

NFC i mobiltelefoner för kollektivtrafikresor.

Lägesbild och möjliga vägval för svensk kollektivtrafik.

Samtrafiken · Trafikförvaltningen, Region Stockholm (SL) · Västtrafik ·
Skånetrafiken · Trafik och samhälle, Region Uppsala (UL), Östgöratrafiken

Samtrafiken

NFC i mobiltelefoner för kollektivtrafikresor

Lägesbild och möjliga vägval för svensk kollektivtrafik

Samtrafiken · Trafikförvaltningen, Region Stockholm (SL) · Västtrafik · Skånetrafiken · Trafik och samhälle, Region Uppsala (UL), Östgöratrafiken

Innehåll

Sammanfattning 2
1. Uppdrag och avgränsning 2
2. Användningsfall och teknikmiljö 3
Två skilda användningsfall 3
Olika utgångslägen 3
Appbaserad HCE och mobilens plånbok 3
3. Lösningar för ID-baserat resande 3
MTS7 eller EMV som mobilt ID via HCE 4
EMV-baserat ID i mobilens plånbok 4
4. Lösningar för enskilda BoB-biljetter 5
5. Översikt över lösningsvägar 5
6. Frågor som styr ett lokalt vägval 6
Befintlig acceptans och investering 6
Befintlig teknik och investering 6
Utgivning av unika identifierare 6
Interoperabilitet utan gemensamt teknikval 7
7. Slutsatser och fortsatt kunskapsbehov 7
Bilagor 8
Begrepp och förkortningar 8

Sammanfattning

Denna utredning beskriver hur närfältskommunikation, NFC (Near Field Communication), i Androidtelefoner och iPhone kan användas för att presentera en biljett eller en identifierare för kontobaserat resande. Den riktar sig till kollektivtrafikbranschen, särskilt de regionala kollektivtrafikmyndigheterna. Tekniken erbjuder flera genomförbara vägar, men deras lämplighet beror på användningsfall, vald teknik för ID-hantering, validatorer, skillnader mellan mobiltillverkare, och samverkan mellan trafikoperatörer. Därför lämnar rapporten ingen gemensam teknikrekommendation för hela branschen.

Två användningsfall behöver bedömas var för sig. Vid ID-baserat resande presenterar mobilen ett rese-ID enligt exempelvis MTS7, BoB-standardens protokoll för rese-ID, eller EMV, en kortstandard som här används som rese-ID. För enskilda biljetter enligt BoB kan mobilen i stället presentera själva biljettinformationen, enligt BOB standarens MTS1-definition.

Det finns olika tekniska sätt att låta mobilen presentera information via NFC. Med HCE (Host Card Emulation) kan en app emulera ett kontaktlöst kort och kommunicera direkt med biljettläsaren. HCE kan användas både för att presentera ett rese-ID och för att överföra biljettinformation enligt MTS1. För ID-baserat resande finns även möjligheten att använda mobilens plånbok, exempelvis Apple Wallet eller Google Wallet, för att presentera rese-ID. Lösningarna har olika tekniska förutsättningar och ger också olika möjligheter att utforma resenärens användarupplevelse.

SL undersöker en plånbokslösning för sitt EMV-baserade rese-ID och ett appbaserat HCE-spår med särskild relevans för presentation av enskilda biljetter. I sin utredning ”Utredning - Införande av NFC-validering i mobila plattformar” (dok. id 3573126, se bilasga 3) beskriver SL även två varianter inom HCE: direkt överföring av biljettdata och användning av ett rese-ID mot ett bakomliggande biljettsystem. Spåren ska därför inte likställas med ett entydigt teknikval för all biljettvalidering. Även vid olika lokala lösningar behöver resor och biljettkontroll över länsgränser fungera.

1. Uppdrag och avgränsning

Uppdraget för denna utredning är att kartlägga vad Apple och Google tillåter och stödjer för NFC-baserad biljettpresentation och identifiering i kollektivtrafik. Utredningen gäller Android och iOS samt digitala enkel- och periodbiljetter och ID-baserat resande med BoB MTS7 eller EMV closed loop. Den behandlar inte ett generellt byte av biljettsystem eller införande av en gemensam nationell lösning.

Arbetet genomfördes i fyra steg. Först gjordes en teknisk studie av användningsfall, kortprotokoll och mobilplattformarnas tillgängliga funktioner (se bilaga 1). Med studien som grund ställde utredningen frågor till Apple och Google och analyserade deras skriftliga svar (se bilaga 2). Därefter vägdes SL:s separata utredning om NFC-validering, inklusive dess workshop, in som ett konkret exempel på en aktörs förutsättningar (se bilagorna 3–4). Bedömningarna i rapporten är en lägesbild; genomförbarhet, kostnad och användarupplevelse behöver prövas i respektive miljö.

2. Användningsfall och teknikmiljö

Två skilda användningsfall

Vid ID-baserat resande presenterar mobilen en identifierare, till exempel enligt BoB MTS7 eller EMV closed loop. Ett bakomliggande system avgör vilka biljetter eller rättigheter som är knutna till ID:t. En enskild biljett enligt BoB MTS1 innehåller däremot biljettdata som kan verifieras vid validering. Om den presenteras via NFC är telefonen bärare av själva biljetten, på motsvarande sätt som Aztec-koden är det vid optisk avläsning. Kraven på protokoll, säkerhet och samverkan skiljer sig mellan fallen.

Olika utgångslägen

MTS7 är ett BoB-protokoll för reseidentifiering. Identifieraren innehåller uppgift om vilken deltagare som har utfärdat den. Varje utgivare kan därför använda egna lokala nummer utan att identifierarna krockar med andra deltagares. SL:s befintliga kortlösning bygger i stället på EMV Kernel 5 i ett slutet system (closed loop), med egen kortutgivning och läsare som redan kan hantera denna kortteknik. Kernel 5 anger vilken variant av EMV-protokollet som används. SL:s utredning är uttryckligen ett beslutsunderlag för ett virtuellt SL-kort, med särskilda mål för spärrmiljö, optisk avläsning och driftstart. Dessa förutsättningar kan inte direkt överföras till andra trafikområden.

Appbaserad HCE och mobilens plånbok

Vid appbaserad HCE ansvarar trafikaktören för hur den egna appen presenterar biljetten eller rese-ID:t när telefonen hålls mot en läsare. Trafikaktören utvecklar och ansvarar då själv för hur appen presenterar biljetten eller rese-ID:t. För ID-baserat resande kan i stället en plånbokslösning användas, där rese-ID presenteras via Apple Wallet eller Google Wallet. Plånboken sköter då presentationen för läsaren och delar av användarupplevelsen. Skillnaden för resenären är framför allt hur rese-ID väljs och om appen behöver öppnas eller telefonen låsas upp. Villkoren varierar mellan iPhone och Android.

3. Lösningar för ID-baserat resande

Apple och Google lyfter fram sina respektive plånbokslösningar som en väg till smidig mobil validering. Båda erbjuder också stöd för appbaserad HCE, men det får mindre utrymme i deras svar. Google varnar uttryckligen för möjliga användarproblem med appbaserad HCE.[SÖ1][SÖ2] Apples svar fokuserar på plånbokslösningen. De Apple-representanter som besvarade utredningens frågor kunde inte besvara de detaljerade HCE-frågorna; de hanteras av en annan avdelning inom Apple. Det innebär inte att HCE saknar stöd: Apple erbjuder HCE i EES för bland annat sluten kollektivtrafik, under förutsättning att appen godkänns och uppfyller Apples villkor. För aktörer som använder MTS7 är skillnaden att de kan bygga stöd för MTS7 via appbaserad HCE, medan stöd i en plånbok kräver att Apple eller Google erbjuder motsvarande funktion; Google bedömer MTS7-stöd som mycket osannolikt inom överskådlig tid, och Apple har inte gett något besked om saken.

MTS7 eller EMV som mobilt ID via HCE

Den tekniska studien bedömer att HCE kan användas för att emulera MTS7 och EMV closed loop i en egen app på Android och, med Apples HCE-behörighet, på iPhone inom Europeiska ekonomiska samarbetsområdet (EES). SL uppger i sin workshopbilaga att den fått Apples HCE-behörighet. Appbaserad HCE på iPhone ger inte samma möjlighet som plånbokens expressläge att validera utan upplåsning. Resenären kan behöva bekräfta sin identitet och välja vilken app som ska användas för kontaktlösa transaktioner. Om appen inte är vald som standard kan den behöva öppnas och förberedas för avläsning inom ett begränsat tidsfönster. Hur detta fungerar i praktiken behöver provas. Apples representanter kunde inte bekräfta alla tekniska antaganden om HCE och har därmed inte bekräftat alla tekniska antaganden. För iPhone innebär Apples HCE-behörighet plattformsstöd för godkända appar inom EES, men MTS7-lösningen har inte verifierats praktiskt.

På Android tillåter Google appbaserad HCE, men varnar för problem när resenären också använder Google Wallet. Telefonen behöver välja rätt app när den hålls mot en NFC-läsare. Det kan påverka om trafikaktörens app aktiveras som avsett. Därför behöver automatisk aktivering och användning vid skärmsläckt läge provas på verkliga telefoner, även när Google Wallet är installerat och används.

EMV-baserat ID i mobilens plånbok

Apple beskriver en serverbaserad lösning, Truth on Server, där ett rese-ID i plånboken identifierar ett konto och biljetterna hanteras i backend. Apple föreslår white-label EMV: ett EMV-baserat kort för trafikaktörens eget slutna system, med utgångspunkt i SL:s befintliga EMV-miljö. Plånbokens expressläge, Express Mode, kan låta resenären validera utan att låsa upp telefonen, om funktionen aktiverats, och kan fungera en begränsad tid med reservström när den ordinarie batteriladdningen är

slut. Den funktionen kallas Power Reserve och är inte samma sak som telefonens vanliga strömsparläge. En integration kräver bland annat läsarberedskap, integrationspartner, backend- och apputveckling. Apples kommersiella villkor och den exakta integrationsvägen behöver klargöras i fortsatt dialog.

Google rekommenderar Google Wallet för kontobaserat resande och uppger stöd för EMV private label, som motsvarar EMV för ett eget slutet system, samt MIFARE-baserade tokens, som bygger på en annan kortteknik. Google bedömer stöd för MTS7 som mycket osannolikt inom överskådlig tid. Enligt Google är dess programmeringsgränssnitt (API) för plånboken kostnadsfria att använda. Det innebär inte att kortutgivning, integrationspartner, läsarändringar eller drift är kostnadsfria. För att kunna validera utan att låsa upp telefonen behöver användaren aktivera en särskild inställning.

4. Lösningar för enskilda BoB-biljetter

För enskilda BoB-biljetter skisserar den första studien MTS1 över NFC: en redan signerad biljett skulle överföras från telefonen till läsaren. Denna variant kan utvecklas med HCE i en app och återanvända delar av MTS1-utgivningen. NFC-transporten är ännu inte specificerad som en färdig BoB-lösning och förutsätter ändringar i läsarna samt prövning av skydd mot kopiering och återspelning. Google framhåller fortsatt optisk biljettpresentation för enkelbiljetter. Denna utredning söker emellertid NFC-alternativ till befintlig Aztec-avläsning; optisk avläsning ingår därför som nuläge och jämförelsepunkt, inte som ett av de undersökta NFC-alternativen.

5. Översikt över lösningsvägar

Lösningsväg	Användningsfall	Viktig förutsättning
ID via MTS7 i egen app	ID-baserat resande	Kräver appbaserad HCE. MTS7-stöd finns i befintliga läsare där detta redan används; funktion och användarflöde behöver verifieras på iPhone och Android.
ID via EMV i egen app	ID-baserat resande	Kan använda läsare som redan stöder EMV. Hantering av säkerhetsnycklar och mobilens kortemulering behöver verifieras.
EMV-ID i Wallet	ID-baserat resande	Kräver EMV-stöd i läsare och system samt administration av EMV-ID:n och plånboksintegration.
BoB MTS1 via NFC i egen app	Enskilda biljetter	Förslag till överföring av signerad biljett. NFC-transport, läsarstöd

		och säkerhet behöver specificeras och prövas.

Tabellen visar de undersökta NFC-vägarna, inte en rangordning. En aktör kan behöva flera parallella sätt att visa en biljett eller identifiera resrätten.

6. Frågor som styr ett lokalt vägval

Befintlig acceptans och investering

Aktörer som redan har EMV-läsare och relevant bakomliggande system kan bygga vidare på den infrastrukturen för en EMV-baserad plånbokslösning. För aktörer som använder MTS7 innebär en sådan lösning att EMV-ID:n behöver administreras och provisioneras, och att läsarna behöver kunna läsa EMV. Det behöver utredas om befintliga läsare kan få sådant stöd genom programvara och certifiering eller om de måste bytas. Underlaget fastställer inte om appbaserad HCE medför avgifter till Apple eller Google. Google uppger att Wallet-API:erna är kostnadsfria, men kostnader för exempelvis kortutgivning, integration, läsaranpassning och drift kan tillkomma. Apples kommersiella villkor, inklusive eventuella provisioner eller krav på integrationspartner, behöver klargöras. SL:s PM prioriterar ett första HCE-spår i den egna appen och beskriver både en biljettpayload via NFC och ett ID som slås upp i backend. Parallellt undersöker SL Wallet för ett digitalt EMV-baserat rese-ID. Det är en lokal bedömning utifrån SL:s mål, läsare och tidplan.

Om MTS7 behålls parallellt med EMV behöver aktören även administrera och förvalta två tekniker. Om MTS7 ersätts påverkas befintliga MTS7-kort och den utrustning som används för dem. De framtida förvaltningsbehoven för de olika alternativen är inte fastställda i detta underlag och behöver utredas.

