Flame Customizable NFC Wallet Functional Specification

Version 2.2.1

15 February 2021

Notices

Following are policies pertaining to proprietary rights and trademarks.

Proprietary Rights

The information contained in this document is proprietary and confidential to Mastercard International Incorporated, one or more of its affiliated entities (collectively “Mastercard”), or both.

This material may not be duplicated, published, or disclosed, in whole or in part, without the prior written permission of Mastercard.

Trademarks

Trademark notices and symbols used in this document reflect the registration status of Mastercard trademarks in the United States. Please consult with the Customer Operations Services team or the Mastercard Law Department for the registration status of particular product, program, or service names outside the United States.

All third-party product and service names are trademarks or registered trademarks of their respective owners.

Disclaimer

Mastercard makes no representations or warranties of any kind, express or implied, with respect to the contents of this document. Without limitation, Mastercard specifically disclaims all representations and warranties with respect to this document and any intellectual property rights subsisting therein or any part thereof, including but not limited to any and all implied warranties of title, non-infringement, or suitability for any purpose (whether or not Mastercard has been advised, has reason to know, or is otherwise in fact aware of any information) or achievement of any particular result. Without limitation, Mastercard specifically disclaims all representations and warranties that any practice or implementation of this document will not infringe any third-party patents, copyrights, trade secrets or other rights.

Table of Contents

1 Introduction

1.1 Revision History

Version	Date	Comments
V1.0	28th Jan, 2016	First draft
V1.4	4 Feb, 2016	Review done up to section 3.3.4.1: - Removed T&C sections and related T&C text - Written Use Cases 3.2 to 3.3.4.1 from the Corporate Wallet SDK point of view - Marked as (INFORMATIVE) text or sections that describes UI/UX and MPA (called wallet application or just application) behaviors including force upgrade sections/parts - Removed section “Delete Card by User” - Updated solution overview - Added Visa/VTS Placeholder - Rewritten the use cases from the Corporate Wallet SDK point of view including MPA
V1.6	7 Feb, 2016	- Resolved Santander comments from Friday, Feb. 5, 2016 - Added comment to add PIN insertion flexibility before or after card provisioning and selection by the user - Added “Delete Card: By User” - Clarified that MPA is equal to MPA and Corporate Wallet SDK
V1.7	8 Feb, 2016	- Updated sections 3.4, 3.4.1, 3.4.2, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, all sections 4.x - Added card status notification to TM form Wallet Server
V1.9	10 Feb, 2016	- Added in section 3.3.2-3.3.3 o the option to insert the Wallet PIN before or after the card selection o card digitization selection by the user - added section 4.7.2, 4.8 - added flow in section 4.1
V1.10	11 Feb 2016	- Added Tap & Pay flows
V1.11	13 Feb 2016	- Removed OTP and replaced IDP with ID&V - Added section 5.4.2 “Manual SUK Replenishment: Mastercard Cards”
V1.12	17 Feb 2016	- Added section 3.8.1 and 3.8.2 - Added assumptions and considerations - Review section 3.3.3 - Added reference [4] and CES APIs in the use cases
V1.13	18 Feb 2016	- Added notification server Twinpush and related APIs - updated section 8.1 Localization - cleaned up section 3.3.2 and 3.33 - Added ID&V User ID and checked in section: 3.3.2, 3.9, 4.1 - Updated assumptions and considerations
V1.14	19 Feb 2016	- Added car art management - Added T&C workaround due to MDES requiring T&C Identifier and T&C Accepted Timestamp during card digitization

		- Added assumptions on Token Manager and MDES BIN ranges - Added section Santander Token Manager used APIs - Added clarification on the card art and images - Added clarification on the default payment app
V1.15-V1.16	26 Feb 2016	- Updated section 3.3.1 on whitelist - Updated section 3.3.3 on card art URL - Updated section 3.16.1 and 3.16.2 on CES notifications - Updated Assumptions - Added configurations section.
V1.17-18	03 March 2016	- Added suspend/unsuspend card: by user - Local PIN validation with PTC - LVT/HVT - Whitebox – Assumption: the whitebox is intended as a container capable to securely store Wallet PIN and PTC - List of Notification - Double RNSId for MPA and CW-SDK (admin and alert messages)
V1.19-20	08 March 2016	- Added Admin and Tenant Portal - Added Reference MPA - Added Token Validation flow chart - Updated reference [4] CES APIs
V1.21	09 March 2016	- Added Alert/Admin Message row in the terminology table - Removed New Card Flag from CES - Added card resume in Reset Wallet PIN Use Cases - Updated notification table - Reviewed Santander (Prudencio) and Nishith's comments - Added IssuerID in tenant parameters
V1.22	11 March 2016	- Removed cardID in section 3.3.3, 4.1 and Assumptions
V1.23	11 March 2016	- Added velocity check to Tap&Pay operations - Fixed PTC
V 1.24	16 Mar 2016	Following changes are incorporated based on the 1. Wallet PIN is removed and replaced by the device unlock 2. CES is being de-scoped for end of June release 3. Green Path changed to Yellow Path with OTP as the only option 4. T&C will be managed by MDES
V1.25 – V1.26.3	22 Mar 2016	- Replaced sTM with CES (Card Enabler System) - Moved Wallet PIN insertion in wallet registration - Reordered yellow path/green path - Added refund in Tap&Pay Use Cases - Merged wallet activation with wallet registration - Added velocity check sections - Removed Local_PTC and Tap&Pay local Wallet PIN validation - Cleaned section 4.1. replaced by section 3.3.3
V1.27	24 Mar 2016	- Updated T&C - Updated Tap and Pay use cases (section 5.2 and 5.3) added single tap and double tap - Updated velocity check section to consider also Tap & Pay use case section 5.1

V1.28	25 Mar 2016	- Updated “delete card by issuer” and “delete card by customer”
V1.29	29 Mar 2016	- Replaced MP-SDK with URPay - Updated overview diagram - Correct TDS registration/unregistration - Transaction receipt and history moved to
V1.30	1 April 2016	- Added that CW-SDK is always setting a default card. - Reviewed Tap&Pay scenarios - Added failures cases in the FCM registration (“FCM_REGISTRATION_FAILED”) - Moved “4.5 Register with TDS – Background Process (Applicable to Mastercard Cards Only)” to - 3.3.2 updated “TOKEN VALIDATION FAILED” to “TOKEN AUTHORIZATION FAILED” to be aligned with CW-SDK API - Added reason code section and logic
V1.31	4 April 2016	- Added additional authentication method “Customer Service” for yellow path - (REL_1.1) Tag Rework: assigned REL_1.1 for UK 1B release and replace REL_1.1 with BACKLOG for functionalities outside 1A and 1B-UK
V1.31.1	4 April 2016	- As per email exchange “Flame – log viewing” on April 4, 2016 the logs are stored in files instead of database. Logs are not viewable by admin portal but exported as syslog. - Updated draw in section 10.10.2
V 1.32	5 April 2016	- Changed Authorization token to ID&V and Data Token - Added Logout use case for portal - Removed reference MPA use cases - Added notes for login use case on portal - Added SPS check for Digitize Cards use case - Incorporated minor feedback from QA
V 1.32	12 April 2016	- Moved CMS-D Registration to Wallet Registration Step - Minor changes to Tap & Pay, Assumptions and T&C
V 1.33	19 April 2016	- Added second call from CW-SDK to WS to set the wallet PIN due to Wallet PIN generation in the WS - Added OTP sent via E-mail - Updated Wallet Server ID&V token and data token validations using SPS and ID&V services deployed by the bank for registration, add card, reset wallet PIN and Authenticate User via ID&V Login Process Use Case. - Merged Tap & Pay use cases - Added section 10.11 - Updated assumptions on L1/L2 authentication
V 1.33.1	21 April 2016	- Aligned velocity check section with San UK assumptions: o Device screen on to perform a transaction o Only android secure unlock methods are supported, there is not configuration to enforce what it’s secure and what not. - Updated Tap & Pay use case references in the velocity check section
V 1.33.2	24 April 2016	- Added buildType - Updated 10.9.2 section as per BRD v1.18 - Added parameter 7 in section 10.4.2 - Updated ID&V/SPS as per new specification from Santander

V 1.33.3	27 April 2016	- Tap&Pay add phone screen on check by the CW-SDK - Added that getTnC retrieves the T&C from MDES via WS/CW-SDK - Updated Terminate Wallet and Add Card concerning deleting the cards and wallet notifying the issuer when app is reinstalled
V1.34	02 May 2016	- Added Synchronization API in section 3.2 - Added assumption on URPay and rephrased assumption on T&C in section 10.1 - Updated section 10.10
V1.34_Visa	11 May 2016	- Added Visa
V 1.35	27 May 2016	- Development changes as well as minor updates.

V 1.36.3	20 June 2016	- Added Prudencio feedback and comments
		- Updated RNSId (RNS_ID) now only registered by the MPA and all push notification received by the MPA
		- Added Brazil velocity check
		- Added changes related
		  o Development team feedback
		  o Clarifications from Fred/Francesco/SAN
		  o Visa decisioning data
		  o Admin Portal use cases and blacklist
		- Updated Force Upgrade
		- Revisited all REL_1B and BACKLOG according to new roadmap
		- Removed Visa manual replenishment UC
		- “Unable to assess” is added to MDES reason codes
		- Added “*we always keep the latest one only if this digitization, and thus additional authentication, was successful*” for calculating decisioning data as agreed with UK

V 1.36.4	23 June 2016	- Clean-up in confcall with Prudencio. - Tenant Admin replaced by Business Admin, as in R1.1 it can only manage blacklist, which fits into Business Admin role in BRD. Tenant Admin will thus be BACKLOG.
V 1.36.5	23 June 2016	- Updated UC Register Wallet for Visa in URPay following discussion with Inside Secure (JIRA ticket MCD02-18)
V 1.36.6	23 June 2016	- Updated notifications section
V 1.36.7	30 June 2016	- Cleaned-up revisions from previous document version - Old wallets deletion is moved to BACKLOG
V 1.36.8	7 July 2016	- Precisions in “CW-SDK Configurations” (including deletion of “minimum LUK threshold” as this cannot be set by MPA but is under control of VTS). - In Admin Portal, remember last 4 passwords is set as BACKLOG. - Unified “notification update” use cases. - For Delete, Suspend, Un-Suspend UC, added that for Visa, it is required to provide reason code to VTS. - Corrected replenishment logic for Visa. It is always under control of VTS.
V 1.36.9	8 July 2016	- Corrected synchronization use case (6.2) - Updated velocity check rule for UK for Visa TfL transaction - -Added changes related to Visa

		- Corrected Format according to Mastercard requirements. - Added rule for Chile Velocity Check.
V1.37	1st Aug 2016	- Added OTP exception flows for Mastercard and Visa cards - Added clarification on Chile velocity check
V1.37.1	3rd Aug 2016	- Changes on SD (Session Duration) for Brazil from 180 sec to 30 sec
V 1.37.1_1	2nd Sep 2016	- Internal changes.
V 1.37.2	9 Sept 2016	Changes after LLD discussions with team regarding release 1.1 and 1.0.1
V 1.37.3	10 Oct 2016	- New requirement included for Digitize a Cards List, use case 4.3.4 and removed note in case 4.3.3 in step 3 doing reference to Digitize Card list requirement as a backlog - CES Notifications will be handled by the Wallet Server, the notes of backlog for notifications deleted in use cases 4.3.3, 4.3.3.1, 4.13, 4.16, 5.2.1, 5.2.2, 5.4.1, 5.4.3, 5.4.4, 11.1, 11.2, 11.3, 11.4.2, 11.5 and 11.5.1 - In section 11.1 Assumptions delete Backlog tag for The Wallet Server shall communicate the RNS_ID to the Santander Card Enabler System to allow alert message push notifications.
V1.37.4	25 Oct 2016	- Changed previous note to modify release name to R1.2 - Removed from case 4.3.3 (Backlog) note for Corporate Wallet SDK will return “CES TIMEOUT” Removed “Visa Note” pointing to R1.1 in Step 11 case 4.3.4
V1.38	2 Nov 2nd, 2016	- Updated doc as per R1.0/1.1: explicit the CardReferenceID - Updated doc as per R1.0/1.1: consequence of unsecure unlock (URPay init error - Updated doc as per R1.2: STM notifications/digitize list - Removed “unsecure unlock” from Chile rule for CDCVM in section 11.9.3.
V 1.38.1	Nov 18th, 2016	- Created new UC 9.1 as per R1.2: Auto recovery procedure - Updated UC 4.2 Launch MPA and 6.1 Tap and Pay as per R1.2:Auto recovery procedure
V 1.38.2	Nov 22nd, 2016	- Updated Use cases as per TUR Limit - Removed Flags and CW-SDK error codes - Updated exception flow for condition when WS fails to send notification to CES - Updated Add Card List to accommodate scenario when RAA card encountered - Added the following logics to the Auto-recovery Use case o Default card o User suspended to suspended state o Yellow path o Wallet pin

V 1.38.3	Dec 2nd, 2016 Dec 6th, 2016 Dec 8th, 2016 Dec 13th, 2016	- Updated the Add card: List UC as per the BRD to reflect the scenario that card list processing should stop when a yellow flow card is discovered - Alternate Flows added in case of connection error between WS and CW-SDK after suspension, Unsuspension, Deletion - - Updated document as per final Auto recovery approach - Updated document as per review comments on auto recovery final approach Updated document as per the document 20161206 – approach for auto recovery Updated Visa Transaction Receipt Use case
V 1.38.4	Dec 14th, 2016 Dec 15th, 2016 Dec 16th, 2016	- Added new Use case to elaborate the Card Mapping scenario during mass card expiry or when user reports card stolen or lost - Updated Use case View Mobile Wallet Dashboard to the extend logic of Auto recovery implementation - Updated Add Card List Use Case to include Yellow Path - Updated Use cases to reflect the start and end logs to be maintained for the auto-recovery process
V 1.39	Dec 19th, 2016	- Final document version planned for R 1.2
V 1.39.1	Dec 22nd, 2016	- Updated Ucs as per confirmation received from Client on CES notification re-transmission - Removed all flags
V 1.39.2	Jan 4th, 2017 Jan 11th, 2017 Jan 16th, 2017 Jan 31st, 2017 Feb 6th, 2017 Feb 8th, 2017	- Added the Refresh scenario to Auto recovery Use case - Added a new section 12.10.3 - Updated Tap and Pay UC, step 5 - Updated notes section in the Transaction history Use Cases - [R 1.0] Updated SDK pre-set up process as per CR Flame – Admin Console – Add blacklist entry v0.6 - [R 1.2] Updated Remapping UC - [R 1.0] Updated Tap and Pay UC as per CR Flame Brazil Improve POS V2 User Experience ID v 0.1 - Updated Velocity Check – Brazil UC as per waiver signed on Jan 20th, 2017 - [R 1.2] Updated Auto recovery UC to incorporate L1 authentication - Updated section 12.6 to reflect L1 auth checks for Autorecovery - Updated Digitize cards Mastercard and Visa to reflect the updation of the DB registration timestamp

		- Updated T&C
V 1.40	Feb 10th, 2017	- Clean-up
V 1.40.1	Feb 14th, 2017	- Removed Wallet Registration TS update from UC Register Wallet
V 1.40.1.1	Feb 14th, 2017	- Removed Wallet Registration TS update from Digitize Mastercard and Visa UC - Updated Remapping UC to remove oldmaskedPAN details being sent to MPA from CW-SDK - Added note in Register Wallet UC for the AppInstanceCreationTimeStamp
1.40.1.2	Feb 15th, 2017	- Added the AppInstanceCreation Timestamp details in Register Wallet UC
1.40.1.3	Feb 16th, 2017	- Added changes to the Remapping UC to reflect the CW-SDK storing the old and new Masked PAN - Updated Notes section for Remapping UC
1.41	Feb 16th, 2017	- Clean up
1.42	Feb 21th, 2017	- Remove comment, accept changes and publish
1.42.1	Feb 21st, 2017	- Added section 12.4 Device Blacklisting – Exception Flow
1.42.1.1	March 1st, 2017	- Updated Exceptional Flow for Add Cards – Digitize Mastercard Cards UC for Provisioning failure scenario - Updated Notes Section for Auto-recovery UC for condition when user deletes token to be suspended after partial recovery
1.42.1.2	March 2nd, 2017	- Updated the Suspend/ Unsuspend UCs as per CR “PCR Visa card suspend handling – v 0.2”
1.42.1.3	March 8th, 2017	- Updated Tap & Pay UC as per Flame terminal V2 V3 CVM limit
1.42.2.1	April 24th, 2017	- Added section Safety net check
1.42.2.2	May 1e, 2017	- Updated notes section for Add Blacklist entry
1.42.2.3.	May 9th, 2017	- Updated Tap & Pay UC, step 5
1.42.2.4	May 15th, 2017	- Updated Auto recovery UC & notes section on the steps to be taken for Issuer suspended case

1.42.2.5	June 2nd, 2017	- Updated notes section for Login and Logout Portal Access Services UCs
1.42.2.6	July 10th, 2017	- Updated UC 4.3.2 Set up CW-SDK: Register Wallet - Added section 12.6 READ_PHONE_STATE Permission required during device fingerprint generation - Added section 12.7 UID management for Chile
1.42.2.7	July 12th, 2017	- Updated section 12.4 Device Blacklisting – Exception Flow
1.42.2.8	September 5th, 2017	- Added the SafetyNet Attestation check in the following use cases: o Step 4 of section 4.2 Launch MPA. o Step 7 of section 4.3.3 Setup Corporate Wallet SDK: Add Card(s). o Step 4 of 'Exceptional Flows' in section 4.3.3 Setup Corporate Wallet SDK: Add Card(s). o Step 6 of section 4.3.4 [rREL_1_2] Set up Corporate Wallet SDK: Add Cards List. o Step 5 of 'Exceptional Flows' in section 4.3.4 [REL_1_2] Set up Corporate Wallet SDK: Add Cards List. o Step 7 of section 9.1 Auto- recovery after URPay init error (due to secure unlock). o Step 7 of 'Exceptional Flows' in section 9.1 Auto-recovery after URPay init error (due to secure unlock). - Added Digitization by List Serialization change in the following use cases: o Updated step 10 Set up Corporate Wallet SDK: Add Card(s) as follows: "Performs further serialized process for digitization for each card included in the list." - Added a new section: [REL_1_2] Digitize by List Serialization.
1.42.2.9	September 20th, 2017	- Modified the CW-SDK initialization to include architecture check: o Section 4.3.1 - Added a new section [REL_1_2] Architecture check
1.43	September 20th, 2017	- Clean-up
1.44	December 23rd, 2017	- Updated the following sections to include limitation of 10 tokens to be sent during delete, suspend and un-suspend operations. Following use cases were impacted: o Section 5.2.1 Delete Card by User o Section 5.2.2 Delete Card by CSR o Section 5.4 Suspend Card [Backlog] o Section 9.1 Auto-recovery after URPay/ URPay init error
2.1	April 26th, 2018	- Added sections o 1.5 Supported Platform and Devices. o 7.6 Tenant Parameters Configuration o 7.7 Key Rotation Mechanism (Technical Story) - S189502

2.3.1	September 3rd, 2013	- Removed sections
		  o Get Payment Card Update (Background Process)
		- Updated sections
		  o Updated GCM to FCM (including registration process) wherever applicable.

2.3.2	7 June 2019	- Added section o Velocity Check for Portugal - Updated Mastercard logo
2.2.0	13 April 2020	- Document name and version changed from Customizable NFC Wallet Functional Specification Version 2.3.2 to Flame Customizible NFC Wallet Functional Specification Version 2.2.0
2.2.1	15 February 2021	- Removed version tags in references to the functionalities. - Removed most INFORMATIVE and REF_MPA tags. - Added new functionalities present in the latest version of our solution. - Reviewed existing features and added corrections.

1.2 Glossary

This section explains and defines words and acronyms that are used throughout this document.

Abbreviation	Description
AES	Account Enablement System
API	Application Programming Interface
ATC	Application Transaction Counter
BRD	Business Requirements Document
CES	Card Enablement System
CMS–D	Credentials Management System Dedicated
CSR	Customer Service Representative
CVC	Card Verification Code
CVM	Cardholder Verification Method
DFP	Device Fingerprint
DPAN	Device Primary Account Number
EMV	Europay Mastercard Visa
FCM	Firebase Cloud Messaging

FI	Financial Institution
FPAN	Funding Primary Account Number
HCE	Host Card Emulation
HSD	High Value Transaction Session Duration
HVT	High Value Transaction
ID&V	Santander Authentication System
LSD	Low Value Transaction Session Duration
LVT	Low Value Transaction
LUK	Limited Use Key
MCBP	Mastercard Cloud Based Payment
MDES	Mastercard Digital Enablement Service
MNO	Mobile Network Operator
MPA	Mobile Payments Application
MPN	Mobile Phone Number
NFC	Near Field Communication
NOT	Number of Transactions
NTU	Notify Token Updated
PAID	Payment Application Instance ID
PAN	Primary Account Number
PIN	Personal Identification Number
QR	Quick Response
RAA	Requires additional authentication
RFID	Radio Frequency Identification
RNS	Remote Notification Service
SBMP	Software-Based Mobile Payments
SD	Session Duration
SDK	Software Development Kit
SPS	Signature Pattern Service
SUK	Single Use Key
TAV	Tokenization Authentication Value
TDS	Transaction Details Service
TER	Tokenization Eligibility Request

TFL	Transport for London
TSP	Token Service Provider
TTL	Time to Live
TUR	Token Unique Reference
TVN	Tokenization Event Notification
VTS	Visa Tokenization System

1.3 Definitions

This section explains some of the key terms and concepts used in this document:

Term	Meaning
Account PAN	The cardholder's real primary account number. PAN used by the issuer to fund the transaction.
Admin / Alert Message	An Admin Message is implemented either as a push notification, web-service or API (e.g. Callback) used for service and administration purposes. Admin Messages can reach the MPA. An Alert Message is a consumer message that means a message that the MPA should display to the user. Example of Alert Messages are Transaction Receipt or New Card(s) Available. See also section "Notifications"
Authorization Token	Token generated by the Santander ID&V system and returned to the MPA after a successful user authentication, see [1]
Business Admin	In Santander BRD terminology, a Business Admin is able to perform "business activities", which in the present specifications represent "updating whitelist/blacklist".
Card Availability	A check to determine whether a card is generally within an available range for digitization.
Card Assets	They are the card images and the product names to be displayed in the Mobile Wallet.
Card Eligibility	A check to determine whether a specific card is eligible for digitization.
Card State – Active	This state indicates that a payment card is successfully approved and provisioned to user. The user will be able to see colored payment card and will be able to use it for payment
Card State – Deleted/Removed	This state indicates that user will no longer be able to use the card. The card deletion/removal can happen in the following ways: • By User: The user wants to discontinue using a payment card and opts to remove it from the wallet. The Wallet Server will delete the card. The card will disappear from the Mobile Wallet. • By CSR: The CSR deletes the card via MDES/VTS portal. The Wallet Server will delete the card and notify the Mobile Wallet. The card will disappear from the Mobile Wallet.

	• By CSR: The CSR deletes the card via MDES portal. The Wallet Server will delete the card and notify the Mobile Wallet. The card will disappear from the Mobile Wallet. • By Wallet Server: This is triggered by execution of the "Terminate Wallet – By Wallet Server" use case wherein a request to register Wallet is received from a device for which an "Active" Wallet exists on the Wallet Server. The Wallet Server terminates the existing record and deletes the associated cards and proceeds with the new registration request.
Card State – Suspended	This state indicates that an authorized administrator (on behalf of bank, issuer) has temporarily suspended the payment card due to certain business or technical reasons. Once the cause is resolved, the card can be resumed. There shall be some appropriate visual indicator to identify the suspended card.
Contactless	A wave-and-pay system that employs RFID technology, and allows shoppers to pay by touching their device against an electronic reader
Credentials Management System	The Credentials Management System (CMS) is responsible for provisioning data from the Account Enablement System, creating Transaction Credentials and delivers them to the Mobile Payment App on the Device. The Credentials Management System is split architecturally into two components: Core – It holds the master keys for the Token, from which Session Keys are derived. Dedicated – which receives the Session Keys from Core, combining them with the Mobile PIN (or other CVM) and the device profile to transform them into Transaction Credentials for use by the Mobile Payment App.
Credentials Management System	Accepts provisioning data from the Account Enablement System, creating Transaction Credentials and delivers them to the Mobile Payment App on the Device.
Digitization	Process of personalization and provisioning of account credentials, cryptographic keys (EMV/Chip), and associated data, into digital devices
Firebase Cloud Messaging	Firebase Cloud Messaging is a service that allows a server to send data to an Android-powered device, and also to receive messages from devices on the same connection. The Firebase Cloud Messaging service handles all aspects of queuing of messages and delivery to the target Android application running on the target device.
Host Card Emulation	Host Card Emulation allows the Device to act as a chip card without requiring a chip based Secure Element. One option to implement HCE is for a software-based application in the Device host to interact with the NFC Controller. The Host Card Emulation functionality may be called by different names depending on the Device Operating System. For simplicity, this document refers to the functionality as Host Card Emulation and does not imply a specific implementation or Device Operating System.
Identification and Verification (ID&V)	Process of identifying and verifying a cardholder prior to initiating digitization.
Issuer	The financial institution that issues payment cards and holds the account or credit line behind the card.
Mastercard Digital Enablement Service	Mastercard digitization and tokenization platform.

Mobile Payment Application (MPA) or Mobile Wallet	A mobile payment Application providing User experience and User interface for Contactless transaction. A Mobile Payment Application can be loaded with more than one Digitized Card.
Device	The Device is a smartphone owned by the User. The Device has the ability to download applications from a standard Mobile Operating System Application Store, such as Google Play.
Device Fingerprint	The device Fingerprint is defined as a list of information collected about the Device and the Mobile Payment Application running on that Device. It is used for the purpose of binding information with the Device and the instance of the Mobile Payment Application actually used by the user.
Masterdata	Refers to the device details of the consumers including vendor, model name, operating system and operating system version. Masterdata is automatically populated into the database and issuers can make use of it to blacklist devices should they want to prevent particular models from using the application (e.g. useful in case of compatibility issues).
Mobile PIN	A personal value shared between User and issuer (not an offline PIN, or mPIN) for a Payment Card. Can be the same as the cardholder's online PIN or any other value. The Mobile PIN is always required for both low and high value transactions initiated with the Mastercard Cloud-Based Payments token. The Mobile PIN is entered by the cardholder into the device, but the Mastercard Cloud-Based Payments Mobile Wallet does not validate the Mobile PIN offline; it is implicitly validated online as part of the cryptogram validation. Incorrect Mobile PIN entry will result in a malformed cryptogram being generated. The Mobile Wallet application does not have any support of offline validation of the Mobile PIN value. The mobile application has no other means than doing an online transaction to know whether the Mobile PIN value was correct or not. That way the issuer (Mastercard) controls the Mobile PIN validation process and can apply some fraud detection techniques (using for example the concept of a Mobile PIN Try Counter). This applicable for Mastercard cards only for the project.
Near Field Communication (NFC)	Near field communication (NFC) is a set of standards for smartphones and similar devices to establish radio communication with each other by touching them together or bringing them into proximity, usually no more than a few inches.
Operator	In Santander BRD terminology, this role is able to modify configuration files of the wallet server and perform other daily operations (such as wallet server restart & backup).
Play Store	Play store is a type of digital distribution platform for mobile apps.
RNS	It stands for Remote Notification Service. It is a platform-specific service used to send data from a server to a Device. Examples of Remote Notification Services include Google Cloud Messaging, Apple Push Notification Service, or Microsoft Push Notification Service.
Suspend	Token has temporarily been suspended and cannot make a payment transaction.
Token	Surrogate value for an Account PAN that is a 13 to 19-digit numeric value that is associated to a consumer's card.
Token Credential	The credential (for example, Card Profile) for the Token as generated by the MDES.

Token Unique Reference (TUR)	Reference returned by MDES to the Wallet Server / Corporate Wallet SDK uniquely identifying the token of a card that has been digitized. It is needed by the Corporate Wallet SDK to fetch the transaction details or the transaction history from TDS.
Token Vault	A repository that maintains the established Token to Account PAN mapping.
Tokenization	Tokenization is the process of replacing the Cardholder Primary Account Number (PAN) with a surrogate value (that is, a Tokenized PAN) that is used in place of the PAN in payment transactions.
Tokenization Authentication Value (TAV)	A cryptographic authentication value that is generated by the Issuer and verified by the Mastercard Enablement System to authorize a digitization request.
Transaction Details Service (TDS)	Service provided by MDES in order to retrieve the transaction details after a transaction has been done or to receive the transaction history related to a specific token (i.e. card) digitized in the MPA.
User	The labels Consumer / User / Cardholder refer to the same entity, a User of the NFC Wallet.
Wallet PIN	Wallet PIN is a personal value shared between User and Wallet server for Wallet User authentication.
Wallet Server	Wallet Server is the entity that manages the Wallet lifecycle, provides services for managing front end experience. Wallet server also provides integration with MDES and VTS Digitization APIs.
Wallet State – Active	The Wallet state transitions to this state from “Stateless” state to “Active” when response to registration is prepared by Wallet Server.
Wallet state – Stateless	The Wallet State is stateless when the wallet has been downloaded and initiated and PIN set up pending.
Wallet State = Terminated/Deactivated – By Server	This is an end-state, in other words, wallet cannot be recovered from this state and user can no longer use the wallet. The Wallet Termination shall deletion/removal can happen in the following ways: • By User: The user has an option to terminate/de-activate the Wallet from the Mobile Wallet. When the user exercises this option, the Wallet Server shall terminate the Wallet record and all the data from the Mobile Wallet shall be removed. The user can once again set up a Wallet by undergoing Wallet set up process without having to download and install the application. • By Wallet Server – The Wallet Server will terminate Wallet record if it a request to register Wallet is received from a device for which an “Active” Wallet exits on the Wallet Server. The Wallet Server terminates the existing record and deletes the associated cards and proceeds with the new registration request.

