDK
```

deactivate MPA  
@enduml

# Getting Transaction History

It is possible that transaction history will be stored on Wallet Server for infinite time. This should be specified during onboarding. If this options is enabled, MPA can retrieve transaction history for given user and filtering. Transactions are returned in corresponding parts for better user experience. If next part is available then response from previous part contain information needed for requesting next part. MPA should check if next part is not empty and then make another request.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
loop next != null
MPA->>SDK: 1. MDC::getTransactionHistory(limit, next?, ...)
activate MPA
activate SDK
SDK ->>WS: 2. getTransactionHistory(limit, next?, userSessionToken, ...)
activate WS
WS ->>SDK: 3. response(transactionHistoryList, next?)
deactivate WS
```

```
SDK --> MPA: 4. result(transactionHistoryList, next?)
deactivate SDK
deactivate MPA
end
@enduml
```

# Payment Token Lifecycle Management via SDK

Payment Token lifecycle management can be done via SDK.

## Delete Device Payment Token via SDK

Device Payment Token can be deleted using SDK.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
User -> MPA: 1. Delete payment token
activate MPA
MPA -> SDK: 2. UCP::delete(paymentInstrumentId, reason)
```

```

deactivate MPA
activate SDK
SDK -> WS: 3. deletePaymentToken(userSessionToken, paymentTokenId, reason)
activate WS
WS -> MDES: 4. Delete token
activate MDES
MDES -> MDES: 5. Delete token mapping
MDES --> WS: 6. response
deactivate MDES
WS --> SDK: 7. response
deactivate WS
alt Request session if required
SDK -> MDES: 8. request session
activate MDES
MDES --> SDK: 9. response
MDES -> WS: 10. sendRemoteNotificationMessage
deactivate MDES
activate WS
WS -> SDK: 11. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
end
SDK -> MDES: 12. delete(tokenUniqueReference)
activate MDES
MDES --> SDK: 13. response
deactivate MDES
SDK -> SDK: 14. Delete transaction credentials, card profile
SDK -> MPA: 15. onPaymentInstrumentStatusChanged(id, status)
deactivate SDK
MPA --> User: 16. show update view
deactivate MPA
@enduml

```

## Delete Static Token via SDK

A User may report that given static token was compromised or just to disable e-commerce payments. To do that SDK exposes method for static token deletion. After token deletion all corresponding data are deleted from acs (see [Unenroll Static Token from ACS](#)).

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F

```



```

skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
User -> MPA: 1. Report compromised static token
activate MPA
MPA -> SDK: 2. UCP::deleteStaticToken(paymentInstrumentId)
activate SDK
SDK -> WS: 3. Delete payment token(userSessionToken, paymentTokenId)
activate WS
WS -> MDES: 4. Delete token
activate MDES
MDES -> MDES: 5. Delete token mapping
MDES --> WS: 6. response
deactivate MDES
WS --> SDK: 7. response
deactivate WS
SDK --> MPA: 8. result
deactivate SDK
MPA --> User: 9. result
deactivate MPA
deactivate WS
...Unenroll static token...
note over WS, MDES #1C1E3F: See Unenroll Static Token Diagram
@enduml

```

## Errors Reporting

Wallet SDK performs some security checks to prevent data stole. When any issue is detected, Wallet SDK reports error to Wallet Server and clears own data - all Payment Tokens, Device and

User information will be removed from local storage. MPA should inform user about incident. MPA should not perform more actions on SDK until application restart. When any SDK method is called it finishes with error and status SECURITY\_EVENT\_OCCURRED.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
SDK -> SDK: 1. detectSecurityIssue
activate SDK
SDK -> SDK: 2. clearSDKData
SDK -> WS: 3. reportSecurityIssue
activate WS
deactivate WS
SDK -> MPA: 4. onSecurityIssueAppeared
deactivate SDK
activate MPA
MPA -> User: 5. show information
deactivate MPA
@enduml
```

## Device Unpairing

Unpairing device clears all modules data and report that fact only if possible to server. If server receives this signal then removes all device data including provisioned device Payment Tokens. If

not then data are cleared locally only - similar like during app uninstallation. This can be used for scenario when MPA does not want to use SDK at all or for scenario when MPA supports switching between users accounts on the same installation. If MPA detects that new User is trying to log into application in case when previous had digitized cards, immediately should clear all data from previous, since SDK stores data in context of one User only.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
MPA -> SDK: 1. MDC::unpairDevice
activate MPA
activate SDK
opt
SDK -> WS: 2. unpairDevice
activate WS
WS --> SDK: 3. response
deactivate WS
end
SDK -> SDK: 4. clearAllData
SDK --> MPA: 5. result
deactivate SDK
deactivate MPA
@enduml
```

# Wallet Server VCP Issuer API

## Initiated

This section describes use cases which are initiated from server VCP Issuer API which is dedicated only for IBANs.