Befintlig teknik och investering

Förutsättningarna för de olika lösningarna beror i hög grad på befintliga läsare, biljettsystem och appar. Appbaserad HCE gör det möjligt att använda MTS7 för att presentera ett rese-ID från mobilen och därmed bygga vidare på befintlig MTS7-infrastruktur. HCE kan även användas med EMV-baserade rese-ID och för att presentera enskilda biljetter enligt MTS1. Förutsättningarna för HCE skiljer sig mellan iPhone och Android och behöver verifieras i den aktuella miljön.

Utgivning av unika identifierare

Om flera utgivare använder ett gemensamt EMV-nummerutrymme måste de avtala om hur unika nummer tilldelas, vem som ansvarar för dem och hur de hanteras över tid. En möjlig modell är att reservera olika nummerintervall; en annan är en gemensam utgivningsfunktion. Kostnad och styrning behöver utredas. Med MTS7 ingår utgivaren i identifieringsstrukturen, vilket gör att deltagarna kan utfärda rese-ID självständigt. För en plånbokslösning är det fortfarande en öppen arkitekturfråga hur

rese-ID kopplas till rätt konto och eventuell fysisk kortidentifierare samt hur detta fungerar mellan olika utgivare.

Interoperabilitet utan gemensamt teknikval

Branschen behöver inte välja en enda mobilteknik för att värna interoperabilitet. Men om resenären ska kunna använda samma biljett eller rese-ID över länsgränser måste berörda parter komma överens om acceptans, verifiering, uppslag, kontroll och ansvar för utgivning. Flera parallella korttekniker kan ge extra krav på läsare och kontrollutrustning. Denna konsekvens bör redovisas för varje tillämpning, även när lokala införanden sker oberoende av varandra. En MTS1-biljett som överförs via NFC kan läsas i ett annat trafikområde om validatorerna där har stöd för MTS1 över NFC; annars kan den inte läsas där.

7. Slutsatser och fortsatt kunskapsbehov

För ID-baserat resande kan appbaserad HCE användas för att presentera ett MTS7- eller EMV-baserat rese-ID. Plattformarnas plånbokslösningar ger tydligare stöd för EMV-baserade identifierare. För enskilda BoB-biljetter är MTS1 över NFC med HCE ett utvecklingsspår som kräver specificering och provning. Användningsfallen behöver därför bedömas var för sig.

Apple och Google framhåller sina plånbokslösningar för en smidig mobilupplevelse, medan appbaserad HCE inte framstår som deras prioriterade väg för framtida kollektivtrafikintegrationer. Google avråder uttryckligen från appbaserad HCE på grund av möjliga kundproblem, medan Apple har hänvisat detaljerade HCE-frågor till ett separat utvecklarteam. Detta är en bedömning av plattformarnas inriktning, inte ett besked om att HCE ska upphöra. Ett EMV-spår löser inte automatiskt frågorna om läsarstöd och utgivning av rese-ID mellan flera organisationer.

Aktörernas befintliga teknikval, framför allt MTS7 respektive EMV, innebär att samma införandeväg inte är lika lämplig för alla. Innan en organisation väljer väg behöver den definiera om målet är ett mobilt rese-ID, presentation av enskilda biljetter eller båda och inventera läsare, kontrollutrustning, utgivning och bakomliggande biljettsystem. Vidare behövs praktiska tester av appbaserad HCE på båda plattformarna, klarhet om plånboksintegrationens villkor och provning av resor mellan trafikområden. Utredningen beskriver möjliga vägar och deras beroenden, men ger inte grund för att rekommendera en och samma införandeväg för alla regionala kollektivtrafikmyndigheter.

SL har också framfört att den fortsatta utformningen av MTS-specifikationen inom BoB-standarden bör möjliggöra dubbelriktad kommunikation mellan mobil och validator, så att validatorn kan ge resenären direkt återkoppling och stödja framtida

funktioner. SL erbjuder sig att ta fram ett designförslag för fortsatt dialog med övriga RKM:er.

Bilagor

Bilaga 1. Teknisk förstudie om NFC-validering
NFC_prestudy_v1.0

Bilaga 2. Svar från Apple och Google på frågorna i förstudien
NFC_Apple_Google_responses

Bilaga 3. SL utredning om införande av NFC-validering i mobila plattformar
Dokument-ID 3573126, version 1.

Bilaga 4. SL workshopunderlag
SL *Bilaga 1 – Sammanställning från Workshop*, dokument-ID 3573131, version 1.

Begrepp och förkortningar

NFC – Near Field Communication, närfältskommunikation mellan exempelvis en telefon och en kortläsare.

HCE – Host Card Emulation, en teknik där en app i telefonen uppträder som ett kontaktlöst kort.

BoB – branschens standarder för biljettdistribution och biljettvisering. MTS1 beskriver digitala biljetter och MTS7 protokollet för reseID.

EMV – kortstandard ursprungligen utvecklade för betalningar. I denna rapport avses closed loop, ett slutet system där kortet används som reseidentifierare.

Validator – utrustning som läser en biljett eller identifierare och kontrollerar rätten att resa.

Plånbok – mobilens funktion för digitala kort, exempelvis Apple Wallet och Google Wallet.

Interoperabilitet – att biljetter, identifierare och system kan fungera tillsammans mellan olika trafikaktörer.

Samtrafiken



REPORT

NFC Validation for Transit Ticketing – Technical Prestudy

Author: David Looberger, david.looberger@stockholmcode.se

Kund: Samtrafiken i Sverige AB

Datum: 2026-03-13

Version: 1.0

Status: Förstudie

Author: David Looberger

Table of Contents

1 Purpose and Scope 5
1.1 Purpose 5
1.2 Use Case 5
1.3 Regulatory Context 6
1.4 Scope 6
2 Background 7
2.1 BoB Background 7
3 Existing Samtrafiken Card Types 8
3.1 MTS7 – Samtrafiken Proprietary Travel Card Application 8
3.2 EMV Kernel 5 Transit Card 10
4 Platform Technical Capabilities 11
4.1 Android HCE (HostApduService) 11
4.2 iOS HCE for EEA (CardSession, iOS 17.4+) 16
4.3 iOS NFC & SE Platform (CredentialSession, iOS 18.1+) 22
4.4 Apple Wallet Express Transit (PassKit) 28
4.5 Google Wallet Smart Tap 32
4.6 Native Wallet Provisioning (Apple Wallet / Google Wallet) 36
4.7 MTS1-over-NFC: Ticket Presentation via NFC Push 39
5 Recommendation 50
5.1 Recommended main option: iOS HCE EEA + Android HCE 50
5.2 Recommendation: MTS1-over-NFC (for operators already issuing MTS1 Aztec barcodes) 51
6 Questions for Apple 53
This document represents a joint initiative by public transport organisations in Sweden, including Samtrafiken i Sverige AB, Trafikförvaltningen (Region Stockholm, SL), Västtrafik (Region Västergötland), Skånetrafiken (Region Skåne) and UL (Region Uppsala). 53
The purpose is to evaluate how NFC-based solutions on iPhone can be used to support these use cases in public transport across Sweden in a consistent and interoperable way. 53
Our evaluation focuses on two primary use cases: 53
1. Digital ticket 53
A single ticket or period ticket stored on the device and presented via NFC for validation. 53
2. Account-based travel identifier 53
A stable identifier presented via NFC and resolved in backend systems, for example based on BoB MTS7 or EMV closed loop. 53
The purpose of this document is to clarify what is technically and commercially possible on iPhone for these use cases, and which implementation paths Apple would recommend in a national public transport context. 53

Example scenario: 53
6.1 Viable iOS NFC Options for Transit Card Validation 53
6.2 Technical Architecture Validation 54
6.3 HCE (CardSession) — Confirming the Recommended Foundation 56
6.4 Express Transit (Apple Wallet Bilateral) — Evaluating the iOS Gap....57
6.5 NFC & SE Platform — Sweden's Exclusion 57
7 Questions for Google 59
7.1 Viable Android NFC Options for Transit Card Validation 59
7.2 Technical Architecture Validation 60
7.3 Android HCE — Confirming the Recommended Approach 62
7.4 Google Wallet — Evaluating Native Integration 63
7.5 Security and Long-Term Commitment 64
8 References 64
8.1 Standards and Protocols....64
8.2 Apple Developer Documentation 65
8.3 Android / Google Documentation 65
8.4 Samtrafiken / BoB....65
8.5 Regulatory 66
8.6 Third-Party Analysis....66
9 Glossary 67

1 Purpose and Scope

1.1 Purpose

This document describes a joint investigation conducted by Samtrafiken i Sverige AB, Trafikförvaltningen (Region Stockholm, SL), Västtrafik (Region Västergötland), Skånetrafiken (Region Skåne) and UL (Region Uppsala).
The purpose of the investigation is to examine how NFC technology in mobile phones (iOS and Android) could be used to support simpler public transport travel in Sweden, based on the BoB standard. The results will be presented to the Swedish public transport sector as a basis for future development and are also intended to support dialogue with Apple and Google regarding platform capabilities.

The investigation focuses on two specific areas: the use of NFC to present a digital travel identity for account-based travel based on EMV closed-loop or BoB MTS7, and the use of NFC to present public transport tickets (single tickets and period tickets).

The target audience for this document is assumed to have some knowledge of NFC, mobile application development and technology related to these areas and standards.

1.2 Use Case

A traveller holds their smartphone near an NFC reader located in a vehicle (bus, tram, train) or at a fixed point of entry (metro turnstile, station gate). The reader validates the digital travel card ID (and any tickets linked to the card) or transit ticket stored or processed by the traveller’s app.

[Traveller's phone] <—NFC—> [Validator/reader in vehicle or station]

(card emulator) (ISO 14443 / NFC-A/B reader)

The phone acts as an **NFC card emulator**. The validator is a fixed or in-vehicle NFC reader device. Only iOS (iPhone) and Android are in scope.

1.3 Regulatory Context

Sweden is a member of the European Economic Area (EEA). The EU Digital Markets Act (DMA) compelled Apple to open HCE-based NFC access for EEA developers from iOS 17.4 (March 2024). Apple additionally announced a voluntary NFC & SE Platform in August 2024, launched with iOS 18.1 in October 2024. The two paths are geographically distinct by policy: the EEA HCE path is available to Swedish operators; the NFC & SE Platform is not — Sweden and all other EEA countries are currently excluded from the SE Platform territory list.

1.4 Scope

In scope:

- iOS (iPhone) NFC card emulation options
- Android NFC card emulation options
- Integration path with BoB
- Requirements for app developers and transit operators

Out of scope:

- Other mobile operating systems
- NFC phone-as-reader (inspector/controller devices)
- QR code or Bluetooth validation alternatives
- Apple Pay / Google Pay payment flows (unless directly reusable)

2 Background

2.1 BoB Background

BoB is a national standard for ticket distribution and validation in Swedish public transport, developed by Samtrafiken in cooperation with the public transport sector. The standard defines how tickets, travel rights and validation information are structured and exchanged to enable interoperability between different ticketing systems, sales channels and validation systems.

BoB already includes support for NFC-based travel identifiers through BoB MTS7. However, the specification does not currently define how NFC functionality in mobile devices (iOS and Android) can be used for presenting tickets or travel identities. This investigation therefore examines how such use of mobile NFC could be introduced in a way that aligns with the existing BoB framework.

Reference: BoB API Documentation

3 Existing Samtrafiken Card Types

Samtrafiken's ecosystem currently uses two distinct NFC card standards on physical smartcards. Any mobile HCE implementation must be capable of emulating one or both.

3.1 MTS7 – Samtrafiken Proprietary Travel Card Application

Used by: Multiple Swedish regional operators.
Specification: MTS7 v1.1.0 (2019-09-05). Bitbucket: samtrafiken/BoB-mts-7-travel-card-application.
AID (Application Identifier):

Field	Length	Value
RID	5 bytes	A0 00 00 07 81
Application	1 byte	1
Version	1 byte	1
PID (Participant Identifier)	2 bytes	Operator-specific
Serial	7 bytes	Unique per PICC

SELECT targets the 6-byte prefix: 00 A4 04 00 06 A0 00 00 07 81 01 00

Physical interface: ISO 14443-4 A or B. Terminals MUST support both.

Commands (CL1 – read-only validation): SELECT, GET (NEXT) DATA, INTERNAL AUTHENTICATE, GET RESPONSE.

Commands (CL2 – full, adds write/lifecycle): GENERAL AUTHENTICATE, PUT (NEXT) DATA, GENERATE ASYMMETRIC KEY PAIR, ACTIVATE FILE, DEACTIVATE FILE, TERMINATE DF. All CL2 write commands require Secure Messaging.

Cryptography: - Authentication: ECDSA P-256 (recommended) or RSA-2048 / PKCS#1 v1.5 - Secure Messaging (CL2 only): AES-128 CBC / SCP03 (Global Platform Card Spec Amendment 3) - Hash: SHA-256 (32-byte Authentication Input to INTERNAL AUTHENTICATE)

CL1 validation APDU flow (3 commands):

1. SELECT BoB Application:
   → 00 A4 04 00 06 A0 00 00 07 81 01 00
   ← FMD containing '6E' DO (application-related data, status)

2. GET NEXT DATA (retrieve MTB – Mobile Ticket Bundle):
   → 00 CC 7F 21 00 00 00
   ← MTB (issuer-signed certificate containing travel entitlements + cardholder public key)
   [If MTB > buffer: status 61xx → terminal issues GET RESPONSE (INS C0) for remainder]

3. INTERNAL AUTHENTICATE (prove card possession):
   → 00 88 00 00 20 [32-byte SHA-256 hash of Authentication Input JSON] 00
   ← ECDSA signature (r||s, big-endian) — terminal verifies against public key in MTB

The Authentication Input JSON contains: pid (terminal participant ID), terminalId, aid, signatureCounter (from tag 93 — replay protection), time, loc (optional GPS).