1.4 References

Sr No.	References
1	Mastercard Digital Enablement Service – API Specification • Version 1.1.3
2	Visa Digital Solutions – API Reference Guide • Version 19.04

3	Santander – 2016_IDV_v1.1
4	Santander Security Architecture Document – version 2.2.1
5	Santander Software Architecture Document – version 2.2.1
6	Flame MWB Admin Portal User Guide – version 1.0.1
7	HSM API Test Utility Developer Guide
8	Flame CW-SDK Integration Guide - v2.2.1
9	Flame CWS & Admin Portal Application Installation Guide

1.5 Supported Platform and Devices

Corporate Wallet SDK runs on Android devices only given the limitation of iOS to use the NFC capabilities to make payments outside of Apple Pay. The Corporate Wallet SDK works in conjunction with a Payment SDK in charge of managing the NFC communication and storing the payment token information.

The minimum Android OS version supported is 4.4 (earlier versions of Android do not support HCE) in devices that include NFC and HCE. The latest Android version supported as of February 2021 is Android 11.

URPay, the Payment SDK used by the Corporate Wallet SDK, only supports ARMv7 (armeabi-v7a) and ARMv8 (arm64-v8a) architectures. For security reasons, URPay requires that users of Android devices enable a secure device unlock method (PIN, Passcode, Pattern or Biometric).

Operating system	**Versions supported**							
Android	4.4 (K)	5.X (L)	6.0 (M)	7.X (N)	8.X (O)	9 (P)	10 (Q)	11 (R)
iOS	Not supported							

In addition to the supported Android versions, CW-SDK makes use of AndroidX support libraries (jetpack). MPAs integrating the CW-SDK should also use the AndroidX libraries and not the traditional and already deprecated support libraries.

2 About Mastercard Wallet Solutions

Mastercard Wallet Solutions offers customizable mobile and online solutions enabling Financial Institutions (FIs), Mobile Network Operators (MNOs), and merchants to deepen digital engagement with their consumers.

In a nutshell, Wallet Solution platform supports the following features, which are essential to drive customer engagement in varying market conditions.

- Development of new, customized mobile and web-based programs.
- Enhancement of existing programs (already in-market) with additional features.
- Integration with the customer's existing back end systems.
- Continuous technology evolution to cope up with the diversified mobile and online platform.

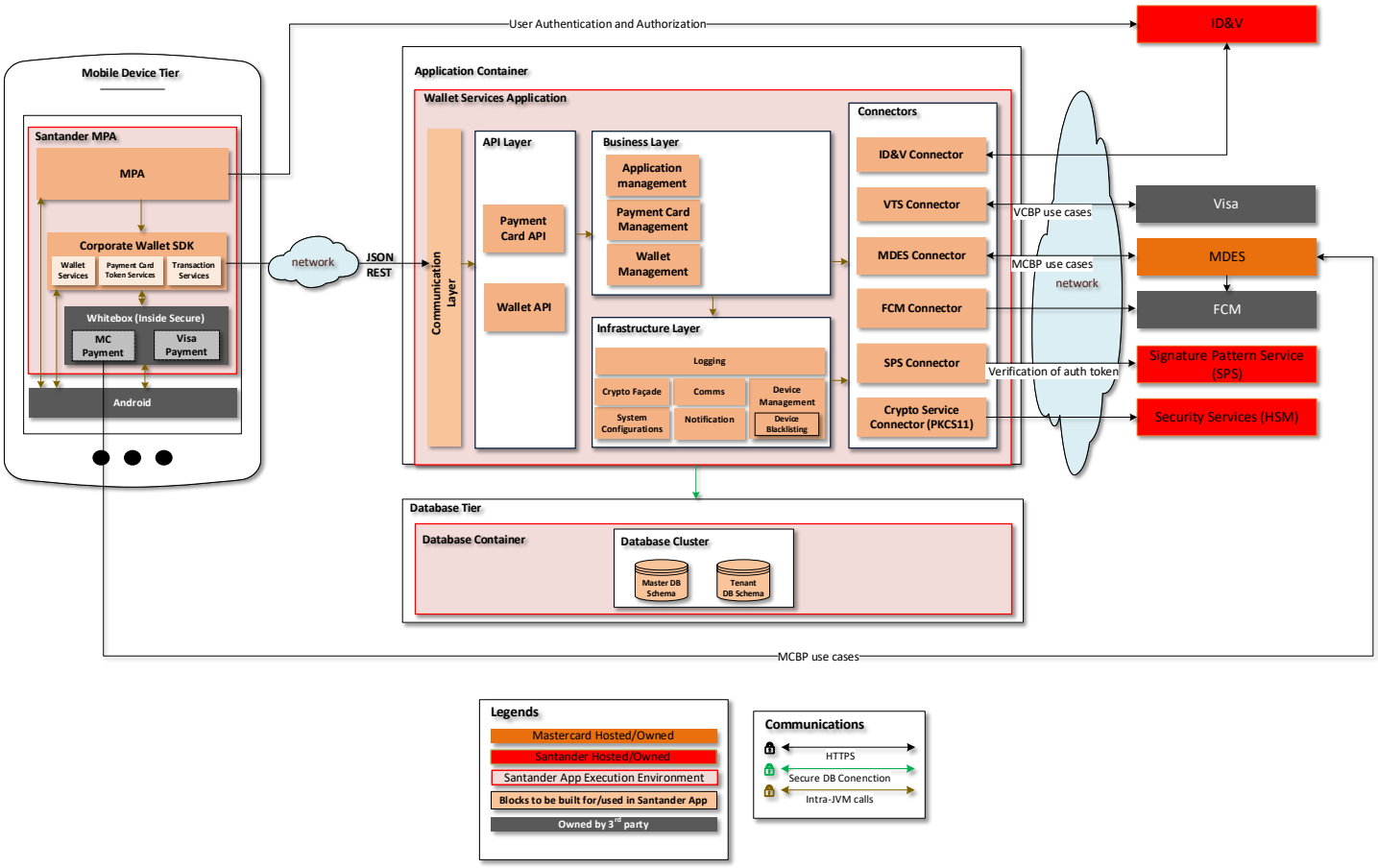
We believe an ideal digital mobile solution should reach beyond its primary function, that is, payment facilities. By incorporating necessary secondary features within a single mobile solution, we try to provide a complete customer experience.

Customers often utilize Mastercard supported payment services such as Masterpass, NFC, QR/Barcode-based transaction systems alongside other digital assets. Mastercard aims to develop solutions which add values to them, foster innovation of new technology and, help driving customer adoption and usage.

3 Solution Overview

Following diagram provides an overview of the proposed solution.

Figure 1. Solution Overview



Sr. No.	Component	Description
1	User	The existing Issuer's customer who is interested in downloading the Mobile Wallet and using it for performing transactions.
2	Mobile Wallet (MPA)	See Definition section. Mobile Wallet will integrate with following: 1. CMS-D/ Mobile Payment APIs: for card and token provisioning. 2. MDES Transaction Detail Service: for transaction details and history services. 3. Wallet Server: for Wallet services. The MPA is composed of:

		- MPA: mainly responsible for the UI, T&C management, Santander ID&V user authentication, and user card selection of cards to be digitized. - Corporate Wallet SDK: the component including Inside Secure's Payment SDK (URPay) supporting payments with Mastercard and Visa cards. It is responsible for the execution of the flows described in this document.
3	Wallet Server	See Definition section. The Wallet Server will support the Wallet lifecycle management and will integrate with MDES Digitization APIs and VTS to enable payment card digitization.
4	MDES	Tokenization and Digitization infrastructure provided by Mastercard. See Definition section.
5	ID&V, SPS, HSM	They are examples of the Santander system(s) with which the Wallet Server and/or MPA shall communicate for specific tasks: • Authenticate user using ID&V. • Verify an authenticated operation with Signature Pattern Service. • Leverage crypto services provided by HSM (Hardware Security Module).
6	VTS	Tokenization and Digitization infrastructure provided by Visa. The communication between the Mobile Wallet and VTS shall always take place via the Wallet Server.

4 Wallet Life Cycle Services

4.1 Download and Install MPA (INFORMATIVE)

This use case describes the process of downloading and installing the MPA onto the User's device.

Actor	1. User 2. App Store 3. MPA
Channels	Device
Pre-Conditions	1. User has a Device with data 2. Internet connection. 3. Application has been published and approved on to the Application Play Store. 4. Device has Google Play Store. 5. Device has NFC and HCE capabilities. 6. User has a valid Google Account.
Trigger Conditions	1. The user wishes to download the MPA.
Start Points	1. The user opens Play Store in the handset. 2. The user opens the Webpage with a link to the Play store or a link within the Issuer Mobile Banking app that directs User to the Play Store to download the MPA.
Post-Conditions	1. User successfully downloads and installs the MPA.
Flow of Events	1. If not logged in, the user logs in to the Google Account 2. After successful, the user open ups the Play Store from the device. 3. The Play Store performs eligibility check based on the filters set while uploading the application to the app store and if checks are passed it displays the application. 4. The user selects the MPA from the Play Store. 5. The Play Store shows the user a list of device services that application will use (for example NFC/Camera, etc.) and prompts the User to consent and grant permission to the application to use those services and initiate the download. 6. The user consents and confirms download of the MPA. 7. The device initiates downloading of the MPA. 8. The MPA gets downloaded and installed on the device and the application icon appears in the application launcher screen.
External Dependencies	1. Play Store. 2. Google Account 3. The Mobile application will be submitted to the Application store by Issuer using its own developer account keys. The application binaries will be provided to Issuer by Mastercard ready to sign using their credentials.
Data Captured	None
Notes	1. If device is not supported, then the Issuer applications will not be shown in the search result on the Play store. plication will be available for download in Issuer countries only. 2. There shall be separate binaries (application bundles) for each country.

	3. Typical filters are as follows: o Wallet Application Configuration ▪ Screen size of the handset ▪ Device Configuration - keyboard, navigation, touch screen ▪ Min & Max OS Version o Play Store Configuration ▪ Application Title ▪ Application Description ▪ Application Type ▪ Application Category ▪ Target Country 4. Setting of the filters and app store upload will be done by Issuer teams.
Business Consideration	1. The device filters will be appropriately set when submitting the application to the Application store so that it can only be downloaded on a set of whitelisted devices. 2. The content for submitting the application to the Play Store (including screen images, promotional information, etc.) will be prepared by Issuer. 3. It is recommended that Issuer create relevant promotional material on their existing website to educate Users about the MPA application and direct them to the Application store for download. Such pages and content are outside the scope of this product. 4. Google play store shall be the only source of application download. 5. There would also be dialogs about setting the Santander NFC Payment Application as the default wallet.it would be on the user to set the Santander Wallet as the default app. 6. From Android M onwards, the user will have capabilities to turn on/off the android permissions.
Exceptional Flows	1. If the user is unable to download the application, an appropriate error is displayed from the Play store, 2. If download is successful and installation is a failure, the user shall have to go to the Play Store and re-download the application. 3. If user doesn't have a Google Account, he/she may need to create one 4. Appropriate error messages will be displayed if the user is unsuccessful to validate his/her Google credentials while opening up the Play Store for downloading the MPA.
Customization	Publication into Play Store and content will be managed by Issuer.
Flow/Sequence Diagram	

4.2 Launch MPA

This use case describes MPA entry into the Corporate Wallet SDK and provides the process of initializing the Corporate Wallet SDK. For the first time MPA launch, the Corporate Wallet SDK establishes secure linkages with the Wallet server to make additional checks before performing the Corporate Wallet SDK setup. On subsequent MPA launches, Corporate Wallet SDK will perform local device checks to ensure that the device environment is eligible and secure to support the Corporate Wallet SDK services.

Actor	1. User 2. MPA 3. Corporate Wallet SDK
Channels	Mobile Wallet
Pre-Conditions	User has a MPA installed on to the mobile phone.
Trigger Conditions	MPA initializes the Corporate Wallet SDK.
Post-Conditions	MPA successfully launches the Corporate Wallet SDK.
Flow of Events	1. The MPA initializes the Corporate Wallet SDK. 2. CW-SDK checks if architecture is supported. For unsupported architectures (x86) then an exception is thrown, without application crash. MPA needs to handle the UX, such as closing application after displaying adequate message. 3. The CW-SDK shall initiate the URPay SDK for Mastercard and Visa. For details on SDK initiation process, please refer the “Notes”. 4. The Corporate Wallet SDK performs various local devices checks such as: a. MPA installed from authorized source (Play Store). i. If the device passes the check then proceed to the next check. ii. If not, then the Corporate Wallet SDK returns an appropriate error b. Rooted device i. If the device passes the check then proceed to the next step. ii. If not, then the Corporate Wallet SDK returns an appropriate error c. SafetyNet Attestation (Asynchronous). i. If device passes the check then proceed to the next step. ii. If not, then appropriate action will be taken depending upon which of the following four scenarios occur: a. Failure of JWS signature verification. b. Failure of nonce verification. c. Technical Error / Unknown Error. d. Failure during JWS payload processing. **Note:** Please refer section SafetyNet for details on the SafetyNet implementation. d. NFC is supported by device e. NFC is active 5. The Corporate Wallet SDK receives minimum upgrade version required of MPA. (It is applicable if upgrade required notification was received from Wallet Server during the previous communication with the Wallet Server; if data connectivity is missing, proceed to next step). a. If yes, please refer “Upgrade Notification” flow. b. Else, proceed to next step. 6. The Corporate Wallet SDK checks if Wallet status is stateless (Note: The status name may change depending on further design discussions) a. If yes, then please refer “Set-up Corporate Wallet SDK: Pre-setup process” flow. b. Else returns to the MPA, proceed to the next step.

	7. The MPA calls the CW-SDK Synchronize that synchronizes with Wallet Server (see Synchronize UC); if data connectivity is missing, proceed to next step. 8. The Corporate Wallet SDK checks (these checks are foreground checks, i.e. sync checks and will be done one by one. If any of checks fails it will be responded by an error): a. If Wallet Status is “Locked”, then returns an appropriate error b. If Wallet Status is “Active” and Digitization is pending, then refer “Exceptional Flow” for “Set up Corporate Wallet SDK: Add Card(s)” c. Else proceed for following checks and returns to MPA: d. If TDS registration is pending, the Corporate Wallet SDK initiates the process for TDS registration of the Wallet Server. For more details please refer “Register with TDS” flow. e. if card deletion is pending, the Corporate Wallet SDK initiates the process for deleting the card from the Wallet Server. For more details please refer “Delete Card: By User” flow. f. if card suspension is pending, the Corporate Wallet SDK initiates the process for suspending the card from the Wallet Server. For more details please refer “Suspend Card: By User” flow. g. if card resume is pending, the Corporate Wallet SDK initiates the process for resuming the card from the Wallet Server. For more details please refer “Unsuspend Card: By User” flow. 9. For Mastercard, if SUK count is lower than the threshold value replenishment required, then it initiates the process for replenishment. For more details please refer “Automatic SUK Replenishment Mastercard Cards” flow 10. For Visa it is controlled by VTS. 11. The MPA displays the wallet dashboard
External Dependencies	None
Data Captured	None
Notes	1. Google play store shall be the only source of MPA download. 2. The rooted device check implemented is built on multiple inputs and checks available. If more checks are available in the future, then it would require changes in the Corporate Wallet SDK and a new integration by the MPA, prompting users to upgrade their existing version. 3. The root check will consider the following criteria with regards to device specs: arm64-v8a, armeabi-v7a. 4. URPay Initiation Process for Mastercard – is done by Corporate Wallet SDK by sending the Mastercard keys (mobile keys) to the URPay. URPay processes the request and sends back the ack to the Corporate Wallet SDK 5. URPay Initiation Process for Visa is done by CW-SDK by sending request to URPay to on-board the device keys. 6. Check for unsupported architecture will be performed on CW-SDK initialization using URPay 2.X. If architecture is unsupported the flow is discontinued and a result is thrown back to the MPA UI.
Business Consideration	

Exceptional Flows	Covered in the “Flow of Events”.
Flow/Sequence Diagram	Please refer to the Sequence Diagrams document for a graphical representation of this flow

4.3 Set up Corporate Wallet SDK

This use case describes the process of setting up the Corporate Wallet SDK for payment. The process is divided into following steps:

1. Pre-setup Process.
2. Register Wallet
3. Activate Wallet
4. Add Card(s)

User Interface and User Experience will be designed to intuitively guide User through above steps.

4.3.1 Set up Corporate Wallet SDK: Pre-setup Process

This use cases addresses process of doing the pre-setup before proceeding with setting up Corporate Wallet SDK.

Actor	1. User 2. Corporate Wallet SDK 3. MPA
Channels	Corporate Wallet SDK
Pre-Conditions	1. Launch Corporate Wallet SDK is complete. 2. User’s device has data connectivity/internet connection.
Trigger Conditions	Corporate Wallet SDK detects that the wallet has not being created (“stateless”).
Start Point	The user launches the Corporate Wallet SDK.
Post-Conditions	The Corporate Wallet SDK successfully completes pre-set up process.
Flow of Events	1. The Corporate Wallet SDK checks if internet is available. a) If yes, then go to next step. b) Else, the Corporate Wallet SDK returns an appropriate error a. The MPA can display an appropriate error. On dismissing the error, the MPA exits. 2. MPA requests, the Corporate Wallet SDK sends request for blacklist check to the Wallet Server. a. The Wallet Server performs the check a) If device is not blacklisted a. The Wallet Server checks if the parameters platform, OS version, OEM and device model are present in the

	masterdata list. Any of these parameters that are not present in the masterdata, are automatically added to the masterdata b. The Wallet Server responds with the SALT value that will be used for Wallet PIN to Mobile PIN transformation. b) Else, sends error response “MOBILE DEVICE BLACKLISTED” to the Corporate Wallet SDK. The MPA can inform the user. The use case should end. 3. The MPA can display the on-boarding screen. 4. The MPA performs the RNS registration to FCM getting the RNS ID. Note: The GCM registration is done by the MPA. The RNS_ID is provided by the MPA to the Corporate Wallet SDK. See section “Push Notifications Handling (FCM)”
External Dependencies	Firebase Cloud Messaging
Data Captured	None
Notes	Concerning push notification using FCM see section “Push Notifications Handling (FCM)”.
Business Consideration	None
Exceptional Flows	None
Flow/Sequence Diagram	Please refer to the Sequence Diagrams document for a graphical representation of this flow

4.3.2 Set up Corporate Wallet SDK: Register Wallet

This use cases addresses process of registering with the Wallet Server.

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. Wallet Server 5. Santander ID&V / SPS 6. CMS-D 7. VTS
Channels	Corporate Wallet SDK
Pre-Conditions	1. User’s device has data connectivity / Internet connection. 2. User is a registered Santander customer with ID&V credentials. **Note** **The Reference MPA delivered with along with the SDK uses mocked or simulated ID&V/SPS credentials.**

Trigger Conditions	Pre-setup process is complete.
Start Point	Pre-setup process is complete.
Post-Conditions	The Wallet Server successfully creates Wallet record and marks it as “active”.
Flow of Events	1. MPA displays ID&V login screen. 2. User enters the ID&V login credentials and proceeds. The user is successfully authenticated by the ID&V System. Please refer “Authenticate User with ID&V Login”. The MPA calls createWallet request providing the Corporate Wallet SDK the RNS_ID, the Wallet PIN (optional) and the Authorization Token. 3. The Corporate Wallet SDK forms the registration request and sends it to the Wallet Server with: a. RNS_ID b. Wallet PIN (optional) **Note: Wallet PIN field is filled if inserted by the user otherwise it is left empty.** c. Device fingerprint d. ID&V token and/or Data token 4. The Wallet Server a. Checks if the ID&V token (L1 validation) is valid using bank deployed services. i. If the Token is invalid the WS will send error message to CW-SDK ‘TOKEN NOT AUTHORIZED’, use case ends. b. Gets the User ID calling the Bank service GetLoggedUID passing the ID&V token. c. Gets the Session ID calling the Bank service GetSecuritySessionUID passing the ID&V token. 5. Wallet server a. Checks if the service is configured for L2 Authentication. If yes, WS checks if the data token has been received or not i. If no data token is received, it would throw an error to CW-SDK, CW-SDK will pass on the error to MPA, use case ends. ii. If data token is received, then b. performs SHA2 on the operations data received from CW-SDK and passes the data token to SPS service for validation of the data token c. Upon successful validation of the data token, SPS returns successful response to WS. i. If the data token Is not valid, SPS will return error codes to WS ii. WS will pass the error message to CW-SDK iii. CW-SDK in turn will pass this message to MPA, use case ends. d. Wallet server continues with the next step. 6. The Wallet Server checks if Wallet with the same device fingerprint exists a. If yes, then removes existing Wallet (please refer “Terminate Wallet – By Wallet Server” use case) and proceed to next step. b. Else, proceed to next step. 7. The Wallet Server generates device keys for VTS. 8. The Wallet Server sends enrol device request to VTS.

	9. VTS enrols the device and sends an acknowledgement to the Wallet Server. 10. Wallet Server fetches the MDES public certificate from the CMS-D and then encrypts it and forwards the same to CW-SDK 11. CW-SDK provides the encrypted the public certificate to URPay 12. URPay generates MDES registration data and provides it to CW-SDK 13. CW-SDK sends MDES registration request to Wallet Server. 14. Wallet Server requests registration with CMS-D. 15. CMS-D processes the request and returns the mobile keyset id and the mobile keys to Wallet Server 16. The Wallet Server a. Creates wallet account and app instance. b. Creates the AppInstanceCreation TimeStamp c. Stores the AppInstanceCreation Timestamp in DB d. Stores the UserID in the wallet record e. Generates the Wallet PIN **Note: Wallet PIN field is returned by the Wallet Server only if not inserted by the user in the MPA. This is the used approach in all Santander deployments till date.** f. Stores the Wallet PIN g. Calculate the HASH_OF_SALTED_PIN 17. The Wallet Server returns to the CW-SDK: a. Device keys (VTS device enrolment), b. Keyset and mobile keys (CMS-D registration), and c. the HASH_OF_SALTED_PIN 18. The CW-SDK transforms the HASH_OF_SALTED_PIN to Secure Mobile PIN as required by MDES using URPay API). Please refer to the SECAD (Security Architecture Document) for further details. 19. The CW-SDK sends the Secure Mobile PIN to the Wallet Server. 20. Wallet Server sends the Wallet PIN to the CMS-D consuming the Set Mobile PIN service. 21. CMS-D processes the request and returns to Wallet Server. 22. Wallet server: a. Marks wallet as “active”. b. Returns to the CW-SDK the wallet status “active”. 23. The CW-SDK: a. Stores the wallet status “active. b. Stores the Secure Mobile PIN and the HASH_OF_SALTED_PIN in URPay. 24. The CW-SDK returns to the MPA: a. The wallet status “active”. b. Notify MPA that MPA is ready to digitize cards (Optional). 25. MPA may want to prompt users to accept the T&Cs.
External Dependencies	None

Notes	1. Inactivity time starts on ID&V login page. In case if the user sends authentication request to ID&V and receives success response with access token and does nothing, auto exit will happen. 2. Concerning the ID&V and SPS, L1 and L2 authentication see [5] 3. Reliability for Corporate Wallet SDK to Wallet Server communication: Please refer section “Corporate Wallet SDK to Wallet Server Communication” 4. Recent Android versions allows having multiple user accounts on the same device. The FS assumes that only 1 user account is present on the mobile device and therefore the Wallet Server limits to have 1 MPA per device based on the fingerprint. This assumption needs to be re-evaluated as new Android version are released. The AppInstanceCreationTimeStamp can be approximated as the first T&Cs approval when the user accepts the T&Cs during wallet registration.
Business Consideration	None
Exceptional Flows	• If MDES (CMS-D) registration: i. Fails, Corporate Wallet SDK shall return “MDES REGISTRATION FAILED”. 1. MPA could display an error. • If VTS device enrolment: i. Fails, Corporate Wallet SDK shall return “VTS REGISTRATION FAILED”. 1. MPA could display an error.
Open Issue	None
Flow/Sequence Diagram	Please refer to the Sequence Diagrams document for a graphical representation of this flow

4.3.3 Set up Corporate Wallet SDK: Add Card(s)

This use case elicits the process of digitizing the card.

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. Wallet Server 5. URPay 6. MDES 7. VTS 8. Santander Card Enabler System / Santander Token Manager
Channels	Corporate Wallet SDK
Pre-Conditions	1. Wallet status is active. 2. The user has authenticated against the Santander ID&V/SPS and the MPA has a valid Authorization Token. 3. User’s device has data connectivity/internet connection.
Trigger Conditions	The Corporate Wallet SDK stores the Wallet status as “Active”.

Start Point	The Corporate Wallet SDK has created the Wallet and its status is “Active”.
Post-Conditions	The card(s) are successfully digitized on the Corporate Wallet SDK.
Flow of Events	1. MPA gets the eligible cards from the Santander backend. 2. MPA can either show the card list to the user that selects the ones to be digitized or digitize all cards automatically. 3. The MPA sends to the Corporate Wallet SDK the card(s) to be digitized 4. The CW-SDK gets the card(s) 5. The Corporate Wallet SDK forms the request to be sent with a. Card details (FPAN, Expiration Date, Card Holder Name) b. Device Fingerprint c. Decisioning Data for the TAR message d. Authorization token 6. The Corporate Wallet SDK a. Collects the “Fraud Information”, see “Fraud Information and Reason Code Association” b. Sends digitization request with “Fraud Information” to the Wallet Server with the card(s) to be digitized, along with ID&V and Data token. 7. Additionally, the Corporate Wallet SDK initiates SafetyNet check asynchronously. If the SafetyNet check is successful, go to next step. If SafetyNet check is not successful, then refer the ‘Exceptional Flows’. 8. The Wallet Server a. Checks if the ID&V token (L1 validation) is valid using bank deployed services. i. If the Token is invalid the WS will send error message to CW-SDK ‘TOKEN NOT AUTHORIZED’, use case ends. b. Gets the User ID calling the Bank service GetLoggedUID passing the ID&V token i. The WS checks if the ID&V token has the same user ID of the wallet record. If the user ID is different then 1. WS will pass the error message to CW-SDK 2. CW-SDK in turn will pass this message to MPA, use case ends. c. Gets the Session ID calling the Bank service GetSecuritySessionUID passing the ID&V token. 9. Wallet server a. Checks if the service is configured for L2 Authentication. If yes, WS checks if the data token has been received or not. i. If no data token is received, it would throw an error to CW-SDK, CW-SDK will pass on the error to MPA, use case ends. ii. If data token is received, then go to next step. b. Performs SHA2 on the operations data received from CW-SDK and passes the data token to SPS service for validation of the data token c. Upon successful validation of the data token, SPS returns successful response to WS. i. If the data token Is not valid, SPS will return error codes to WS ii. WS will pass the error message to CW-SDK

iii. CW-SDK in turn will pass this message to MPA, use case ends.

d. Wallet server continues with the next step.

10. Performs further **serialized** process for digitization for each card included in the list:
    a. For digitization of Mastercard card please refer section “Set up Corporate Wallet SDK: Add Cards: Digitize Mastercard Card".
    b. For digitization of Visa card please refer section “Set up Corporate Wallet SDK: Add Cards: Digitize Visa Card".
11. On completion of digitization of the card the Wallet Server:
    a. Stores the updates
    b. Prepares the response to confirm card addition request with:
        i. TUR(s) and token(s) of the successful digitized Mastercard card(s),
        ii. Tokens with corresponding PANenrolmentId for successful digitized Visa card(s).
        iii. Failure response for card that could not be digitized.
        iv. Card (masked FPAN with identifier for Visa/Mastercard), - response can be without any card in case no eligible card have been received
        v. Error/Failure event for these card(s) (Approved or Rejected)
        vi. Wallet State
    c. Sends response to the Corporate Wallet SDK.
12. The Corporate Wallet SDK receives the response.
13. The Corporate Wallet SDK performs different steps (depending on the type of card) to activate card.
    a. For Mastercard cards
        i. Stores TUR
        ii. if not done before, Initiates TDS registration process in background for Mastercard
        iii. Store failure/error events for display after assets are fetched.
    b. For Visa cards
        i. Stores tokens. For details for storing Visa tokens please refer the “Notes” section.
        ii. Receives ack for storing tokens in URPay SDK and sends it to VTS via Wallet Server. Then, VTS sends response to the URPay SDK via Wallet Server.
        iii. Marks token as “Active”.
        iv. Store failure/error events for display after assets are fetched.
14. The Corporate Wallet SDK does the following:
    a. In case no cards are eligible: the Corporate Wallet SDK shall return an appropriate error to the MPA.
        i. MPA could display an empty dashboard or a relevant message to the end user.
    b. In case of timeout between MDES and Wallet Server, the Corporate Wallet SDK shall return an appropriate error “(for exact timeout strategy please refer MDES API Spec 1.1.3) to the MPA.
        i. MPA shall display an empty dashboard.

	c. In case of timeout between VTS and Wallet Server, the Corporate Wallet SDK shall return an appropriate error to the MPA. i. MPA shall display an empty dashboard. d. MPA should choose how to display the dashboard and notify end users something went wrong. e. If the card has been digitized: **i.** The Corporate Wallet SDK sets the first successfully digitized card as the default card. **NOTE:** • **The use case "Set Default Card" can be executed after this point to let the user selected the default payment card.** • **The default card is the first digitized card irrespective from the card network (Mastercard or Visa).** ii. The Corporate Wallet SDK returns the digitized card to the MPA. f. If card has not been digitized: The Corporate Wallet SDK returns an appropriate error to the MPA . 15. MPA displays an updated UI indicating whether the card has been successfully digitized or not with an option to "Retry". If the user opts to retry but the session has expired, MPA shall navigate to the Santander ID&V login screen given the ID&V token is a pre-requisite to digitize the cards. **NOTE: "Retry" is not considered for the Reference API and it is replaced by the "Add Additional Card(s)" Use Case.**
External Dependencies	None
Data Captured	Tokens
Notes	1. Process for storing Visa token in URPay: The CW-SDK sends the request to store provisioned token to URPay. URPay stores it and sends response with token key to the CW-SDK. The CW-SDK then sends request to construct provision ack to the URPay. The URPay processes the request and sends response with provision ack request to the CW-SDK. 2. For Reliability for Corporate Wallet SDK to Wallet Server communication: Please refer section "Corporate Wallet SDK to Wallet Server Communication". 3. For CW-SDK error codes, please refer to CW-SDK API
Business Consideration	None
Exceptional Flows	1. In case a timeout happens at the Corporate Wallet SDK, an error is returned. On subsequent launch, there are two possible scenarios: a. If add card request from the Corporate Wallet SDK did not reach the Wallet S appropriate error. MPA should display a relevant screen and/or an empty dashboard. User will have to add the card by selecting an option from the MPA. b. The request reaches the Wallet Server and it completes the digitization process for cards. In that case the Wallet Server shall send the request to

provision for Visa cards to VTS with VPAN Enrolment Id. On reception of the response the Wallet Server shall form a response with tokens for Mastercard cards and Visa cards and failure status for cards for which digitization was not successful. This response is sent to the Corporate Wallet SDK.