## IBAN Digitization via Server

IBAN can be also digitized via Server API for static token. After digitization static token is enrolled to ACS(see [Enroll Static Token to ACS](#)).

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "AHI App" as AHIAPP
participant "AHI Server API " as AHIS
participant "Wallet Server" as WS
participant MDES
User -> AHIAPP: 1. openBankApplication
activate AHIAPP
AHIAPP --> User: 2. requireAuthentication
User -> AHIAPP: 3. presentAuthCredentials
AHIAPP -> AHIS: 4. authenticate
```

activate AHIS  
AHIS -> AHIS: 5. authenticate  
AHIS --> AHIAPP: 6. response  
AHIAPP -> AHIAPP: 7. isPbAEnabled  
AHIAPP --> User: 8. presentPbAT&C  
AHIAPP -> AHIS: 9. digitizeBankAccountNumber  
AHIS -> AHIS: 10. signDataAsJwt\r (userId, iban(bank account number),\r  
countryCode,\remail/phone)  
AHIS -> WS: 11. digitizeIbanForStatic(signedAccountInfo)  
activate WS  
WS -> WS: 12. createUserRecordAndIban  
WS -> MDES: 13. tokenize(iban(bank account number)\r countryCode, ahild)  
activate MDES  
MDES --> WS: 14. response  
deactivate MDES  
WS --> AHIS: 15. response  
deactivate WS  
AHIS --> AHIAPP: 16. response  
deactivate AHIS  
AHIAPP -> User: 17. digitization succeed  
deactivate AHIAPP  
...Enroll static token to ACS...  
note over WS, MDES #1C1E3F: See Enroll Static Token diagram  
@enduml

## Web/E-commerce Transaction via Server

In this process Wallet Server provides for given User Id and IBAN id static payment data needed for e-commerce transaction.

@startuml  
skinparam ParticipantPadding 30  
skinparam BoxPadding 30  
skinparam noteFontColor #FFFFFF  
skinparam noteBackgroundColor #1C1E3F  
skinparam noteBorderColor #1C1E3F  
skinparam noteBorderThickness 1  
skinparam sequence {  
ArrowColor #1C1E3F  
ArrowFontColor #1C1E3F  
ActorBorderColor #1C1E3F  
ActorBackgroundColor #FFFFFF  
ActorFontStyle bold  
ParticipantBorderColor #1C1E3F