Key data objects: 7F21 = cardholder certificates (MTBs); 6E = application-related data; 93 = digital signature counter (3-byte, increments on every successful INTERNAL AUTHENTICATE, reset on re-personalisation).

Terminal requirements: ISO 14443 A and B; ISO 7816-4; SHOULD support 424 kbps and extended APDU ≥1024 bytes.

3.2 EMV Kernel 5 Transit Card

Used by: Storstockholms Lokaltrafik (SL), the largest member of Samtrafiken.
Kernel: EMV Kernel 5 (the EMV contactless kernel specification defined by JCB; used here by SL for closed-loop transit, not for JCB payment).
AID: A0 00 00 08 38 10 10 (SL-registered, separate from the standard JCB payment AID A0 00 00 00 65).
BIN prefix: 97523124 (owned and used by SL).
Card number format: 19 digits (PAN), prefixed with BIN 97523124.

This card type uses **EMV Contactless** protocol over ISO 14443-4, structured according to EMV Kernel 5. SL maintains its own registered AID and card number range, using the kernel infrastructure for closed-loop transit — not for payment.

Key EMV contactless commands in transit validation:

- SELECT (by AID): Application selection
- GET PROCESSING OPTIONS (GPO): Retrieve Application Interchange Profile and AFL
- READ RECORD: Read card data records (card identifier, expiry, keys, certificates)
- GENERATE AC: Card generates an Application Cryptogram (ARQC for online auth, TC for offline) proving possession of the card's diversified key

Cryptography: Triple-DES or AES for Application Cryptogram generation (standard EMV symmetric key scheme). RSA for offline data authentication certificates (if SDA/DDA/fDDA supported).

Physical interface: ISO 14443-4 (NFC-A or NFC-B), EMV Contactless Book D Level 1 timing.

4 Platform Technical Capabilities

4.1 Android HCE (HostApduService)

Android has offered mature, production-grade HCE since Android 4.4 (API 19, 2013). HostApduService routes ISO/IEC 7816-4 APDUs directly to the host CPU over ISO-DEP (ISO/IEC 14443-4), with no hardware Secure Element required.

4.1.1 Capability Summary

Attribute	Details
Available from	Android 4.4 (API 19)
Protocol	ISO 14443-4 / ISO 7816-4 APDUs — NFC-A mandatory; NFC-B HCE is optional per the Android spec and absent from essentially all real-world devices
Fee	None — no gating process, no commercial agreement required
AID routing	System-managed prefix/exact matching; CATEGORY_OTHER for transit
Screen-off NFC	Reliable from Android 12+; requireDeviceScreenOn="false" attribute available since API 21
Lock-screen NFC	Supported by default (requireDeviceUnlock="false")
Background operation	Service is started and bound by the OS on AID match; app does not need to be in foreground
Key security	Android Keystore — TEE-backed (Android 4.3+) or StrongBox (Android 9+)
Observe Mode	Android 15+ (API 35) — app can gate transactions on reader-field detection

4.1.2 HostApduService Methods

The service implements three key methods:

Method	Description
processCommandApdu(commandApdu, extras)	Called on the **main thread** when an APDU arrives from the reader. Return response bytes synchronously, or return null and use sendResponseApdu() asynchronously.
sendResponseApdu(responseBytes)	Sends a response asynchronously — required for any non-trivial processing (crypto, DB lookup).
onDeactivated(reason)	Called when the NFC field is lost (DEACTIVATION_LINK_LOSS) or the reader deselects the AID (DEACTIVATION_DESELECTED). Use for session cleanup.

Threading: processCommandApdu() runs on the main thread. All long-running operations (cryptographic signing, backend calls) must be dispatched to a background thread/coroutine, with sendResponseApdu() called when complete.

Performance targets: ~1 KB recommended total data per transaction; ~300 ms typical total round-trip time. Target 3–5 APDU commands for a complete transit validation.

4.1.3 AID Registration and Routing

AIDs are declared statically in an XML resource (apduservice.xml) referenced by the manifest. For transit use, category="other" is the correct category:

```
<host-apdu-service android:requireDeviceScreenOn="false"
                   android:requireDeviceUnlock="false">
  <aid-group android:category="other">
    <aid-filter android:name="YOUR_TRANSIT_AID"/>
  </aid-group>
</host-apdu-service>
```

Android supports exact match, prefix match (trailing *), and longest-match-wins resolution. If multiple services register the same AID in CATEGORY_OTHER, Android resolves the conflict as follows: if the user's default wallet app has registered the AID, that app is invoked; if not, and exactly one other service has registered the AID, that service is invoked; if more than one non-default service has registered the AID, Android presents a disambiguation dialog. CardEmulation.setPreferredService() bypasses this entire resolution chain when active — it is not a step within it but a foreground override that takes precedence unconditionally. UIDs are

randomly generated per tap — they must not be used for session identity.
Using CATEGORY_OTHER for transit AIDs avoids all conflicts with payment apps (which use CATEGORY_PAYMENT). The transit operator does not need the ROLE_WALLET (default wallet) role.

4.1.4 Background and Screen-Off NFC

Android Version	Screen-Off	Lock-Screen	Notes
≤ 9	Not supported	Supported (default)	NFC controller and application processor are powered off when the screen is off — a hardware-level power constraint, not a software gate
10–11	Depends on Secure NFC user setting	Supported	Secure NFC toggle affects availability
12+	Supported — set requireDeviceScreenOn="false"	Supported	Explicit opt-in for screen-off transit taps
15+	Supported + Observe Mode	Supported + Observe Mode	App controls transaction gating per tap

With requireDeviceScreenOn="false" (available since API 21; reliable screen-off NFC behaviour from Android 12+), the HCE service responds even when the device screen is completely off — enabling true pocket tap-and-go without user interaction, directly equivalent to a contactless card.

Secure NFC caveat: Users can enable “Secure NFC” in Android Settings (Android 10+). When Secure NFC is ON, all card emulators are disabled with the screen off, regardless of the manifest setting. Transit apps should inform users of this setting.

Observe Mode (Android 15+): The NFC controller polls but does not respond, allowing the app to authenticate the user before committing to a transaction. The app calls nfcAdapter.setObserveModeEnabled(true), processes polling frames to detect the reader field, optionally prompts biometrics, then disables observe mode to allow the APDU exchange.

4.1.5 Android Keystore for Credential Security

Android Keystore provides hardware-backed cryptographic key storage. Private keys never leave secure hardware (TEE or StrongBox). Key security levels:

Level	Description	Android Version

TEE-backed	Keys in Trusted Execution Environment	Android 4.3+
StrongBox	Dedicated SE with independent CPU; tamper-resistant	Android 9+ (API 28+)

Supported algorithms in StrongBox: RSA 2048, AES 128/256, ECDSA/ECDH P-256, HMAC-SHA256.

Key attestation: The Keystore can generate an attestation certificate chain rooted in a Google-provisioned CA, proving to the transit backend that the credential is bound to genuine hardware. This allows the server to verify credential authenticity without contacting the device at transaction time.

Screen-off constraint: User-authentication-bound keys (requiring biometric before use) cannot be used during screen-off transactions, since no UI is available for biometrics. For screen-off transit taps, use session tokens pre-authorized during the last active session, with a short-lived validity window.

Recommended model for Samtrafiken: A long-lived TEE/StrongBox-backed ECDSA P-256 identity key (for attestation and provisioning), plus a short-lived session token pre-derived during the user's last authenticated session, used for signing during screen-off APDU exchanges.

4.1.6 Credential Provisioning

Android HCE imposes no provisioning model — the operator's app is fully responsible. Nothing is provisioned at install time; the app is an empty shell until the user registers.

Trigger: User registers or logs in to the operator's account within the app.

Flow:

1) The app generates an ECDSA P-256 key pair via **Android Keystore**. The private key is bound to the TEE or StrongBox and never leaves secure hardware.
2) The app sends the public key and optionally a **hardware attestation certificate chain** to the operator's backend. The attestation proves to the backend that the key is genuinely hardware-bound on an unmodified Android device.
3) The backend registers the public key against the user's account and issues a signed **credential bundle** (for MTS7: an MTB containing travel entitlements and the device public key, signed by the issuer).
4) The signed credential is stored in the app's encrypted private storage on the device.

At tap time: The app signs the reader's challenge (INTERNAL AUTHENTICATE) using the locally stored private key. No network call is needed at tap time — validation is fully offline.

Refresh: The signed credential bundle is time-bounded and refreshed over the network (e.g. at app launch or via background push). The long-lived key pair remains on-device; only the credential contents are updated.

Re-provisioning: Reinstalling the app generates a new key pair. A new credential must be issued by the backend; old credentials bound to the previous key are invalidated.

Event	Outcome
App install	Nothing — no credentials created
User registration / login	Key pair generated; credential issued by backend
Periodic refresh	Credential bundle updated; keys unchanged
App reinstall	New key pair; full re-provisioning required

4.2 iOS HCE for EEA (CardSession, iOS 17.4+)

Driven by the EU Digital Markets Act (DMA), Apple introduced HCE-based NFC for developers established in the EEA with iOS 17.4 (March 2024). Apple's commitments to the European Commission took effect 17 July 2024 and are binding for 10 years, EU-wide.

4.2.1 Platform Requirements and Capability Summary

Attribute	Details
Available from	iOS 17.4 (March 2024); out-of-EEA dev/test requires iOS 18.2+
Geographic scope	EEA only — user's Apple Account must be registered in the EEA
Technology	Host Card Emulation (software, no SE hardware required)
Fee	Free of charge (DMA commitment)
Supported use cases	Payments, closed-loop transit, car keys, home/hotel keys, corporate badges, membership, event tickets
Device requirement	iPhone XS or later
Protocol	ISO 14443-4 (transport) / ISO 7816-4 APDU (application)
Framework	CoreNFC
Entitlement	com.apple.developer.nfc.hce — must request via Apple per Bundle ID
Key APIs	CardSession, CardSession.isEligible, NFCPresentmentIntentAssertion, Field Detect
Simulator support	Not supported — physical iPhone and physical NFC reader required for all testing

For Samtrafiken, the applicable use-case category is **Closed-loop Transit**. Eligibility requires being a transport operator or having a binding agreement with one for transit tickets.

4.2.2 CardSession API

CardSession (in CoreNFC) is the central class for HCE on iOS. Before presenting any HCE UI, the app must check CardSession.isEligible (returns Bool). If false, all HCE-related features must be hidden or disabled — this guards against unsupported devices, iOS versions below 17.4, and users whose Apple Account is not in the EEA.
CardSession exposes an **async event stream** rather than a delegate:

```
for await event in cardSession.eventStream {
   switch event {
   case .sessionStarted: // NFC sheet shown
   case .readerDetected: cardSession.startEmulation()
   case .received(let cardAPDU):
      let responseData = processAPDU(cardAPDU.payload)
      try await cardAPDU.respond(response: responseData)
   case .readerDeselected: cardSession.stopEmulation(status: .success)
   case .sessionInvalidated(let reason): // error or user cancelled
   }
}
```

Each incoming APDU triggers a .received event. The response Data passed to cardAPDU.respond(response:) must include the 2-byte ISO 7816-4 status word (SW1SW2) at the end (e.g. 0x90 0x00 for success).

APDU size constraint: Response APDUs under approximately 1694 bytes work reliably in practice; errors may occur at or above that threshold. This is an empirically observed limit reported by developers — Apple does not document a specific figure.

Standard transit ticketing exchanges are well within this limit. Chunked responses using GET RESPONSE (ISO 7816-4) can work around it if needed.

4.2.3 AID Registration and Routing

On iOS, AID routing is controlled **at entitlement time**, not at runtime.

The app declares handled AIDs in the .entitlements plist under:
com.apple.developer.nfc.hce.iso7816.select-identifier-prefixes

This is an array of hex-encoded AID or RID **prefix** strings. The system performs prefix matching: an entry of A000000396 matches any AID beginning with those bytes.

When an NFC reader sends a SELECT (by AID) command (ISO 7816-4, INS 0xA4, P1 0x04), the iOS NFC subsystem routes it to the matching app's CardSession.

If multiple apps register the same AID, the user's **default contactless app** (set in device Settings) takes precedence for Field Detect scenarios. Only one app can be the system default contactless app at a time.

For Samtrafiken, the AID(s) used must be:

1) defined in the BoB NFC protocol specification,
2) registered in the Apple entitlement request, and
3) declared in the app's .entitlements file. The AID must match what the vehicle-side readers issue in their SELECT commands.

4.2.4 Three Activation Modes

iOS HCE supports three modes for initiating a transaction:

Mode	Trigger	Available to	Auth required
Field Detect	Phone enters NFC reader's RF field	Default contactless app only	May require Face ID/Touch ID on locked device
Double-click	User double-clicks side or Home button	Default contactless app only	May require authentication
In-App Presentment	User opens app and taps a button	Any eligible app (foreground)	No additional auth (app is already open)

For seamless transit tap-in/tap-out, **Field Detect** is the required mode — the user holds the phone to the reader without unlocking or opening an app.

This requires the transit app to be the user's designated default contactless app.

Non-default apps can only transact in **In-App Presentment** mode, requiring the user to open the app and initiate the transaction within a **15-second window** before holding the phone to the reader.

The 15-second window is defined by the NFCPresentmentIntentAssertion, which expires after 15 seconds and has a 15-second cooldown before a new one can be acquired.

Key implication for transit UX: If the operator's transit app is not set as the user's default contactless app, the tap-and-go experience is degraded. the default app slot will typically be competed for by other third-party HCE apps (e.g., a bank's standalone card app).