2. In case if there is a time out at Corporate Wallet SDK, the Corporate Wallet SDK shall return appropriate error messages to MPA. Please refer to CW-SDK API for exact error codes.
3. In the cases above, the MPA display an error message. When the user dismisses the message, the application exits.
4. If SafetyNet attestation is not successful, then depending upon the following scenarios, appropriate action is performed:
    a. **JWS signature verification failure:** CW-SDK will send an appropriate exception to MPA indicating that the device appears to be tampered and the MPA will exit. CWS will maintain a failure counter for each such occurrence. After 'n' such occurrences (*where 'n' is configured as "SafetyNet Counter" tenant configuration*), wallet instance will be terminated.
    b. Nonce verification failure: Appropriate exception is returned by CW-SDK to MPA.
    c. Technical Error / Unknown Error: Appropriate exception is returned by CW-SDK to MPA.
    d. JWS Payload Processing Error: For this failure scenario, several validations will be performed against the following parameters:
        i. CTS Profile
        ii. Basic Integrity
        iii. Package Name, Cert Digest, APK Digest

    Appropriate response will be generated / action will be performed based on validation results. Please refer section Safety Net Check for details on the validations performed.

    There will be 4 different actions (paths) as a result of validations performed on the above listed parameters. Please refer to the below table:

CTS Profile	Basic Integrity	PackageName, Cert Digest, Apk Digest	Action
False	True	Null	**Path 1**
False	True	Match	**Path 1**
False	False	Null	**Path 2**
False	False	Match	**Path 2**
False	False	Mismatch	**Path 3**
False	True	Mismatch	**Path 3**

True	True	Mismatch	**Path 3**
True	True	Null	**Path 4**
True	True	Match	**Path 4**

1. **Path 1**:
    a. CWS to deliver proper response to CW-SDK when receiving response corresponding to Path 1.
    b. CW-SDK will provide response to MPA.
    c. CWS resets the counter which keeps the count of occurrences of path 2.
    d. In parallel to step c, (Recommended) MPA shall prompt message to notify user that his device may be impacted by tampered device status and require user acceptance of the risk.
    e. (Recommended) If user accepts the risk then proceed further and maintain the acceptance to prevent repeated prompts in future. Else, exist the app.
    f. CW-SDK provides the acknowledgement of user acceptance of Risk (if provided by MPA or automatic acknowledgement response) to CWS.
    g. CWS resets counter of occurrences.

2. **Path 2:**
    a. When SafetyNet validation falls under condition described in Path 2
    b. CWS will maintain a Counter to Server to measure this failure case for “Active” wallets.
    c. CWS will Increment the counter on each occurrence and MPA will prompt a message to the user to notify that his device looks tampered and exit the app.
    d. If the Counter reaches n, terminate the wallet instance, n value will be defined in “SafetyNet Counter” tenant configuration.

3. **Path 3:**
    a. When values that coincide with what is detailed in chart for Path 3 wallet instance will be terminated.

4. **Path 4:**
    a. When SafetyNet validation falls under condition described in Path 4.
    b. CWS resets counter of occurrences
    c. MPA continues with BAU.

Customization	Addressed in the “Flow of Events”
Flow/Sequence Diagram	Please refer to the Sequence Diagrams document for a graphical representation of this flow

4.3.3.1 Set up Corporate Wallet SDK: Add Card(s): Digitize Mastercard Card

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. Wallet Server 5. MDES 6. CMS-D 7. Issuer 8. Santander Card Enabler System / Santander Token Manager
Channel	Corporate Wallet SDK
Pre-Conditions	1. User's device has data connectivity/internet connection. 2. Mastercard card is available for digitization.
Trigger Condition	Digitization request has been initiated by the Wallet Server.
Start Point	Same as above.
Post Conditions	The card gets successfully digitized
Flow of Events	1. The Wallet Server sends request to MDES to perform eligibility check. **Note:** The Digitize by List method contains two calls to MDES: a. Check eligibility. b. Digitize The Check Eligibility requests must be performed in a synchronous manner. The Digitize requests can be performed asynchronously. All calls should use the responseHost received from the initial Check Eligibility call. 2. CWS logs the perform eligibility check start event. 3. The MDES system performs eligibility check. If the check passes, then proceed to the step 5. 4. Else, MDES system sends error response to the Wallet Server. a. If the error is due to the card not being eligible, the Wallet Server notifies the Santander Card Enabler System with "CARD NOT ELIGIBLE" (DigitizationFailed Notification) b. If the error is due to the device not being eligible, the Wallet Server notifies the Santander Card Enabler System with "DEVICE NOT ELIGIBLE" (DigitizationFailed Notification) c. If the error is due to the device and the card not being eligible, the Wallet Server notifies the Santander Card Enabler System with "DEVICE AND CARD NOT ELIGIBLE" (DigitizationFailed Notification) d. The use case ends, please returns to the "Set up Corporate Wallet SDK: Add Card(s)" or "Set up Corporate Wallet SDK: Add Card List" flow. 5. The MDES system sends Eligibility receipt and T&C Id to the Wallet Server. 6. CWS logs the perform eligibility check end event. 7. Wallet server stores the T&C Id returned along with the TER response.

8. If the Wallet Server
    a. Receives the response and if it is success response then go to next step.
    b. Else, (i.e. in case of communication failure) then:
        i. The Wallet Server records error event for response, and
        ii. The use case ends, please returns to the “Set up Corporate Wallet SDK: Add Card(s)” or “Set up Corporate Wallet SDK: Add Card List” flow.
9. CWS compares the T&C ID received as part of check eligibility response.
    a. If the T&C ID is new
        i. The Wallet Server suppresses T&C approval by user.
        ii. CWS generates the T&CAcceptedTimestamp
        iii. CWS updates the T&CAccepted Timestamp at DB
        iv. Go to next step
    b. Else if T&C is same as previous
        i. The Wallet Server suppresses T&C approval by user
        ii. Go to next step
10. Wallet server sends request with Eligibility receipt, T&C Id, T&CAcceptedTimestamp, Registration Id and reason codes (for the selection of the reason codes see “Fraud Information and Reason Code Association (MDES)”) to digitize card to MDES system and logs the digitization start event
    a. If request is sent successfully, then go to next step.
    b. Else, retry once. If retry fails, then send an error message to Corporate Wallet SDK. The use case ends.
11. The MDES system sends token authorization request (TAR) to the Issuer.
12. The Issuer system processes the request response with following to the MDES system.
    a. Approved (Green Path)
    b. Additional Authorization Required (Yellow Path)
    c. Declined (Red Path)
13. The MDES system forwards the response to the Wallet Server with asset Id.
14. If the Wallet Server
    a. Receives the response and processes it:
        i. (Green Path) If it is success response. Execute the provisioning step 33 to 38, then go to step 16.
        ii. (Red Path) Else if it is an unsuccessful response, Wallet Server records failure event for response, prepares and send response to the Corporate Wallet SDK. Returns to “Set up Corporate Wallet SDK: Add Card(s)” or “Set up Corporate Wallet SDK: Add Card List”
        iii. (Yellow Path) Else if Additional Authentication Required, execute the provisioning step 34 to 39, and
            a. If called from “Set up Corporate Wallet SDK: Add Card List, then, return to “Set up Corporate Wallet SDK: Add Card List”
            b. Else (called from “Set up Corporate Wallet SDK: Add Card”) prepares the response to be sent to the Corporate Wallet SDK. Go to step 18.
    b. Else, (i.e. in case of communication failure) then record error event for response and prepare response to be sent to the Corporate Wallet SDK.

15. MDES:
    a. Creates token mapping
    b. Sends Tokenization Complete Notification (TCN) to the issuer and
    c. Sends Notify Token Update (active) to the Wallet Server, and returns to “Set up Corporate Wallet SDK: Add Card(s)” or “Set up Corporate Wallet SDK: Add Card List”

NOTE: Yellow Path: follow steps below in case of yellow path (OTP or Customer Service activation method). This was removed from scope but maintained here as a reference.

16. The Corporate Wallet SDK returns to the MPA Additional Authentication Required
17. MPA should let the user selects the activation method.
    a. If user selects OTP via SMS go to next step
    b. If user selects OTP via Email go to next step
    c. If user selects “Customer Service” go to step 28
18. The MPA sends Request_OTP via SMS/Email to the CW-SDK.
19. The CW-SDK sends Request_OTP via SMS/Email to the Wallet Server.
20. The Wallet Server sends Request Activation Code to MDES
21. MDES creates the Activation Code (OTP) and sends an Activation Code Notification (ACN) with the OTP to the Issuer.
22. The issuer sends the OTP via SMS/Email to the User.
23. User enters the OTP in the MPA.
24. MPA calls the Activate Card providing the OTP to the CW-SDK.
25. CW-SDK calls the Activate Card providing the OTP to the Wallet Server.
26. The Wallet Server calls the Activate card with the Activation Code (OTP) to MDES
27. MDES verifies the Activation Code. Got to step 31.
28. User contacts Customer Service
29. Customer Service authenticates the user.
30. Issuer sends activate token to MDES.
31. MDES:
    a. Creates token mapping
    b. Sends Tokenization Complete Notification (TCN) to the issuer and
    c. Sends Notify Token Update (active) to the Wallet Server, and returns to “Set up Corporate Wallet SDK: Add Card(s)” or “Set up Corporate Wallet SDK: Add Card List”

NOTE: Provisioning - Green Path or Yellow Path.

32. MDES sends request to Provision Card Profile with RNS_ID to the CMS-D.
    a. If the sending request fails then removes the digitized card, sends response to the Wallet Server and Wallet Server sends the response to Corporate Wallet SDK. Corporate Wallet SDK returns an appropriate error. At this point, MPA may display an error message. The use case ends.

       NOTE: provision from MDES to CMS-D is done in both green and yellow path. In case of yellow path, the token mapping is not created until activation.

	b. Else, go to next step. 33. The CMS-D sends a notification (with Registration code, Encrypted Sessions code) to Corporate Wallet SDK using RNS_ID via the MPA: a. If sending notification fails then CMS-D retries, if it still fails then CMS-D notifies MDES, the MDES removes the digitized card and sends notification to Wallet Server. The Wallet Server forwards the error response to the Corporate Wallet SDK. Corporate Wallet SDK notifies the MPA with an appropriate error response and the MPA displays error and navigates to card addition screen. b. Else, go to next step. 34. The Corporate Wallet SDK decrypts the session code using the Mobile Keys and sends a request to Provision the card profile to CMS-D system. 35. The CMS-D system sends Card Profile to the Corporate Wallet SDK. 36. The Corporate Wallet SDK notifies the provisioning result to the CMS-D system. 37. The CMS-D system a. Sends an acknowledgement to the Corporate Wallet SDK. b. Notifies provisioning result to the MDES system.
Notes	1. T&C acceptance timestamp is not related, and it cannot be correlated to any consumer acceptance, being the consumer acceptance done independently by the MPA and being the wallet Server not aware of such consumer acceptance from the MPA.
Exception flows	1. OTP is Invalid or Expired a. MDES verifies the OTP entered by the user b. MDES verifies the OTP entered by the user. If MDES finds that the OTP is invalid or expired, it will send INCORRECT_CODE or EXPIRED_CODE to CWS c. CWS would send the error code to CW-SDK d. CW-SDK in turn will send an appropriate error as per the information received from WS as part of activate API response to MPA e. (INFORMATIVE) – if OTP is expired, MPA will ask user to request a new OTP and then follow step 17 of the main use case. f. (INFORMATIVE) – if OTP is invalid, MPA will ask user to enter OTP once again. Maximum 3 retries are allowed, after which user will be requested to ask for a new OTP. 2. OTP has not been received by the user in step 22 of main use case a. If user has not received an OTP from the issuer, then MPA should have an option for the user to request OTP once again and then follow step 18 of main use case. 3. In case of timeout between WS and CES, WS will retry to send the notification to up to 3 times and will not pursue any steps to ensure guaranteed delivery of notification 4. In case the provisioning fails and there is a mismatch in the provisioning states at the CW-SDK, MDES or WS, following are the steps to be followed: a. [R 1.0] i. Based on MDES portal status, CSR can: 1. Delete the token and ask user to re-digitize 2. OR ask user to re-install the application. ii. If the card is not activated within a pre-determined time frame, 1. MPA asks CSR to delete the card

	2. MPA performs the digitize/digitize List UC b. [R 1.2, R2.1, R2.2, R2.2.1] i. CW-SDK sends provisioning failed notification to MPA ii. MPA gets the provisioning failed notification iii. MPA sends a delete token request to CW-SDK iv. If token deletion is successful 1. MPA performs L1 authentication 2. MPA performs the digitize/digitize List UC v. Else go to step 4b iii

4.3.3.2 Set up Corporate Wallet SDK: Add Card(s): Digitize Visa Card

Actor	1. User 2. Mobile Wallet 3. Wallet Server 4. VTS
Channel	Mobile Wallet
Pre-Conditions	1. User's device has data connectivity/internet connection. 2. Visa card is available for digitization.
Trigger Condition	Digitization request has been initiated by the Wallet Server.
Start Point	Same as above.
Post Conditions	The card gets successfully digitized
Flow of Events	1. The Wallet Server sends enrolPAN request to VTS. 2. CWS logs the digitization start event. 3. The VTS system performs enrol PAN. If the enrol PAN is successful, Wallet Server parses the response, then go to step 4. If the payment instrument is non tokenizable then wallet server sends notification to CW-SDK that card is not tokenizable. 4. The use case ends, please return to the "Set up Corporate Wallet SDK: Add Card(s)" or "Set up Corporate Wallet SDK: Add Card List" flow. 5. Wallet server stores the T&C ID returned along with the enrol PAN response. 6. If the Wallet Server a. Receives the response and if it is success response then go to next step. b. Else, (i.e. in case of communication failure) then: i. the Wallet Server records error event for response, and ii. The use case ends, please returns to the "Set up Corporate Wallet SDK: Add Card(s)" or "Set up Corporate Wallet SDK: Add Card List" flow. 7. WS compares the T&C ID received as part of enrol PAN response a. If the T&C ID is new

	i. The Wallet Server suppresses T&C approval by user. ii. WS generates the T&CAccepted Timestamp iii. WS updates the T&C AcceptedTimestamp in DB iv. Go to next step b. Else if T&C is same as previous i. The Wallet Server suppresses T&C approval by user ii. Go to next step 8. Wallet server sends to VTS the ProvisionToken request with all mandatory parameters (please refer VTS specification for the parameters) to provision the token to VTS system. a. If request is sent successfully, then go to next step. b. Else, retry once. If retry fails, then send an error message to Corporate Wallet SDK. The use case ends. 9. The VTS system sends token authorization request (TAR) to the Issuer. 10. The Issuer system processes the request response with following to the VTS system. a. Approved (Green Path) b. Declined (Red Path) **Note: Yellow Path for Visa is not supported and was removed from scope.** 11. The VTS system forwards the response to the Wallet Server 12. If the Wallet Server a. Receives the response and: i. (Green Path) If it is success response. go to step 29 ii. (Red Path) Else if it is an unsuccessful response, Wallet Server records failure event for response, prepares and send response to the Corporate Wallet SDK. Returns to “Set up Corporate Wallet SDK: Add Card(s)” or “Set up Corporate Wallet SDK: Add Card List” b. Else, (i.e. in case of communication failure) then record error event for response and prepare response to be sent to the Corporate Wallet SDK. 13. Wallet server informs VTS of successful provisioning by calling provisioning confirm API. a. returns to “Set up Corporate Wallet SDK: Add Card(s)” or b. returns to “Set up Corporate Wallet SDK: Add Cards List” 14. The Use Case ends.
External Dependencies	VTS
Notes	1. T&C acceptance timestamp is not related, and it cannot be correlated to any consumer acceptance being the consumer acceptance done independently by the MPA and being the wallet Server not aware of such consumer acceptance from the MPA.
Exceptional Flows	1) OTP is Invalid or Expired a. VTS verifies the OTP entered by the user b. If VTS finds that the OTP is Invalid or Expired, it would send the a code 401 notAllowed to WS c. WS would send the error code to CW-SDK d. CW-SDK in turn will send an appropriate error as per the information received from WS as part of activate API response to MPA

	e. If OTP is expired or Invalid, MPA will ask user to request to re-enter OTP and then follow step 14 of the main use case. f. In case if OTP retries are exceed as per the maximum attempts parameter, then for that token activate via OTP will be disabled as per Visa Specs. Hence MPA will need to ask user to choose other option except OTP to activate the token. (Please refer page 93 of Visa specs Version 3.6) 2) OTP has not received by the user in step 18 of main use case a. If user has not received an OTP from the issuer then MPA should have an option for the user to request OTP once again and then follow step 17 of main use case. 3) In case of timeout between WS and CES, WS will retry to send the notification to up to 3 times and will not pursue any steps to ensure guaranteed delivery of notification

4.3.4 Set up Corporate Wallet SDK: Add Cards List

This use case elicits the process of digitizing several cards in one call. It is assumed that digitization will follow green path. If issuer responds by yellow path for any card, at the end of the digitization process, MPA will request Cardholder to activate them. If issuer responds by red path, the corresponding card is not digitized, but digitization continues for the other cards.

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. Wallet Server 5. URPay 6. MDES 7. VTS
Channels	Corporate Wallet SDK
Pre-Conditions	1. Wallet status is active. 2. The user has authenticated against the Santander ID&V/SPS and the MPA has a valid Authorization Token. 3. User's device has data connectivity/internet connection.
Trigger Conditions	The Corporate Wallet SDK stores the Wallet status as "Active" and more than one card is selected/sent for digitization.
Start Point	The Corporate Wallet SDK has stored the Wallet status as "Active".
Post-Conditions	The card(s) are successfully digitized on the Corporate Wallet SDK.

Flow of Events	1. The MPA gets the eligible cards. 2. Then, MPA shows the card list to user and user selects more than one card for digitization. 3. The MPA sends to the Corporate Wallet SDK the cards to be digitized. 4. The CW-SDK forms the request for the selected cards to be sent with a. Cards details (FPAN, Expiration Date, Card Holder Name) b. Device Fingerprint c. Decisioning Data for the TAR message d. Authorization token 5. The Corporate Wallet SDK a. Collects the “Fraud Information”, see “Fraud Information and Reason Code Association” b. Sends digitization request with “Fraud Information” to the Wallet Server with the card(s) to be digitized, along with ID&V and Data token. 6. Additionally, the Corporate Wallet SDK initiates SafetyNet check asynchronously. If the SafetyNet check is successful, go to next step. If SafetyNet check is not successful, then refer the ‘Exceptional Flows’. 7. CWS checks if the ID&V token (L1 validation) is valid using bank deployed services. i. If the Token is invalid the WS will send error message to CW-SDK ‘TOKEN NOT AUTHORIZED’, use case ends. 8. Gets the User ID calling the Bank service GetLoggedUID passing the ID&V token 9. The WS checks if the ID&V token has the same user ID of the wallet record. If the user ID is different then i. WS will pass the error message to CW-SDK ii. CW-SDK in turn will pass this message to MPA, use case ends (not pursuing digitization for other cards). 10. CWS checks if the service is configured for L2 Authentication. If yes, WS checks if the data token has been received or not. 11. If no data token is received, it would throw an error to CW-SDK, CW-SDK will pass on the error to MPA, use case ends. 12. If data token is received, then go to next step. 13. Performs SHA2 on the operations data received from CW-SDK and passes the data token to SPS service for validation of the data token 14. Upon successful validation of the data token, SPS returns successful response to WS. 15. If the data token Is not valid, SPS will return error codes to WS 16. WS will pass the error message to CW-SDK. 17. CW-SDK in turn will pass this message to MPA, use case ends (not pursuing digitization for other cards). 18. Wallet server continues with the next step. 19. Performs further serialized process of digitization for each card included in the list: a. For digitization of Mastercard cards please refer section “Set up Corporate Wallet SDK: Add Cards: Digitize Mastercard Card". The Check Eligibility requests must be performed in a synchronous manner to avoid TSP meeting a lock condition on PAID for different PAN being digitized. The Digitize requests can be performed asynchronously. All calls should use the responseHost received from the initial Check Eligibility call.

b. For digitization of Visa card please refer section “Set up Corporate Wallet SDK: Add Cards: Digitize Visa Card".

20. On completion of digitization of the card the Wallet Server:
    a. Stores the updates.
    b. Prepares the response to confirm card addition request with:
        i. TUR(s) and token(s) of the successful digitized Mastercard card(s),
        ii. Tokens with corresponding PANenrolmentId for successful digitized Visa card(s).
        iii. Failure response for card that could not be digitized.
        iv. Card (masked FPAN with identifier for Visa/Mastercard), - response can be without any card in case no eligible card have been received
        v. Error/Failure event for these card(s) (Approved or Rejected)
        vi. Wallet State
    c. Pass to next card, go to step 9.
21. When all cards are digitized or one or more cards are found requiring yellow path, the Wallet Server sends the response to CW-SDK

22. The CW-SDK performs different steps (depending on the type of card) to activate each card.
    a. For Mastercard cards
        i. Stores TUR
        ii. Initiates the Initial Replenishment process
        iii. if not done before, Initiates TDS registration process in background for Mastercard ()
        iv. Store failure/error events for display after assets are fetched.
    b. For Visa cards
        i. Stores tokens. For details for storing Visa tokens please refer the “Notes” section.
        ii. Receives ack for storing token in URPay SDK and sends it to VTS via Wallet Server. The VTS sends response to the URPay 2.0 SDK (via Wallet Server).
        iii. Marks token as “Active”.
        iv. Store failure/error events for display after assets are fetched.
23. The Corporate Wallet SDK does the following:
    a. If cards belong to Yellow Path, CW-SDK returns to the MPA the list of cards with Additional Authentication Required as response
        i. For each card where Additional Authentication Required, the MPA starts the additional authentication:
            1. If Card is Mastercard,
                a. Refer Use Case - Set up Corporate Wallet SDK: Add Card(s): Digitize Mastercard Card, Step 17
            2. Else If Card is Visa,
                a. Refer Use Case – Set up Corporate Wallet SDK: Add Card(s): Digitize Visa Cards, Step 13

	b. Else, in case no cards from the list are eligible: Corporate Wallet SDK shall return an appropriate error to the MPA. i. The MPA may display an empty dashboard. **CardReferenceID must be generated by the MPA. It acts as a reference parameter. CardReferenceID must be unique for the same PAN across all devices (WalletID) of the same consumer (userID).** c. In case of timeout between MDES and Wallet Server, at any point of the Card List digitization, WS will mark the card as appropriate, notify CW-SDK and proceed with the next card. The Corporate Wallet SDK shall forward the response to MPA with the list of cards along with the failure reasons (if applicable) i. MPA may show the cards which have been successfully digitized d. In case of timeout between VTS and Wallet Server, at any point of the Card List digitization, WS will mark the card as appropriate, notify the CW-SDK and proceed with the next card. The Corporate Wallet SDK shall forward the response to MPA with the list of cards along with the failure reasons (if applicable). i. MPA may display cards which have been successfully digitized and differentiate them from those that were not. e. If the card has been digitized: i. CW-SDK sets the first successfully digitized card as the default card. **Note: the use case "Set Default Card" can be executed after this point to let the user selected the default payment card.** **Note: The default card is the first digitized card irrespective from the card schema.** ii. CW-SDK returns to the MPA the digitized cards f. If no card has been digitized: The Corporate Wallet SDK returns to the MPA an appropriate error 24. MPA may display an updated UI with different indicators indicating whether the cards have been successfully digitized or not with option to "Retry" the digitization of the whole list. If the user opts to retry, MPA will attempt, ensuring the session is still valid or redirecting the user to the Santander ID&V Login screen if not. **Note: There's no specific Retry API. A retry mechanism can be implemented following the "Add Additional Card(s)" Use Case.**
External Dependencies	None
Data Captured	Tokens
Notes	1. Process for storing Visa token in URPay: CW-SDK sends the request to store provisioned token to URPay. URPay stores it and sends response with token key to the CW-SDK. The CW-SDK then sends request to construct provision ack to URPay. URPay processes the request and sends response with provision ack request to the CW-SDK. 2. For Reliability purposes, from Corporate Wallet SDK to Wallet Server communication: Please refer section "Corporate Wallet SDK to Wallet Server Communication".
Business Consideration	None

Exceptional Flows

1. In case a time out happens at the Corporate Wallet SDK, an error is returned. On subsequent launch, there are two possible scenarios:
    a. if add card request from the Corporate Wallet SDK did not reach the Wallet Server, in that case the Corporate Wallet SDK returns an appropriate error.
    b. The request reaches the Wallet Server and it completes the digitization process for cards. In that case the Wallet Server shall send the request to provision for Visa cards to VTS with VPAN Enrolment Id. On receipt of the response the Wallet Server shall form a response with tokens for Mastercard cards and Visa cards and failure status for cards for which digitization was not successful. This response is sent to the CW-SDK.

2. In case if there is a time out at Corporate Wallet SDK, the Corporate Wallet SDK shall return appropriate error messages to MPA. Please refer to CW-SDK API for exact error codes.
3. In the cases above, the MPA may display an error message. When the user dismisses the message, the application may exit.
4. If SafetyNet attestation is not successful, then depending upon the following scenarios, appropriate action is performed:
    a. **JWS signature verification failure:** CW-SDK will send an appropriate exception to MPA indicating that the device appears to be tampered and the MPA will exit. CWS will maintain a failure counter for each such occurrence. After 'n' such occurrences (*where 'n' is configured as "SafetyNet Counter" tenant configuration*), wallet instance will be terminated.
    b. **Nonce verification failure:** Appropriate exception is returned by CW-SDK to MPA.
    c. **Technical Error / Unknown Error:** Appropriate exception is returned by CW-SDK to MPA.
    d. **JWS Payload Processing Error:** For this failure scenario, several validations will be performed against the following parameters:
        i. CTS Profile
        ii. Basic Integrity
        iii. Package Name, Cert Digest, APK Digest

    Appropriate response will be generated / action will be performed based on validation results. Please refer section Safety Net Check for details on the validations performed.

    There will be four different actions (paths) as a result of validations performed on the above listed parameters. Please refer the below table:

CTS Profile	Basic Integrity	PackageName, Cert Digest, Apk Digest	Action
False	True	Null	**Path 1**
False	True	Match	**Path 1**
False	False	Null	**Path 2**

False	False	Match	**Path 2**
False	False	Mismatch	**Path 3**
False	True	Mismatch	**Path 3**
True	True	Mismatch	**Path 3**
True	True	Null	**Path 4**
True	True	Match	**Path 4**

1. **Path 1**:
    a. CWS to deliver proper response to CW-SDK when receiving response corresponding to Path 1.
    b. CW-SDK will provide response to MPA.
    c. CWS resets the counter which keeps the count of occurrences of path 2.
    d. In parallel to step c, (Recommended) MPA shall prompt message to notify user that his device may be impacted by tampered device status and require user acceptance of the risk.
    e. (Recommended) If user accepts the risk then proceed further and maintain the acceptance to prevent repeated prompts in future. Else, exist the app.
    f. CW-SDK provides the acknowledgement of user acceptance of Risk (if provided by MPA or automatic acknowledgement response) to CWS.
    g. CWS resets counter of occurrences.

2. **Path 2:**
    a. When SafetyNet validation falls under condition described in Path 2
    b. CWS will maintain a Counter to Server to measure this failure case for “Active” wallets.
    c. CWS will Increment the counter on each occurrence and MPA will prompt a message to the user to notify that his device looks tampered and exit the app.
    d. If the Counter reaches n, terminate the wallet instance, n value will be defined in “SafetyNet Counter” tenant configuration.

3. **Path 3:**
    a. When values that coincide with what is detailed in chart for Path 3 wallet instance will be terminated.

4. **Path 4:**
    a. When SafetyNet validation falls under condition described in Path 4.
    b. CWS resets counter of occurrences
    c. MPA continues with BAU.

Customization	Addressed in the “Flow of Events”
Flow/Sequence Diagram	The card(s) are successfully digitized on the Corporate Wallet SDK.

4.4 Upgrade Notification

These uses cases describe the process to manage the scenario in which a newer version of the MPA is available and user is required to upgrade the MPA to be able to access the MPA.

4.4.1 Upgrade Notification – During Launch

This use case describes the process wherein new version of MPA availability is discovered when the MPA is launched and it is downloaded by the user.

Actor	1. User 2. MPA (MPA and Corporate Wallet SDK) 3. Wallet Server 4. Google Play
Channels	Mobile Wallet
Pre-Conditions	1. New MPA version availability information is available in Wallet Server 2. New MPA version is available for download. 3. User’s device has data connectivity/internet connection.
Trigger Conditions	1. During any communication synch service call with the Wallet Server, the minimum required MPA version is discovered and it is communicated back to the Corporate Wallet SDK
Post-Conditions	1. The user downloads the new version of MPA.
Start Point	The user launches the MPA Or performs double tap for payment at POS, thereby invoking “Launch Wallet”
Flow of Events	1. The Corporate Wallet SDK returns to the MPA the minimum required version of MPA (see “Launch MPA” use case) from the wallet server 2. MPA checks if the user has already upgraded to the latest version or not, if the user is on the latest version, use case ends, if not then follow step 3 3. The MPA displays a message regarding availability of new MPA version. Upon pressing the button, user is taken to the Google Play Store. This is a responsibility of MPA. 4. If the user a) Initiates the download process, then go to next step. b) Else, (i.e. if user clicks “Back”) the MPA exits. 5. If internet is available a) Then go to next step, b) Else, display an appropriate error message. 6. The user is redirected to the Google play for downloading the app.

	7. Once the download process is complete, the user launches the MPA.
External Dependencies	Santander Wallet Server Admin must update on a per-tenant basis the latest version of available MPA.
Data Captured	None
Notes	Upgrade Notification does not need re-provisioning or SUKs / LUK replenishment
Business Consideration	None
Exceptional Flows	Addressed in “Flow of Events”
Customization	Addressed in “Flow of Events”

4.4.2 Upgrade Notification – During Communication with Wallet Server

This use case describes the process wherein new MPA version is discovered when the Corporate Wallet SDK communicates with the Wallet Server and the new MPA version is downloaded by the user.