```
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "Merchant App" as MA
participant "AHI App" as AHIAPP
participant "AHI Server API " as AHIS
participant "Wallet Server" as WS
participant Zapp
participant "Acquirer Gateway" as AG
User -> MA: 1. checkOut
activate MA
MA -> AHIAPP: 2. openAhiApp
activate AHIAPP
AHIAPP --> User: 3. requestCredentials
User -> AHIAPP: 4. presentCredentials
AHIAPP -> AHIS: 5. authenticate
activate AHIS
AHIS --> AHIAPP: 6. response
AHIAPP -> AHIAPP: 7. prepareOrderUI
AHIAPP --> User: 8. requestToConfirmOrderAndAuthorizePayment
User -> AHIAPP: 9. confirmPayment
AHIAPP -> AHIS: 10. confirmPayment
AHIS -> AHIS: 11. signDataAsJwt(userId, ibanId(sha256(bank account number)))
AHIS -> WS: 12. getStaticTokenPaymentData(signedIbanInfo)
activate WS
WS -> WS: 13. getStaticTokenUniqueReferenceForMdes(userId, ibanId)
WS -> Zapp: 14. searchToken(tokenUniqueReference)
activate Zapp
Zapp --> WS: 15. response(tokenPAN, expDate, CVC2)
deactivate Zapp
WS --> AHIS: 16. response(tokenPAN, expDate, CVC2)
deactivate WS
AHIS --> AHIAPP: 17. response
deactivate AHIS
AHIAPP-->MA: 18. response
deactivate AHIAPP
MA -> AG: 19. submitAuthTransaction(tokenPAN, expDate, CVC2)
activate AG
AG-> AG: 20. bauAuthTransactionFlow
AG--> MA: 21. confirmAuthStatus
deactivate AG
MA--> User: 22. confirmOrder
```

deactivate MA  
@enduml

# Payment Token Lifecycle Management via Server

This section describes payment token lifecycle management performed via Server.

## Delete Device Payment Token via Server

Device Payment Token can be deleted using server API.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant AHI
AHI -> WS: 1. deletePaymentToken(paymentTokenId)
activate AHI
activate WS
WS -> MDES: 2. Delete token
activate MDES
```

```

MDES -> MDES: 3. Delete token mapping
MDES --> WS: 4. response
deactivate MDES
WS --> AHI: 5. response
deactivate AHI
WS -> SDK: 6. deliverMessage(paymentTokenDeleted)
deactivate WS
NOTE LEFT: See: Handle Message From Server
activate SDK
deactivate WS
deactivate MDES
alt Request session if required
SDK -> MDES: 7. request session
activate MDES
MDES --> SDK: 8. response
MDES -> WS: 9. sendRemoteNotificationMessage
deactivate MDES
activate WS
WS -> SDK: 10. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
deactivate MPA
end
SDK -> MDES: 11. delete(tokenUniqueReference)
activate MDES
MDES --> SDK: 12. response
deactivate MDES
SDK -> SDK: 13. Delete transaction credentials, card profile
SDK -> MPA: 14. onPaymentInstrumentStatusChanged(id, status)
deactivate SDK
MPA --> User: 15. show update view
deactivate MPA
@enduml

```

## Delete Static Payment Token via Server

Static token may also be deleted via Server API. After token deletion, static token related data are unenrolled from ACS(see [Unenroll Static Token from ACS](#)).

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1

```



```

skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "Wallet Server" as WS
participant MDES
participant AHI
User -> AHI: 1. Report compromised static token
activate AHI
AHI -> WS: 2. deleteStaticToken(paymentTokenId)
activate WS
WS -> MDES: 3. Delete token
activate MDES
MDES -> MDES: 4. Delete token mapping
MDES --> WS: 5. response
deactivate MDES
WS --> AHI: 6. result
deactivate AHI
deactivate WS
deactivate WS
...Unenroll static token...
note over WS, MDES #1C1F3E: See Unenroll Static Token Diagram
@enduml

```

## Suspend Device Payment Token via Server

Payment Token can be suspended using server API.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {

```

```

ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant AHI
AHI -> WS: 1. suspendPaymentToken(paymentTokenId)
activate AHI
activate WS
WS -> MDES: 2. suspendToken(tokenUniqueReference)
activate MDES
MDES -> MDES: 3. suspendToken
MDES --> WS: 4. response
deactivate MDES
WS --> AHI: 5. response
deactivate AHI
WS -> SDK: 6. deliverMessage(paymentTokenSuspended)
deactivate WS
activate SDK
NOTE LEFT: See: Handle Message From Server
SDK -> SDK: 7. suspendToken
SDK -> MPA: 8. onPaymentInstrumentStatusChanged(id, status)
deactivate WS
deactivate SDK
deactivate MPA
@enduml

```

## Suspend Static Payment Token via Server

Static token may be suspended via Server.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30

```

```

skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "Wallet Server" as WS
participant MDES
participant "AHI" as AHI
AHI -> WS: 1. suspendStaticToken(paymentTokenId)
activate AHI
activate WS
WS -> MDES: 2. Suspend token
activate MDES
MDES -> MDES: 3. Suspend token
MDES --> WS: 4. response
deactivate MDES
WS --> AHI: 5. response
deactivate AHI
deactivate WS
deactivate WS
@enduml

```

## Unsuspend Device Payment Token via Server

Payment Token can be unsuspended using server API.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {

```

```

ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant AHI
AHI -> WS: 1. unsuspendPaymentToken(paymentTokenId)
activate AHI
activate WS
WS -> MDES: 2. unsuspendToken(tokenUniqueReference)
activate MDES
MDES -> MDES: 3. checkPermissions
MDES -> MDES: 4. unsuspendToken
MDES --> WS: 5. response
deactivate MDES
WS --> AHI: 6. response
deactivate AHI
WS -> SDK: 7. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
activate SDK
deactivate WS
SDK -> SDK: 8. activateToken
SDK -> MPA: 9. onPaymentInstrumentStatusChanged(id, status)
deactivate WS
deactivate SDK
... Replenishment ...
note over SDK, MDES #1C1E3F: Just after token activation transaction credentials replenishment is
performed by SDK\r. See Transaction Credentials Automatic Replenishment diagram
@enduml