Unlike Apple Pay Express Transit, HCE has **no Express Mode** — there is no lock-screen, no-authentication transaction path for HCE apps even as the default app.

Relationship to Apple Wallet and bank cards: The default contactless app designation applies exclusively to CardSession HCE apps. It has no effect on Apple Wallet or the payment cards stored in it.

Bank cards in Apple Wallet operate via the hardware Secure Element (SE), not via CardSession, and the NFC controller routes payment AIDs (Visa, Mastercard, etc.) directly to the SE regardless of which app holds the default contactless app slot.

A user can therefore have the transit app set as their default contactless app while continuing to use their bank cards in Apple Wallet for payments without any conflict or interference.

The default contactless app slot is global, not per use-case: iOS provides a single default contactless app slot shared by all HCE apps — there is no per-AID or per-category default.

An HCE app that has registered a specific AID can still be activated via AID-driven routing without holding the default slot: iOS routes the tap to the app when a reader selects its registered AID.

The distinction matters for transit. AID-driven routing requires the reader to first issue a SELECT command for the app's AID before the app is activated, which works correctly in normal operation.

Field Detect activates the app the moment the phone enters any NFC field, before AID negotiation, giving a marginally more robust and faster startup. For the most demanding validators (strict timing requirements), Field Detect as the default app is preferred; AID-driven routing is a viable fallback if the user has another app occupying the default slot.

4.2.5 Entitlement Request Process

The entitlement is granted **per Bundle ID**. Every app using CardSession requires a separate request. The process:

1. Submit form to Apple
2. Provide: app name, Bundle ID, use case, full list of AIDs/RID prefixes, Account Holder credentials
3. Apple issues a **development entitlement** first; a **separate distribution entitlement** request is required for App Store submission (AIDs must be resubmitted)
4. App Store review requires: test login credentials, at least one test transit credential, and a screenshot/video of a real NFC transaction at a terminal

Apple does not publish an SLA. Developer forum experience indicates **several weeks** for approval. If denied, a dispute mechanism exists: email HCEEntitlementdispute@apple.com within 15 business days, with copy to the EC Monitoring Trustee (ApplePay.EC-commitments@alcis-advisers.com).
Following that submission, there is a mandatory 20-business-day Consultation Period during which both parties attempt to resolve the matter. Only if the Consultation Period ends without resolution can the dispute be referred to the NFC Appeal Board (chair@nfc-appeal-board.com).
Naming constraints: Apple prohibits using the terms “NFC”, “Core NFC”, or “near-field communication” in user-facing UI copy, and prohibits use of Apple Pay/Wallet marks in the HCE app.

4.2.6 Key Limitations

Limitation	Detail
No Express Mode	No locked-screen, no-authentication transactions; SE Platform required for that
EEA-only	Only available to users with EEA Apple Account; not available to travellers with non-EEA accounts
No SE hardware security	Keys and credentials in app software keychain; no hardware-backed key storage
15-second window	NFCPresentmentIntentAssertion expires after 15 s with 15 s cooldown (non-default apps)
No simulator	Physical iPhone XS+ and physical NFC reader required for all testing
Response APDU size cap	~1694 bytes (empirically observed; not an Apple-documented limit); chunked responses needed for larger payloads
Per-bundle entitlement	Each app Bundle ID needs separate entitlement; distribution requires re-approval

4.2.7 Credential Provisioning

The provisioning model mirrors Android HCE in structure — app-controlled, triggered by user registration, no credentials at install time — but differs in the security properties of key storage.
Trigger: User registers or logs in to the operator's account within the app.

Flow:

1. The app generates a key pair and stores the private key in the **iOS Keychain** (software-backed; no hardware SE is accessible to CardSession apps).
2. The app sends the public key to the operator's backend.
3. The backend issues a signed credential bundle (MTB or equivalent) bound to that public key.
4. The credential is stored in the app's Keychain or encrypted local storage.

At tap time: The app's CardSession handler signs the reader's challenge using the Keychain-stored private key. Fully offline once provisioned.

Key difference from Android: The iOS Keychain is software-backed for HCE apps. There is no equivalent to Android Keystore's hardware attestation — the backend cannot cryptographically verify that the key is resident on a specific hardware device.

This is the “No SE hardware security” limitation noted in the Key Limitations section.

Refresh and re-provisioning: Same pattern as Android HCE — credential bundle refreshed periodically; reinstalling the app requires full re-provisioning with a new key pair.

Event	Outcome
App install	Nothing — no credentials created
User registration / login	Key pair generated (software Keychain); credential issued by backend
Periodic refresh	Credential bundle updated; keys unchanged
App reinstall	New key pair; full re-provisioning required

4.3 iOS NFC & SE Platform (CredentialSession, iOS 18.1+)

Apple’s voluntary global extension of NFC access using the hardware Secure Element (SE), launched with iOS 18.1 in October 2024.

4.3.1 Capability Summary

Attribute	Details
Available from	iOS 18.1 (October 2024)
Geographic scope	Limited list of territories; **Sweden/EEA not confirmed as supported** as of research date
Technology	Hardware Secure Element (SE) — isolated silicon, tamper-resistant
Fee (EEA)	**Undisclosed commercial fees** — DMA FRAND commitments apply to HCE (CardSession) only; Apple has not confirmed zero-fee access for the NFC & SE Platform in the EEA
Fee (non-EEA)	Undisclosed commercial fees per commercial agreement
Supported use cases	Payments, closed-loop transit, car/home/hotel keys, corporate badges, student IDs, loyalty, event tickets, government IDs (announced; not yet available)
Device requirement	iPhone XS or later
Protocol	ISO 14443-4 / ISO 7816-4 (APDUs via Java Card applet in SE)
Entitlement keys	com.apple.developer.secure-element-credential; com.apple.developer.secure-element-credential.default-contactless-app
API SDK	Gated — distributed only to enrolled NFC & SE Platform partners under NDA
Simulator support	**Not supported — physical hardware required**

Critical finding: Sweden is not confirmed in the NFC & SE Platform's supported territory list as of iOS 18.1–18.4. The EEA HCE route (CardSession, iOS 17.4+) covers Sweden today; the SE Platform does not yet.

4.3.2 Four API Groups

The platform exposes four API groups:

1. **Provisioning API** — Initiates credential installation: Apple's servers download the signed applet to the device SE, create a dedicated SE memory partition, then pass control to the issuer's servers for personalization (injecting keys and card data).
2. **Transaction Authorization API** — Handles presentment: prompts the user to authenticate via Face ID, Touch ID, or passcode; Secure Enclave sends signed authentication data to the SE; SE transmits credential data over NFC.
3. **Life Cycle Management API** — Manages credential updates (issuer sends commands to the app which forwards to the SE applet) and deletion (user or remote wipe).
4. **Presentment Intent Assertion API** — Same 15-second assertion mechanism as HCE; prevents the default contactless app from intercepting the tap. Foreground-only, expires after 15 seconds, 15-second cooldown.

4.3.3 Applet Model

An **applet** is a Java Card application running inside the hardware SE. The applet implements the card emulation protocol logic. Key constraints:

- The developer/issuer writes and certifies the applet; Apple does not write it.
- The applet must pass evaluation by an **independent, accredited third-party security lab** before Apple signs and hosts it.
- Apple signs all applets and stores them on Apple servers; developers cannot sideload applets.
- Each credential has an isolated SE memory partition; Apple manages SE memory allocation.
- Updates to the applet require re-certification if significant changes are made.
- A **TSM (Trusted Service Manager)** is required to personalize the SE partition with credential-specific data over a secure channel.

The SE is certified by both **EMVCo** and **Common Criteria**.
It runs the Java Card platform.

4.3.4 Onboarding Process

Step	Action
1	Submit request to Apple
2	Sign NFC & SE Platform commercial agreement + NDA with Apple
3	Onboard to Apple Business Register (ABR); request entitlement for the specific app
4	Commission independent security lab to evaluate the applet binary
5	Deliver signed applet bundle (with SP-TSM, Team ID, etc.) to Apple; Apple signs and hosts
6	Implement all four API groups; test with physical hardware
7	**App Store review: provide test credentials, NFC transaction video, Presentment Intent Assertion video**

Estimated timeline for a new entrant: **several months to over one year**.
Indirect costs include: third-party security lab certification fees (industry estimates: tens of thousands to hundreds of thousands of EUR) and TSM integration.

4.3.5 Express Mode and Background NFC

A critical distinction: Express Mode (tap-and-go with no waking, unlocking, or authentication — including in Power Reserve) is an **Apple Wallet exclusive**.
It is **not available** to third-party NFC & SE Platform apps.
Third-party SE Platform apps behave as follows:

Scenario	Behaviour
App in foreground + Presentment Intent Assertion held	Transaction handled by SE credential; device must be unlocked and app in foreground
App is designated default contactless app	Field-detect activates and auto-launches the app; double-click is an alternative launch path; authentication is required before the transaction proceeds
App in background, not default	Tap is not handled by the third-party app; the system default contactless app (if set) may handle it, or the tap goes unhandled
Power Reserve (low battery)	**Not supported for third-party SE credentials**

The practical transit UX for SE Platform is thus essentially equivalent to HCE: both require the user to authenticate per transaction. Neither provides passive tap-and-go without the app open or user action.

4.3.6 Comparison with HCE for Samtrafiken's Use Case

Dimension	NFC & SE Platform	HCE (CardSession, EEA)
SE hardware security	Yes — tamper-resistant, air-gapped silicon	No — software keychain
Express Mode	No — Apple Wallet exclusive	No
Authentication per tap	Required (double-click + biometric)	Required (same)
Territory (Sweden)	Not yet available	Available (iOS 17.4+, EEA)
Fee (EEA/Sweden)	Undisclosed commercial fees (no DMA zero-fee commitment for SE Platform)	Zero
Applet development	Required (Java Card)	Not required
Security lab certification	Required	Not required
TSM required	Yes	No
Time to market	Months to >1 year	Weeks to months
Transit use-case suitability	Sufficient (overkill for closed-loop)	Sufficient

Verdict: For Samtrafiken's closed-loop transit use case, HCE is the pragmatic choice. The SE Platform offers no meaningful UX advantage over HCE for third-party apps, adds major onboarding complexity, and is not currently available in Sweden.

If high-security credential storage (comparable to EMV payment cards) becomes a requirement in a future phase, the SE Platform would be revisited.

4.3.7 Credential Provisioning

Provisioning for the SE Platform is fundamentally different from HCE: credential material is written directly into the hardware SE via a **Trusted Service Manager (TSM)**, not stored by the app. The app itself never holds or sees the cryptographic keys.

Trigger: An explicit user action in the operator's app (e.g. "Activate card", "Add to phone").

Flow:

1. The user initiates provisioning in the operator's app; the app authenticates the user with the backend.
2. The backend notifies the operator's **SP-TSM (Service Provider Trusted Service Manager)**, which holds the authority to personalize the operator's SE applet partition.
3. The SP-TSM opens a **Secure Messaging channel** (typically GlobalPlatform SCP03) to the SE via Apple's SE management infrastructure, over the air.
4. The TSM sends personalization data — card-specific keys, account identifiers, initial state — to the SE applet via encrypted STORE DATA commands. The communication is end-to-end encrypted between TSM and SE; neither the app, iOS, nor the operator's server can read the key material in transit.
5. The SE applet is activated. The operator's app can now communicate with it via CredentialSession.

Key storage: All cryptographic keys reside exclusively inside the SE after personalization. They are not accessible to the application processor, the OS, or the operator's backend.

Updates: Any credential update (e.g. key rotation, entitlement change) requires a new TSM-mediated OTA operation. The app triggers it; the backend and SP-TSM execute it.

Device migration: SE data may or may not survive a device backup/restore depending on Apple's SE management policy. In most SE-based payment/transit deployments, the credential does not transfer automatically to a new device — re-provisioning is required.

Event	Outcome
App install	Nothing — SE applet slot exists but is unpersonalized
User initiates provisioning	TSM personalizes SE applet over OTA secure channel
Credential update	New TSM-mediated OTA operation required
New device	**SE data typically does not transfer; re-provisioning required**

4.4 Apple Wallet Express Transit (PassKit)

Apple Wallet supports two distinct pass types for transit. Only one is relevant for NFC tap-to-ride at a fixed reader.

4.4.1 Two Pass Types: What Works for NFC

Standard boarding pass (.pkpass **format):** Visual/barcode pass representing a single trip. Supports QR codes and barcodes for gate scanning.
Does **not** support NFC tap at a reader — not viable for Samtrafiken's use case.

Native NFC transit card (Express Transit programme): Uses the device's Secure Element. Behaves like a physical contactless transit card. Supports Express Transit (tap-and-go without waking, authenticating, or opening the app).

As of mid-2024 (latest found data), approximately 27 transit systems globally have native Wallet support; only about 12 of those support Express Transit.

Access is by bilateral Apple partnership only — not self-service.

For Samtrafiken's use case (fixed NFC reader in vehicle/station), only the native NFC transit card integration is relevant.

4.4.2 Express Transit — How It Works

Express Transit allows tap-and-go with **no device wake, no authentication, no app launch**. The transit card in the SE is always "live" in the NFC field. This is the key UX differentiator vs. HCE-based options.

Technically, transit card data (including balance if applicable) is provisioned into a dedicated applet on the SE. The operator holds the cryptographic keys to perform balance transactions against that applet. From Apple's security guide: "*If the card balance is stored on the device, it's encrypted and stored to a designated applet in the Secure Element. The operator has the keys to perform cryptographic operations on the card data for balance transactions.*".

Express Transit works in **Power Reserve** mode (device battery near-empty). Basic Express Transit support requires iPhone 8 or later; Power Reserve specifically requires iPhone XS or later.