Actor	1. User 2. MPA (MPA and Corporate Wallet SDK 3. Wallet Server 4. Google Play
Channels	Mobile Wallet
Pre-Conditions	1. New MPA version availability information is available with the Wallet Server. 2. New MPA version is available for download. 3. User’s device has data connectivity/Internet connection.
Trigger Conditions	Sync service call with the Wallet Server (for example occurs at MPA launch)
Post-Conditions	The user downloads the new version of MPA.
Start Point	Any synch service call with the Wallet Server Or user launches the MPA (as it triggers synch service call too) Or user performs double tap for payment at POS thereby invoking “Launch Wallet”
Flow of Events	1. Minimum required version of MPA is discovered by the Corporate Wallet SDK and provided to Wallet Server 2. Wallet server will check with the version received from CW-SDK and version available with the property file from admin portal. If there is a new version available, then the steps are similar to that of “Upgrade Notification – During Launch” flow 3. Corporate Wallet SDK communicates the minimum version back to the MPA and provides adequate notification as part of the response in synch service call. 4. MPA may check if the user has already upgraded to the latest version or not, if the user is on the latest version, use case ends, if not then follow step 5 5. The MPA displays a message regarding availability of new MPA version. 6. If the user a) Initiates the download process, then go to next step. b) Else, (i.e. if user clicks “Back”) the MPA exits. 7. If internet is available

	a) Then go to next step, b) Else, display an appropriate error message. 8. The user is redirected to the Google play for downloading the app. 9. Once the download process is complete, the user launches the MPA.
External Dependencies	Santander Wallet Server Admin must update on a per-tenant basis the minimum upgrade version of available MPA.
Data Captured	None
Notes	Upgrade Notification does not need re-provisioning or SUKs/LUK replenishment
Business Consideration	None
Exceptional Flows	Addressed in “Flow of Events”
Customization	Addressed in “Flow of Events”
Flow/Sequence Diagram	

4.5 View Mobile Wallet Dashboard

This use case describes the process of viewing the MPA dashboard following the Launch Wallet use case. The MPA dashboard provides a snapshot of number of cards added in the Wallet and the default card. User can initiate a transaction from the MPA dashboard either by selecting a default card or selecting any other card in the MPA.

Actor	1. User 2. MPA 3. Corporate Wallet SDK
Channels	MPA
Pre-Conditions	1. User has launched the MPA. 2. Wallet state is “Active”. 3. The user has added/tokenized at least one card.
Trigger Conditions	1. User wants to view the dashboard.
Post-Conditions	1. User successfully views the MPA dashboard.
Flow of Events	1. The MPA calls getLocalCardList API from the Corporate Wallet SDK to get the list of cards. 2. The Corporate Wallet SDK returns the list of cards that are digitized with the following details from local Database: a. Card details (including masked FPAN, assets) b. Default card c. Card status (digitized, suspended, unsuspended…) d. Credit/Debit 3. MPA displays the dashboard with following: a. Default card b. Other payment cards (starting with the first card successfully digitized)

	4. User successfully views the dashboard.
External Dependencies	None
Data Captured	None
Notes	getLocalCardList API only returns the information related to currently digitized cards or only the cards that were recovered post auto recovery. Non-digitized or deleted cards are not returned by the getLocalCardList API.
Business Consideration	None
Exceptional Flows	None.
Customization	Wallet dashboard design.
Flow/Sequence Diagram	

4.6 Set up MPA after Delete App/Factory Reset

This use case describes the process of setting up the Mobile Wallet following one of the events:

1. Delete App
2. Device factory reset

If the device fingerprint is already associated to a wallet, the Wallet Server terminates this wallet before proceeding with the use case.

Actor	1. User 2. MPA 3. Corporate Wallet SDK 4. Wallet Server 5. CMS-D
Channels	1. MPA
Pre-Conditions	1. MPA (and Corporate Wallet SDK instance) on the phone has been deleted or is no longer in use.
Trigger Conditions	1. User deletes the MPA, or User does the factory reset on the phone that deletes the MPA. 2. User re-installs the MPA
Post-Conditions	The MPA is successfully set up.
Flow of Events	1. User goes through installing, launching and setting up the MPA again. 2. The user continues setting up the Wallet.
External Dependencies	None
Data Captured	Wallet terminated status when device fingerprint is already associated

Notes	With the introduction of Android 10, which is supported in all Wallet versions, it's no longer possible to maintain the device fingerprint across application installs, so it's not possible to match the device with previous wallets accounts, reason why a new one gets created every time the application is installed or the application data is cleared.
Exceptional lows	None
Customization	None
Flow/Sequence Diagram	

4.7 Terminate/Deactivate Wallet: by MPA

This use case explains the scenario wherein the User opts to terminate/de-activate the Wallet instance (for this device) from the Corporate Wallet SDK.

Actor	1. MPA 2. Corporate Wallet SDK 3. Wallet Server 4. MDES (including CMS-D and TDS) 5. VTS
Channels	Corporate Wallet SDK
Pre-Conditions	1. The MPA has launched the Corporate Wallet SDK. 2. The device has internet connectivity.
Trigger Conditions	1. The MPA requests to terminate/deactivate the Wallet for this device.
Post-Conditions	1. The Wallet Server terminates the Wallet instance (for this device) on the Wallet Server. 2. The Corporate Wallet SDK removes all the existing data. The user can start a fresh registration from the application.
Start Point	The user selects "Deactivate/Terminate" Wallet from the menu.
Flow of Events	1. MPA displays confirmation screen. 2. If the user confirms; the MPA prompts user to: a. Authenticate using CDCVM (Consumer Device Cardholder Verification Method) If the user is successfully authenticated, then proceed to the next step. 3. The MPA requests to Terminate/Deactivate the wallet to the Corporate Wallet SDK. The Corporate Wallet SDK: a) Performs TDS un-registration (applicable to Mastercard cards). If failure/error is encountered, then the step is skipped. b) Deletes the Wallet data. c) Sends request to the Wallet Server to terminate the Wallet. 4. The Wallet Server a) Initiates the Wallet termination process

	b) For Mastercard cards performs the following steps. a. Sends request to delete the tokens to MDES. The MDES deletes the token and sends response to the Wallet Server. **NOTE: When MDES delete the token, the issuer is notified via Notify Token Update.** c) For Visa cards performs the following steps a. The Wallet Server sends Token Life-Cycle Management—Delete Token request to delete token to VTS. The VTS removes the token and sends response to the Wallet Server. b. The Wallet Server repeats the process for all Visa cards. d) Marks the Wallet Instance as “Terminated”.
External Dependencies	1. CMS-D 2. MDES 3. VTS
Data Captured	Wallet instance (i.e. wallet for this device) terminated status
Notes	
Business Consideration	
Exceptional Flows	1. In case of failure from CW-SDK while deleting wallet data or any other reliability scenario, a retry strategy should be implemented. The recommendation is to get aligned with MDES retry strategy as per MDES API Spec 1.1.3.
Customization	Addressed in “Flow of Events”.
Flow/Sequence Diagram	

5 Card Services

The use cases below highlight all scenarios needed to enable digitized cards in the Corporate Wallet SDK.

5.1 Add Additional Card(s)

The use case for adding additional card(s) is composed of several main steps:

- ID&V authentication and/or SPS authentication to get the ID&V and data token.
- Selection by the MPA of the cards to be digitized or Digital by Default.
- Card digitization done by the Wallet Server.

The following table depicts the process at high level that needs to be performed for adding additional card(s).

Event	Yes/ No	Steps/processes to be performed
Mastercard card(s) added previously	Yes	• Digitize the card(s).
	No	• Register the Mobile Application with the CMS-D. • Perform URPay initiation.
Visa card(s) added previously	Yes	• Digitize the card(s).
	No	• Enrol device with the VTS. • Perform URPay initiation.

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. Wallet Server 5. CMS-D 6. VTS 7. Santander ID&V/SPS
Channels	Corporate Wallet SDK
Pre-Conditions	1. User has launched the MPA and User is on the MPA dashboard. 2. The user has authenticated against the Santander ID&V/SPS and the MPA has valid ID&V token and/or data token. 3. The Wallet Server activates the Wallet and the Corporate Wallet SDK stores the Wallet status as “Active”. 4. User’s device has data connectivity.
Trigger Conditions	1. User wants to add card(s).
Start Point	User selects the option to update the list of cards.
Post-Conditions	The Wallet Server successfully digitizes the selected cards.
Flow of events	See “Set up Corporate Wallet SDK – Add Card(s)”
External Dependencies	None

Data Captured	
Notes	None
Business Consideration	None
Exceptional Flows	In case if there is a time out at Corporate Wallet SDK, the Corporate Wallet SDK shall return specific error codes as defined in CW-SDK API Document.
Open Issue	None
Flow/Sequence Diagram	

5.2 Remove/Delete Card

5.2.1 Delete Card: By User

This use case is only available if the MPA is NOT implementing “Digital By Default” but “User Card Selection”.

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. Wallet Server 5. MDES 6. CMS-D (For Mastercard Cards only) 7. VTS
Channels	MPA
Pre-Conditions	1. User has successfully set up the MPA and has at least one card. 2. User has launched the MPA and User is on the MPA dashboard. 3. User’s device has data connectivity. 4. MPA is not implementing “Digital by Default”
Trigger Conditions	User selects the option to remove the card in the MPA.
Post-Conditions	Card is deleted from the Corporate Wallet SDK.
Start Point	Card settings screen
Flow of Events	1. User selects the option to remove card in the MPA. 2. MPA displays a confirmation screen. 3. If User confirms, then proceed to the next step. 4. MPA requests the list of cards to the Corporate Wallet SDK. 5. The Corporate Wallet SDK responds with the list of cards containing: a. Card details b. If the card is the default one c. The status of the card

6. MPA checks if there are more than 1 active card(s) present.
    a) If yes, checks if the card to be deleted is default.
        a. If yes, then the MPA displays a message indicating that user is trying to delete the default card.
    b) Else, displays a message that after deleting the card user will not be able to perform payment transaction from the wallet.
7. User reads the message and proceeds.
8. MPA requests the Corporate Wallet SDK to delete the card.
9. The Corporate Wallet SDK checks if Internet is available
    a) If yes, then go to next step.
    b) Performs TDS de-registration for that specific token using MDES API.
    c) Else, returns an appropriate error

    MPA displays an appropriate error. On dismissing the error, the application navigates back to the card settings screen.
10. There shall be different set of steps performed depending on the type of card.
    a) For Mastercard card following steps are performed
        a. The Corporate Wallet SDK sends request to delete the Card to the Wallet Server.
        b. The Wallet Server receives the request
        c. WS logs the delete card start event
        d. If the number of cards (tokens) to be deleted is less than or equal to 10:
            - CWS sends the delete card (token) request to MDES.

        Else, if number of cards (tokens) to be deleted is greater than 10, for every batch of 10 cards (tokens),
            - CWS sends the delete card (token) request to MDES.
        e. MDES deletes the token mapping and sends response back to the Wallet Server.
        f. WS logs the delete card end event.
        g. The Wallet Server forwards the response to the Corporate Wallet SDK.
        h. The Corporate Wallet SDK receives response. Proceed to next step.
    b) For Visa card following steps are performed
        a. The Corporate Wallet SDK sends request to delete the Card to the Wallet Server.
        b. The Wallet Server receives the request, logs the delete card start event and sends “Token Life-Cycle Management—Delete Token” request to VTS.
        c. The VTS deletes the token and sends response back to the Wallet Server.
        d. WS logs the delete card end event.
        e. The Wallet Server forwards the response to the Corporate Wallet SDK.
11. The Corporate Wallet SDK:
        a. Deletes all credentials
        b. Returns successful response
12. MPA should check if more than 1 active card(s) is/are present.
    a) If so, MPA checks if default card has been deleted.

	a. In case the default card is not deleted then displays a message indicating the card (ending with last 4 digits) has been deleted and screen navigates to dashboard screen. b. Else, it will display a message indicating the default card (ending with last 4 digits) has been deleted The MPA sets the new Default Card using the “Set Default Card” flow. Go to next step. Assumption here is that the setting of default card is managed by MPA. b) Else, displays message indicating the card (ending with last 4 digits) has been deleted and application navigates to dashboard screen. 13. The user proceeds. 14. The MPA updates the dashboard.
External Dependencies	1. MDES 2. CMS-D 3. VTS
Data Captured	None
Notes	In the messages displayed to the user, the last 4 digits of D-PAN should be shown For Visa, it is required to provide a reason code to VTS. As per Visa specs, the Wallet Server will always provide “CUSTOMER_CONFIRMED” For Mastercard, a limit of 10 TURs has been placed on every batch call being made for card deletion
Business Consideration	None
Exceptional Flows	1. During the deletion of Mastercard cards, if timeout happens at Corporate Wallet SDK, then an appropriate error is returned. a. MPA shall display an appropriate error message. On dismissing the error, the MPA shall navigate back to the card screen. There shall be a visual indicator indicating that card deletion is in progress. On subsequent launch, the “Get Payment Card Update” flow will be invoked. As a result of which there are two scenarios possible 2. Processing is done by the Wallet Server but due to communication failure the response is not received by the Corporate Wallet SDK: In this case, as a result of the sync process, the card shall be removed from the Wallet. 3. In case of communication error, timeout or connection failure between MDES and WS, in the event following deletion, resulting in the cards being deleted at MDES but not at CW-SDK and MPA, the following needs to be done: a. MDES will automatically retry 3 times with up to a 5-second wait between each try. If the call has not succeeded after the initial retries, MDES will attempt a second round of 3 retries with increasing time intervals between each retry. Between attempts the system will wait 15 minutes, 30 minutes, and then 2 hours. In the case of a 503, the Retry-After header will be respected if present and will count as a retry". 4. In case of communication error, timeout, or connection failure between VTS and WS there will be retries with exponential backoff strategy where retry count and delay are configurable:

	a. Default retry count is 3. County can configure using vts.connection.retry.count in system level config key. b. Delay for next retry is extracted from exception info from VTS. If it does not exist, immediate retry will be triggered.
Customization	None
Flow/Sequence Diagram	

5.2.2 Delete Card: By CSR

Actor	1. User 2. CSR 3. Corporate Wallet SDK 4. Wallet Server 5. MDES 6. CMS-D (For Mastercard cards only) 7. VTS 8. Santander Card Enabler System / Santander Token Manager
Channels	Corporate Wallet SDK
Pre-Conditions	1. User has successfully set up the Wallet. 2. User's device has internet connectivity.
Trigger Conditions	The user calls up CSR to initiate the Card deletion process.
Post-Conditions	Card is deleted/removed from the Corporate Wallet SDK.
Start Point	CSR initiates the card deletion process on VTS/MDES portal.
Flow of Events	1. User calls Santander CSR to delete the payment card. 2. For Mastercard cards, the Santander CSR logs into the Santander Card Enabler System (CES) to delete the card calling the MDES Customer Service APIs. 3. For Visa cards, the Santander CSR logs into the Santander Card Enabler System (CES) / Santander Token Manager (STM) to delete the card calling the VTS Customer Service APIs. MDES/VTS deletes the card token mapping and notifies the Wallet server. 4. The Wallet Server: a) Deletes the token. b) Updates the Corporate Wallet SDK by sending a notification using the RNS_ID 5. The Corporate Wallet SDK receives the notification through the MPA a) In case of Mastercard cards, i. The Corporate Wallet SDK deletes all credentials ii. Performs TDS de-registration using MDES API call for that specific token. b) In case of Visa cards, the Corporate Wallet SDK deletes all credentials. 6. The Corporate Wallet SDK checks if more than 1 active card(s) is/are present.

	a) If yes, then go to next step. b) Else, the Corporate Wallet SDK returns successful response to the MPA. MPA displays notification in the notification tray indicating the card deletion. Go to step 10. 7. MPA sets the default card using the “Set Default Card” flow. 8. Optionally, MPA may display a notification in the notification tray indicating that card ending with last 4 digits has been deleted and another card has been set as default card. 9. MPA displays updated dashboard.
External Dependencies	None
Data Captured	None
Notes	
Business Consideration	None
Exceptional Flows	1. In case of communication error, timeout or connection failure between MDES and WS, in the event following Deletion, resulting in the cards being deleted at MDES but not at CW-SDK and MPA, the following needs to be done: o MDES will automatically retry 3 times with up to a 5-second wait between each attempt. If the call has not succeeded after the initial retries, MDES will attempt a second round of 3 retries with increasing time intervals between each retry. Between attempts the system will wait 15 minutes, 30 minutes, and then 2 hours. In the case of a 503, the Retry-After header will be respected if present and will count as a retry" 2. In case of communication error, timeout, or connection failure between VTS and WS there will be retries with exponential backoff strategy where retry count and delay are configurable: o Default retry count is 3. County can configure using vts.connection.retry.count in system level config key. o Delay for next retry is extracted from exception info from VTS. If it does not exist, immediate retry will be triggered.
Customization	None
Flow/Sequence Diagram	

5.3 Set Default Card

This use case allows the MPA to set the chosen card as default/preferred. The default card selection will be used during NFC payment.

Actor	1. MPA 2. Corporate Wallet SDK
Channels	Corporate Wallet SDK

Pre-Conditions	1. MPA has an active Corporate Wallet SDK. 2. MPA has launched the Corporate Wallet SDK.
Trigger Conditions	1. User selects default option for a card of choice. 2. Or MPA decides to establish the card as default
Post-Conditions	The default card is set.
Start Point	Dashboard
Flow of Events	1. The MPA request to set the card as default. 2. The Corporate Wallet SDK sets the selected card as default and clears the existing card from default. 3. The use case ends.
External Dependencies	None
Data Captured	Preferred Card selection.
Notes	1. The default card shall always be the first card on the dashboard. 2. The default card is set on the wallet app instance. Each *app instance* will have a separate default card.
Business Consideration	None
Exceptional Flows	None
Customization	None
Flow/Sequence Diagram	

5.4 Suspend Card

5.4.1 Suspend Card – Santander Initiated

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. CSR 5. MDES 6. VTS 7. Wallet Server
Channels	Corporate Wallet SDK
Pre-Conditions	1. MPA has successfully set up the Corporate Wallet SDK. 2. The card to be suspended is active. 3. Wallet Server has NOT initiated a Suspend Token Request for that particular token
Trigger Conditions	1. Customer care suspends card. 2. User calls CSR to request card suspension.

Post-Conditions	1. The card is suspended in the Corporate Wallet SDK.
Flow of Events	1. The user calls CSR to suspend the card. 2. The CSR authenticates the user and logs into the Santander CES/STM to suspend the card calling the MDES/VTS Customer Service APIs. 3. The MDES/VTS suspends the card token and sends the confirmation to the Wallet Server. 4. For Mastercard cards, a. The Wallet Server updates the card status to “Suspended”. 5. For Visa cards, a. Wallet Server verifies whether it had initiated a Suspend Token request for that particular token & Wallet Server has received a response from VTS i. WS marks card as Suspended by User b. If Wallet Server had initiated a Suspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Suspended by User c. If Wallet Server had NOT initiated a Suspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Suspended by Issuer 6. Wallet Server sends a notification to the Santander CES/STM (suspend notification API). 7. Wallet Server sends a notification to the MPA using RNS_ID. 8. MPA, in turn, sends the notification details to CW-SDK. Please refer Push Notification Handling in the document. 9. The Corporate Wallet SDK updates the card status and marks card as “Suspended”. 10. The Corporate Wallet SDK checks if 1 or more active card(s) is/are present. a. If yes, then go to next step. b. Else, returns successful response to the MPA. 11. MPA may display a notification if the notification tray indicating card suspension event. 12. MPA selects another default card using the “Set Default Card” flow. 13. A notification is displayed in the notification tray indicating the default card suspension event and another card has been set as default card. 14. The UI for suspended card is updated.
External Dependencies	1. MDES 2. VTS
Data Captured	Card status
Business Consideration	None
Exceptional Flows	1. In case of communication error, timeout or connection failure between MDES and WS, in the event following Suspension, resulting in the cards being suspended at MDES but not at CW-SDK and MPA, the following needs to be done: o MDES will automatically retry 3 times with up to a 5-second wait between each attempt. If the call has not succeeded after the initial retries, MDES will attempt a second round of 3 retries with increasing time intervals between each retry. Between attempts the system will wait 15 minutes, 30 minutes, and then 2 hours. In the case of a 503, the Retry-After header will be respected if present and will count as a retry.

	2. In case of communication error, timeout, or connection failure between VTS and WS there will be retries with exponential backoff strategy where retry count and delay are configurable: o Default retry count is 3. County can configure using vts.connection.retry.count in system level config key. o Delay for next retry is extracted from exception info from VTS. If it does not exist, immediate retry will be triggered.

5.4.2 Unsuspend Card – Santander Initiated

Actor	1. User 2. CSR 3. MDES 4. VTS 5. Wallet Server
Channels	Corporate Wallet SDK
Pre-Conditions	1. MPA has payment card(s) in the Corporate Wallet SDK that are suspended. 2. User's device has internet/data connectivity. 3. Wallet Server had NOT initiated a Resume Token Request for that particular token.
Trigger Conditions	Customer care initiates unsuspend.
Post-Conditions	Card is unsuspended.
Flow of Events	1. User calls CSR to unsuspend the card. 2. The CSR authenticates the user and logs into the Santander CES/STM to unsuspend the card, calling the MDES/VTS Customer Service APIs. 3. The MDES/VTS unsuspends the card token and sends the information to the Wallet Server. 4. For Mastercard cards, a. The Wallet Server updates the cards status as "Active". 5. For Visa cards, a. If Wallet server had initiated a Unsuspend Token request for that particular token AND WS has received a successful response from VTS i. WS marks card as Suspended by User b. If Wallet server had initiated a Unsuspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Unsuspended by User c. If Wallet server had NOT initiated a Un suspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Unsuspended by issuer 6. Wallet Server sends notification to the Corporate Wallet SDK using RNS_ID via MPA. 7. The Corporate Wallet SDK updates the card status and marks card and token as active and returns successful response to the MPA.

	MPA may display a notification. When user navigates to the dashboard/clicks on the notification, the UI is updated.
External Dependencies	1. MDES 2. VTS
Data Captured	Card status
Business Consideration	None
Exceptional Flows	1. If data connectivity is not available in event 4c, Corporate Wallet SDK status will be updated when Wallet resumes the data connectivity. 2. In case of communication error, timeout or connection failure between MDES and WS, in the event following Un-Suspension, resulting in the cards being Un-suspended at MDES but not at CW-SDK and MPA, the following needs to be done: a. MDES will automatically retry 3 times with up to a 5-second wait between each attempt. If the call has not succeeded after the initial retries, MDES will attempt a second round of 3 retries with increasing time intervals between each retry. Between attempts the system will wait 15 minutes, 30 minutes, and then 2 hours. In the case of a 503, the Retry-After header will be respected if present and will count as a retry. 3. In case of communication error, timeout, or connection failure between VTS and WS there will be retries with exponential backoff strategy where retry count and delay are configurable: a. Default retry count is 3. County can configure using vts.connection.retry.count in system level config key. b. Delay for next retry is extracted from exception info from VTS. If it does not exist, immediate retry will be triggered.

5.4.3 Suspend Card – User Initiated

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. MDES 5. VTS 6. Wallet Server
Channels	MPA
Pre-Conditions	1. User has successfully set up the MPA and has at least one card. 2. User has launched the MPA and User is on the MPA dashboard. 3. User's device has data connectivity. 4. The card to be suspended is active. 5. MPA is not implementing "Digital by Default".

Trigger Conditions	1. User selects the option to suspend card via MPA
Post-Conditions	1. The card is suspended in the Corporate Wallet SDK.
Flow of Events	1. MPA requests the list of cards from the Corporate Wallet SDK. 2. The Corporate Wallet SDK responds with the list of cards containing: a. Card details. b. If the card is the default one. c. The status of the card. 3. MPA checks if there are more than 1 active card(s) present. a) If yes, checks if the card to be suspended is default. a. If yes, then the MPA displays a message indicating that user is trying to suspend the default card b. Else, go to step no. 5. b) Else, MPA may display an informative message indicating that after suspending the card user will not be able to perform payment transaction from the wallet. 4. User proceeds. 5. MPA requests to suspend the card to the Corporate Wallet SDK. 6. The Corporate Wallet SDK checks if internet is available. a) If yes, then go to next step. b) Else, returns appropriate error MPA may displays an appropriate error. On dismissing the dialog, the application navigates back to the card settings screen. 7. The Corporate Wallet SDK sends request to suspend the Card to the Wallet Server. 8. WS receives the request and logs the Suspend Card start event 9. For Visa Cards, the Wallet Server receives the request and sends it to VTS. 10. For Mastercard cards, WS sends the suspend card request to MDES 11. MDES/VTS suspends the card token mapping and sends response back to the Wallet Server. 12. For Mastercard card, a. The Wallet Server updates card status to Suspended 13. For Visa cards, a. If Wallet server had initiated a Suspend Token request for that particular token AND WS has received a successful response from VTS i. WS marks card as Suspended by User b. If Wallet server had initiated a Suspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Suspended by User c. If Wallet server had NOT initiated a Suspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Suspended by issuer 14. WS logs the Suspend Card event. 15. The Wallet Server forwards the response to the Corporate Wallet SDK. 16. The Corporate Wallet SDK a) Receives response, and

	b) Updates the card status and marks card as “Suspended”. 17. The Corporate Wallet SDK checks if 1 or more active card(s) is/are present. a) If yes, then go to next step. b) Else, returns successful response to the MPA. MPA displays notification in the notification tray indicating the card suspension event. 18. MPA selects another default card using the “Set Default Card” flow. 19. Customer is notified indicating of the card suspension event and that another card has been set as default card. 20. MPA updates the UI
External Dependencies	1. MDES 2. VTS
Data Captured	Card status
Notes	For Visa, it is required to provide a reason code to VTS. As per Visa specs, the Wallet Server will always provide “CUSTOMER_CONFIRMED”.
Business Consideration	None
Exceptional Flows	1. In case of communication error, timeout or connection failure between MDES and WS, in the event following Suspension, resulting in the cards being suspended at MDES but not at CW-SDK and MPA, the following needs to be done: o MDES will automatically retry 3 times with up to a 5-second wait between each attempt. If the call has not succeeded after the initial retries, MDES will attempt a second round of 3 retries with increasing time intervals between each retry. Between attempts the system will wait 15 minutes, 30 minutes, and then 2 hours. In the case of a 503, the Retry-After header will be respected if present and will count as a retry. 2. In case of communication error, timeout, or connection failure between VTS and WS there will be retries with exponential backoff strategy where retry count and delay are configurable: o Default retry count is 3. County can configure using vts.connection.retry.count in system level config key. o Delay for next retry is extracted from exception info from VTS. If it does not exist, immediate retry will be triggered.

5.4.4 Unsuspend Card – User Initiated

Actor	1. User 2. CSR 3. MDES 4. VTS 5. Wallet Server
Channels	Corporate Wallet SDK

Pre-Conditions	1. MPA has payment card(s) in the Corporate Wallet SDK that are suspended. 2. User has launched the MPA and User is on the MPA dashboard. 3. User's device has Internet/data connectivity.
Trigger Conditions	User selects the option to unsuspend a card via MPA.
Post-Conditions	Card is unsuspended.
Flow of Events	1. User selects option to unsuspend a card in the MPA. 2. If user confirms, either a. User authenticates using CDCVM If the user is successfully authenticated, then proceed to the next step. 3. The MPA requests to unsuspend the card to the Corporate Wallet SDK. 4. The Corporate Wallet SDK checks if internet is available a. If yes, Go to next step. b. Else, returns appropriate error Note: MPA may display an appropriate error and navigates back to the card settings screen on dismissal by user. 5. The Corporate Wallet SDK sends request to unsuspend the Card to the Wallet Server. 6. For Visa Cards, a. If Wallet server had initiated an Unsuspend token request for that particular token AND WS has received a successful response from VTS i. WS marks card as Suspended by User b. If Wallet server had initiated a Unsuspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Unsuspended by User c. If Wallet server had NOT initiated a Unsuspend Token request for that particular token AND WS has received a notification from VTS concerning the same token i. WS marks card as Unsuspended by Issuer 7. For Mastercard cards, WS sends the suspend card request to MDES 8. MDES/VTS unsuspend the card token mapping and sends response back to the Wallet Server. 9. The Wallet Server. a. Updates the card status to "Active". b. Sends response back to CW-SDK 10. The Corporate Wallet SDK updates the card status, marks card and token as "Active" and returns appropriate response back to MPA. 11. MPA may display a notification to let user know the card was unsuspended. When user navigates to the dashboard, the UI should be updated.
External Dependencies	1. MDES 2. VTS
Data Captured	Card status
Notes	If a card has been suspended by the issuer (see "Suspend Card – Santander Initiated" User Case), it can only be resumed by the issuer (see "Unsuspend Card – Santander Initiated" User

	Case). In other words, the “Unsuspend Card – User Initiated” Use Case cannot unsuspend a card suspended using the “Unsuspend Card – Santander Initiated” Use Case. For Visa, it is required to provide a reason code to VTS. As per Visa specs, the Wallet Server will always provide “CUSTOMER_CONFIRMED”.
Business Consideration	None
Exceptional Flows	1. If in Flow of event 4b, data connectivity is not available, Corporate Wallet SDK status will be updated when wallet resumes the data connectivity. 2. In case of communication error, timeout, or connection failure between MDES and WS, in the event following Unsuspension, resulting in the cards being unsuspended at MDES but not at CW-SDK and MPA, the following needs to be done: a. MDES will automatically retry 3 times with up to a 5-second wait between each attempt. If the call has not succeeded after the initial retries, MDES will attempt a second round of 3 retries with increasing time intervals between each retry. Between attempts the system will wait 15 minutes, 30 minutes, and then 2 hours. In the case of a 503, the Retry-After header will be respected if present and will count as a retry 3. In case of communication error, timeout, or connection failure between VTS and WS there will be retries with exponential backoff strategy where retry count and delay are configurable: a. Default retry count is 3. County can configure using vts.connection.retry.count in system level config key. b. Delay for next retry is extracted from exception info from VTS. If it does not exist, immediate retry will be triggered.