```

## Unsuspend Static Payment Token via Server

Static token may be unsuspended via Server.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "Wallet Server" as WS
participant MDES
participant "AHI" as AHI
AHI -> WS: 1. unsuspendStaticToken(paymentTokenId)
activate AHI
activate WS
WS -> MDES: 2. Unsuspend token
activate MDES
MDES -> MDES: 3. Unsuspend token
MDES --> WS: 4. response
deactivate MDES
WS --> AHI: 5. response
deactivate AHI
deactivate WS
deactivate WS
@enduml

```

## Getting Payment Tokens

AHI can retrieve information about Payment Tokens via Server.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30

```

```

skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "Wallet Server" as WS
participant "AHI " as AHIS
AHIS -> WS: 1. getPaymentTokens(ibanId)
activate WS
activate AHIS
WS --> AHIS: 2. response(paymentTokens)
deactivate WS
deactivate AHIS
@enduml

```

## Update Email/Phone

With static token User credentials(phone/email) are enrolled to ACS. User can change those credentials and this change needs to be updated on ACS side.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF

```

```

ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "AHI " as AHIS
participant "Wallet Server" as WS
participant ACS
AHIS -> WS: 1. updateUser(userId, phone/email)
activate WS
activate AHIS
WS --> AHIS: 2. response
deactivate AHIS
WS -> WS: 3. getAllEnrolledStaticTokensForGivenUser
loop For every enrolled static token
WS -> ACS: 4. update(FPAN, DPAN, phone/email)
activate ACS
ACS --> WS: 5. response
deactivate ACS
end
deactivate WS
@enduml

```

# Wallet Server Initiated

## Remove Static Tokens Without CVC2

Wallet Server checks after IBAN digitization for static token whether static token has CVC2 available on the Zapp side, since there might be scenarios where even static token is created on MDES side, it is not created on the Zapp side. If it is not available for some period of time, Wallet Server deletes static token, unenrolls from ACS and User have to digitize IBAN for static token once again.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30

```

```

skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant "Wallet Server" as WS
participant MDES
participant Zapp
WS -> WS: 1. getStaticTokensWithoutConfirmedCVC2OnZapp
activate WS
WS -> Zapp: 2. searchToken(tokenUniqueReference)
activate Zapp
Zapp --> WS: 3. response
deactivate Zapp
alt CVC2 available on Zapp
WS -> WS: 4. confirmStaticTokenHasCVC2
else CVC2 not available
alt CVC2 not available for some period of time
WS -> MDES: 5. deleteStaticToken
activate MDES
MDES --> WS: 6. response
deactivate MDES
deactivate WS
note over WS, MDES #1C1E3F: Unenroll Static Token from ACS
end
end
@enduml

```

## Removing Not Provisioned Device Tokens



Wallet Server checks periodically device Payment Tokens and verify if provisioning is completed. These Payment Tokens which have provisioning status in progress for long period of time are deleted automatically and from User perspective process needs to be started again.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
WS -> WS: 1. find not provisioned payment tokens for a long period of time
activate WS
loop Not provisioned payment tokens for a long period of time
WS -> MDES: 2. Delete token
activate MDES
MDES -> MDES: 3. Delete token mapping
MDES --> WS: 4. response
deactivate MDES
WS -> SDK: 5. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
activate SDK
alt Request session if required
SDK -> MDES: 6. request session
activate MDES
MDES --> SDK: 7. response
```

```
MDES -> WS: 8. sendRemoteNotificationMessage
deactivate MDES
activate WS
WS -> SDK: 9. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
end
SDK -> MDES: 10. delete(tokenUniqueReference)
activate MDES
MDES --> SDK: 11. response
deactivate MDES
SDK -> SDK: 12. Delete transaction credentials, card profile
SDK -> MPA: 13. onPaymentInstrumentStatusChanged(id, status)
deactivate SDK
end
deactivate MPA
@enduml
```

# Wallet Server Admin API Initiated

## Admin Device Deletion

During this process all data related to given device are deleted. Payment Tokens are deleted asynchronously.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
}
```

```
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant "Admin Panel" as AP
AP -> WS: 1. deleteDevice(deviceInstallationId)
activate AP
activate WS
WS --> AP: 2. response
deactivate AP
loop All device Payment Tokens for given device
WS -> MDES: 3. delete token
activate MDES
MDES --> WS: 4. response
deactivate MDES
deactivate WS
end
@enduml
```