4.4.3 Bilateral Partnership Process

Transit operators must apply to Apple's transit partner programme at learn.wallet.apple/transit. The process is not self-service:

- Apple evaluates the transit authority case-by-case and negotiates a bilateral agreement
- Commercial terms, data handling, and technical integration details are negotiated and not publicly disclosed
- The operator's card management system is connected to Apple's transit provisioning API

- Fewer than 30 systems globally have this integration — Apple is highly selective

The nearest European analogue is the Île-de-France Mobilités Navigo card for iPhone and Apple Watch, launched in May 2024. In Sweden, Skånetrafiken has **open-loop** Apple Pay Express Transit (riders tap their Visa/Mastercard via Apple Pay) — this is EMV payment contactless, **not a provisioned BoB-style closed-loop transit card**.

4.4.4 BoB Compatibility

BoB Requirement	Apple Wallet Express Transit
NFC tap at reader (ISO 14443)	Yes — SE-backed, always-on
ISO 7816-4 APDU exchange	Possible only if Apple certifies a custom BoB applet
BoB APDU protocol (MTS7)	SE path (bilateral): requires Apple to certify a custom BoB/MTS7 JavaCard applet; known bilateral precedents involve open or widely standardised schemes (FeliCa, Calypso). EEA HCE path: full APDU protocol freedom, no applet cert required — but no Express Mode, no native Wallet card.
Operator card lifecycle control	Partial — mediated through Apple's API
Multi-operator interoperability	Requires per-operator partnership with Apple; no self-service path

Key constraint (SE/bilateral path): Apple's bilateral transit card programme has historically been based on established, publicly documented card schemes — FeliCa (Suica/PASMO), Calypso (Navigo), MIFARE-based systems.

Implementing the full BoB/MTS7 APDU protocol via this path would require Apple to certify a custom JavaCard applet for a proprietary, undocumented protocol — a significantly higher bar than certifying an open or widely standardised scheme.

The EEA HCE path (CardSession, iOS 17.4+) removes this constraint entirely: full APDU protocol freedom is available without SE applet certification, but at the cost of Express Mode and native Apple Wallet integration.

4.4.5 Operator Control Limitations

- **Card lifecycle is mediated:** All provisioning, blocking, and expiry operations go through Apple's API — not direct SE access.
- **UX is Apple-controlled:** Apple controls the Wallet UI, card art template, and Express Transit behavior.
- **No journey tracking by Apple:** Apple explicitly does not track users' journeys. Card

data is encrypted in SE. Journey data reaches the operator via fare readers, not Apple's servers.

- **Platform lock-in:** Dependent on Apple maintaining the programme, API, and SE provisioning infrastructure.
- **Android not covered:** Apple Wallet is iOS-only. Android requires a separate, completely different NFC stack — there is no symmetric approach.
- **Not DMA-mandated:** The native Wallet transit card programme is not mandated by the Digital Markets Act. Apple's HCE obligation (CardSession) is separate and does not force Apple to open the Wallet transit programme.

4.4.6 Credential Provisioning

Provisioning into Apple Wallet Express Transit uses Apple's **PassKit provisioning sheet** — an OS-level UI flow that mediates between the operator's card management system and Apple's SE infrastructure.

The user initiates it; the operator's backend and Apple's servers execute it.

Trigger: User taps an "Add to Apple Wallet" button in the operator's app or on a web page.

Flow:

1. For payment-card-style provisioning, the operator's app calls PKAddPaymentPassViewController (or a web equivalent), which presents an Apple-branded card preview and confirmation sheet. (Native transit card provisioning in the bilateral partner model may use a different flow — for some systems, users can add the card directly via the Wallet app's transit card UI without involvement of the operator's app.)
2. PassKit contacts Apple's provisioning server, which generates a one-time encrypted provisioning request and passes it to the operator's card management system via the app.
3. The operator's card management system generates card-specific data: account/transit identifier (and any PAN for EMV-based cards), initial keys, and entitlement state. This data is encrypted under Apple's public key so that only Apple's SE infrastructure can decrypt it.
4. The encrypted payload is returned to PassKit, which forwards it to Apple's SE management infrastructure.
5. Apple's SE infrastructure personalizes the transit applet in the device's SE with the card-specific data.
6. The card appears in the user's Apple Wallet and is immediately active for Express Transit taps.

Key principle: The operator defines the card data; Apple controls the SE. The operator's backend never has access to the in-SE private key material — only the initial personalization data it provides.

Balance and entitlement updates: If the card stores a value balance, it is encrypted in the SE applet. The operator holds the cryptographic keys and can push balance/entitlement updates via Apple's transit API without user interaction.

Device migration: When a user switches to a new iPhone, Wallet cards do not auto-transfer. The user re-provisions via the operator app (a new provisioning flow against the new device's SE). Apple Wallet contents are not included in iCloud backups. However, transit cards are associated with the user's iCloud account; on a new device the user can re-add the card via "Cards Found For You" or "Previous Cards" in the Wallet app — provided the old device has been removed first. This is an iCloud account-linked re-provisioning flow, not a backup restore.

Event	Outcome
App install	Nothing — provisioning is a separate explicit step
User taps "Add to Apple Wallet"	PassKit provisioning sheet; SE personalized by Apple's TSM

Balance / entitlement update	Operator backend pushes update via Apple's transit API; no user action
New device	Re-provisioning required; no automatic SE transfer

4.5 Google Wallet Smart Tap

Google Wallet is Google's digital wallet for Android. For transit NFC, it uses **Smart Tap** — a **Google-proprietary NFC protocol** that runs over the standard Android HCE stack.

> ***Note:*** *Smart Tap is the NFC mechanism for Google Wallet passes (loyalty cards, transit passes, boarding passes). It is distinct from Google Wallet's separate* ***closed-loop transit card emulation*** *programme, which supports MIFARE DESFire, MIFARE Plus, and ITSO/CMD2 (card emulation path), or EMVCo Private Label Card (PLC) tokens (account-based path). That programme is gated and requires a bilateral partnership with Google. Neither MTS7 nor EMV Kernel 5 SL qualifies for the card emulation programme (MTS7 is neither MIFARE nor ITSO; EMV K5 SL is a JCB-specific EMVCo-standardised kernel, not MIFARE and not an EMVCo PLC participant).*

4.5.1 Smart Tap Protocol Overview

Smart Tap allows Google Wallet passes to transmit their redemption value to NFC reader terminals. The flow:

1. The NFC terminal identifies itself to Google Wallet using a Collector ID.
2. Google Wallet matches passes whose class definition includes that Collector ID.
3. The pass's smartTapRedemptionValue (a developer-defined string payload) is **encrypted** and transmitted to the terminal over NFC.
4. The terminal decrypts the value and processes validation.
5. The backend can update the pass object via the Google Wallet Objects API to reflect validation.

The transit operator defines the smartTapRedemptionValue string (e.g., a ticket serial number or cryptogram). All NFC communication is mediated by Google Wallet; the transit operator does not write APDU handlers.

4.5.2 Express Mode in Google Wallet

Google Wallet has an **Express Mode** setting that allows transit taps without opening the Wallet app — the device can complete a Smart Tap transaction while the screen is locked or off, subject to the device's Secure NFC setting. This is a user-controlled setting within Google Wallet; the transit operator cannot mandate or configure it.

4.5.3 Key Constraints for Transit Operators

1. **Terminal certification required:** All validators must implement the Smart Tap protocol and be provisioned with Collector IDs. This requires a partnership with Google and/or a

Smart Tap terminal provider — existing ISO 14443-4 validators cannot be used without upgrades.

2. **No direct APDU control:** The operator only controls the smartTapRedemptionValue payload string. No custom cryptographic protocol at the APDU level.
3. **Google dependency:** Pass issuance, updates, and revocation all go through Google's servers. Policy or API changes by Google affect the operator.
4. **Privacy:** Tap events and pass state updates flow through Google's infrastructure — potential GDPR data minimisation considerations.
5. **No iOS symmetry:** Google Wallet Smart Tap is Android/Wallet-specific. The iOS analogue would be Apple Wallet PassKit (a different system), breaking cross-platform protocol symmetry.

4.5.4 Smart Tap vs. Pure HCE for Samtrafiken

Dimension	Pure HCE (HostApduService)	Google Wallet Smart Tap
Protocol	Custom APDU — operator-defined, BoB-compatible	Smart Tap proprietary — string payload only
Reader infrastructure	Any ISO 14443-4 / ISO 7816-4 validator	Requires Smart Tap-certified terminals
Operator control	Full	Limited to payload string and pass metadata
Screen-off	Full control (requireDeviceScreenOn="false")	User-controlled in Google Wallet settings
No-app-needed flow	Transit app must be installed	Google Wallet pre-installed on most Android devices
Google dependency	None	High — APIs, servers, Smart Tap partnerships
BoB crypto compatibility	Full — operator designs APDU command set	Not compatible — no custom APDU layer
GDPR / data control	**Operator controls all data flows**	**Personal travel data via Google infrastructure**

Verdict for Samtrafiken: Pure HCE is the appropriate Android path for a BoB-aligned, multi-operator, interoperable transit NFC solution.
Google Wallet Smart Tap is not suitable as the primary NFC validation mechanism because it requires Smart Tap terminal deployment, does not support BoB-compatible crypto at the APDU layer, and creates Google dependency.

Google Wallet passes may be a complementary channel for pass distribution (no-app-needed experience) in a later phase.

4.5.5 Credential Provisioning

Google Wallet Smart Tap does not use on-device hardware credentials. Passes are stored in **Google's cloud** and synced to the device. There are no device-specific keys generated at provisioning time.

Trigger: User taps a "Save to Google Wallet" link or button (issued by the operator via app, website, or email).

Flow:

1. The operator generates a pass object via the **Google Wallet Objects API** and issues a signed JWT link or “Save to Wallet” button.
2. The user taps the link → Google Wallet opens and displays a pass preview.
3. The user confirms “Add”. Google’s servers retrieve and validate the pass object from the operator’s API.
4. The pass is stored in Google’s cloud and synced to the Google Wallet app on the user’s device via Google Play Services.
5. The pass is now available for Smart Tap NFC presentation at readers.

No device-bound keys: Unlike HCE or SE-based provisioning, Smart Tap passes are not cryptographically bound to a specific device. The smartTapRedemptionValue presented over NFC is computed and signed by Google’s servers based on the current pass state. The same pass syncs to all the user’s Android devices automatically.

Updates: The operator calls the Google Wallet Objects API to update pass state (mark as used, update balance, change validity). Changes propagate to all synced devices automatically — no user action required.

Revocation: The operator can expire or deactivate a pass via the API. Google syncs the revocation to the device on next connection.

Event	Outcome
App install	Not required — Google Wallet is pre-installed on most Android devices
User taps “Save to Google Wallet”	Pass saved to Google’s cloud; synced to device
Operator updates pass state	API call; syncs to all user devices automatically
User switches device	Pass automatically available on new device via Google account sync

4.6 Native Wallet Provisioning (Apple Wallet / Google Wallet)

Native wallet provisioning is a fundamentally different approach from HCE: instead of the phone running an APDU application layer that the transit operator controls, the transit credential is provisioned *into* the platform wallet itself (Apple Wallet or Google Wallet), which then handles the NFC tap. This section describes both platforms' capabilities and their constraints for Samtrafiken's card types.

4.6.1 Apple Wallet Native Transit Provisioning

Apple Wallet supports two distinct native provisioning paths for transit credentials:
Path 1A: Bilateral Apple Wallet Transit Partnership

The transit operator enters a direct commercial integration agreement with Apple.

The card's applet runs on the iPhone Secure Element, provisioned via Apple Pay's TSM infrastructure. The card appears natively inside the Apple Wallet app.

Express Transit (tap without authentication, with ECP-enabled readers) is available. Fewer than 30 transit systems globally have achieved this. Examples: Tokyo Suica, New York OMNY, Paris Navigo.

The **Navigo precedent** (May 2024) established that a European transit authority can achieve a bilateral Apple Wallet integration with a proprietary closed-loop card: Île-de-France Mobilités integrated the Navigo card — based on the open Calypso standard — after a multi-year negotiation with Apple. Calypso's publicly documented, internationally certified open standard status is likely a contributing factor.

Samtrafiken's MTS7 and EMV Kernel 5 cards do not have equivalent open standard status.

Path 1B: NFC & SE Platform (iOS 18.1+)

Apple's NFC & SE Platform allows third-party transit operators to provision a custom applet into the iPhone Secure Element without a bilateral Wallet partnership.

The credential is managed via the operator's own app (not visible in Apple Wallet). Express Mode is not available to NFC & SE Platform apps — Apple's documentation confirms that all in-app NFC transactions require the iPhone to be unlocked and the app to be in the foreground, with Express Mode remaining exclusive to native Apple Wallet cards. The operator's app must therefore be explicitly invoked before each tap.

Territory eligibility: **Sweden and all other EEA countries are not listed in any NFC & SE Platform tier**. The supported territory list has expanded since the initial iOS 18.1 launch — iOS 18.4 added Argentina, Chile, Colombia, Dominican Republic, Hong Kong, Israel, Kazakhstan, Malaysia, Mexico, Peru, Saudi Arabia, Singapore, South Africa, Taiwan, and the United Arab Emirates — but no EEA country has been added.

Apple has made the two access paths geographically distinct by policy: in the EEA, Apple opened HCE (CardSession, iOS 17.4+) under DMA obligations, while the NFC & SE Platform (hardware SE access) is available in markets such as the United States, United Kingdom,

Canada, Japan, Australia, and Switzerland.

As a result, EEA developers currently have access to one path or the other, not both.