5.5 Remap card

This use case describes the process of card issuer remapping an existing FPAN to a different (or same) FPAN, while keeping same tokens.

This can occur in case of card being reported as lost or stolen or card expired.

Actor	1. Corporate Wallet SDK 2. MPA 3. Wallet Server 4. TSP (MDES or VTS) 5. Issuer
Channels	MDES, VTS
Pre-Conditions	1. User has successfully set up the MPA and has at least one card. 2. Corporate Wallet SDK in the active state. 3. Card reported lost/stolen or card expires.
Trigger Conditions	Issuer sends Remap notification to TSP (MDES/VTS)

Post-Conditions	The card with the new last 4 digits is updated and available at MPA
Start Point	TSP receives Remap notification from Issuer
Flow of Events	1. TSP notifies WS of the card remapping activity with the new last 4 digits (new masked FPAN) 2. WS updates the new last 4 digits using the TUR for MDES and the vProvisionedTokenID for VTS 3. WS sends success notification to CW-SDK 4. CW-SDK stores the new masked FPAN 5. CW-SDK notifies the MPA including a. The old masked FPAN b. The new masked FPAN c. The cardReferenceId 6. MPA displays the updated card with the new last 4 digits to user
External Dependencies	Issuer MDES, VTS
Data Captured	• Last 4 digits of FPAN (old and new Masked FPAN) • CardReferenceId
Notes	1. In case the TSP is not notified of the Remap activity changing the FPAN, and if user manually seeks to digitize new card via MPA, the issuer will decline the TAR since the new FPAN is already considered digitized by the Issuer 2. In cases where remapping takes place, Auto replenishment is not triggered, since the SUKs/LUK are linked with the token and not the FPAN. So, after remapping, the LUK/SUK count does not need to be revised or reset 3. The solution designed assumes that the Issuer will not allocate more than 1 card with same last 4 digits to a particular user
Business Consideration	None
Exceptional Flows	None
Customization	
Flow/Sequence Diagram	

5.6 Register with TDS – Background Process (Applicable to Mastercard cards only)

The process of registering with TDS results in generation of an “Authentication Code” which shall be stored in the Corporate Wallet SDK and will be leveraged for further communications with the TDS.

Actor	1. Corporate Wallet SDK 2. MPA 3. Wallet Server

	4. MDES 5. TDS
Channels	Corporate Wallet SDK
Pre-Conditions	1. MPA has successfully launched the Corporate Wallet SDK. 2. Corporate Wallet SDK in the active state. 3. Card has been added. 4. User's device has data connectivity/internet connection.
Trigger Conditions	One card has been activated.
Post-Conditions	Corporate Wallet SDK successfully registers with the TDS
Start Point	The Corporate Wallet SDK has received the card status as "Active" and request for replenish has been initiated.
Flow of Events	1. The Corporate Wallet SDK sends request to get transaction/registration code (with TUR) to the TDS. 2. The TDS processes the request and a) Sends registration code1 to the Corporate Wallet SDK. b) Sends registration code 2 to the MDES. MDES forwards the request to Wallet Server. 3. The Corporate Wallet SDK sends request to get registration code 2 to the Wallet Server. a) If the Corporate Wallet SDK receives the response, then go to next step. b) Else, retry once (after 2 sec). If the retry fails, then flush the registration code 1. Use case ends. 4. The Corporate Wallet SDK combines registration code 1 and 2 and creates a hash. 5. The Corporate Wallet SDK sends registration request to the TDS (TUR, hash). 6. The TDS processes the request and sends "Authentication Code" to the Corporate Wallet SDK. 7. The Corporate Wallet SDK a) Stores authentication code.
External Dependencies	MDES TDS
Data Captured	None
Notes	None
Business Consideration	None
Exceptional Flows	In case the registration process fails, the Corporate Wallet SDK shall maintain the state and shall complete the registration process in the background if the Wallet is active, card is active and MPA has been launched.
Customization	
Flow/Sequence Diagram	

5.7 View Transaction Receipt / Transaction History

5.7.1 View Transaction Receipt / Transaction History: Mastercard card

This use case illustrates a User who is able to view the transaction history from TDS.

Actor	1. User 2. MPA 3. Corporate Wallet SDK 4. Wallet Sever 5. MDES 6. Transaction Detail Service
Channels	Corporate Wallet SDK
Pre-Conditions	1. User has successfully set up the MPA and registered with TDS. 2. User's device has internet connectivity. 3. User has launched the Wallet and User is on the Wallet dashboard.
Trigger Conditions	User wishes view transaction history from the card or has conducted a transaction at point of sale.
Post-Conditions	User is able to receive the transaction history.
Start Point	1. User has conducted a transaction. 2. Wallet dashboard.
Flow of Events	**Scenario A: View Transaction Receipt** 1. User makes a payment using the MPA. Mastercard network stores transaction details in TDS. 2. TDS notifies Wallet server via MDES. The Wallet Server notifies the Corporate Wallet SDK using RNS_ID via MPA to receive new transaction details. 3. If data connection is not available on the device, Corporate Wallet SDK returns an appropriate error to the MPA. MPA displays appropriate error message to the user and use case ends. 4. The Corporate Wallet SDK receives the notification from the Wallet Server and sends the request to fetch transaction history to TDS service (with unique token identifier (TUR) and authentication code obtained in use case "Register with TDS"). 5. TDS service processes request and sends response with transaction details. 6. The Corporate Wallet SDK returns the transaction details to the MPA MPA displays the transaction details (Merchant Name, Timestamp, Currency, Amount, Status). **Scenario B: View Transaction History** 1. User selects an option to view the transaction history on card screen, either a. The user authenticates using CDCVM On successful authentication please go to next step. 2. The Corporate Wallet SDK sends the request to fetch transaction history to TDS service (with unique token identifier and authentication code obtained in use case "Register with TDS"). 3. TDS processes the request and sends the response with transaction details. 4. The Corporate Wallet SDK returns the transaction details to the MPA.

	5. MPA displays the transaction history for the card in the context (last 30 days) (Note: The latest 10 transactions shall be displayed).
External Dependencies	None
Data Captured	None
Notes	For scenario B if the card in the context is changed, the history shall be displayed for the new card in context. The history of all the previous transactions will no longer be available in case of auto-recovery.
Business Consideration	
Exceptional Flows	If anything fails due to system unavailability, the process should be restarted.
Customization	
Flow/Sequence Diagram	

5.7.2 View Transaction Receipt / Transaction History: Visa card

This use case describes the process in which a User is able to view the transaction history from VTS.

Actor	1. User 2. MPA 3. Corporate Wallet SDK 4. Wallet Sever 5. VTS
Channels	Corporate Wallet SDK
Pre-Conditions	1. User's device has internet connectivity. 2. User has launched the Wallet and User is on the Wallet dashboard.
Trigger Conditions	User wishes view transaction history from the card.
Post-Conditions	User is able to view the transaction history.
Start Point	1. Wallet dashboard
Flow of Events	**Scenario A: View Transaction Receipt** 1. User makes a payment using the MPA. 2. The VTS notifies Wallet Server. 3. WS notifies CW-SDK using RNS_ID 4. CW-SDK receives notification and requests Wallet Server for transaction history. 5. The Wallet Server sends "Transaction History Retrieval" request to VTS with vProvisionedTokenID, count, lastUpdatedTimestamp and encryptionMetaData . 6. VTS service processes request and sends response with transaction details to Wallet Server. 7. Wallet Server sends the response received from VTS to CW-SDK

	8. CW-SDK sorts the response based on transaction date. 9. The Corporate Wallet SDK returns the details of the most recent transaction by date to the MPA MPA displays the transaction details (Merchant Name, Timestamp, Currency, Amount, Status). **Scenario B: From transaction history option** 1. User selects an option to view the transaction history on card screen, either a. User authenticates using CDCVM On successful authentication please go to next step. 2. The Mobile Wallet sends the request to fetch transaction history to the Wallet Server. 3. The Wallet Server sends “Transaction History Retrieval” request to VTS with vProvisionedTokenID, count, lastUpdatedTimestamp and encryptionMetaData. 4. VTS service processes request and sends response with transaction history to Wallet Server. 5. Wallet Server sends transaction history to Corporate Wallet SDK. 6. The Corporate Wallet SDK returns the transaction history to the MPA. 7. MPA displays the transaction history.
External Dependencies	VTS
Data Captured	None
Notes	For scenario B, if the card in the context is changed, the history shall be displayed for the new card in context. MPA should display the transaction receipt or history matching the device locale. User will be unable to retrieve the transaction history for the cards which are in ‘Suspend’ state. VTS determines the number of transactions to be sent using the following logic: • For Scenario A: Client sends actual timestamp obtained from WS through PUSH --> server sends count =1 and time stamp to VTS to get 1 transaction in case of tap and pay • For Scenario B: Client sends 0 as time stamp --> server sends count =10 and time stamp =0 to VTS to get last 10 transaction. For Scenario A: Transaction Receipt after tap and Pay • The availability of the correct transaction receipt, after each Tap and Pay activity is dependent on Visa successfully providing the latest transaction details, since the data being provided does not contain an identifier by which to correctly correlate the Tap and pay activity and the Transaction receipt data being shared. The system is thus limited to utilize the timestamp of each transaction and assume the last timestamp is the latest one. The history of all the previous transactions will no longer be available in case of auto-recovery.
Business Consideration	
Exceptional Flows	If anything fails due to system unavailability, MPA will initiate the request once again.
Customization	
Flow/Sequence Diagram	

5.8 Get Remaining Available Payments

5.8.1 Get Remaining Number of SUKs: Mastercard card

This use case illustrates how the MPA gets the number of remaining SUKs (the number of transactions credentials that are remaining) in order to make payments.

Actor	1. MPA 2. Corporate Wallet SDK
Channels	Corporate Wallet SDK
Pre-Conditions	1. MPA has an active Corporate Wallet SDK. 2. MPA has launched the Corporate Wallet SDK.
Trigger Conditions	Get remaining SUKs from a card of choice.
Post-Conditions	The remaining SUKs are retrieved.
Start Point	Dashboard
Flow of Events	1. The MPA request to get the number of SUKs for a specific card. 2. The Corporate Wallet SDK returns the number of SUKs for a specific card. 3. Use case ends.
External Dependencies	None
Data Captured	None
Notes	None
Business Consideration	None
Exceptional Flows	None
Customization	None
Flow/Sequence Diagram	

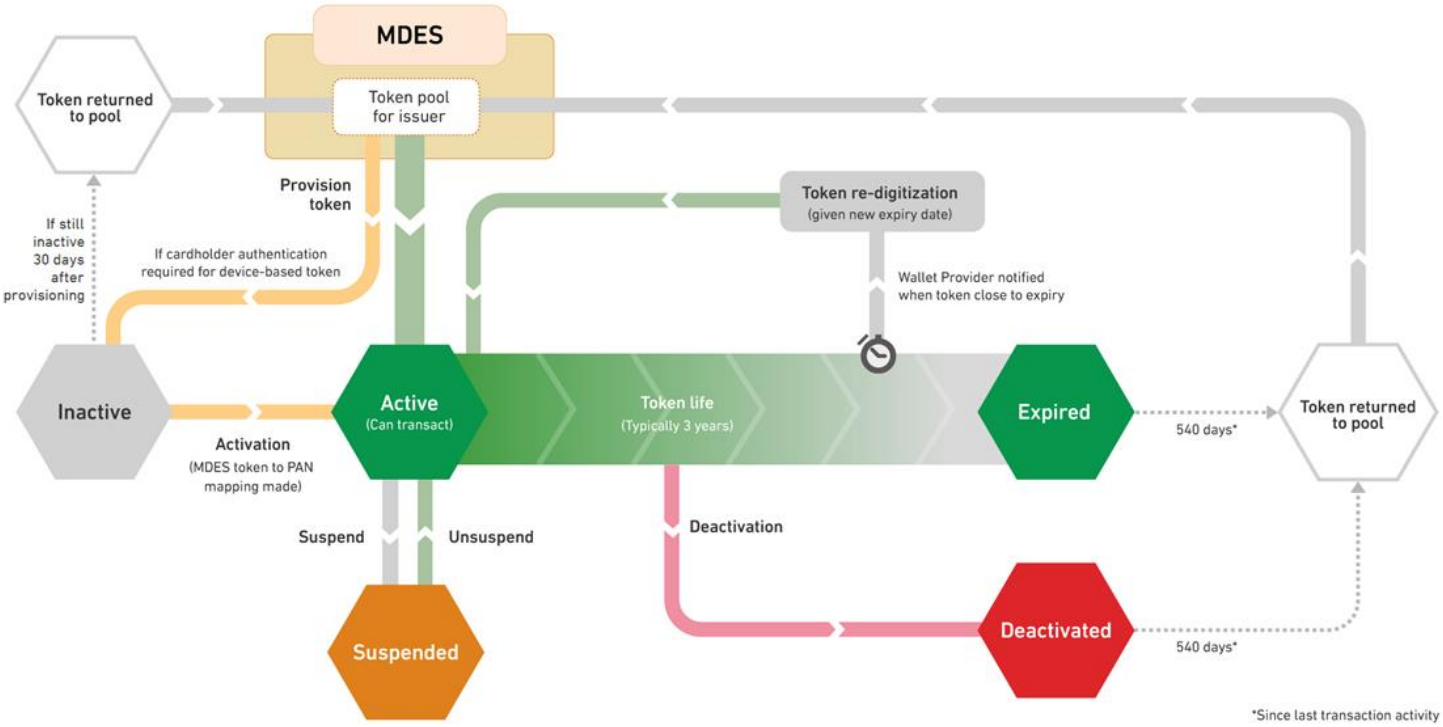
5.8.2 Get LUK limits: Visa Card

This use case illustrates how the MPA gets the LUK limits (device TTL-Time to Live or device NOT – Number of Transactions) in order to perform payments.

Actor	1. MPA 2. Corporate Wallet SDK
Channels	Corporate Wallet SDK
Pre-Conditions	1. MPA has an active Corporate Wallet SDK. 2. MPA has launched the Corporate Wallet SDK.
Trigger Conditions	Get remaining LUK limits from a card of choice.
Post-Conditions	The remaining LUK limits are retrieved.
Start Point	Dashboard
Flow of Events	1. The MPA request to get the LUK limits for a specific card. 2. The Corporate Wallet SDK returns the LUK NOT limit for a specific card. 3. The use case ends.
External Dependencies	None
Data Captured	None
Notes	Only the NOT part of the LUK can be obtained from URPay. The TTL is not available. Note that the NOT is a soft limit, the issuer can decide on a hard limit (i.e. it may still be possible to make payments when remainingNumberOfTransactions=0 depending on the issuer configuration). Visa replenishment lifetimes and replenish counts are primarily driven by the Wallet Server. The Visa Card profile contains the LUK TTL and maximum payment (soft) limit and Visa requests that any replenishment operation respects these limits. The URPay SDK will send a Replenishment required intent if either the maximum payment limit is close or the LUK TTL is about to expire.
Business Consideration	None
Exceptional Flows	None
Customization	None
Flow/Sequence Diagram	

5.9 Token Services

The following diagram represents the Token lifecycle:



5.9.1 MDES token expiry

MDES Tokens which are successfully provisioned, get an expiry of 37 months. Afterwards, Tokens are held for an additional 540 days to ensure any clearing and exception processing activity can occur successfully and they're eventually returned to the pool for future use by the issuer.

When a token reaches it expiry date, it becomes invalid for making payments and the only course of action is to re-provision the existing card or a different card into the digital wallet. To prevent this inconvenience, MDES introduced a new mechanism to renew tokens which are about to expire, so that Token Requestors (i.e. the Santander Wallet) can manage the renewal transparently for cardholders.

When a Token is within 30 of expiring, MDES notifies the Wallet so that it can request the re-digitization of the token before it actually expires. Token re-digitization extends the life of the token, making it valid for another three years, avoiding any inconvenience to the cardholder. The re-digitized token has the same TUR and token PAN. If multiple re-digitizations are attempted on the same token, an additional month will be added to the previous expiration date.

When a token is re-digitized successfully, MDES sends the issuer a Tokenization Event Notification (TVN) message and/or a Notify Token Updated (NTU) message - if the issuer chose to receive them. These requests include the reason for the token update and the new token expiration date. Key considerations:

1. Tokens are identified 30 days prior to expiry and notified to the Token requestor by MDES.
2. The Wallet must initiate the re-digitization request.
3. There's no change to the Token provisioning flow.
4. The wallet provider is notified when (re)provisioning is complete. New Tokens are ready to be consumed.
5. Old credentials must be discarded, and new ones should be requested.
6. Re-digitization shall work without user intervention.

5.9.2 Re-digitization of MDES tokens (AN3819)

In July 2020, Mastercard introduced a mandate coming into effect by 1 November 2020 whereby the following use cases shall be supported by consuming the re-digitization API:

Use Case	Situation Currently	What's Changing	Impact to existing tokens
Token Expiry. Token is nearing expiry. One month before token expiry, Mastercard will request that the token requestor re-digitize the token.	MDES supports this via existing NotifyTokenUpdated (NTU) call to token requestor. Issuers can opt in to receive Tokenization Event Notifications (TVNs) or NTU notifications.	No change (this is currently the only listed use case for re-digitization using the Redigitize API).	• TUR and Token stays the same. • Expiration date is extended by three years from the date of the re-digitization.
Attribute Change. Issuer performs an attribute change at the BIN account-range level impacting their MDES enabled ranges. Some device tokens may need to have their data refreshed to match the new attributes.	Partially supported. It is currently possible to change some attributes of an account range, provided that there is no impact on the tokens (for example, brand product code is allowed, but country code change is not).	Mastercard will support the ability for an issuer to make attribute changes at the BIN account-range level for their MDES enabled BIN account ranges. Details of when this functionality will be available will be published in a separate bulletin announcement.	Depending on the change requested, Mastercard may need to update the chip data on device-based tokens. In this event: • TUR and Token stays the same. • Expiration date is extended by three years from the date of the re-digitization.
BIN Account-Range Split. Issuer performs a BIN account-range split. Some existing tokens may need to be updated to ensure that they are linked to the correct funding BIN account ranges internally.	Not supported, it is not possible to split an MDES enabled BIN account range and keep the tokens active.	Mastercard will support the ability for an issuer to split their MDES-enabled BIN account ranges. Details of when this functionality will be available will be published in a separate bulletin	• TUR stays the same. • Tokens for account ranges that belong to a different funding range after a BIN account-range split

		announcement. Issuers can opt in to receive TVN or NTU notifications.	will be replaced with a new token DPAN in the correct token account range. • Expiration date is extended by three years from the date of the re-digitization.
PAN Update in Different Account Range. Issuer updates the cardholder's PAN via the Automatic Billing Updater (ABU), the Customer Service API/Portal, 0302 network message, R311, or the Payment Account Management (PAM) API, and the new PAN is in a different BIN account range.	Not supported, it is not possible to issue a new card to a cardholder on a different BIN account range and keep the tokens active.	Mastercard will support changes to the cardholder's PAN when the new PAN is in a different account range. Details of when this functionality will be available will be published in a separate bulletin announcement. Issuers can opt in to receive TVN or NTU notifications.	• TUR stays the same. • Token is replaced (new DPAN). • Expiration date is extended by three years from the date of the re-digitization.

Key considerations:

1. Wallet provider must contact their Mastercard representative to request the re-digitization notifications are enabled. Typically, this would be done following the wallet providers update to their MDES implementation to support re-digitization.
2. In addition to implementing the enhancements for re-digitization, the wallet provider must also support:
    a. Encrypted content in notifyTokenUpdated API calls from MDES.
    b. Payment Application Instance ID (PAID) in body of message rather than in the URL.

Actor	1. User 2. MPA 3. Corporate Wallet SDK 4. URPay 5. Wallet Sever 6. MDES 7. RNS (FCM)
Pre-Conditions	1. A Token is provisioned into a User’s Wallet 2. The Token is reaching its expiry
Trigger Conditions	MDES sends a notification through the Notify Token Updated outbound call to the Wallet provider to indicate a token is within 30 days of expiring
Post-Conditions	Token is re-digitized transparently without user intervention
Start Point	1. MDES sends a NTU to the Wallet provider to indicate a Token, identified by a TUR is within 30 days of expiring.

Flow of Events	1. CWS receives the outbound call from MDES, looks up in the database the token which is expiring and stores a flag to indicate that re-digitization is required for the token. 2. CWS sends a push notification to the CWS via RNS 3. MPA receives the push notification and sends the payload to the CW-SDK. 4. CW-SDK triggers the synchronize service to update the status of the tokens and wallet. 5. CWS returns the token and wallet status to the CW-SDK. 6. CW-SDK iterates over the tokens and marks that re-digitization is required for the token which is about to expire. 7. CW-SDK calls CWS to re-digitize the token 8. CWS sends a re-digitize call to MDES 9. MDES returns the response to the CWS 10. CWS propagates the response to the CW-SDK 11. CW-SDK receives the response from CWS and calls the re-digitize API exposed by URPay. 12. URPay confirms it's ready to perform the re-digitization. 13. CW-SDK provides the payload received by CWS with the card information. 14. URPay stores the information into the payment SDK and sends a confirmation to CW-SDK. 15. CW-SDK updates the token details into the DB. 16. MDES sends a push notification to URPay via RNS to re-provision the token that has just been re-digitized, given the old credentials must be discarded after the re-digitize API is called. 17. URPay handles the push notification and notifies CW-SDK that it is ready for provisioning. 18. CW-SDK calls the provision API exposed by URPay. 19. URPay establishes a session with the CMS-D to provision (or re-provision) the token. 20. CMS-D provides the token profile to URPay. 21. URPay stores the card profile and associated credentials into its storage and notifies CMS-D provisioning was successful. 22. CMS-D acknowledges the response. 23. URPay then notifies CW-SDK of the result of the operation: a. If the operation is successful i. The flag indicating re-digitization is needed for the token is removed from CW-SDK database. ii. Use case ends b. If the operation is unsuccessful i. CW-SDK will attempt again. Go to step 18.
External Dependencies	MDES RNS
Data Captured	None
Notes	1. For Reliability for Corporate Wallet SDK to Wallet Server communication: Please refer section "Corporate Wallet SDK to Wallet Server Communication". 2. For CW-SDK error codes, please refer to CW-SDK API
Business	

Consideration	
Exceptional Flows	If connectivity between MDES and CWS fails due to system unavailability, MDES will send additional notifications to re-digitize the token as the date approaches so CWS can perform the re-digitization. In case the token is expired, users will not be able to transact with the token in their wallet, and they will have to remove the card and digitize it again.
Customization	
Flow/Sequence Diagram	See Sequence Diagrams for further details on this flow.

5.9.3 VTS token expiry

Just like MDES tokens, Visa tokens also have an expiry date, which is 3 years after the primary card expires. The purpose of this feature is to support the re-digitization of expiring Visa tokens. Unlike MDES tokens, VTS tokens have an expiry based on the expiry of the primary card. When a primary card is about to expire, Issuers need to call the Lifecycle API.

VTS notifies the Token Requestor (TR) by a Card Metadata Update notification. The TR is expected to call the Get Card Metadata API to get the latest Info (new Expiration Date or New Last 4 PAN number) for the given token and updates it DB. TR will notify to CWSDK, who would then initiate a sync request and update token in its DB and notify MPA.

When this API is called, and either the FPAN expiry gets extended or the FPAN changes, the DPAN gets its expiry extended by 3 years after the new FPAN expiry.

In case there is an FPAN1 expiring in 2021 which has a DPAN1 expiring in 2024. When FPAN2 is issued, for example, expiring in 2025, Issuer needs to call the lifecycle API and then FPAN2 (2025) would be linked to DPAN1 (2024). When the lifecycle API is called, that will increase the DPAN1 expiry to FPAN2 expiry + 3 years, that is, the expiry of DPAN1 would also be extended until 2028.

To summarize, there wouldn't be cases where a token expires before the physical card does and since Issuers need to take action, the Wallet Server simply needs to be ready to handle the Card Metadata Update notification.

For further details please refer to the PAN / Token expiry flow in the Sequence Diagrams document.

5.9.4 VTS re-perso service

As a part of the upgrade of the VTS API specification, starting with VTS 19.04, a new service needs to be implemented to manage the token re-personalization: TOKEN_REPERSO_ADVICE.

There are cases when the token personalization profile gets changed by Issuers or businesses which does not occur often. In this case, Visa will update those personalization profiles into the VTS system. This personalization profile must be updated on the device as well otherwise transactions using those tokens will fail. The only solution, until the API was introduced, was to delete the existing token and digitize the card again.

With the new API, user experience is improved and there is no need to delete or digitize the card again. Ideally, without any user interaction, the personalization profile should be replaced so that the Tap & Pay operation will not stop.

When Visa notifies the Wallet Provider of the TOKEN_REPERSO_ADVICE, Wallet Server should notify CW-SDK which, in turn, should request URPay to start the re-perso flow. Given the VTS implementation requires Server to Server communication, the request should be triggered from CWS.

At this point CW-SDK sends a request to CWS that will ultimately reach out to VTS to re-personalize one of the existing tokens. The response from VTS will be passed onto the CW-SDK and URPay that will store the data received from Visa.

Lastly, upon a successful provisioning into URPay, CW-SDK confirms the provisioning to VTS through the CWS, to which VTS responds with an acknowledgement which is propagated to the CWS, CW-SDK and URPay.

At this point, the token has been successfully provisioned, database is updated to remove the need to re-provision the token and the use case ends.

For further details, please refer to the VTS re-perso flow in the Sequence Diagrams and the VTS API Specification v19.04.

6 Transaction Services

The Tap and Pay use cases below assumes Mastercard and Visa transactions only.

6.1 Tap and Pay payments and refunds

This use case illustrates a User is able to make the NFC based payment over HCE using the card digitized in to their MPA when the phone is locked, unlocked or manually selecting card and initiating the payment.

Actor	1. User 2. MPA 3. Corporate Wallet SDK 4. POS Terminal 5. Wallet Server		
Channels	MPA		
Pre-Conditions	1. User has successfully set up the MPA, User has launched the MPA, User is on the Wallet dashboard, and The Wallet is in “active” state	1. User has successfully installed and set-up the MPA.	
Trigger Conditions	1. User opens the MPA and selects the card for payment or refund.	1. User wishes to do a tap and pay transaction with a phone which is: a) unlocked and b) MPA is in foreground	1. User wishes to do a tap and pay transaction with: a) a locked or unlocked mobile phone and b) MPA not in foreground.
Post-Conditions	User is able to make payment using the default or selected card.		
Start Point	1. Card Screen	1. User taps the mobile phone against the NFC terminal	
Flow of Events	1. The user selects the option to pay / to make a payment via NFC on the card screen.	1. The user first taps the phone on the NFC terminal.	
	2. The MPA requests to the Corporate Wallet SDK to initiate the transaction.	2. If phone screen is off, a) Then there may be no activity in MPA and/or terminal. The use case ends.	

		Note: There are some phones that enable HCE even when the phone screen is off. b) Else go to next step.	
	3. The Corporate Wallet SDK checks if NFC is active. a) If yes, then go to next step. b) Else, Corporate Wallet SDK returns to the MPA. MPA can prompt User to switch on the NFC functionality. The user switches it on and selects “Back” button to navigate to the MPA. The use case ends.	3. If NFC is off, a) Then there is no activity in MPA and/or terminal. The use case ends. b) Else go to next step.	
	4. The CW-SDK checks If “MPA is set as default app for NFC”, a) If yes, then go to next step. b) Else, if it is Android Lollipop or above and ‘Favor foreground application’ selected, and CW-SDK has overridden default setting, a. then go to next step b. Else, Corporate Wallet SDK returns to the MPA. MPA can prompt user to select current application as default payment application. The use case ends.	4. If “MPA is set as default app for NFC”, a) If yes, then go to next step. b) Else, If it is Android Lollipop or above and ‘Favor foreground application’ selected, and CW-SDK has overridden default setting, a. then go to next step b. Else, Corporate Wallet SDK returns to the MPA. MPA can prompt user to select current application as default payment	4. If “MPA is set as default app for NFC”, a) If yes, then go to next step. b) Else, the use case ends.

		application. The use case ends.	
		5. CW-SDK checks for screenlit up, if true: a) then getConsent event will be notified by CW-SDK to MPA b) else MPA receives onTransactionError callback with specific error code from CW-SDK MPA checks if device is locked, if true: a) then use case stops b) else go to next step	
	6. If the wallet is “locked”, a) Then, the CW-SDK informs the MPA The MPA informs the user that the wallet is locked. The use case end. Else go to the next step.		
	7. Corporate Wallet SDK to check if SUK or LUK are available: a) If yes, Corporate Wallet SDK returns the number of available SUKs or the availability of LUK, then go to next step. b) If no SUKs/LUKs are available, checks if internet connectivity is available to replenish: a. If Internet is not available, Corporate Wallet SDK an appropriate error to MPA. Application is in foreground or background: MPA displays message indicating the user to switch on the internet. The use case ends. b. If internet is available, and replenishment starts or is in progress, then Corporate Wallet SDK returns an appropriate error to MPA 8. MPA displays a message indicating the user that update is in progress and the user will be notified later. The use case end.		
	9. if CVM Always Switch is On, then directly go to Step b.ii else CW-SDK does the velocity check only if the ‘CVM Always switch is Off, if the velocity check passes (i.e. no CVM required, see “Velocity Check” section) , a. Then, the Corporate Wallet SDK extracts the Mobile PIN from the Payment SDK b. Else, Corporate Wallet SDK returns to the MPA and: i. In case the Wallet PIN is used: 1. MPA prompts User to enter the Wallet PIN. 2. The user enters the Wallet PIN. 3. The MPA provides the Wallet PIN to the Corporate Wallet SDK. 4. The CW-SDK receives the Mobile Pin from the MPA ii. Or (XOR) in case the CVM is used		

		1. The MPA triggers the CVM. 2. The user enters the CVM, and the CVM is validated. 3. The MPA indicates to the Corporate Wallet SDK that the CVM has been successfully validated. The CW-SDK extracts: Mobile PIN from the whitebox module for Mastercard payments. 4. CW-SDK (via URPay MP SDK) generates the cryptogram. In case of Mastercard card the Mobile PIN is either coming from whitebox or from MPA (which then needs to provide it for this operation).
	10. The Corporate Wallet SDK: a. Activates card for payment/refund. b. returns an appropriate error to the MPA, and c. MPA asks the user to tap. If the user Taps the Mobile phone against the terminal within the 3 minutes timer: i. Then go to next step. 11. Else, the use case ends.	11. If user has inserted the CVM or the Mobile PIN, a. Then go to the next step b. Else skip the next step
	12. Corporate Wallet SDK sends the cryptogram to the terminal via NFC HCE. 13. If the contactless terminal accepts transaction request, a. Then go to next step. b. Else, Corporate Wallet SDK returns an appropriate error to the MPA. c. The MPA can displays an error on the Mobile Wallet. The MPA navigates to the dashboard screen. 14. The contactless terminal d. Sends an acknowledgment to the Corporate Wallet SDK. The Corporate Wallet SDK returns an appropriate error to the MPA e. Sends authentication request to the Processing network. f. Prints transaction receipt.	