## Admin Device Token Deletion

Device Payment Token can be deleted via admin panel.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
}
```

```
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant "Admin Panel" as AP
AP -> WS: 1. deletePaymentToken(paymentTokenId)
activate AP
activate WS
WS -> MDES: 2. Delete token
activate MDES
MDES -> MDES: 3. Delete token mapping
MDES --> WS: 4. response
deactivate MDES
WS --> AP: 5. response
deactivate AP
WS -> SDK: 6. deliverMessage(paymentTokenDeleted)
NOTE LEFT: See: Handle Message From Server
deactivate WS
activate SDK
deactivate MDES
alt Request session if required
SDK -> MDES: 7. request session
activate MDES
MDES --> SDK: 8. response
MDES -> WS: 9. sendRemoteNotificationMessage
deactivate MDES
activate WS
WS -> SDK: 10. deliverMessage(mdesRemoteMessage)
NOTE LEFT: See: Handle Message From Server
deactivate WS
end
SDK -> MDES: 11. delete(tokenUniqueReference)
activate MDES
MDES --> SDK: 12. response
deactivate MDES
SDK -> SDK: 13. Delete transaction credentials, card profile
SDK -> MPA: 14. onPaymentInstrumentStatusChanged(id, status)
deactivate SDK
MPA --> User: 15. show update view
deactivate MPA
@enduml
```

# Admin Device Token Suspension

Payment Token can be suspended via admin panel.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant "Admin Panel" as AP
AP -> WS: 1. suspendToken(paymentTokenId)
activate WS
activate AP
WS -> MDES: 2. suspend token
activate MDES
MDES -> MDES: 3. Suspend token
MDES --> WS: 4. response
deactivate MDES
WS --> AP: 5. response
deactivate AP
WS -> SDK: 6. deliverMessage(paymentTokenSuspend)
NOTE LEFT: See: Handle Message From Server
deactivate WS
activate SDK
SDK -> SDK: 7. suspend
```

```
SDK -> MPA: 8. onPaymentInstrumentStatusChanged(id, status)
deactivate SDK
deactivate MPA
@enduml
```