Dimension	Path 1A: Bilateral Wallet Partnership	Path 1B: NFC & SE Platform
Credential in Apple Wallet	Yes	No — operator app only
Express Transit	Yes (ECP readers required)	No
Commercial model	Direct bilateral agreement with Apple	Commercial licence fees to Apple
Timeline	Multi-year (Navigo: years of negotiation)	Faster; application via Apple developer programme
Sweden eligible	Yes if partnership secured	No — Sweden/EEA currently excluded
Applet certification	Apple manages in partnership	Independent lab certification + Apple signing
Card protocol flexibility	**Proven for proprietary schemes (Navigo/Calypso, Suica/FeliCa)**	**Any APDU set subject to security certification**

4.6.2 Google Wallet Native Transit Provisioning

Google Wallet supports a **closed-loop transit card provisioning** programme distinct from Google Wallet passes (see the Google Wallet Smart Tap section).

Transit credentials can be provisioned into Google Wallet itself, enabling tap-and-go using the Google Wallet app. For open-loop payment credentials, Google Wallet is HCE-backed (software tokens, limited-use tokens pre-loaded for offline operation).

Native closed-loop transit cards use card-type-specific middleware — MIFARE emulation for MIFARE-based systems, EMVCo PLC tokens for account-based systems — which is architecturally distinct from the HCE payment token stack. Suica and PASMO on Google Wallet operate via a separate Japan-specific FeliCa/Osaifu-Keitai integration path, not the MIFARE or EMVCo tracks described below.
For non-Japanese markets, the programme supports **two protocol tracks** (the Japan-specific FeliCa/Osaifu-Keitai path is a separate integration outside this framework):

Track A: MIFARE-family card emulation. Google Wallet supports NFC emulation of MIFARE DESFire and MIFARE Plus cards (the latter using the CMD2 command set). This emulation is handled by Google Pay Services (a privileged system app), not the standard Android HCE layer — which is why this track requires a formal Google partnership and cannot be replicated by an operator's own HostApduService app.

Track B: EMVCo Private Label Card (PLC). Google Wallet supports an account-based token

model where the credential acts as a closed-loop account identifier using EMV contactless conveyance. The back-end recognises it as a closed-loop account, not a payment. Requires account-based fare calculation on the operator's back-end.

Track B uses the same wire protocol as open-loop EMV payment contactless: **ISO 14443-4 + EMV APDUs** (SELECT AID → GPO → READ RECORD, with Offline Data Authentication). The key difference from Track A is that no card-protocol emulation is required — the phone presents an EMV-format token to a standard EMV contactless reader, and the reader routes it to the transit backend instead of a payment network.

Validators must be **EMV Level 1 and 2 certified** and support ODA.

No other protocol tracks are supported. Arbitrary ISO 7816-4 applets, custom AIDs, and non-EMV/non-MIFARE protocols cannot be provisioned into Google Wallet's native transit programme without a bilateral partnership.

Onboarding is gated: operators must submit an interest form to Google, sign an NDA, receive review approval, and sign a bilateral commercial agreement. Commercial terms are not public. The estimated integration timeline from decision to production is lengthy — likely more than a year.

4.7 MTS1-over-NFC: Ticket Presentation via NFC Push

MTS1 defines the Mobile Ticket Bundle (MTB): a CBOR-encoded, ECDSA-signed ticket payload that Samtrafiken participants currently issue as Aztec 2D barcodes (MTS1 §3.2). Section 3.1 of the MTS1 specification states explicitly that raw transport interfaces *“MAY include NFC, Bluetooth or transfer over IP.”*

This section examines what that NFC transport would look like technically, how it compares to the travel card HCE approaches described in the preceding platform sections, and when it is appropriate to use.

4.7.1 Concept and Transport Mechanism

MTS1-over-NFC is a **push model**: the phone holds a pre-signed MTB in memory and transmits it to the reader at tap time. The reader verifies the issuer’s ECDSA signature offline. No reader-to-phone challenge is issued.
This is architecturally the opposite of travel card HCE (MTS7 and EMV Kernel 5), which are **pull models**: the reader drives a multi-step APDU exchange, issuing challenges that the phone must answer with live cryptographic operations.

An MTS1 MTB is a binary structure comprising:

- CBOR-encoded ticket payload (entitlements, validity, fare data): 200–600 bytes - JWS protected header: ~20–40 bytes
- ECDSA P-256 (ES256) issuer signature (raw): 64 bytes
- Optional HMAC-SHA256 device signature: 32 bytes
- CBOR container wrapper: ~10–20 bytes

Typical uncompressed size: 400–800 bytes. Large multi-product bundles may reach 1500–2000 bytes. ZLIB-compressed (as used for Aztec): 300–600 bytes typical.

Two viable NFC transport mechanisms:

(a) NDEF Type 4 Tag emulation (AID D2760000850101**)**

The phone implements an NFC Forum Type 4 Tag (T4T) application in its HCE service. The reader executes the standard T4T reader state machine:

1. SELECT APPLICATION by AID D2760000850101
2. SELECT FILE — Capability Container (file ID E103) → READ BINARY CC
3. SELECT FILE — NDEF Data File → READ BINARY NDEF (loop, 256 bytes per call until complete)

The MTB is wrapped in an NDEF record stored as the NDEF Data File payload. Total round trips: 5–7 APDUs.
This approach uses an NFC Forum-standardised AID and is recognisable to readers with generic NDEF reader capability. However, most deployed transit validators are not certified against NFC Forum Device Requirements and do not implement the T4T reader state machine.

Adding T4T reader mode requires a new firmware/software library in the validator, not a configuration change.

(b) Proprietary BoB AID, simple APDU transport

A new sub-AID is allocated in Samtrafiken's registered A000000781 namespace (e.g. A000000781010200) specifically for MTS1 NFC transport. The protocol is minimal:

1. Reader → Phone: SELECT by AID
2. Phone → Reader: MTB bytes (full response if ≤ 256 bytes, or 61 xx + GET RESPONSE chain per ISO 7816-4 §8.3.2 if larger)

Total round trips: 2–4 APDUs for any practical MTB size. The GET RESPONSE chaining mechanism (61 xx status word → reader issues 00 C0 00 00 <xx>) handles arbitrarily large payloads within standard ISO 7816-4 (up to 65535 bytes with extended-length APDU support; short-form GET RESPONSE delivers up to 256 bytes per call). No new NFC protocol library is needed in validators — only a new AID entry in the existing routing table.
This approach is directly analogous to MTS7's SELECT → GET NEXT DATA flow, and is the recommended transport.

4.7.2 Platform Feasibility

Attribute	Android HCE	iOS CardSession (EEA)
NDEF T4T emulation	Fully implementable in HostApduService; no platform cap	Technically possible; AID D2760000850101 is a generic NFC Forum AID — entitlement category unclear; Apple approval requires negotiation
Proprietary BoB AID	Same HostApduService mechanism as MTS7; no gating	AID in A000000781 namespace = closed-loop transit category; same entitlement path as MTS7; clearly approvable
APDU response size	No hard platform cap; heap-bound; several KB confirmed in practice	~1694 bytes per response (empirically observed by developers; not an Apple-documented limit); compressed MTB (300–600 bytes) fits in a single response; very large MTBs (>1694 bytes) require chunked reads
Screen-off operation	requireDeviceScreenOn="false" (available since Android 5.0/API 21) — same capability as MTS7	No screen-off for any CardSession app; Express Mode exclusive to Apple Wallet — same limitation as MTS7
Background operation	HostApduService wakes automatically on AID match; app not required in foreground	Field Detect mode (default contactless app) activates on NFC field entry; same as MTS7
MTB readiness at tap	MTB must be pre-loaded in the HCE service memory; app must have performed any necessary refresh before tap	Same requirement; CardSession receives APDU events only after field detection

iOS APDU chunking for large MTBs: If the MTB exceeds ~1694 bytes, the phone returns the first chunk in the SELECT response and subsequent chunks in response to GET RESPONSE commands.

Each response is individually capped at ~1694 bytes, so chunked reads are possible. For the typical compressed MTB (300–600 bytes), chunking is not required.

Proprietary AID entitlement: The A000000781 namespace is already in use for MTS7 (where it would be registered as a closed-loop transit AID). Registering an additional sub-AID in the same namespace under the same entitlement request is a straightforward extension of an existing pattern.

4.7.3 MTS1 Credential Provisioning

MTS1-over-NFC provisioning follows the same pattern as MTS7 HCE:

Step	MTS1-over-NFC
Install time	Nothing — app is an empty shell
User registration/login	Backend issues a signed MTB; if Device Signature is used, the app generates an HMAC key stored in Android Keystore / iOS Keychain
At tap time	The HCE service returns the pre-stored MTB; if Device Signature is enabled, the HMAC is computed fresh over the MTB + a Device Signature Protected Header containing the current timestamp (t)
Refresh	MTB is time-bounded by the exp field; app refreshes before expiry (push notification or background refresh at app launch)
Re-provisioning	Reinstalling the app requires a new MTB to be issued; if Device Signature is used, a new HMAC key is generated

No TSM (Trusted Service Manager) or SE applet installation is required. The issuance infrastructure (ECDSA signing, MTB structure) for operators already issuing MTS1 Aztec barcodes is reused without modification — only the delivery transport changes from optical to NFC.

Issuer public keys are distributed via DNS TXT records (_keyid._mtbkey.domainName, DNSSEC-protected) and must be pre-cached in validators. Validators with no connectivity at tap time rely on a periodic key refresh cycle. This is a new infrastructure dependency compared to MTS7, where the device public key is embedded in the MTB itself.

If Device Signature validation is required, validators need access to the MTS2 Key Derivation Key (KDK) — a symmetric key shared across all BoB Participants who validate Device Signatures. The KDK infrastructure is defined in the BoB specification but is not widely deployed as of 2026. Deploying Device Signature validation at scale is a separate operational project from deploying MTS1-over-NFC itself.

4.7.4 Security Model Comparison

MTS1-over-NFC, MTS7, and EMV Kernel 5 use different cryptographic mechanisms to achieve their security properties. The following table summarises the key dimensions.

Dimension	MTS1 without Device Signature	MTS1 with Device Signature	MTS7 HCE	EMV Kernel 5 HCE
Forgery resistance	ECDSA P-256 issuer signature — computationally infeasible to forge	Same	ECDSA P-256 issuer + cardholder signatures	AES/3DES session key cryptogram
Ticket cloning (copy MTB)	Copy works — static blob verifies on any device	HMAC under device-specific key — copy is useless without the HMAC key	Useless without ECDSA private key (hardware-backed)	Useless without AES master key (hardware-backed)
Replay protection	Static blob — replay is viable within validity window; requires backend duplicate detection	t timestamp in Device Signature Protected Header — validators enforce narrow time window (e.g. ±30 s); offline replay blocked	Fresh validator nonce per tap (INTERNAL AUTHENTICATE); signatureCounter secondary layer	ATC (Application Transaction Counter) monotonically increments; stale ATC rejected
Key extraction risk (SW keystore)	N/A (no device key)	HMAC key extractable from software storage; enables Device Signature forgery	ECDSA key extractable; enables impersonation of that device's specific ticket	Session key extractable; enables cryptogram forgery for that device
Key extraction risk (HW-backed)	N/A	Not feasible (Android Keystore StrongBox / iOS Keychain)	Not feasible (StrongBox / Secure Enclave)	Not feasible
Offline fraud	Relies entirely on	Timestamp-	Challenge-	ATC-based;

resistance	short validity window and eventual duplicate detection	bounded; offline replay blocked by ±30 s window	response intact offline; cloned MTBs useless even without connectivity	offline TC path secure
Relay attack susceptibility	**Trivially relayable (static blob)**	**prx field in DevSig header may detect relay; otherwise relayable if clock is controlled**	**Also susceptible to NFC relay (key never leaves device); relay attack requires live proxying**	**Same relay susceptibility as MTS7**

Conclusion: MTS1 with Device Signature reaches near-parity with MTS7 for replay and cloning protection, provided the Device Signature HMAC key is hardware-backed and validators enforce a narrow timestamp window. MTS1 without Device Signature has a meaningfully weaker fraud model — suitable only for low-value or low-risk scenarios, or where backend duplicate detection is reliably operational. For high-value period tickets or scenarios requiring strong offline protection, travel card HCE (MTS7 or EMV K5) remains the stronger choice.

Note that relay attacks (NFC relay/Ghost Tap class) are a threat to all three approaches at the NFC layer. The Device Signature prx field is intended to capture NFC proximity data that would differ in a relay scenario, but practical deployment of this mitigation is hardware-capability-dependent (IACR ePrint 2011/618, Resecurity: NFC Fraud Wave).

4.7.5 UX and Performance Comparison

Transaction latency:

Protocol	APDU round trips	Approximate total latency
MTS1 push (no Device Sig)	2	~80–150 ms
MTS1 push (with Device Sig)	2–3	~100–200 ms
MTS7 CL1 (challenge-response)	3	~200–400 ms
EMV Kernel 5 (full flow)	**4**	**~300–500 ms**

MTS1's two-step protocol provides a meaningful latency advantage.
MTS7 adds ~200–500 ms for ECDSA P-256 signing during INTERNAL AUTHENTICATE: StrongBox on Android is the slowest path, with TEE-backed keystores and Apple Secure Enclave both being significantly faster. EMV GENERATE AC adds AES/3DES key diversification overhead. All protocols are within the ~500 ms threshold considered acceptable for transit gate operations (Android HCE documentation: ~300 ms typical for ~1 KB exchange).

For MTS1 with a compressed MTB below 256 bytes (Le=00 short-form, meaning up to 256 bytes expected), the SELECT response carries the full MTB in a single APDU — potentially a single round trip after SELECT, making it the lowest-latency NFC tap of the three approaches.