	15. The MPA can display a confirmation that transaction was submitted successfully and invokes View Transaction History use case. 16. After a transaction history notification is received by the MPA. The MPA requests the transaction receipt from the Corporate Wallet SDK. The Corporate Wallet SDK checks if internet is available. a. If yes, then invoke depending on the type of card the “View Transaction Receipt: Mastercard Card” or the “View Transaction Receipt: Visa Card” use case. b. Else, the Corporate Wallet SDK returns an appropriate error to the MPA i. MPA display message indicating internet unavailability. The use case ends.
External Dependencies	Processing Network
Data Captured	From terminal: Transaction amount, Transaction currency, CVM requested, MCC (to identify a transport type merchant) From TDS: Transaction Receipt information.
Notes	1. When the MPA requests the Corporate Wallet SDK to fetch transaction details, the MPA will display the message (indicative ‘Please wait, fetching transaction details). In case of no response from TDS, after 20-30 seconds the MPA should time out and the earlier message should be replaced with the message ‘Unable to obtain transaction details. Please check the terminal to see if your transaction was complete’ 2. Certain smartphone OEMs have made changes to the standard behaviour wherein if the phone is locked and phone screen is off; and if the phone is tapped against the terminal; the screen is invoked, and the transactions goes through. 3. Message if token is over “Information update is required; please establish a data connection as soon as you can”. 4. MPA should display POS tap transaction amount confirmation. 5. The MPA must listen to catch the device unlock (CVM) event in order to handle the pre-arm counter used in the velocity check 6. The sequence of CW-SDK checks may change during LLD. 7. For more details about “Always CVM” setting, refer Velocity Check. 8. In case there is no default card available at time of Tap and Pay, the CW-SDK will send and exception to MPA and use case will end. 9. It is the responsibility of the MPA UI to set and update correctly the mobile CVM limit so that it matches the terminal CVM 10. For Chile, CVM Always switch and Velocity Check rules will be bypassed. Also, if the phone screen is on but the phone is locked, the transaction won’t be allowed.
Business Consideration	1. None
Exceptional Flows	1. In case CW-SDK discovers that TDS registration is not done earlier, before requesting the transaction history, CW-SDK will be do the TDS registration and then invoke that service. Please refer to TDS registration use case for more details on registration. 2. In case the merchant uses a V2 POS/Terminal, during a Tap & Pay, the CW-SDK immediately notifies the MPA UI. After receiving this notification with the information that a V2 terminal has been tap, the MPA UI shall:

	○ In case of locked screen and the screen is ON: show a message on top of the lock screen to inform user that he needs to unlock the device just before tapping the (V2) terminal the second time. ○ In case the device is unlocked (even when the user has another application in foreground) and when the 1st tap happens more than 30 seconds after the user authentication (e.g. device unlock): show a message in foreground to inform user that he needs to authenticate just before tapping the (V2) terminal the second time. Depending on the UI and when the 2nd tap happens the MPA UI may show the user authentication screen.
Customization	None

6.2 Tap and Pay refunds

This use case covers the scenario wherein a User wants to be refunded for a previous purchase made with their Mobile Wallet.

Depending on the country and region, the refund process is triggered differently, sometimes directly in the POS by searching for the transaction reference and others by tapping on the POS with the smartphone and a card selected, acting like a regular Tap and Pay payment described in 6.1 but with a positive amount to be added to the recipient card.

Actor	1. User 2. MPA 3. Corporate Wallet SDK 4. POS Terminal 5. Wallet Server		
Channels	MPA		
Pre-Conditions	1. User has successfully set up the MPA, User has launched the MPA, User is on the Wallet dashboard, and The Wallet is in “active” state	1. User has successfully installed and set-up the MPA.	
Trigger Conditions	1. User opens the MPA and selects the card for refund.	1. User wishes to do a Tap and Pay transaction to get a refund with a phone which is: a) unlocked and b) MPA is in foreground	1. User wishes to do a tap and pay transaction to get a refund with: a) a locked or unlocked mobile phone and b) MPA not in foreground.
Post-Conditions	User is able to tap her phone against the POS using the default or selected card in order to receive a refund		

Start Point	1. Card Screen	1. User taps the mobile phone against the NFC terminal
Flow of Events	Refunds tapping in a POS don't differ from a regular Tap and Pay payment. Please refer to Use case 6.1 for a detailed description of the flow.	

6.3 Token Replenishment

6.3.1 Automatic SUK Replenishment: Mastercard cards

This use case is used to replenish the transaction credentials when the count of available tokens (SUKs) on the device goes below the minimum threshold.

Actor	1. MPA 2. Corporate Wallet SDK
Channels	Corporate Wallet SDK
Pre-Conditions	1. MPA has an active Corporate Wallet SDK. 2. MPA has launched the Corporate Wallet SDK
Trigger Conditions	Transaction credentials are below minimum threshold.
Start Point	1. Change Wallet PIN 2. Reset Wallet PIN 3. Tap & Pay
Post-Conditions	Corporate Wallet SDK automatically replenishes the transaction credentials.
Flow of Events	1. The Corporate Wallet SDK detects that transaction credentials have reached the minimum threshold limit. 2. The Corporate Wallet SDK checks if internet connectivity is available on the phone a) If yes, go to next step. b) Else, the Corporate Wallet SDK returns an appropriate error MPA may choose to display a message prompting users to switch on the data connectivity. a. If user switches on data connectivity, then go to next step. b. Else, the use case ends. 3. The Corporate Wallet SDK sends request to establish secure session to CMS-D. 4. The CMS-D a) Sends an acknowledgement to the Corporate Wallet SDK. b) Sends request to RNS_ID via MPA to send notification to Corporate Wallet SDK. 5. The Corporate Wallet SDK sends request to replenish token to CMS-D. 6. The CMS-D sends the transaction credentials to the Corporate Wallet SDK. 7. The Corporate Wallet SDK a. Securely stores the transaction credentials.

	b. Calls the Callback “CardReadyToPay” to the MPA when there is a full replenishment from 0 to maximum SUKs. 8. The use case ends.
External Dependencies	CMS-D
Data Captured	Tokens
Notes	Thresholds to be defined and configured by MPA.
Business Consideration	None
Exceptional Flows	The Corporate Wallet SDK shall inform the MPA that tokens are below threshold limit.
Customization	None
Flow/Sequence Diagram	

6.3.2 Automatic LUK Replenishment: Visa cards

This use case is used to replenish the transaction credentials for Visa cards.

Actor	1. MPA 2. Corporate Wallet SDK
Channels	Corporate Wallet SDK
Pre-Conditions	1. User has successfully set up the Wallet. 2. User’s device has data connectivity/internet connection.
Trigger Conditions	VTS notifies the Wallet Server that notifies CW-SDK via MPA that replenishment is needed
Post-Conditions	Corporate Wallet SDK replenishes the LUK
Flow of Events	1. VTS notifies the Wallet Server that notifies CW-SDK via MPA that replenishment is needed. 2. The Corporate Wallet SDK checks if internet connectivity is available on the phone a) If yes, go to next step. b) Else, the Corporate Wallet SDK returns an appropriate error MPA may choose to display a message prompting user to switch on the data connectivity. a. If user switches on data connectivity, then go to next step. b. Else, the use case ends. 3. The Corporate Wallet SDK sends request to Wallet Server to replenish the LUK. 4. The Wallet Server forwards request to the VTS. 5. VTS processes request and sends the response to Wallet Server. 6. The Wallet Server forwards the response to the Corporate Wallet SDK. 7. The Corporate Wallet SDK securely stores the LUK information.

	8. The Corporate Wallet SDK confirms to Wallet Server that LUK replenishment has been successful. 9. The Wallet Server confirms to VTS that LUK replenishment has been successful calling “Active Account Management—Confirm Replenishment”.
External Dependencies	VTS
Data Captured	LUK information
Notes	None
Business Consideration	None
Exceptional Flows	None
Customization	None
Flow/Sequence Diagram	

7 Other Use Cases / Sub Processes

7.1 Authenticate User via ID&V Login Process

This use cases describes the process to authenticate against Santander's ID&V system in order to perform multiple use cases in the solution.

Actor	1. User 2. MPA 3. Santander ID&V / SPS 4. CW-SDK
Pre-Conditions	1. User's device has data connectivity/Internet connection. 2. User has initiated either of following processes: a) Set up Mobile Wallet: Register Wallet
Trigger Conditions	Mentioned in pre-conditions
Start Point	Mentioned in pre-conditions
Post-Conditions	User successfully is successfully authenticated against Santander Identity and Verification System

Flow of Events	1. MPA displays login screen whereby the user can enter the Issuer online login credentials. 2. User enters the login credentials. 3. MPA sends the request with login credentials to the Santander ID&V to fetch ID&V token. 4. The Santander ID&V processes the request and sends identity token to the MPA. 5. In case the use case requires authentication level 2 (L2): a. The MPA gets the hash SHA2 (operation, parameter1, paramN, WalletId, DeviceId) from the CW-SDK b. MPA sends to the Signature Pattern Service (SPS) the SHA2 hash and the identity token and performs the authentication level 2 (L2). c. SPS i. If authentication passes, generates the data token and returns it to the MPA. Go to next step. ii. Else, sends an error response to the MPA. The user case ends. d. MPA receives the response MPA stores the ID&V token and/or data token in cache memory.
External Dependencies	Santander ID&V
Data Captured	ID&V login credentials
Notes	1. See the References table for more information on the Santander ID&V System. 2. No critical data is stored on the Wallet Server 3. Expiry for data token will be as defined under SPS Specification. 4. MPA manages the authorization token session
Business Consideration	None

Flow of Events	1. MPA displays login screen whereby the user can enter the Issuer online login credentials. 2. User enters the login credentials. 3. MPA sends the request with login credentials to the Santander ID&V to fetch ID&V token. 4. The Santander ID&V processes the request and sends identity token to the MPA. 5. In case the use case requires authentication level 2 (L2): a. The MPA gets the hash SHA2 (operation, parameter1, paramN, WalletId, DeviceId) from the CW-SDK b. MPA sends to the Signature Pattern Service (SPS) the SHA2 hash and the identity token and performs the authentication level 2 (L2). c. SPS i. If authentication passes, generates the data token and returns it to the MPA. Go to next step. ii. Else, sends an error response to the MPA. The user case ends. d. MPA receives the response MPA stores the ID&V token and/or data token in cache memory.
Exceptional Flows	None
Open Issue	None
Flow/Sequence Diagram	

7.2 Synchronize Corporate Wallet SDK - Wallet Server (Checks for Mobile Client Communication)

With every request from Corporate Wallet SDK to Wallet Server, Wallet Id and Device fingerprint will piggy back the request using which Wallet Server shall perform a series of checks to ensure that the Wallet Server and Corporate Wallet SDK are in sync with the Wallet states. Additionally, if a need for T&C update has been identified, then user is informed to take the appropriate action.

Actor	1. Corporate Wallet SDK 2. Wallet Server
Channels	Corporate Wallet SDK
Pre-Conditions	1. User downloaded and launched the MPA and Corporate Wallet SDK is launched by MPA. 2. User's device has data connectivity / Internet connection. 3. Wallet is active.
Trigger Conditions	1. MPA calls synchronize method of CW-SDK. 2. Occurs also for every communication between CW-SDK and WS.
Post-Conditions	If there is change in Corporate Wallet SDK state, then Corporate Wallet SDK is updated accordingly, and an appropriate screen is displayed.
Flow of Events	1. The Corporate Wallet SDK communicates (sends request with timestamp) with Wallet Server 2. The Wallet Server performs the checks.

	a) Device fingerprint check a. If device fingerprint check fails, the use case ends. b. Else, proceed to next check. b) Blacklist check a. If check fails, then the Wallet Server sends response to the Corporate Wallet SDK. The Corporate Wallet SDK return a corresponding message to the MPA that should manage the behaviour (e.g. application may exit). b. Else, then proceeds to the next check. c) MPA version (see upgrade notification use case) 3. The Wallet Server fetches the following items: a) Card token state b) Wallet state 4. The Wallet Server a) Prepares response with Wallet status, Card token status, minimum MPA version, and potentially these notifications if required a. Blacklisted device b. Incorrect fingerprint c. MPA upgrade. b) Sends response to the Corporate Wallet SDK. 5. The Corporate Wallet SDK receives response and stores the timestamp. 6. If Wallet Status is “Active”, then continue with the process for which the communication with Wallet Server was initiated.
External Dependencies	None
Data Captured	None
Notes	1. In case the Corporate Wallet SDK receives a failure as response (for specific error codes as mentioned in CW-SDK API Document), it is recommended that MPA retries, calling the synchronize service via CW-SDK.
Exceptional Flows	None
Customization	None
Flow/Sequence Diagram	

7.3 Device Blacklisting – Exception Flow

The blacklisting check is performed at registration, during sync or while performing any of the critical CWS services. In case the device is found to be blacklisted while checking for device eligibility, the following steps are performed:

a. WS returns appropriate error message to CW-SDK.

b. CW-SDK forwards the response to MPA.

c. MPA may display an appropriate message to User.

The MPA can also call the Terminate Wallet API to delete the wallet instance and data from that particular blacklisted device.

Given below is the list of CW-SDK APIs which include the Device Blacklisting check:

1. checkDeviceEligibility
2. sync
3. startRecovery
4. create wallet
5. register with MDES & VTS
6. digitize
7. digitizebyList
8. getTerms
9. requestActivation
10. activate
11. delete
12. resume
13. suspend

7.4 SafetyNet Check considerations

The SafetyNet validation detects tampered or compromised devices. With SafetyNet, Google is able to notify applications whether the device is compatible with the Compatibility Test Suite (CTS). CTS is a suite of tests a device must pass, prior to its commercialization, to ensure certain security requirements are met and for the Google Play Services to be included in the software.

This means, SafetyNet is about assuring that the device is safe to run the application.

The SafetyNet check is performed in the background once the CW-SDK is initialized. In cases where the SafetyNet verification fails, the Wallet Server terminates the wallet instance.

SafetyNet check is made asynchronously and it is possible for the MPA to re-use the session to create a new Wallet again after terminating a previous one and also make other API calls provided by the CW-SDK, if the session is not terminated or app is closed.

If a new Wallet is created in such situation, the digitize card can be initiate and the card will be provisioned should the digitization is successful. Also, the wallet will be marked as “Active” and user will be able to transact with the payment cards available in the app. The user will also be able to perform tap and pay transactions if he does not launch the app.

In order to resolve the conditions described above, the following solution is implemented:

- Once the user launches the app and the CW-SDK is initialized, the SafetyNet check is called asynchronously . The check, however, is not initiated before accessing the local CW-SDK APIs in which there are no server communications (e.g. `/getWalletInfo`).
- Once the SafetyNet check response is received (success / failure), it is stored in memory.
- If the response returned is failure, an exception is returned to the MPA. The MPA is not able to access any of the APIs within the same session anymore. The wallet instance also remains terminated on the server side.
- If the response returned is successful, once the MPA terminates the session, force closes the application or the device is restarted, the result is cleared from memory and on initialization of the CW-SDK the next time, the steps above are repeated.
- Below is a summary of the flow of events when SafetyNet check is performed:

Actor	1. User 2. MPA 3. CW-SDK 4. CWS 5. Google SafetyNet
Channels	MPA
Pre-Conditions	User has Internet connectivity
Trigger Conditions	User launches MPA
Post-Conditions	SafetyNet check is performed and a result is returned to the CWS and CWS
Flow of Events	1. User launches MPA. 2. CW-SDK is initialized. 3. If CWS Public Key, Nonce or the Server Timestamp are not available in runtime memory in the CWS: a. If there’s network connectivity, those are requested to the CWS b. Else, CW-SDK returns an error message to the MPA. MPA may choose to display a message to user prompting for Internet connectivity 4. CWS derives all these 3 values and provides a response to the CW-SDK. 5. CW-SDK stores the values in runtime memory. 6. CW-SDK makes a call to the Google Play Services to obtain the SafetyNet attestation result (JWS payload) passing the Nonce value obtained from the CWS. 7. Google Play Services respond with a the SafetyNet attestation result (JWS Payload). 8. CWS invokes the verifyNonce service provided by the CWS in order to validate the attestation result.

	9. Corporate Wallet Server uses the HSM to decrypt the payload and verifies it. 10. CWS responds to the CW-SDK with the result of the validation. a. If Payload verification is successful, use case ends. b. Else, i. If User consent is required, CW-SDK will raise an exception to the MPA asking for user consent. 1. If User accepts the risk, response is propagated to CW-SDK and CWS. Corporate Wallet Server logs the acceptance and confirms the CW-SDK that user consent is successful. Use case ends. 2. If User does not accept the risk, MPA should indicate User the application can't be used. Use case ends. ii. If CWS detects the device could potentially be tampered, CW-SDK returns the corresponding error code to the MPA that will show a proper message and it will be terminated. Use case ends. iii. If SafetyNet verification fails, CWS terminates the Wallet. CW-SDK also marks the Wallet as terminated, wipes all stored data, and propagates the result to the MPA that should present a proper message to the user. Use case ends.
External Dependencies	Google SafetyNet
Data Captured	None
Notes	Please refer to the Security Architecture Document for more details on SafetyNet
Exceptional Flows	
Customization	None
Flow/Sequence Diagram	Please refer to the Sequence Diagrams for a graphical representation of this flow

7.5 Encrypted payload between MPA and CW-SDK

Santander Portugal had raised a security concern on the card info being passed in plain text from MPA when CW-SDK APIs are called. Although this is the regular way of operating with SDKs, Santander Portugal InfoSec team requires this interface to receive encrypted data.

As a resolution, card info provided by the Santander backend should come encrypted using an asymmetric encryption scheme known as Elliptic Curve Integrated Encryption Scheme (ECIES). Under this scheme, a keypair will be generated in the HSM by the security officers at Santander countries.

Before the Santander backend provides the list of cards for a user, it will encrypt the data using the public key using a function provided by the HSM, that shall return an encrypted payload. This payload will then be transferred to the MPA which can then pass it to the CW-SDK using a polymorphic method.

CW-SDK will send the payload to the CWS that will detect it's in encrypted form. To decrypt it, CWS needs to make a call to the HSM that will decrypt the card information using the private key store within it. The decrypted content will be returned to the CWS that will be able to perform the required checks and make the corresponding calls to the TSPs to digitize the cards.

This encryption mechanism requires additional calls to the HSM which increases the time required to digitize the cards, having an impact in the user experience.

Given other countries have not raised any concerns to pass the card information to the CW-SDK in plain text, the solution was made configurable so that issuers can decide to either continue passing card info in clear text or adapt using encrypted card info

When encryption of card info is enabled, validations on the card number, expiry and card type will be only performed at CWS. CW-SDK will ignore the validations. CWS will propagate relevant errors to CW-SDK and further they would be displayed to users

Encrypted card info is supported in both single card digitization and digitization by list.

7.6 Tenant Parameters Configuration

Tenants can configure the subset of features available in the context of this Wallet Solution. As of today,

- **HCE Only:** In this feature subset, the Wallet Solution shall be able to perform only contactless payments.
  a. Applicable for only Android devices.
  b. Not applicable for web and HCE blacklisted devices.
  c. Tenant must be MDES or VTS enabled or both.

In the past, there were additional configurations to support Masterpass only or both (HCE and Masterpass) for converged wallets but these two configurations were removed when Masterpass was removed from scope.

7.7 Purging of inactive tokens

Due to the restrictions introduced on Android Q whereby the access to certain APIs and identifiers is restricted for non-privileged (system) applications, CWS will not be able to detect the same Device Fingerprint after it has migrated to Android Q.

This will result in Wallet records being inaccessible from MPA to manage their status, hence some tokens will remain active but will not be accessible by any means, for example:

- After digitization on a device, user uninstalled and re-installed the same app and have performed the digitization again. Before Android Q, it was providing the same DFP so CWS was able to delete the previous app instance. But same won't be possible with Android Q, which leads to the creation of dormant app instance records on the wallet server.

- The user simply uninstalls the application after digitization and never uses the application. This will also lead to create dormant app instance records.

Please refer to section MPA Re-install, Upgrade and Factory Reset for the different scenarios.

As recommended by Mastercard Security Officers, a solution is required to suspend and delete records that are inaccessible. A new feature has been implemented in the new Admin Portal that will allow tenants to configure a period for suspension or deletion of inactive tokens for all tokens.

Each tenant is able to configure the period of time for suspension or deletion given in the number of months, taking as baseline the last day of activity presented in the token. Configuration can be applied for either suspend or delete tokens or both at the same time.

Once tokens are deleted, the Application Instance will be deleted, and the Wallet terminated. This applies only when the delete option is enabled.

Suspension and deletion process will be performed automatically after the defined number of months, with 0 used to indicate that no suspension or deletion will take place. A scheduler has been implemented and will be triggered when the inactivity period configured is met, making calls to CWS, MDES and VTS to complete the process.

When a period of suspension/deletion is modified, the newer period will always prevail even if it's lower than the period entered previously. For example, if the deletion of tokens was configured within 6 months from the last day of activity and suddenly this value is reduced to 2 months, all the tokens matching inactivity for 2 months will be deleted.

Deletion and suspension will be reflected in the metrics reports.

Default Settings

The default settings displayed on Admin Portal, which shall be editable by Admin User are:

- Suspension: 3 months
- Deletion: 6 months

7.8 Java utility for HSM crypto operations

A utility was created at the sole discretion of Mastercard to reduce troubleshooting time by performing a standalone test of crypto operations with HSM, using the HSM API's.

This utility is an executable jar built on Spring boot and its use is limited to non-production environments.

The following cryptographic operations are supported for testing:

1. Loading key store
2. Getting certificates from key store
3. Encryption and Decryption using AES algorithm
4. Generating a secure and random code

5. Wrapped encryption using AES algorithm
6. Unwrapping of key

Please refer to Flame HSM API Test Utility Developer Guide for technical details

7.9 Validation of load balancing via HSM library

When Santander countries started implementing their Wallets, the default setup had one HSM paired up with multiple CWS instances. As CWS instances would be scaled horizontally, Santander realized a need to balance load for crypto operations. This led to a setup of having more than one HSM in the backend. Realsec (HSM vendor) implemented client-side load balancing, wherein their client library at CWS would act as a load balancer and would decide which HSM would be reached out for crypto operations when multiple HSM servers are available.

The decision algorithm implemented in the library was designed and is proprietary to Realsec. No updates were required at CWS.

Load balancing can be achieved by configuring the set of available HSM servers and setting load balancing flag to true in the Realsec library as below at each CWS instance

```
[CONEXION]
SERVERTYPE=REMOTE
IPADDRESS=81.XX.YYY.ZZZ <---Server 1
PORT=11111
SECURITY=true

SERVERTYPE=REMOTE
IPADDRESS=81.ZZ.YYY.ZZZ <---Server 2
PORT=11112
SECURITY=true

[PROFILE]
PATH_AUTH=0
CONNECTIONS_PER_SERVER=1
LOAD_BALANCING=true <---Enable load balancing
```

7.10 Mocked SDK mode

Santander Corporate requested a feature that would enable MPA developers to validate API calls from MPA to SDK with mocked (dummy) responses. Mastercard upgraded their SDK to make it configurable to work as either a real SDK or a mocked SDK.

In the mocked SDK mode:

1. SDK will provide MPA with mocked responses for successful and error scenarios.
2. Only input data validations will be considered to evaluate success/error scenarios.
3. Business exceptions are out of scope.
4. SDK does not transact with any systems like MDES, VTS, FCM, URPay, CWS etc.
5. MPA developer needs to invoke API's in correct sequence.

To enable mocked SDK mode, kindly refer to the Flame CW-SDK Integration guide for further details.

7.11 CWS Health Status / Corporate Wallet version

CWS provides an API that can be called to prove the system is up and running, useful to Santander monitoring systems that can make periodic requests to ensure there's no disruption and even automate the behavior should the Corporate Wallet server stops responding.

When this service is called, CWS returns its version number, something that can also be useful for troubleshooting during deployments, considering different versions may be deployed in the different servers and environments Santander may have.

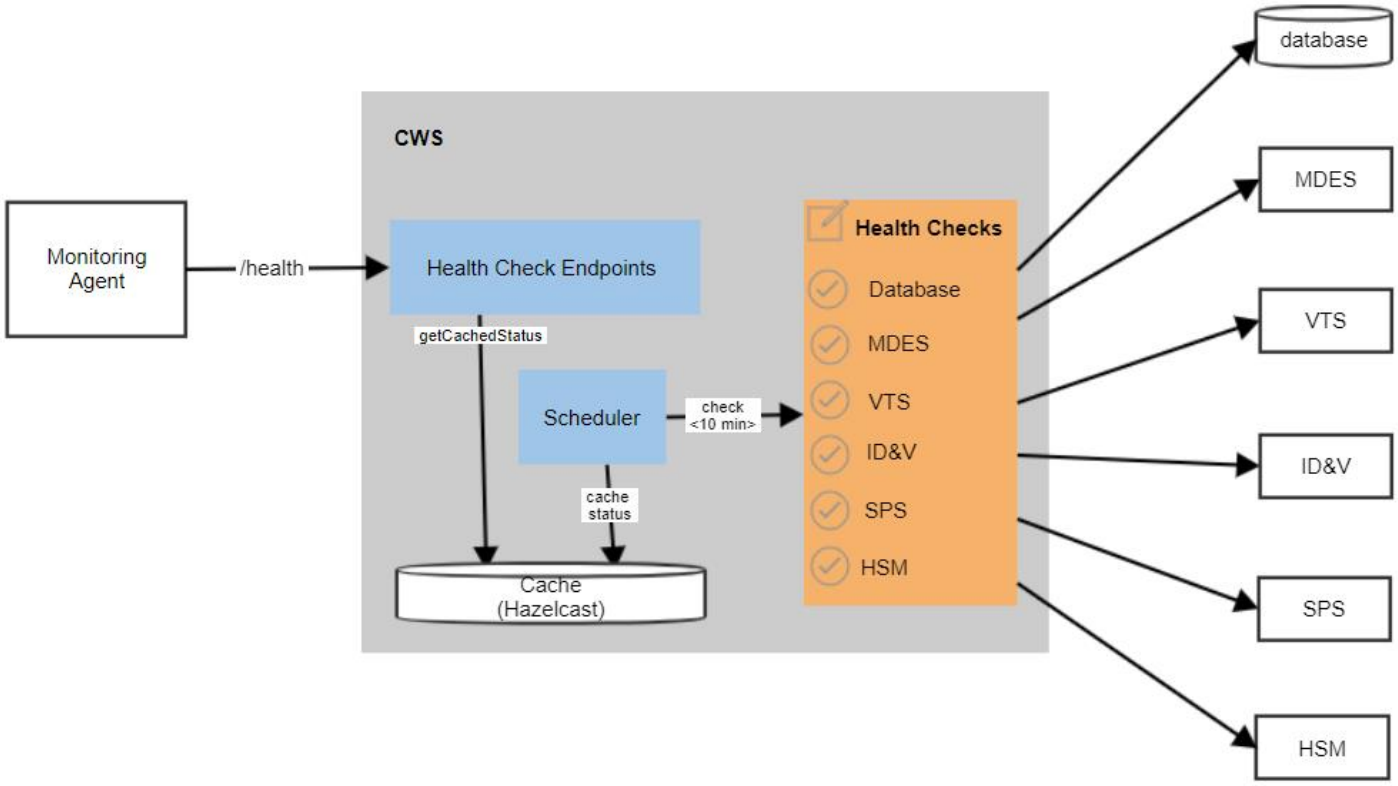
The CW-SDK version, however, can be checked in the build.gradle file included.

For details about the health check API please refer to the Flame CWS & Admin Portal Application Installation Guide.

7.12 Ecosystem Health Check

In addition to the existing health check API which returns the CWS version, an ecosystem health check has been incorporated. The Ecosystem Health Check is a lightweight health check service that validates the status of all services which are part of the wallet ecosystem, such as MDES, VTS, ID&V, SPS, HSM and the databases by sending requests to their endpoint. Then, CWS performs the necessary checks and returns an indication of its status.

This service can be used by Santander to be notified should any of the services goes down and can be used by their monitoring or log aggregation systems.



Entity	Description
Monitoring Agent	A monitoring agent which calls application's health check endpoint and does further analysis and reporting.
Health Check Endpoints	A set of REST endpoints which return cached status of health of one or more internal/external component. Sample APIs include (1) individual component status (b) whole eco-system status
Scheduler	Runs at predefined interval and performs functional checks of internal/external components.
Cache	Stores (in-memory) result/status of health of internal/external components.
Health Checks	Service layer (a separate service for each component health check) performing health check.

At a high level:

- CWS performs a number make the checks against all the systems they are connected to: DB, MDES, VTS, ID&V, SPS, HSM.
- Ensure connectivity is checked in terms of:
  - System availability
  - Response time for health check request.
- Timing / intervals of the checks should be defined via configuration file.

- Minimum time for check – 5 mins.
- The configurations are kept in database/configuration home.
- Endpoints can be configured.
- Checks can be enabled or disabled for specific systems.

The output of this service is logged to a file or to the standard output so that the monitoring agent used in each country can read this input and act accordingly.

Please refer to the Flame CWS Admin Portal Application Installation Guide for further details.

7.13 Decoupling of URPay from CWSDK

Since 2016, the URPay library was an integral part of the CW-SDK, thus making CW-SDK a complete layer of abstraction for the MPA. In release 1.0 and 1.2, there were multiple incidents to upgrade or replace the URPay library.