# Admin Device Token Unsuspension

Payment Token can be unsuspended via admin panel.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant "Admin Panel" as AP
AP -> WS: 1. unsuspendToken(paymentTokenId)
activate WS
activate AP
WS -> MDES: 2. unsuspend token
activate MDES
MDES -> MDES: 3. Unsuspend token
MDES --> WS: 4. response
deactivate MDES
WS --> AP: 5. response
```

deactivate AP

WS -> SDK: 6. deliverMessage(paymentTokenUnsuspend)

NOTE LEFT: See: Handle Message From Server

deactivate WS

activate SDK

SDK -> SDK: 7. activate

SDK -> MPA: 8. onPaymentInstrumentStatusChanged(id, status)

deactivate SDK

deactivate MPA

... Replenishment ...

note over SDK, MDES #1C1E3F: Just after token activation transaction credentials replenishment is performed by SDK\r. See Transaction Credentials Automatic Replenishment diagram

@enduml

## Admin IBAN Deletion

IBAN is stored in context of given User. Because of that IBAN is always deleted in context of User which belongs to. If other User also has the same IBAN then it is not deleted.

@startuml

skinparam ParticipantPadding 30

skinparam BoxPadding 30

skinparam noteFontColor #FFFFFF

skinparam noteBackgroundColor #1C1E3F

skinparam noteBorderColor #1C1E3F

skinparam noteBorderThickness 1

skinparam sequence {

ArrowColor #1C1E3F

ArrowFontColor #1C1E3F

ActorBorderColor #1C1E3F

ActorBackgroundColor #FFFFFF

ActorFontStyle bold

ParticipantBorderColor #1C1E3F

ParticipantBackgroundColor #1C1E3F

ParticipantFontColor #FFFFFF

ParticipantFontStyle bold

LifeLineBackgroundColor #1C1E3F

LifeLineBorderColor #1C1E3F

}

actor User

participant MPA

participant "Wallet SDK" as SDK

participant "Wallet Server" as WS

participant MDES

participant "Admin" as AP  
AP -> WS: 1. deleteIban(userId, ibanId)  
activate AP  
activate WS  
WS -> WS: 2. deleteIbanForUser  
WS --> AP: 3. response  
deactivate AP  
WS -> WS: 4. getAllPaymentTokensForUserAndIban  
loop All Payment Tokens for given User and IBAN  
WS -> MDES: 5. Delete token  
activate MDES  
MDES -> MDES: 6. Delete token mapping  
MDES --> WS: 7. response  
deactivate MDES  
alt Device Payment Token  
WS -> SDK: 8. deliverMessage(paymentTokenDeleted)  
NOTE LEFT: See: Handle Message From Server  
deactivate WS  
activate SDK  
deactivate MDES  
alt Request session if required  
SDK -> MDES: 9. request session  
activate MDES  
MDES --> SDK: 9. response  
MDES -> WS: 14. sendRemoteNotificationMessage  
deactivate MDES  
activate WS  
WS -> SDK: 10. deliverMessage(mdesRemoteMessage)  
NOTE LEFT: See: Handle Message From Server  
deactivate WS  
end  
SDK -> MDES: 11. delete(tokenUniqueReference)  
activate MDES  
MDES --> SDK: 12. response  
deactivate MDES  
SDK -> SDK: 13. Delete transaction credentials, card profile  
SDK -> MPA: 14. onPaymentInstrumentStatusChanged(id, status)  
deactivate SDK  
MPA --> User: 15. show update view  
deactivate MPA  
else Static Payment Token  
note over WS, MDES #1C1E3F: Unenroll Static Token from ACS  
end  
end  
@enduml



# Admin Static Token Deletion

Static Payment Token may be deleted via admin panel.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "Wallet Server" as WS
participant MDES
participant "Admin Panel" as AP
AP -> WS: 1. deleteStaticToken(paymentTokenId)
activate AP
activate WS
WS -> MDES: 2. Delete token
activate MDES
MDES -> MDES: 3. Delete token mapping
MDES --> WS: 4. response
deactivate MDES
WS --> AP: 5. response
deactivate AP
deactivate WS
deactivate WS
...Unenroll static token...
note over WS, MDES #1C1E3F: See Unenroll Static Token Diagram
@enduml
```

# Admin Static Token Suspension

Static Payment Token may be suspended via admin panel.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "Wallet Server" as WS
participant MDES
participant "Admin Portal" as AP
AP -> WS: 1. suspendStaticToken(paymentTokenId)
activate AP
activate WS
WS -> MDES: 2. Suspend token
activate MDES
MDES -> MDES: 3. Suspend token
MDES --> WS: 4. response
deactivate MDES
WS --> AP: 5. response
deactivate AP
deactivate WS
deactivate WS
@enduml
```

# Admin Static Token Unsuspension

Static Payment Token may be unsuspended via admin panel.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "Wallet Server" as WS
participant MDES
participant "Admin Panel" as AP
AP -> WS: 1. unsuspendStaticToken(paymentTokenId)
activate AP
activate WS
WS -> MDES: 2. Unsuspend token
activate MDES
MDES -> MDES: 3. Unsuspend token
MDES --> WS: 4. response
deactivate MDES
WS --> AP: 5. response
deactivate AP
deactivate WS
deactivate WS
@enduml

```

## Admin User Deletion

During this process all data related to given User are deleted. Payment Tokens are deleted asynchronously.

```

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30

```

```

skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor User
participant MPA
participant "Wallet SDK" as SDK
participant "Wallet Server" as WS
participant MDES
participant "Admin Panel" as AP
AP -> WS: 1. deleteUser(userId)
activate AP
activate WS
WS -> WS: 2. delete ibans, devices
WS --> AP: 3. response
deactivate AP
loop All Payment Tokens for given User
WS -> MDES: 4. delete token
activate MDES
MDES --> WS: 5. response
alt Static Token
note over WS, MDES #1C1E3F: Unenroll Static Token from ACS
end
deactivate WS
deactivate MDES
end
@enduml

```

---

Revision #90

Created 17 March 2023 11:25:51 by Jagoda Mazurek

Updated 27 April 2023 14:54:28 by Krzysztof Boczek