User experience on Android: Both MTS1 and MTS7 HCE services wake from the background automatically on AID selection. Screen-off tap-and-go is supported by both via requireDeviceScreenOn="false" (available since Android 5.0/API 21). There is no protocol-level UX difference from the user's perspective.

User experience on iOS: Neither MTS1 nor MTS7 HCE supports Express Mode (screen-off, no-authentication tap) — this is exclusive to Apple Wallet. Both require the app to be registered as the default contactless app for Field Detect activation. The latency advantage of MTS1 (~100 ms faster) is perceptible but does not affect the structural UX constraints.

MTB refresh dependency: The pre-signed MTB must be current at tap time. If the MTB has expired (per the exp field), the tap will fail. This is the same dependency as MTS7, where the credential bundle is also time-bounded. Operators must implement background refresh (push notification or periodic poll) before the expiry window closes.

4.7.6 Comparison with Travel Card HCE (MTS7 and EMV Kernel 5)

Dimension	MTS1-over-NFC (push)	MTS7 HCE (card emulation)	EMV Kernel 5 HCE (card emulation)
Protocol model	Push — phone sends pre-signed blob	Pull — reader drives multi-step APDU challenge-response	Pull — standard EMV contactless flow
NFC round trips at tap	2–4	3	4
Forgery resistance	ECDSA P-256 issuer sig	ECDSA P-256 issuer + per-tap challenge	AES/3DES session key cryptogram

		sig	
Ticket cloning (no device sig)	Static blob is copyable and will verify on any device	Cryptographically infeasible (private key required)	Cryptographically infeasible (session key required)
Ticket cloning (with device sig)	Copying MTB insufficient; HMAC key also needed	N/A (per-device ECDSA key inherent)	N/A
Replay protection	Timestamp window + duplicate detection (DevSig) or validity window only (no DevSig)	Fresh validator nonce per tap + signatureCounter	ATC monotonic counter
Key extraction risk	No device key (no DevSig) or HMAC key (symmetric, shared risk)	ECDSA private key (asymmetric; hardware-backed)	AES/3DES master key (hardware-backed)
iOS platform (CardSession)	Supported; proprietary BoB AID recommended	Supported; AID A000000781010101	Supported; AID A0000008381010
Android platform	Supported; HostApduService	Supported; HostApduService	Supported; HostApduService
Screen-off tap (Android)	Yes (Android 12+)	Yes (Android 12+)	Yes (Android 12+)
Express tap (no-auth, iOS)	No — not available to HCE apps	No — not available to HCE apps	No — not available to HCE apps
BoB specification exists	No — NFC transport binding for MTS1 is not yet defined in BoB	Yes — MTS7 CL1 specification exists	Yes — via EMV Kernel 5 / SL card specification
Reader infrastructure	Requires new AID + simplified read handler in validators; ECDSA verify same as MTS7	Existing MTS7 validators	Requires EMV Kernel 5-capable validators
Validator complexity	Low — SELECT + read binary; no live crypto on validator	Medium — ECDSA verify + challenge generation	High — EMV online/offline crypto paths
Credential provisioning	No TSM; reuses MTS1 issuance infra;	No TSM; backend issues MTB with	Commercial TSM may be required;

	DNS/DNSSEC key distribution; MTS2 KDK for DevSig	device public key; per-device ECDSA keypair	EMV card credential personalisation
Multi-ticket bundle (multi-operator, single tap)	Yes — single MTB can carry tickets from multiple operators	No — one AID per operator card	No — one AID per operator card
Existing deployments	None (NFC path); MTS1 Aztec widely deployed in Sweden	BoB-compliant deployment; limited field pilots as of 2026	SL (Stockholm) closed-loop transit card

4.7.7 Assessment

Technical feasibility: MTS1-over-NFC is technically feasible on both Android and iOS via HCE. On both platforms, the proprietary BoB AID approach (simple SELECT → MTB response) is the recommended transport: it is shorter (fewer round trips), simpler to implement in validators, and has a clear entitlement path on iOS.
The fundamental architectural trade-off: MTS1-over-NFC shifts the per-tap replay and anti-cloning burden from a live cryptographic challenge (MTS7) to the Device Signature mechanism and short validity windows. The security model is:

- Without Device Signature: weakly bounded by validity window; suitable for low-value single-use tickets or where backend duplicate detection is operationally reliable

- With Device Signature: near-parity with MTS7 for replay and cloning, provided HMAC keys are hardware-backed and validators enforce the timestamp window; the MTS2 KDK infrastructure adds a shared-symmetric-key risk not present in MTS7's asymmetric model

- **No current validator support**: the NFC transport binding is not yet defined in BoB and is therefore not implemented in any existing validators.

When MTS1-over-NFC is the better choice:

- The operator already issues MTS1 Aztec barcodes, has no current NFC-based travel card and wants to add NFC with minimal new infrastructure — the issuance pipeline, ECDSA signing, and MTB structure are unchanged
- Protocol simplicity is valued — simpler validator logic to implement
- Multi-operator single-tap ticketing is required — a single MTB can bundle tickets from multiple Participants, reducing the need for the passenger to manage multiple card emulation services
- Validators are being newly deployed (greenfield), where the AID routing table and ECDSA verification can be defined from scratch

When travel card HCE (MTS7 or EMV K5) is the better choice:

- The strongest per-tap replay protection is required, particularly for high-value or long-validity tickets where the window of exploitation from a compromised MTB is large
- Existing reader infrastructure is APDU-based and already deployed for MTS7/EMV, where retrofitting a new AID is straightforward but the MTS7 challenge-response logic is already present
- The MTS2 KDK infrastructure for Device Signature is not yet deployed and operators are unwilling to accept MTS1 without Device Signature for the ticket value levels being issued

BoB specification gap: MTS1 NFC transport is not yet defined in the BoB specification. The required specification work is:

- A new sub-AID in the A000000781 namespace for MTS1 NFC
- An APDU command set (SELECT AID → MTB response with GET RESPONSE chaining)
- Fragmentation rules for MTBs exceeding the ISO-DEP frame size
- Guidance on Device Signature inclusion, timestamp window enforcement, and issuer key caching in validators

This is a smaller specification effort than defining a new APDU card application from scratch, since the MTS1 MTB format and ECDSA/Device Signature semantics are already defined in MTS1 and MTS2.

The prerequisite for production deployment is a **BoB MTS1 NFC transport sub-specification and validator firmware updates to add the new AID.** The security level deployed (with or without Device Signature) should be aligned to the value and fraud risk profile of the tickets being issued.

5 Recommendation

5.1 Recommended main option: iOS HCE EEA + Android HCE

Primary recommendation: iOS HCE (CardSession API, iOS 17.4+, EEA) combined with Android HCE (HostApduService).

5.1.1 Rationale

1. The only viable option today. iOS NFC & SE Platform is blocked because Sweden is not in Apple's supported territory list. Apple Wallet Express Transit requires a bilateral Apple partnership that is not self-service and may take years. Native wallet provisioning is blocked because neither MTS7 nor EMV Kernel 5 are compatible with Google Wallet's programme, and Apple Wallet requires either a multi-year bilateral negotiation or Sweden to be added to the SE Platform territory list. Only iOS HCE (EEA) + Android HCE can be started immediately on both platforms.
2. Full BoB compatibility. iOS HCE + Android HCE allows Samtrafiken to define the APDU command set, ticket encoding, and cryptographic protocol entirely within the BoB standard. No Apple or Google intermediation at the NFC protocol layer.

The same BoB NFC profile runs on both platforms.

3. Operator autonomy and zero platform fees. HCE is free on both platforms. The iOS entitlement is DMA-protected — non-discriminatory, with a formal dispute resolution mechanism. Samtrafiken retains control of credential management, key infrastructure, revocation, and data flows.

4. Platform symmetry. A single BoB NFC APDU specification covers both iOS and Android apps. Reader infrastructure does not need to differentiate between platforms. This simplifies both development and long-term maintenance.

5. Android express transit is fully achievable. With android:requireDeviceScreenOn="false" and android:requireDeviceUnlock="false" (Android 12+), Android achieves genuine pocket tap-and-go without user interaction — directly equivalent to a physical contactless card. This covers the majority of the Swedish market.

6. iOS UX limitation is acceptable. The absence of Express Mode on iOS HCE (user must authenticate, or must set the transit app as the default contactless app) is a real limitation but is manageable. Field Detect mode (when the transit app is the default contactless app) provides near-seamless validation with authentication. Most high-frequency commuter devices in the Swedish market are Android; iOS users are a significant but secondary segment. The iOS UX can be documented and communicated.

5.1.2 Known Risk: iOS Authentication at Tap

The most significant ongoing risk is iOS authentication friction. If the operator's transit app is not the user's default contactless app, the user must open the app and tap a button within 15

seconds before holding the phone to the reader. Even as the default app, Face ID/Touch ID is required on locked devices. This should be: - Evaluated in a real-world PoC at transit validators before full deployment - Communicated clearly to users in onboarding - Raised explicitly with Apple in the meeting (see Section 7)

5.2 Recommendation: MTS1-over-NFC (for operators already issuing MTS1 Aztec barcodes)

Recommendation: For operators who currently only issue MTS1 Aztec barcodes and want to add NFC support with minimal new infrastructure, MTS1-over-NFC (proprietary BoB AID, simple SELECT → MTB push transport) is a viable complement to the primary HCE recommendation.

5.2.1 Rationale

1. Reuses existing issuance infrastructure. The MTS1 MTB format, ECDSA signing, and issuance pipeline are unchanged. Only the delivery transport changes from optical to NFC. No TSM, no new credential personalisation, and no per-device key provisioning ceremony beyond what is already in place for MTS1 Aztec.
2. Lowest validator complexity. The validator logic is: SELECT AID → read MTB → verify ECDSA P-256 issuer signature. There is no live challenge-response, no counter state, and no per-session key management. For greenfield validator deployments, this is the simplest NFC profile to implement.

3. Lowest tap latency. With a compressed MTB under 256 bytes, the full ticket is delivered in 2 APDU round trips (~80–150 ms). This is faster than MTS7 (3 round trips, plus StrongBox signing overhead) and EMV Kernel 5 (4+ round trips).

4. Multi-operator single-tap capability. A single MTB can bundle tickets from multiple Participants, which is not possible with travel card HCE (one AID per operator card). This is particularly relevant for Samtrafiken's multi-operator national context.

5. Same platform support as MTS7. MTS1-over-NFC uses the same HCE mechanism on both Android and iOS. The proprietary BoB AID (A000000781 namespace) follows the same iOS entitlement path as MTS7 and can be registered in the same entitlement request.

5.2.2 Conditions and Prerequisites

MTS1-over-NFC has some prereuisites that has to be met:

- **No BoB specification exists** for the MTS1 NFC transport. A sub-specification defining the AID, APDU command set, fragmentation rules, and Device Signature guidance must be written before production deployment.
- **No validator support exists.** All deployed validators require firmware updates to recognise the new AID and implement the MTB read and ECDSA verification logic.
- **Security model requires alignment.** MTS1 without Device Signature is only appropriate for low-value or single-use tickets with reliable backend duplicate detection. For high-value or long-validity tickets, Device Signature (MTS2) must be implemented —

and the MTS2 KDK infrastructure is not yet widely deployed.

5.2.3 When to prefer MTS1-over-NFC over MTS7 HCE

Scenario	Preferred approach
Operator has existing MTS7 infrastructure	MTS7 HCE — validator and issuance infrastructure already exists
Operator issues only MTS1 Aztec today, no travel card	MTS1-over-NFC — reuses existing issuance pipeline
High-value period tickets requiring strong per-tap replay protection	MTS7 HCE — challenge-response with hardware-backed key
Multi-operator single-tap ticketing required	MTS1-over-NFC — single MTB can bundle multiple operator tickets
Greenfield validator deployment	Either; MTS1-over-NFC has lower validator implementation complexity

6 Questions for Apple

This document represents a joint initiative by public transport organisations in Sweden, including Samtrafiken i Sverige AB, Trafikförvaltningen (Region Stockholm, SL), Västtrafik (Region Västergötland), Skånetrafiken (Region Skåne) and UL (Region Uppsala).

The purpose is to evaluate how NFC-based solutions on iPhone can be used to support these use cases in public transport across Sweden in a consistent and interoperable way.

Our evaluation focuses on two primary use cases:

1. Digital ticket

   A single ticket or period ticket stored on the device and presented via NFC for validation.

2. Account-based travel identifier

   A stable identifier presented via NFC and resolved in backend systems, for example based on BoB MTS7 or EMV closed loop.

The purpose of this document is to clarify what is technically and commercially possible on iPhone for these use cases, and which implementation paths Apple would recommend in a national public transport context.

Example scenario:

A traveller taps their iPhone at a public transport validator (e.g. gate, onboard reader, or platform validator).

The interaction must be fast and reliable, and support one or both of the following use cases:
- digital ticket validation
- account-based travel identifier validation

Open-loop bank card payments are out of scope.

6.1 Viable iOS NFC Options for Transit Card Validation

What NFC mechanisms does Apple make available on iPhone that could support:
(a) digital ticket presentation, and
(b) account-based travel identifier presentation,
in a public transport validation scenario such as Sweden?

There are two transit card standards that validators in Swedish buses, trams, and stations need to read via NFC. Both are currently deployed on physical smartcards; the

goal is to enable passengers to use iPhone as a direct replacement.