With each incident, the entire CW-SDK package had to be re-built and delivered . This led to recurrent efforts awas time consuming. Hence, we took up a feature to decouple the CWSDK and URPay. This way whenever there was a change in URPay, library the change and replacement of the URPay would be transparent to CWSDK considering that no change in existing URPay API flow. This change was released in R2.1 in Q3 2018. This allows MPA to initialize URPay first and if successful, then only initialize the CWSDK and continue with the launch MPA use case. Decoupling also allows replacing the URPay completely with another vendor's library in a relatively easy manner than earlier.

8 Portal Access Services

The Flame Wallet is complemented by a web-based application to manage and operate key aspects of the Wallet. This interface is intended for Santander employees in charge of managing the Wallet.

These users will get a local account created by a system administrator (a local account, not connected to any SSO or LDAP that may exist in their environment) that will be used to access the MWB Admin Portal to manage aspects such as:

- Global Configuration
- Tenant Management
- User Management
- Roles and Permissions
- Device Blacklist Management
    - OS Management
    - Device Management
        - Manual entry or by CSV file
- Application configuration
    - Required for every application published in the Google Play Store.
- Access to Wallet reports
- Token purging configuration

For an in-depth description of the different features and use cases, please refer to the Flame MWB Admin Portal User Guide included with the product.

9 Metrics and Reporting

In order to better assess the product performance in the markets, Corporate Wallet exposes APIs to collect information from the client and stores it in the CWS database.

This information will later be processed by a standalone application operating on the CWS database that generates reports in CSV format with information such as the total number of wallet instances, active wallets, transaction details, etc.

The Reporting application can be configured to enable/disable reporting at CWS as well as to define the frequency to generate different types of reports which are stored in the filesystem, in a folder of choice for the system administrator. In addition to the filesystem, reports are also available through the Admin Portal, considering the challenges Santander countries face when accessing the filesystem in production systems.

Reports consider the following states in the database:

- Wallet states
  - Active
  - Terminated
- Card States
  - Active
- Token States:
  - Inactive (when digitization response is received but provisioning fails),
  - Active
  - Suspended
  - Deactivated (equivalent to Terminated)

There are three types of reports the CWS can generate:

1. Cumulative
2. Periodic
3. Detailed

Report Type	Reports
Cumulative	- Wallet Instances Report - Cards Cumulative Report - Token Status Cumulative Report - Digitization Methods Status Cumulative Report
Periodic	- Wallet Instances Weekly (Periodic) Report - Cards Weekly (Periodic) Report
Detailed	- Detailed Wallet Instance Report - Detailed Card Report - Detailed Transaction Report

9.1 Cumulative Reports

Cumulative reports contain cumulative data (gradual increase) from the date of system deployment.

For example, if a cumulative report runs monthly, then every month the generated report will containing the requested information from system deployment date till date plus all the previous generated records so that Santander can have the historic view in a single CSV.

The following cumulative reports can be generated:

- **Wallet instances reports**, with information of the created, active and terminated wallet instances.
- **Cards**, with information of the cards activated.
- **Token Status**, with information of the active, inactive, deactivated and suspended tokens.
- **Digitization path**, with information of green and yellow path
    - Note: yellow path is not in scope but can be detected given tokens remain in inactive state. CWS does not have awareness of red path, as the cards don't get digitized.

9.2 Periodic Reports

Periodic reports contain wallet, card and token information in a specific period of time, be it a week, a month or a quarter. These reports also contain information of the previous period of time.

The following periodic reports can be generated:

- **Wallet instances reports**, with information of the created, active and terminated wallet instances.
- **Cards**, with information of the cards activated.
- **Token Status**, with information of the active, inactive, deactivated and suspended tokens.
- **Digitization path**, with information of green and yellow path.
    - Note: yellow path is not in scope but can be detected given tokens remain in inactive state. CWS does not have awareness of red path, as the cards don't get digitized.

9.3 Detailed Reports

Detailed reports containing comprehensive details of events that occurred in a specific period of time.

The following detailed reports can be generated:

- **Wallet instance**, with information of the created, active and terminated wallet instances along with the Payment App Instance ID and Device ID
- **Card**, with information about the Payment App Instance ID, Device ID, PUR, TUR, Pan Suffix, Token status and last update.
- **Transactions**, with information of the Payment App Instance ID, Device ID, PAN suffix, Transaction date, Transaction amount, Merchant name, Transaction Status, etc.

10 Auto-recovery (only in R2.1)

10.1 Auto- recovery after URPay init error (due to secure unlock)

The wallet solution SHALL be able to recover automatically all the cards for the user anytime as soon as the user has changed the security settings from mobile secure lock to unsecure lock that triggers a URPay exception (NATIVE_INITIALIZATION_ERROR) making unusable all card tokens of the wallet client.

This recovery procedure is called auto-recovery.

The auto-recovery should not need customer to select again cards to be digitized. The customer will however have to login using the ID&V credentials to restore the same digitized cards.

Note: this functionality has recently been removed from URPay SDK R2.2 which works with Santander Corporate Wallet versions R2.2 and R2.2.1. As a result, auto-recovery is only available in Santander Corporate Wallet version R2.1

Actor	1. User 2. Corporate Wallet SDK 3. MPA 4. MDES 5. VTS 6. Wallet Server 7. URPay 8. TDS 9. Santander Card Enabler System 10. Santander ID&V
Channels	Corporate Wallet SDK
Pre-Conditions	1. User has completed MPA launch and Wallet state is Active
Trigger Conditions	1. User changes settings to unsecure unlock
Post-Conditions	1. Restoration of previously digitized cards
Flow of Events	1. CW-SDK catches the URPay init error. 2. CW-SDK resets URPay a. If reset URPay successful, go to next step. b. Else Use Case Ends. 3. CW-SDK notifies MPA. 4. (INFORMATIVE) MPA verifies whether or not the secure unlock has been set calling the performEnvironmentalChecks API. a. (INFORMATIVE) If secure unlock not set, the MPA informs user to set back a secure unlock. b. Else if secure unlock is set i. (INFORMATIVE)The MPA displays ID&V login page.

ii. (INFORMATIVE)The user enters the ID&V login credentials and proceeds, go to next step

5. MPA triggers auto recovery by requesting the CW-SDK to process the startRecovery API.
6. CW-SDK requests WS to provide the PIN and SALT values using the getPIN API passing the ID&V credentials.
7. Additionally, CW-SDK initiates SafetyNet check asynchronously. If the SafetyNet check is successful, go to next step. If SafetyNet check is not successful, then refer the 'Exceptional Flows'.
8. WS receives the request and logs the getPIN start event
9. The Wallet Server checks if the ID&V token (L1 validation) is valid using bank deployed services.
    a. If the Token is invalid the WS will send appropriate error message to CW-SDK, use case ends.
    b. Else go to next step
10. The Wallet Server gets the User ID calling the Bank service GetLoggedUID passing the ID&V token
11. The WS checks if the ID&V token has the same user ID of the wallet record.
    a. If the user ID is different then
        i. WS will pass the error message to CW-SDK
        ii. CW-SDK will in turn pass the message to MPA, use case ends
    b. If the user ID is same, go to next step
12. WS processes the getPIN request and responds with the values
13. WS logs the getPIN end event
14. CW-SDK generates the RGK and the MPIN
15. CW-SDK requests WS to register with MDES and VTS and fetch all tokens passing the ID&V credentials
    a. Repeat steps 8 to 10
16. WS logs the register with MDES, register with VTS, fetch_all_token start event
17. WS forwards the register request to MDES and VTS as necessary
18. MDES and VTS process the request and send success response
19. WS receives the success response and logs the register with MDES and register with VTS end events
20. WS processes the fetch_all_token request (i.e. the WS fetches the latest Card Snapshots like status, suspended by etc for all wallet cards)
21. WS logs the fetch_all_tokens end event
22. WS sends the response to CW-SDK
23. CW-SDK backups the previous default card details
24. CW-SDK updates the current state of the cards to "To be Deleted" and the target state as the desired state
25. CW-SDK stores the card snapshots of the card states
26. CW-SDK requests WS to delete the cards
    a. Refer use case - Delete Cards: By User, Step 10
27. CW-SDK updates the current states for all cards as appropriate based on response received from WS
    a. If there is at least a card where Current state is "Deleted" and target state is "Active" or if Current state is "Deleted" and target state is "User Suspended".go to next step

b. Else if Current state is “Deleted” and target state is “Issuer Suspended” Use Case ends

28. CW-SDK requests WS to digitize all cards by calling the DigitizeListPUR API (including vPanEnrollmentID for Visa) passing the ID&V credentials
    a. Repeat steps 8 to 10
    b. Refer use case – “Set Up Corp Wallet SDK: Add Cards List”, Step 8
29. WS logs the digitization end event
30. WS sends the response to CW-SDK
31. CW-SDK call the sync function
32. WS provides response
33. CW-SDK updates the current card status as appropriate (for all applicable cards)
    a. If current card state is “Active” and target state is “User Suspended”
        i. CW-SDK calls the Suspend API
            1. Refer use case – Suspend Card: User Initiated, step 9
34. WS sends the response to CW-SDK
35. CW-SDK updates current card status
36. CW-SDK sets previous default card as default
37. CW-SDK sends the response for the startRecovery API call to MPA (COMPLETE or PARTIAL)
    a. If the startRecovery API response is COMPLETE (this also includes RAA cards but with target state different from “User Suspended)
        i. (INFORMATIVE) MPA displays dashboard to user
        ii. Use Case ends
    b. Else If startRecovery API response is PARTIAL, then
        i. If response to startRecovery API is PARTIAL due to one or more RAA cards and with Target state as “User Suspended” then for each RAA card execute the following steps:
            1. The MPA lets the user select the activation method for the RAA card
                a. Refer use case – Set Up Corp Wallet SDK: Add Cards: Digitize Mastercard Card Step 15 for Mastercard Cards or Set Up Corp Wallet SDK: Add Cards: Digitize Visa Step 12 for Visa Cards
            2. If RAA card is successfully activated,
                a. CW-SDK marks current card state as “Active”
                b. Repeat steps 32 a to 36
            3. Else if card is not successfully activated,
                a. Go to step 36 b i 1
        ii. Else if response to startRecovery API is PARTIAL due to auto-recovery failing for any other reason
            1. (INFORMATIVE) MPA updates auto-recovery retry counter
            2. If auto-recovery retry counter is equal to or greater than the limit, then
                a. MPA calls the Reset API
                b. CW-SDK deletes all local data
                c. CW-SDK sends success response to MPA

	d. (INFORMATIVE) MPA asks user to setup the CW-SDK. Refer Use cases: 4.3.1 Set up Corporate Wallet SDK: Pre-setup Process e. Use Case ends 3. Else MPA triggers auto recovery again 4. Go to step 5
External Dependencies	1. MDES 2. VTS
Data Captured	Tokens
Notes	1. Logging of Auto recovery start and end events – There will be no specific Auto recovery start and end event logs maintained by the WS. The WS will only log specific API events that are requested by CW-SDK. 2. The technical interpreter can utilize the getPIN start event to decipher that the auto-recovery process has been initiated and can make an informed judgement on the exact performance of the process by looking at the various logs on the respective start and end events. 3. The count of the no. of retries has to be maintained at the MPA since the limit of the no. of retries will be governed as per the country specific requirements. 4. On Android OS version equal to or above 6.0.1, auto recovery should not be needed because Matrix HCE/ /URPay 2.0 should not throw initialization error in this case, considering that the Android keystore in those versions behaves differently from lower versions. 5. In case of Partial recovery, if the user deletes a token (via CSR or manually) which had target state as Suspended, then the recovery table will be updated and the token will no longer be recovered in case the MPA initiates auto recovery again. 6. In case the cards have target state as “Issuer Suspended”, following steps need to be taken as per the country of deployment: - Brazil – User will need to re-select and re-digitize the card manually. - Chile – The issuer would have to immediately re-digitize the card without any user intervention. 7. If SafetyNet attestation is not successful (done Asynchronously), then depending upon the following scenarios, appropriate action is performed: a. **JWS signature verification failure:** CW-SDK will send an appropriate exception to MPA indicating that the device appears to be tampered and the MPA will exit. CWS will maintain a failure counter for each such occurrence. After ‘n’ such occurrences (*where ‘n’ is configured as “SafetyNet Counter” tenant configuration*), wallet instance will be terminated. b. **Nonce verification failure:** Appropriate exception is returned by CW-SDK to MPA. c. **Technical Error / Unknown Error:** Appropriate exception is returned by CW-SDK to MPA. d. **JWS Payload Processing Error:** For this failure scenario, several validations will be performed against the following parameters: i. CTS Profile ii. Basic Integrity iii. Package Name, Cert Digest, APK Digest

Appropriate response will be generated / action will be performed based on validation results. Please refer section Safety Net Check for details on the validations performed.

There will be four different actions (paths) as a result of validations performed on the above listed parameters. Please refer the below table:

CTS Profile	Basic Integrity	PackageName,Cert Digest,Apk Digest	Action
False	True	Null	**Path 1**
False	True	Match	**Path 1**
False	False	Null	**Path 2**
False	False	Match	**Path 2**
False	False	Mismatch	**Path 3**
False	True	Mismatch	**Path 3**
True	True	Mismatch	**Path 3**
True	True	Null	**Path 4**
True	True	Match	**Path 4**

1. Path 1:
   i. CWS to deliver proper response to CW-SDK when receiving response corresponding to Path 1.
   ii. CW-SDK will provide response to MPA
   iii. WS resets the counter which keeps the count of occurrences of path2.
   iv. In parallel of step 3, (Recommended) MPA shall prompt message to notify user that his device may be impacted by tampered device status and require user acceptance of the risk.
   v. (Recommended) If user accepts the risk then proceed further and maintain the acceptance to prevent repeated prompts in future. Else, exist the app.
   vi. CW-SDK provides to CWS the acknowledgement of user acceptance of Risk (if provided by MPA or automatic acknowledgement response).
   vii. CWS resets counter of occurrences.

2. Path 2:
   i. When SafetyNet validation falls under condition described in Path 2
   ii. WS will maintain a Counter to Server to measure this failure case for “Active” wallets.
   iii. WS will Increment the counter on each occurrence and MPA will prompt a message to the user to notify that his device looks tampered and exit the app.

	iv. If the Counter reaches n, terminate the wallet instance, n value will be defined in "SafetyNet Counter" tenant configuration. 3. Path 3: i. When values that coincide with what is detailed in chart for Path 3 wallet instance will be terminated. 4. Path 4: i. When SafetyNet validation falls under condition described in Path 4. ii. CWS resets counter of occurrences. iii. MPA continues with BAU.
Business Consideration	None
Exceptional Flows	1. In case of timeout between WS and CES, WS will retry to send the notification to up to 3 times and will not pursue any steps to ensure guaranteed delivery of notification 2. In case the card to be suspended belongs to Yellow Path: Since an inactive card cannot be suspended, the card will have to be activated by user intervention. Only once the cards is active can the suspend activity be carried out for User Suspended cards. 3. In case the default card belongs to the Yellow Path: Once the user activates the card, the same card will be set as default. In case the user sets a green path card as default before activating the yellow card, the new card will now act as the new default card and the yellow card even after activated, will cease to be the so.

11 Replenishment Approach

11.1 Replenishment Approach: Mastercard cards

The following table elicits the triggers for replenishment along with the initiator and a brief description. Please refer “Launch Wallet”, “Add Card”, “Reset PIN/Change PIN” flows.

Trigger Point	Description	Initiator
When SUKs number is below threshold	Replenishment happen as soon as the number of SUKs is below the SUKs threshold.	Corporate Wallet SDK
Manually triggered	The user is triggering the replenishment from the MPA	MPA

Notes:

- Replenishment will always be done in the background (as a background process).
- **In order to perform a replenishment, the MPA must be launched and active.**
- **Corporate Wallet SDK will inform MPA once replenishment is done via callback.**

11.2 Replenishment Approach: Visa cards

The following table elicits the triggers for replenishment along with a brief description. Please refer “Launch Wallet” and “Tap & Pay” flows.

Trigger Point	Description	Initiator
When LUK is below threshold	Replenishment happens as soon as reaching device TTL (Time to Live) or device NOT (Number of Transactions).	Corporate Wallet SDK
When in response to VTS notification	Replenishment happens in response to VTS notification to CW-SDK via Wallet Server.	VTS
Manually triggered	Not supported by Visa specifications	N/A

12 NFR, Assumptions and configurations

12.1 Localization

Corporate Wallet SDK only supports English language, and the support of other languages or localization is delegated to the MPA. However, it is assumed that MPA will send the data required to CW-SDK in English.

12.2 Security

Santander Corporate Wallet is by design, a secure solution relying on a secure technology stack and components which have been approved by Mastercard and Visa.

In addition to using securing components, the solutions must adhere to the strict security requirements from the payment schemes, including but not limited to being compliant with OWASP Web Security Testing and passing an SBMP (Software-Based Mobile Payments) security evaluation with an independent Security Lab approved with every major release.

For further details about the security please refer to the SECAD (Security Architecture Document).

12.3 Assumptions and Considerations

- Only a single user in a single device is supported. Multiple user profiles on one single device are not supported.
- In order to perform a refund operation, users shall perform either:
    - A Tap and Pay operation with any token of the same card that has been used for the purchase, or
    - Use the plastic card related to the same digitized card that has been used for the purchase.

  Alternatively, in some markets, the refund operation can be triggered from the merchant through the POS without interaction with the card or mobile device.
- If the same Santander customer is downloading and installing the MPA on 2 mobile devices, the Wallet Server will creates 2 separate wallet application instances.
    - A consequence of this is that change handset flow is not applicable.
- The RNS_ID (Wallet Server RNS Registration ID) shall be communicated from the Wallet Server.
- The MPA is responsible to receive the FCM push notifications via RNS_ID and to dispatch the relevant ones to the Corporate Wallet SDK.
- The MPA is registering to FCM getting the different registration IDs for Wallet Server and MDES.
- The MDES RNS Registration ID is sent to the WS and to MDES.
- Concerning the T&C, Santander must upload during on-boarding (BPMS) the T&C also in MDES. MDES provides the T&C content and T&C Id to the Wallet Server in the Card Eligibility Check response together with the eligiblityReceipt. T&C content is suppressed by the Wallet Server after the 1st digitized card and if the T&Cs are not changed. However T&C Identifier and the T&CAcceptedTimestamp are mandatory parameters to be passed to MDES in the next service call called Digitization.

- The ID&V authentication level (L1 or L2) per specific Wallet Server Mobile web-service is a tenant configuration. The Use Case Authentication level (L1 or L2) is checked by the Wallet Server using the SPS and the ID&V services from the bank. The authentication level is defined as:
    - L1 → Login required (user and password), MPA authenticate the users connecting to the ID&V back-end that returns the ID&V token. The ID&V token is sent from the MPA via the CW-SDK to the Wallet Server and validated by the Wallet Server using ID&V services provided by the back.
    - L2 → Second factor authentication, the MPA performs the authentication connecting to the SPS that generates the signature data token. The data token is sent from the MPA via the CW-SDK to the Wallet Server and it is verified by the WS using the SPS.

  The Corporate Wallet SDK is not aware of the authentication level, but it is responsible to send the ID&V token, the data token and the operational data to the Wallet Server.

  MPA is responsible to execute the authentications with the right authentication level (L1 and L2) required by the consumed WS services (see use case “Authenticate User via ID&V Login Process”).
- The LVT currency thresholds for 3 currencies are provided by the MPA to the CW-SDK. The Corporate Wallet SDK gets the currencies and the LVT currency thresholds during the “Get Payment Card Update – Background Process” use case.
- Retry strategy between WS and MDES will be as per MDES Specification 1.1.2 and will be applicable for all services.
- Retry strategy for Visa: there are 2 options:
    - VTS is not able to reach WS, then it will retry 3 times with increased delay between retries.
    - Wallet Server will retry 3 times if it fails to contact VTS.
    - VTS is able to reach WS but gets an error. No further retries will be made.
- T&C are managed at MPA level (i.e. MPA must validate T&C from MDES and VTS on CW-SDK, and can also manage their own T&C with the Santander back-end). All Santander Wallet eligible cards have the same T&C/T&C ID. If T&C ID is different from the one that has been received originally (e.g. new T&C ID due to the updated T&C), user has to explicitly accept it only once.
- The following items are taken as assumption concerning the cards received by the MPA used for digitization:
    - MPA receives the eligible cards with details from a Santander system not described in this document.
        - Santander should respond to MPA about eligible cards based in unique Wallet instance id.
        - Santander is already taking care of threshold value of number of tokens allowed per FPAN during digitization.

12.4 Android permissions required

The following Android permissions are required for the SDK to work and are declared in the Android Manifest. Depending on the Android OS version where the MPA is running, they may be requested to users at installation time on the Google Play Store or during runtime.

For further details, please refer to the CW-SDK integration guide.

12.4.1 INTERNET

Required to open network sockets and communicate with the Wallet Server.

This permission and the ACCESS_NETWORK_STATE permission are normal permissions, which are granted at install time and don't need to be requested at runtime.

12.4.2 ACCESS_NETWORK_STATE

Required to let the SDK know about the network state and know when there are connectivity changes.

12.4.3 ACCESS_WIFI_STATE

Required to let the SDK know about the Wi-Fi state and know when there are connectivity changes.

12.4.4 NFC

Required to make payments with the device.

12.5 Configurations

12.5.1 Wallet Server Configurations managed via Portal

Sr No	Configuration Item	Description	Value
1	Device Blacklist	Update device blacklist	Country dependent

12.5.2 Wallet Server Configurations managed via Configuration file

The table below indicates tenant configurations on a configuration file that shall be accessible by Operator. When this configuration will be accessible via Portal then will be manageable by Tenant Administrator Profile. Note that any configuration change requires Wallet Server restart that requires system admin access.

Sr No	Configuration Item	Description	Value
1	Web service Conf for Signature Pattern Service	This configuration will enable a tenant aware configuration on the portal as to which service to be called for the token validation service that a wallet server needs to call for validating the data token	Country Dependent
2	Service Specific Authentication Preferences	This configuration will be enabling a tenant aware configuration on the portal which will tell which Authentication level is applicable for Wallet services, Services will be highlighted later. Note: L2 is disabled by default.	L1, L2

12.5.3 CW-SDK Configurations

Sr No	Configuration Item	Description	Value
1	LVT Threshold Value / Currency Code	The map of country specific LVT Threshold value and Currency code managed at the MPA level for phase. 3 sets of (LVT Threshold, Currency code) are be managed	Country specific. e.g. SPAIN $50, USD.
2	Minimum SUK Threshold	This value needs to be embedded into the MPA and passed to CW-SDK. bundled with the application	Issuer Dependent Value
3	Max Consecutive Transaction	Number of transactions after which user must be authenticated using the Wallet PIN (BACKLOG). There is one value for LVT, and one value for HVT. These parameters are named “LvtCounterThreshold” and “HvtCounterThreshold” in CW-SDK API	MPA specific
4	Max Total Amount	Max sum of the amounts of the transactions after which user must be authenticated using the Wallet PIN (BACKLOG) This parameter is named “AccumulatedAmtThreshold” in CW-SDK API	Country specific
5	Session Time Length	Session time after which user must be authenticated using the Wallet PIN (BACKLOG). There is one value for LVT, and one value for HVT. These parameters are named “LvtSessionDuration” and “HvtSessionDuration” in CW-SDK API	Country specific
6	walletServerURL	Host URL of Wallet Server	Country specific
7	environment	This flag would be used by MPA to inform SDK about the environment in which the MPA application is running. It can be ‘development’, ‘sit’, ‘uat’ or ‘production’. Note: Certain checks can be performed only when the application is in specific environment. (e.g. application installed from authorized source can be performed only when the application is in production environment). It can further be used for troubleshooting purpose as well. This flag would let SDK know in which environment application is running.	Country specific
8	WalletServer_FCM_SenderId	FCM sender Id for Wallet Server	MPA specific
9	MDES_FCM_SenderId	FCM sender Id for MDES	MPA specific

10	LogLevel	Logging level like debug, info, warning, error, etc.	MPA specific

12.6 Notifications

Here below notifications are listed:

Notification	Admin/Alert Message	Push Notification / web-service / API	Description
WS→ MPA → CW-SDK	Alert	Push Notification (RNS_ID)	- Transaction Receipt
	Admin	Push Notification (RNS_ID)	- Card Suspended – Santander Initiated - Card Unsuspended – Santander Initiated - Card Deactivated – By CSR
WS→ CW-SDK	Admin	Web-service	- Card Suspended – User Initiated - Card Unsuspended – User Initiated - Card Deactivated – By User
CW-SDK→MPA	Admin	API (Call Back)	- Upgrade notification - First Tap (Double Tap Experience) - Card Activated - Card Suspended – Santander Initiated - Card Unsuspended – Santander Initiated - Card Deactivated – By CSR
MDES/CMS-D → MPA → CW-SDK	Admin	Push Notification (RNS_ID)	- Provisioning - Request Session (Replenishment) - Request Session (Delete Token) - Request Session (Reset Mobile PIN)

12.6.1 Push Notifications Handling (FCM)

This section describes how the MPA and the CW-SDK shall implement and manage the Firebase Cloud Message (FCM) push notifications.

As per Google recommendation only one FCM receiver shall be present per Android application. The MPA is the master of the FCM push notifications. The MPA:

- Performs the FCM registration,
- Receives all push notifications from the Android framework and
- Dispatches to CW-SDK related push notifications from MDES and Wallet Server sent for CW-SDK.

1. MPA does the registration with FCM using two different SENDER-IDs for Wallet Server, MDES in two different calls.
   NOTE: MPA must know theSender-IDs for the Wallet Server, MDES and other systems for which the FCM IDs are required.

2. In response to the registration with FCM, MPA receives different (registration tokens) for all Sender-IDs.
3. MPA passes the Wallet Server RNS Registration ID to CW-SDK in the createWallet API.
4. CW-SDK passes this RNS_ID to Wallet Server.
5. Wallet Server stores this Wallet Server RNS Registration ID for sending messages to the CW-SDK via the MPA in future.
6. MPA passes MDES RNS Registration ID to CW-SDK in request for MDES Registration.
7. Wallet Server passes the same MDES RNS Registration ID to MDES /VTS during registration and MDES/VTS stores it for sending messages to the CW-SDK via the MPA in future.
8. At a time when any of the Sender (Wallet Server, MDES/VTS) sends a message to MPA/CW-SDK using FCM, all notifications are received by the MPA.
9. When notification is received at MPA, the MPA also gets the information about from which Sender this notification is coming. In case the notification is not meant for MPA (for example when it is sent from the Wallet Server or MDES /VTS) then the MPA passes this notification to CW-SDK using handleNotification API. Along with the notifications, the MPA also passes the information about who is the Sender in 'from' parameter.
10. Using 'from' parameter the CW-SDK is able to identify that whether the notification is meant for CW-SDK (from=WalletServer's SenderId) or for URPay (MDES' SenderId).

12.6.2 Notification for card deletion to various systems

In the Wallet eco-system the deletion of the card (token/D-PAN) will be triggered by the Corporate Wallet SDK,. The Wallet Server processes the request and, as a result of which the card is deleted at MDES which is finally propagated to the Issuer system. On reception of the delete card event, the Issuer system sends a notification (via SMS/Email) to its customers for deletion of the card. The below matrix provides details on when the systems will be notified on card deletion.

Event	Description	Systems Notified		
		WS	MDES	VTS
Add card post Wallet Registration	User adds Card (token) in Wallet	Yes	Yes	Yes
Suspend card – By Issuer	Card is suspended by the Issuer	Yes	Yes	Yes
Un-suspend card – By Issuer	Card is un-suspended by the Issuer	Yes	Yes	Yes
Suspend card – By User	Card is suspended by the User	Yes	Yes	Yes
Un-suspend card – By User	Card is un-suspended by the User	Yes	Yes	Yes
Delete card – By Issuer	Card is deleted by the Issuer	Yes	Yes	Yes

Event	Description	Systems Notified		
		WS	MDES	VTS
Delete card – By User	Card is deleted by the User	Yes	Yes	Yes
Lock Wallet – while MDES Tap & Pay	Cards are suspended by MDES/VTS	Yes	Yes	Yes
Terminate/Deactivate/Reset Wallet – By User	User deactivates/resets the Wallet and card gets deleted.	Yes	Yes	Yes
Un-install	When user uninstalls the Wallet, the event will not be registered at the WS and hence no other system will be aware of the un-installation. The card available in the Wallet and the Wallet records are still available in the Wallet record. Also, the card in the MDES (token mapping) is not deleted. Hence, the Issuer system will also be unaware. E.g. User uninstalls wallet A having card A on device A	No	No	No
Factory Reset/Delete User Profile	Same as above	No	No	No

12.7 Authentication Level vs Use Cases

Use Case	WS Mobile Web-Service	Default Authentication Level (Tenant Configurable)	Description
Set up Corporate Wallet SDK: Register Wallet	Register Wallet	L1	
Set up Corporate Wallet SDK: Activate Wallet / Add Additional Card(s)	Digitize Cards	L1	The Authorization token used in this API is the same one used in the Register Wallet and Card List Request
Add Additional Card(s)	Card List Request	L1	
Auto recovery	Recover cards	L1	

12.8 CVM Authentication

Use Case	CDCVM / Online (Wallet Server)	Description
Unsuspend Card: by User	Online	Unsuspension of a token by the user
View Transaction History	Online	
Terminate Wallet	Online	
Tap & Pay	Local	Contactless payment in POS

12.9 Authorization Token Verification by Wallet Server

See reference [5].

12.10 Velocity Check

Velocity checks refer to the rules that can be defined when it comes to making contactless payments that provide an indication whether Android-based Cardholder Verification Method (CVM) is needed or not.

These rules are defined based on the transaction type (low value, high value), the number of them and the period of time when transactions can be made since the last authentication:

- Number of LVT transactions from last secure unlock action (#LVT)
- Number of HVT transactions from last secure unlock action (#HVT)
- Session duration for LVT (#LSD)
- Session duration for HVT (#HSD)
- Amount spent during last secure unlock action (#Amount)

In general, CDCVM or PIN input will be required if:

- A Santander country has configured PIN/CDCVM always.
- A LVT transaction is made but the LVT counter has been exceeded.
- It is the firsts HVT transaction.
- The device has been unlocked recently, remain unlocked and session has expired.
- It is a HVT and the number of HVT is higher than a limit from last unlock.
- Total amount spent since last secure unlock is greater than a limit.

Each Santander country has defined their own velocity checks which are reproduced here for tracking purposes.

General Note on Velocity Check Rules:

If the user unlocks the device using any secure unlock method to access any other app (for example, email) and thereafter tries to make a payment using the MPA, then the scenario will be handled as follows-

- The user will not be prompted for secure unlock again if he/ she tries to make the payment within configured time after making secured unlock of the device.

12.10.1 Velocity Check - UK

The velocity check rules provide the indication if Android-based Cardholder Verification Method (CVM) is needed or not.

In this section the device unlock is the CVM method. Valid device unlock methods are:

- Device pin,
- Device passcode,
- Device pattern and
- Device biometric.

These are the methods that Android considers secure unlock methods, which we can enforce through app permissions. Only Android secure unlock methods are supported, there is not configuration to enforce what it's secure and what not.

The following definitions applies:

- MPA will provide up to 3 threshold couples i.e. (Threshold amount, Currency). When tapping phone on terminal, terminal will also provide the currency, amount.
- If the currency does not belong to any of the list, transaction will be considered as HVT Transaction.
- An LVT is a transaction with amount <= £30 (threshold provided by MPA during initialization, see above).
- An HVT is a transaction with amount > £30 OR amount=0 from terminal, OR no amount provided.
- LVT and HVT velocity checks are decreased at the time of payment tap irrespective of the result of the transaction. In case of double tap scenarios the first tap shall not be considered as payment tap and it shall not decrease the LVT and HVT velocity check.

The velocity check is composed of several velocity check and a time counter listed below:

- An LVT velocity check value initially set to 3 and reset to 3 after each successful CVM.
    - This is reduced on every transaction (LVT or HVT) excluding Transport for London (TFL).
    - This is reset to initial value (3) when the CVM is successfully.
- An HVT velocity check value initially set to 2 and reset to 2 after each successful CVM.
    - This is only reduced when a HVT is made.
    - This is reset to initial value (2) when the CVM is successfully.
- A 180 second pre-arm counter during which HVT can be made.
    - This is reset when the CVM is successfully.

- This counter remains active and counting down even when the device has subsequently been locked. This means you can make a high value payment within 3 minutes of unlocking your device, even if the device is within a locked state when making the payment.

The LVT, HVT velocity check set values and the pre-arm counter set value are configurable tenant parameters.

Here below the applicable rules for the Tap & Pay velocity check. Note that the following rules assumes that the device screen is on (lit).

A TfL (Transport for London) transaction will be recognized according to following criteria:

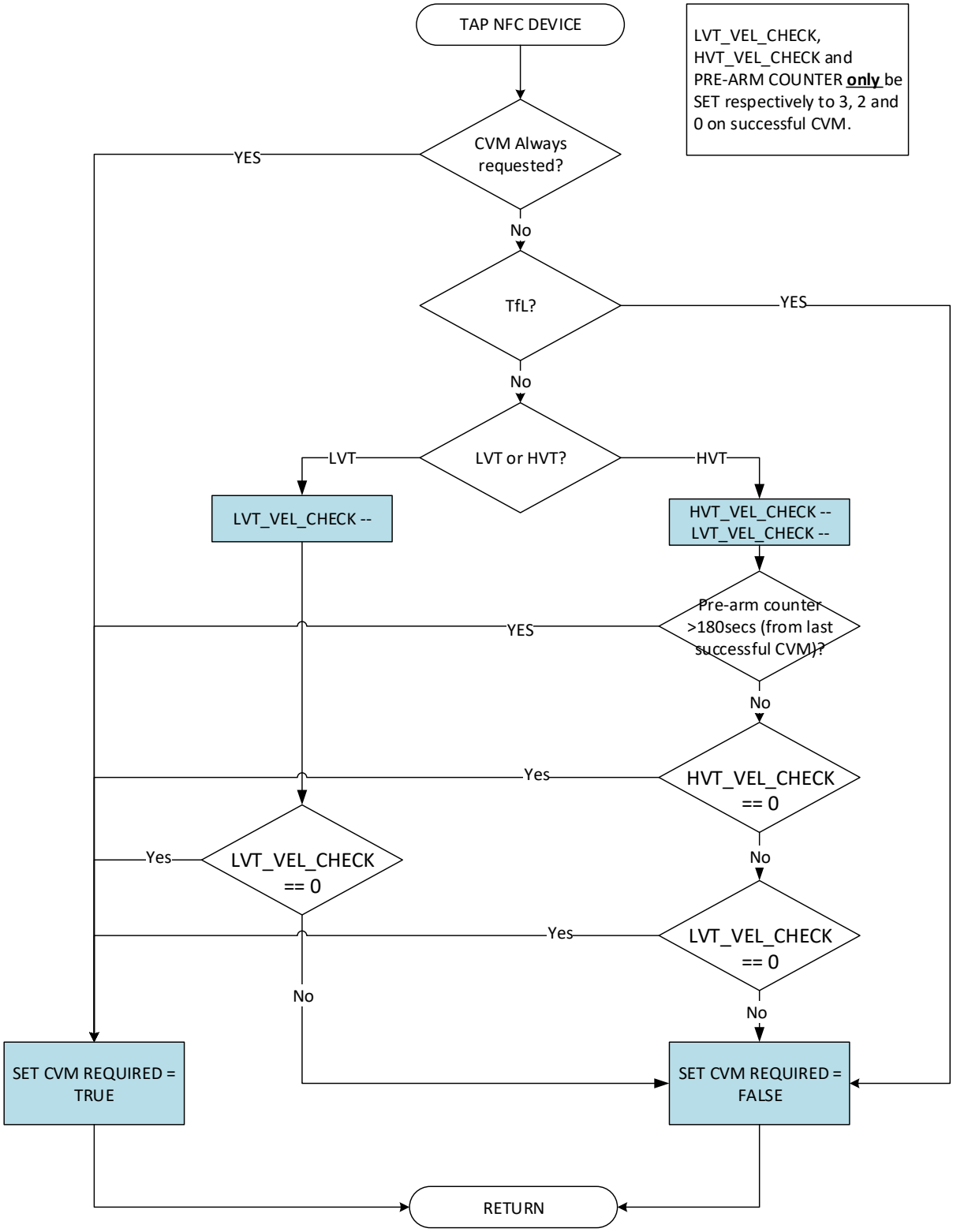
- For Mastercard: terminal amount equal to 0 (or no amount received) and MCC equals to TRANSIT, info provided by URPay SDK to CW-SDK.
- For Visa: ODA transaction detected by URPay SDK, info provided to CW-SDK.

In case the user wishes to do a tap and pay transaction with a phone which is unlocked and MPA is in foreground, or user wishes to do a tap and pay transaction with a locked or unlocked mobile phone and MPA not in foreground (In other words user does a tap and pay transaction without selecting the card for payment on the MPA), the following rules apply during the payment tap:

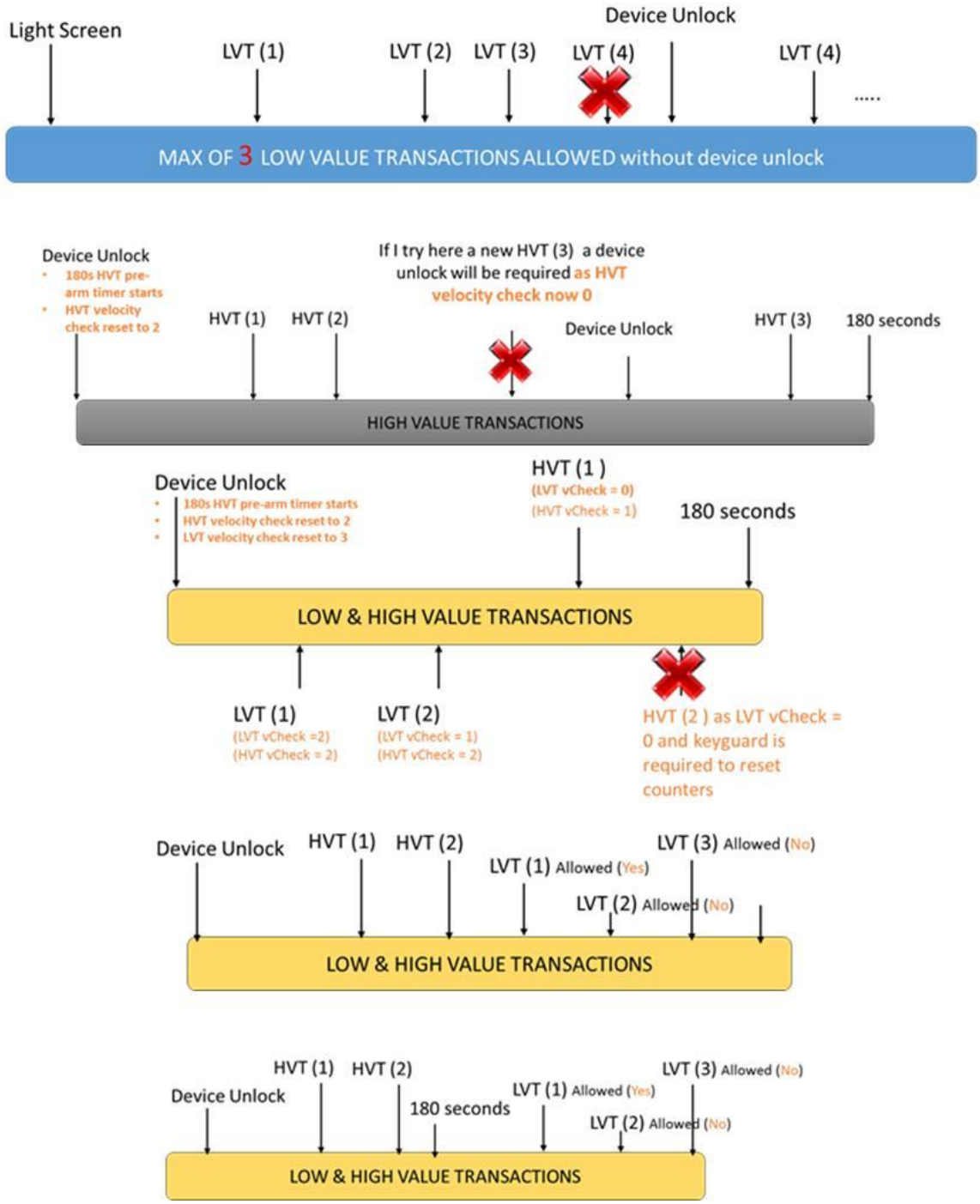
- A HVT requires no CVM when all below conditions are satisfied at the same time:
  - LVT velocity check value is above 0
  - HVT velocity check value is above 0, including:
    - Terminal amount is equal to 0 or no terminal amount received
    - And not a TfL transaction
  - It happens within 180 seconds of the CVM being entered
- A HVT requires CVM in all other cases.
- A LVT requires no CVM when:
  - LVT velocity check value is above 0.
- A LVT requires CVM in all other cases.
- A TfL transaction requires no CVM

The CVM can be always (Called “Always CVM”) requested for all transactions, this can either be a user preference or a tenant configuration.

The following flow summarizes the UK velocity check in case the user wishes to perform a Tap and Pay transaction with a phone which is unlocked and MPA is in foreground, or user wishes to do a Tap and Pay transaction with a locked or unlocked mobile phone and MPA not in foreground (In other words user does a Tap and Pay transaction without selecting the card for payment on the MPA):



The diagrams below shows several examples of velocity check:



12.10.2 Velocity Check - Brazil

The velocity check rules provide the indication if Android-based Cardholder Verification Method (CVM) is needed or not.

In this section the device unlock is the CVM method. Valid device unlock methods are:

- device pin,
- device passcode,
- device pattern and
- device biometric.

These are the methods that Android considers secure unlock methods, which we can enforce through app permissions. Only android secure unlock methods are supported, there is not configuration to enforce what it's secure and what not.

The following definition applies:

MPA will provide up to 3 threshold couples i.e. (Threshold amount, Currency). When tapping phone on terminal, terminal will also provide the currency, amount

- If the currency does not belong to any of the list, transaction will be considered as HVT Transaction.
- An LVT is a transaction with amount <= 50 Reales (threshold provided by MPA during initialization, see above).
- An HVT is a transaction with amount > 50 Reales OR amount=0 from terminal, OR no amount is provided.
- LVT and HVT velocity check are decreased at the time of payment tap irrespective of the result of the transaction. In case of double tap scenarios the first tap shall not be considered as payment tap and it shall not decrease the LVT and HVT velocity check.

The velocity check is composed of several velocity check and a time counter listed below:

- A LVT velocity check value initially set to 3 and reset to 3 after each successful CVM.
    - This is reduced on every transaction (LVT or HVT)
    - This is reset to initial value (3) when the CVM is successfully.
- A HVT velocity check value initially set to 1 and reset to 1 after each successful CVM.
    - This is only reduced when a HVT is made.
    - This is reset to initial value (1) when the CVM is successfully.
- A 30 second pre-arm counter during which LVT and HVT can be made.
    - This is reset when the CVM is successfully.
    - This counter remains active and counting down even when the device has subsequently been locked. This means you can make a low value payment or a high value payment within the duration agreed between issuer and Mastercard scheme (on Jan 20th 2017, current waiver allows any value up to 180 sec) of unlocking your device, even if the device is within a locked state when making the payment.

The LVT, HVT velocity check set values and the pre-arm counter set value are configurable tenant parameters.

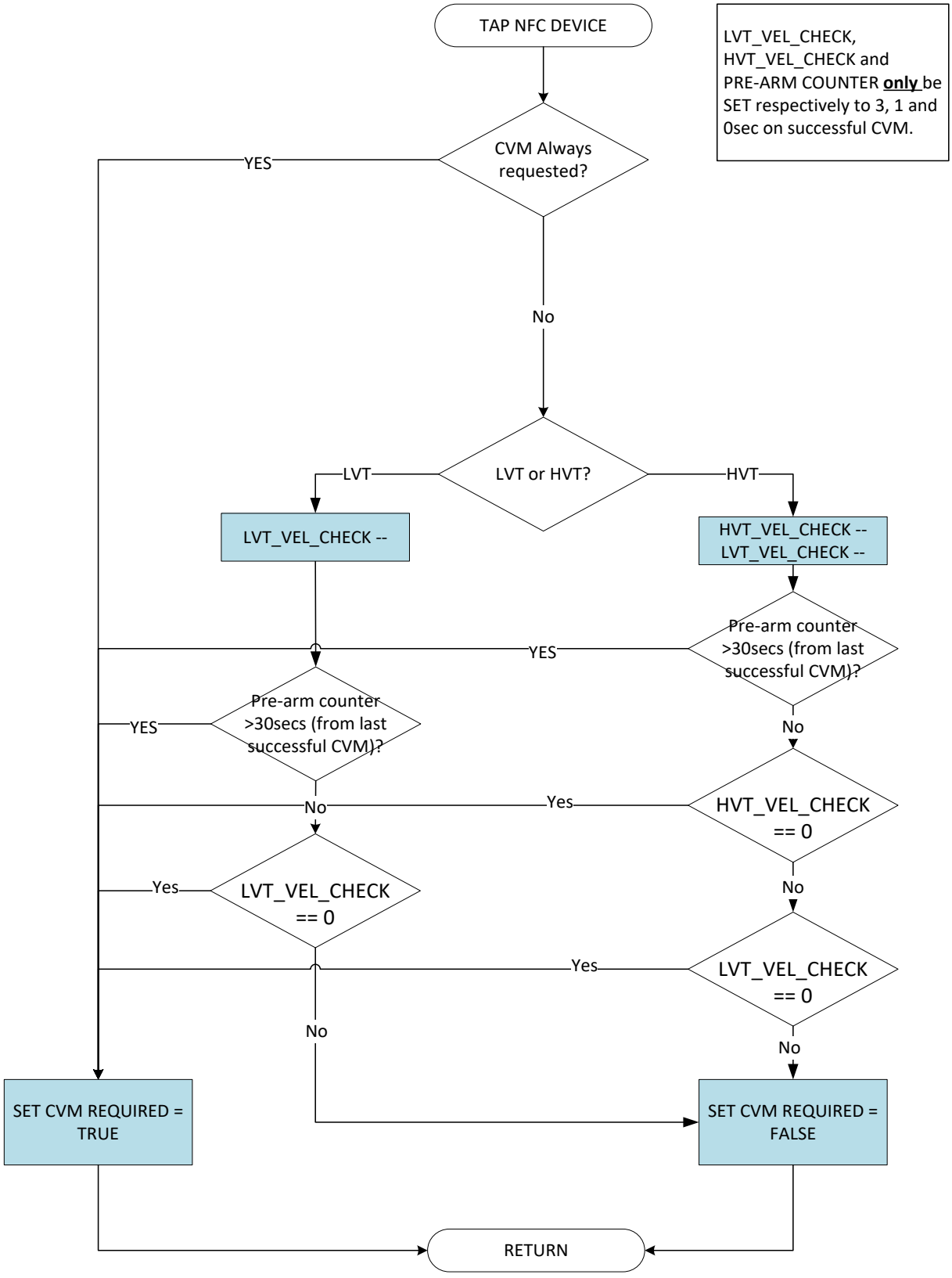
Here below the applicable rules for the Tap & Pay velocity check. Note that the following rules assumes that the device screen is on (lit).

In case the user wishes to do a tap and pay transaction with a phone which is unlocked and MPA is in foreground, or user wishes to do a tap and pay transaction with a locked or unlocked mobile phone and MPA not in foreground (In other words user does a tap and pay transaction without selecting the card for payment on the MPA), the following rules apply during the payment tap:

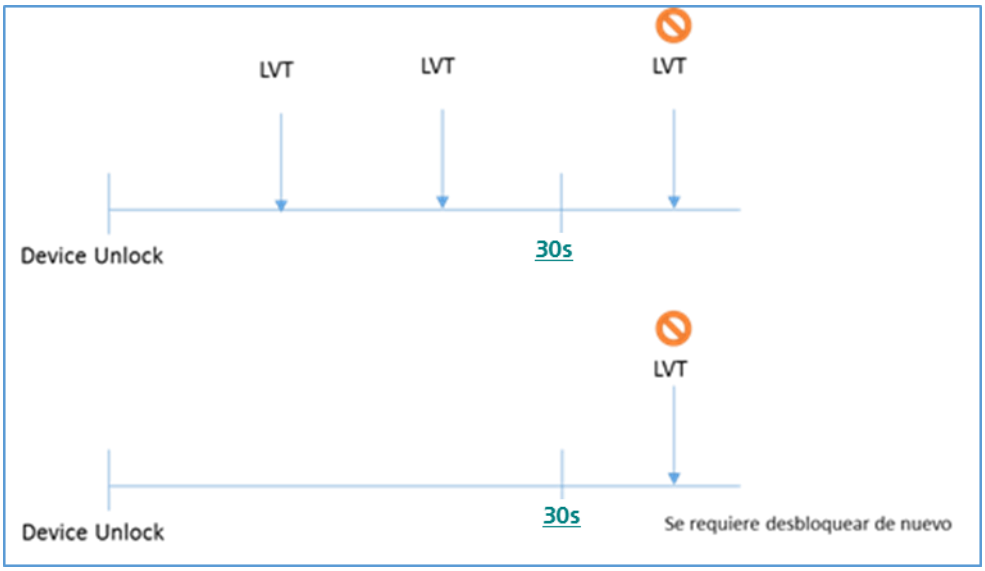
- A HVT requires no CVM when all below conditions are satisfied at the same time:
    - LVT velocity check value is above 0.
    - HVT velocity check value is above 0, including:
        - No terminal amount received.
- It happens within 30 seconds of the CVM being entered.
- A HVT requires CVM in all other cases.
- A LVT requires no CVM when:
    - LVT velocity check value is above 0
    - It happens within 30 seconds of the CVM being entered.
- A LVT requires CVM in all other cases.

The CVM can be always (called “Always CVM”) requested for all transactions, this can either be a user preference or a tenant configuration.

The following flow summarize the Brazil velocity check in case the user wishes to do a tap and pay transaction with a phone which is unlocked and MPA is in foreground, or user wishes to do a tap and pay transaction with a locked or unlocked mobile phone and MPA not in foreground (In other words user does a tap and pay transaction without selecting the card for payment on the MPA):



The Diagram below shows 2 examples of velocity check:



12.10.3 Velocity Check – Chile

Chile will need to support secure unlock for their transactions since Chile has mandatory requirements of entering PIN @ POS, i.e. Hence the velocity check rules for Chile would be governed by Terminal and velocity check rules won't be applied on the CW-SDK.

Support of unsecure unlock is not possible, due to security constraint from the URPay.

12.10.4 Velocity Check – Mexico

Mexico will request user to authenticate with CDCVM for all the transactions, hence velocity check rules won't be applied on the SDK.

12.10.5 Velocity Check – Portugal

Portugal will request user to authenticate with CDCVM for all the transactions, hence velocity check rules won't be applied on the SDK.

12.11 Fraud Information and Reason Code Association (MDES)

12.11.1 In release 1.0.1

Wallet Server will provide MDES with the following static hardcoded decisioningData in all digitize requests:

```
"decisioningData": {
      "recommendation": "APPROVED",
      "recommendationAlgorithmVersion": "01",
      "deviceScore": "5",
      "accountScore": "5",
```

```
      "recommendationReasons": [
         "GOOD_ACTIVITY_HISTORY"
      ]
}
```

These decisioning data are not managed (i.e. enabled/disabled) with a tenant parameter.

12.11.2 In release 1.1 and beyond

A device fingerprint is generated in CW-SDK and collected by the Wallet Server during device fingerprint check, as described in UC named “Synchronize Corporate Wallet SDK - Wallet Server (Checks for Mobile Client Communication)”. Digitization will be possible only if such check does pass first.

Then, if this check passes, in order to allow Santander fraud management to decide digitization approval, the Wallet Server will fill the “recommendation reasons” of the digitize request towards MDES. Those reasons will then be provided by MDES to the Santander issuer IT as “wallet provider reason codes” within the TAR message.

These decisioning data are not managed (i.e. enabled/disabled) by tenant parameter and all countries will receive the same decisioning data set described in this section.

The tables below indicate which “reason code” will be provided according to which logic. Some reason codes will be generated by Wallet Server based on data fetched by CW-SDK from the Android device, while others will be based on information known to the server database itself.

With the introduction of Android 10, some of the identifiers obtained through regular Android permissions, no longer became available, having to change the mechanism used to generate the DFP (Device Fingerprint).

In the table below, the items contained in the column “Fraud Information” shall be collected by the CW-SDK from the Android device.

Group	Fraud Information	Android permission
Device/OS	Device locale	No specific permission needed

Optionally, MPA can request additional permissions to the user to capture the email addresses.

The Wallet Server will send the digitization request to MDES with 0, 1 or more “reason codes” matching the logic in the table below. The group named “Wallet” corresponds to an information that the Wallet Server will generate from its own database, without fetching Android-related information from the CW-SDK.

Please note that some parameters are no longer collected by CW-SDK due to restrictions introduced in Android 10. However, the implementation is still valid, given those parameters can be obtained from MPA and passed in the calls to the CW-SDK APIs.

Group	Fraud Information	Will be stored in WS (yes/ no)	Logic to raise a "reason code" and to included it in the digitization request to MDES	Reason Code	BPMS code	Bit Number in DE124 subfield 8
Accounts	Account (Gmail Address)	Yes	Store email (E1) + timestamp at register. Fetch email (E2) at digitization. Rules: A) If (E2 is different from E1) && if (timestamp < 3 months), then (raise a reason code) and log (E2 is different from E1, timestamp < 3 months) B) Update email to (E2) + timestamp Note: with the reason code we communicate any change in the email.WS Log helps in case of issues. Note2: email address is no longer collected by CW-SDK and can be fet	Account Recently Changed Or Unable to Assess	04	4
Telephony	MNO ISO Country	Yes	Store country (C1) at register. Fetch country (C2) at digitization. A) If C2 is different from C1 then raise a "reason code" and log (C2 is different from C1) B) Update country to (E2) + timestamp Note: with the reason code we communicate any	Outside Home Territory Or Unable to Assess	0E	14

			change in the country, and we always keep the latest one only if this digitization, and thus additional authentication, was successful. WS Log helps in case of issues.			
Telephony	SIM Serial Number	Yes	Store SIM ID (S1) + timestamp at register. Fetch SIM ID (S2) at digitization. Rules: A) If (S2 is different from S1) then (raise a reason code) and log (S2 is different from S1) B) Update SIM ID to (S2) + timestamp Note: with the reason code we communicate any change in the SIM ID, and we always keep the latest one only if this digitization, and thus additional authentication, was successful. WS Log helps in case of issues.	Account Too New Or Unable to Assess	02	2
Device/OS	Device locale	Yes	Store locale (L1) + timestamp at register. Fetch locale (L2) at digitization. Rules: A) If (L2 is different from L1) then (raise a reason code)and log (L2 is different from L1) B) Update Locale	Account Too New Since Launch Or Unable to Assess	01	1

			to (L2) Note: with the reason code we communicate any change in the Locale, and we always keep the latest one only if this digitization, and thus additional authentication, was successful. WS Log helps in case of issues.			
Wallet	-	Yes	If the WalletID has at least 1 suspended token, then raise reason code.	Has suspended tokens	07	7
Wallet	-	Yes	If the WalletID waited a long time (180 days) between register & digitize, then raise a reason code. Note: it should not happen in “digital by default” scenario	Inactive Account	06	6
Wallet	-	Yes	If a new userID is replacing a previous one for same device, then raise a reason code.	Too many different cardholders	0B	11
Wallet	-	No	If device details cannot be fetched at time of digitization to compare with data at time of registration (for example consumer changed application permissions)	Unable to assess	0F	15

Wallet server will be calculating recommendation as per the specification provided by MDES and as a general rule if one or more reason codes suggest a yellow path (require add auth), the default recommendation from Wallet Server will be yellow path (require add auth).

The digitize request will not include device score or account score decisioning data.

12.12 Fraud Information and encRiskDataInfo (VTS)

The following codes for user ID&V risk data will be generated as part of response to TAR message while digitizing a Visa card, the logic for generation of the reason codes will be as follows.

As against MDES these codes aren't to be calculated for sending recommendation but just to compile the values and to be sent in response to PanEnrollments. Please refer to Visa API Specification for more details.

Other Wallet Providers' Reason Code	Attribute	Required to be produced by WS	Logic to generate the reason code
A0	Wallet Pan Age	No	NA
A4	Wallet Account first created	Yes	Consideration would be the new wallet ID creation for a new device. This reason code will be generated at time of first card digitization.
A5	Last Account Change	Yes	A new device id for a specific user id will be considered as last account change, value will be sent as Boolean between digitization attempts.
AA	Account activity	Yes	Waiting time too long between registration and digitization for a particular wallet id would generate this reason code,
A9	Suspended card in the Secure Element/Account	Yes	For WS, we will count number of suspended token for a specific wallet ID at time of digitization.
AB	Device Lost mode	No	NA
A6	Provisioning Attempts	Yes	Number of provisioning attempts during last 24 hours for a specific wallet ID
A7	Distinct Cardholder names	Yes	If a new user ID is replacing previous one for the same device fingerprint at time of digitization, we shall generate this reason code.
A8	Device or Wallet Account Country	Yes	MNO ISO Country code to be captured during digitization, if during subsequent digitization the country is different, raise this reason code.

A1	Wallet Name Match	No	NA
A2	Account to device binding age	Yes	Number of days between wallet id creation and current date,
A3	User Account first created	Yes	Number of days since first create wallet occurred for the user id.
AC	Transactions in last 12 Months	Yes	Count of number of VTS transactions for this use rid.
AD	Active Tokens	Yes	Count of number of active tokens at time of digitization for that user id.
AF	Tokens on All devices	Yes	Number of tokens irrespective of status for that user id.
AE	Devices Same User Id with token	No	NA

Note that the wallet cannot contain 2 Visa cards with 2 different cardholder names considering the this is a closed wallet (i.e. no other cards except Santander cards can be digitized) provisioned with cards from a particular user ID.

12.13 MPA Re-install, Upgrade and Factory Reset

The following table provides information on the Wallet Server actions and changes on identifiers (Device Fingerprint, New Wallet ID assigned, New Wallet Reference ID, New PaymentAppInstanceID).

Use Case	Wallet Server Actions	Identifiers
User Deletes and Re-installs MPA	After the introduction of Android 10, non-privileged applications lost access to the non-resettable device identifiers such as the IMEI or serial number which were used earlier to generate the Device Fingerprint and recognize app re-installs from the same device. As a result, when a user uninstalls and re-install the same application in the same device, Device Fingerprint will be different and Wallet server will not be able to detect and terminate the previous instance and will create a new one. This will lead to the creation of dormant app instance records on the Wallet Server with its associated payment tokens that should be removed periodically. Wallet Server finds no wallet record with the same device fingerprint	Changes on Device Fingerprint. - New Wallet ID assigned - New Wallet Reference ID - New PaymentAppInstanceID

	• The Wallet Server creates a new wallet record. • New registration is initiated • Customer needs to re-select the cards to be re-digitized	
Delete and Re-install due to Mobile Factory Reset OR Restore MPA on the same device	Wallet Server finds no wallet record with the same device fingerprint • The Wallet Server creates a new wallet record. • New registration is initiated • Customer needs to re-select the cards to be re-digitized **NOTE: There will be dormant app instance records on the Wallet Server with its associated payment tokens that should be removed. Wallet Server terminates the old wallet record after X number of days without any hit from a CW-SDK.**	Device Fingerprint will change: - New Wallet ID assigned - New Wallet Reference ID - New PaymentAppInstanceID
MPA Upgrade	Wallet Server does nothing, the wallet record is kept. • No registration, digitization or replenishment is needed	No changes on Device Fingerprint - Wallet ID is kept
Restore on different Device	Wallet Server finds no wallet record with the same device fingerprint • The Wallet Server creates a new wallet record. • New registration is initiated. • Customer needs to re-select the cards to be re-digitized. **NOTE: Santander Corporate Wallet incorporates a utility to terminate and delete old wallet records after a period of time. Use of this utility must be configured and triggered by the issuer.**	Device Fingerprint is brand new: - New Wallet ID assigned - New Wallet Reference ID - New PaymentAppInstanceID

13 Functional Specification Approval

I have carefully assessed this Functional Specification document. This document has been completed in accordance with the requirements and clearly reflects the specification of the project.

Issuer	Mastercard
Signature	Signature
Name	Name
Title	Title
Date	Date