The two card standards:

- **MTS7** (Used by 7 PTAs including Skånetrafiken, and chosen by Västtrafik for future implementation): open standard (BoB) for challenge-response protocol over ISO 14443-4 NFC-A, AID prefix A000000781. The validator initiates a 3-step APDU exchange — SELECT, GET NEXT DATA (retrieves a signed credential bundle), INTERNAL AUTHENTICATE (the phone proves possession by signing a validator-issued challenge with an ECDSA P-256 key). The card design does not involve Apple Wallet.
- **EMV Kernel 5 SL** (Stockholm public transport, SL, and chosen by UL for future implementation): closed-loop transit contactless card, AID A000000838101010, using the JCB-defined EMV Kernel 5 APDU flow (SELECT, GET PROCESSING OPTIONS, READ RECORD, GENERATE AC). Used for transit access control, not payment. The card design does not involve Apple Wallet.

Questions:

- What iPhone NFC mechanisms does Apple make available that could support:
  (a) digital ticket validation, and
  (b) account-based travel identifier validation,
  at a transit gate?
- Which of those mechanisms would Apple recommend for each of these use cases —
  (a) digital ticket and
  (b) account-based travel identifier —
  in a national public transport context such as Sweden?
- Are there integration paths or platform capabilities public transport authorities in Sweden should be aware of that are not covered in Apple's public developer documentation?

6.2 Technical Architecture Validation

2. Validating our understanding of CardSession HCE for transit card emulation:

The public transport authorities in Sweden want to use the iOS HCE (EEA) CardSession API (com.apple.developer.nfc.hce entitlement, iOS 17.4+, available in Sweden as an EEA country) to emulate two transit card standards on iPhone. Both flows run over ISO 14443-4 NFC-A, require no Apple Wallet integration, and keep all transaction data between the phone app and the transit validator — no Apple servers involved.

The two card standards:

- **MTS7** (Used by 7 PTAs including Skånetrafiken, and chosen by Västtrafik for future implementation): proprietary ECDSA P-256 challenge-response protocol, AID prefix A000000781, 3-step APDU flow: SELECT → GET NEXT DATA (returns a signed certificate bundle up to ~1700 bytes) → INTERNAL AUTHENTICATE (32-byte SHA-256 challenge).

- **EMV Kernel 5 SL** (Stockholm public transport, SL): closed-loop contactless applet, AID A000000838101010, standard EMV APDU commands: SELECT, GET PROCESSING OPTIONS, READ RECORD, GENERATE AC.

Questions:

- Is this use case — account-based travel identifier using MTS7 or EMV — within the intended scope of the CardSession entitlement for transit operators?
- Is our understanding of the API capabilities technically accurate for supporting an account-based travel identifier at a transit gate — particularly APDU routing by AID prefix, response buffer limits, and the activation model (Field Detect as default contactless app, or In-App Presentment via NFCPresentmentIntentAssertion)?
- Are there constraints or behaviours not reflected in the public developer documentation that public transport authorities in Sweden should be aware of before committing to this architecture?

3. **Validating our understanding of MTS1-over-NFC on iOS (CardSession):**

The public transport authorities in Sweden are evaluating a NFC presentation model called MTS1-over-NFC. MTS1 defines a Mobile Ticket Bundle (MTB): a CBOR-encoded, ECDSA P-256-signed (ES256) ticket payload currently delivered as an Aztec 2D barcode. The proposal is to deliver the same pre-signed MTB over NFC using a push model — the phone holds the MTB in memory and transmits it at tap time; no reader-to-phone challenge is issued.

The proposed iOS implementation:

- **Entitlement**: same com.apple.developer.nfc.hce entitlement as MTS7, with an additional sub-AID in the A000000781 namespace (e.g. A000000781010200) registered alongside the MTS7 AID
- **APDU flow**: 2-step — reader sends SELECT by AID; phone returns the full MTB in the SELECT response (single response if ≤ ~1694 bytes; 61 xx / GET RESPONSE chaining per ISO 7816-4 §8.3.2 for larger payloads)
- **Payload size**: typically 300–600 bytes (ZLIB-compressed); up to ~1500–2000 bytes for large multi-product bundles
- **Activation and screen behaviour**: identical to MTS7 — Field Detect (default contactless app) or In-App Presentment; no Express Mode

Questions:

- Is registering multiple sub-AIDs within the same A000000781 namespace under a single CardSession entitlement within Apple’s intended scope for the closed-loop transit entitlement category?
- Is our understanding of the ~1694-byte per-response limit accurate, and is ISO 7816-4 GET RESPONSE chaining (61 xx status → reader issues 00 C0 00 00 <Le>) supported as described in the CardSession API?
- Is there anything in the CardSession APDU handling model that would prevent a digital ticket use case, based on a push-model protocol (SELECT → return binary blob, no challenge-response), from functioning correctly?

6.3 HCE (CardSession) — Confirming the Recommended Foundation

4. **Commercial terms and obligations:** Apple states that the CardSession HCE entitlement is available free of charge under DMA commitments. Are there any indirect obligations, certification requirements, or ongoing compliance costs that the public transport authorities in Sweden should budget for? Are the terms identical for all EEA transit operators, or can Apple negotiate deviations?

5. **Entitlement process — realistic timeline for a national transit authority:** From submission of the entitlement request to receiving approval for a production app, what is the realistic calendar timeline for a Swedish public transit standards body? Is there a dedicated review track or point of contact for national-scale operators rather than individual developers?

6. **User experience at the validator — the authentication question:** Today, a traveller using HCE on iOS must either have the operator's transit app set as their default contactless app (Field Detect) or open the app within 15 seconds before tapping (In-App Presentment). In practice, how do existing European transit operators using iOS HCE handle this friction for:
   (a) digital ticket use cases and
   (b) account-based travel identifier use cases?

   Does Apple have UX data or field experience showing this is acceptable to commuters, or is it considered a significant barrier?

7. **Impact on existing Apple Pay and bank card users:** If a traveller sets the operator's transit app as their default contactless app, will their bank cards in Apple Wallet continue to work for payments without any change to their setup? We want to be confident that choosing the operator's transit app as default does not disrupt payment behaviour or confuse users.

8. **Long-term access guarantee:** The DMA commitments run for 10 years. In practical terms, what governance or dispute mechanism exists to ensure that the entitlement terms, API stability, and access conditions do not change in ways that would strand an operator who has invested in an HCE-based production deployment?

9. **Data privacy — Apple's access to transit transaction data:** In the HCE architecture, APDU commands pass between the reader and our app running on the phone. Apple's servers are not involved in the transaction. Can Apple confirm this data flow, and provide documentation or a formal statement the public transport authorities in Sweden can reference in their GDPR data processing assessments?

10. **Reference operators — lessons from early European HCE deployments:** Which European transit operators or national transport authorities have already launched with CardSession HCE? What practical challenges did they encounter, and would Apple facilitate an introduction so the public transport authorities in Sweden can learn from their experience?

6.4 Express Transit (Apple Wallet Bilateral) — Evaluating the iOS Gap

11. **The Express Transit gap:** The most significant iOS limitation under the recommended iOS HCE (EEA) + Android HCE approach is that HCE apps require authentication per tap, even when set as the default contactless app. As the default contactless app, the phone activates on NFC field-detect without a prior in-app gesture — but Face ID/Touch ID is still required on a locked device. As a non-default app, an additional prior in-app gesture (invoking NFCPresentmentIntentAssertion) is also required. Neither path delivers the passive tap-and-go that physical cards and Android (screen-off) provide. For both digital ticket and account-based travel identifier use cases, the main usability gap on iPhone compared to physical cards and Android is the lack of passive “tap-and-go” (Express Mode equivalent). Is it realistic to achieve a similar user experience for either of these use cases on iPhone, and if so, through which Apple-supported path?

12. **Requirements to become an Apple Wallet transit partner:** This initiative represents public transport authorities in Sweden acting jointly at national scale — comparable in scale to Île-de-France Mobilités (Navigo). What are Apple’s minimum requirements for a national transit body — such as a jointly coordinated Swedish initiative — to apply for a bilateral Apple Wallet transit card partnership? What evidence of ridership, technical readiness, or standard maturity does Apple need to see?

13. **Timeline and cost of a Wallet partnership:** If public transport authorities in Sweden, acting through a jointly coordinated national initiative, expressed interest in pursuing an Apple Wallet native transit card (via bilateral partnership or SE Platform), what is the realistic end-to-end timeline from initial expression of interest to a passenger-facing deployment? What are the cost categories involved — certification, TSM, commercial licence, ongoing fees?

6.5 NFC & SE Platform — Sweden’s Exclusion

15. **Sweden’s exclusion from NFC & SE Platform — roadmap and timeline:** Sweden is not currently in Apple’s list of supported territories for the NFC & SE Platform. Is Apple able to share a roadmap or indicative timeframe for when Sweden — and the broader EEA — will be added? This is a material factor in the long-term platform strategy of public transport authorities in Sweden.

16. **SE Platform vs. HCE for transit — security and UX difference:** The NFC & SE Platform stores credentials in hardware rather than software, which is a security advantage. However, our research indicates the user experience at the transit gate would be essentially the same as HCE — authentication is still required per tap, and Express Mode is not available to SE Platform apps. Can Apple confirm this, and explain what meaningful operational or security advantage the SE Platform offers over HCE for a closed-loop transit scenario, beyond key storage?

17. **Apple’s recommended path for a Swedish national transit operator:** The public transport authorities in Sweden is deciding between three Apple paths — HCE

(CardSession, available now), SE Platform (blocked by territory exclusion), and bilateral Apple Wallet partnership (multi-year negotiation).

From Apple's perspective, which path is most realistic for supporting:
(a) digital ticket and
(b) account-based travel identifier
for a national public transport implementation in Sweden within a 2–3 year horizon?

If the answer is HCE now with a Wallet partnership later, what would public transport authorities in Sweden need to demonstrate or prepare — for each of these use cases — in the HCE phase to be in the strongest position for a future Wallet conversation?

18. **MTS1-over-NFC entitlement — sub-AID bundling:** Public transport authorities in Sweden are evaluating a NFC transport that pushes pre-signed MTS1 ticket bundles using a new sub-AID in the registered A000000781 namespace (e.g. A000000781010200), distinct from the MTS7 AID (A000000781010101). Can this sub-AID be registered in the same entitlement request as MTS7, or does it require a separate entitlement application? Is a push-model ticket presentation (SELECT → return binary blob, no challenge-response) within the intended scope of the closed-loop transit entitlement category?

7 Questions for Google

This document represents a joint initiative by public transport organisations in Sweden, including Samtrafiken i Sverige AB, Trafikförvaltningen (Region Stockholm, SL), Västtrafik (Region Västra Götaland), Skånetrafiken (Region Skåne) and UL (Region Uppsala).

The purpose is to evaluate how NFC-based solutions on Android devices can be used to support public transport across Sweden in a consistent and interoperable way.

Our evaluation focuses on two primary use cases:

1. Digital ticket
   A single ticket or period ticket stored on the device and presented via NFC for validation.
2. Account-based travel identifier
   A stable identifier presented via NFC and resolved in backend systems, for example based on BoB MTS7 or EMV closed loop.

Android-based implementations already support NFC card emulation via HCE. The purpose of these questions is therefore to clarify practical capabilities, limitations, and recommended implementation patterns for these use cases in real-world public transport environments.

Example scenario:

A traveller taps their Android device at a public transport validator (e.g. gate, onboard reader, or platform validator).

The interaction must be fast and reliable, and support one or both of the following use cases:

- digital ticket validation
- account-based travel identifier validation

The solution must work consistently across Android devices and OEMs in real-world public transport environments.

Open-loop bank card payments are out of scope.

7.1 Viable Android NFC Options for Transit Card Validation

1. **What NFC mechanisms does Android provide that can reliably support:**

   (a) digital ticket presentation, and

   (b) account-based travel identifier presentation

   in a public transport validation scenario such as Sweden?

   Public transport authorities in Sweden operate two transit card standards that validators in buses, trams, and stations across Sweden need to read via NFC. Both are currently

deployed on physical smartcards; the goal is to enable passengers to use an Android device as a direct replacement.

The two card standards:

- **MTS7** (BoB-based national travel card used by multiple public transport authorities in Sweden): open standard (BoB) for challenge-response protocol over ISO 14443-4 NFC-A, AID prefix A000000781. The validator initiates a 3-step APDU exchange — SELECT, GET NEXT DATA (retrieves a signed credential bundle), INTERNAL AUTHENTICATE (the phone proves possession by signing a validator-issued challenge with an ECDSA P-256 key). The card design does not involve Google Wallet.
- **EMV Kernel 5 SL** (Stockholm public transport, SL): closed-loop transit contactless card, AID A000000838101010, using the JCB-defined EMV Kernel 5 APDU flow (SELECT, GET PROCESSING OPTIONS, READ RECORD, GENERATE AC). Used for transit access control, not payment. The card design does not involve Google Wallet.

Questions:

- What Android NFC mechanisms does Google make available that could support:
  (a) digital ticket validation, and
  (b) account-based travel identifier validation
  at a transit gate?
- Which of those mechanisms would Google recommend for each of these use cases —
  (a) digital ticket and
  (b) account-based travel identifier —
  in a public transport system such as Sweden, taking into account real-world device and OEM variation?
- Are there integration paths, limitations, or platform capabilities that public transport authorities in Sweden should be aware of, including any OEM-specific behaviours or inconsistencies not covered in the public Android developer documentation?

7.2 Technical Architecture Validation

2. Validating our understanding of Android HCE for transit card emulation:

Public transport authorities in Sweden plan to use Android Host-based Card Emulation (HostApduService, android.nfc.cardemulation, available since Android 4.4) to emulate two transit card standards. All APDU exchanges occur directly between the validator reader and the app — no Google servers are involved in the transaction.

The two card standards:

- **MTS7** (BoB-based national travel card used by multiple public transport authorities in Sweden): open standard (BoB) using an ECDSA P-256 challenge-response protocol over ISO 14443-4 NFC-A, AID prefix A000000781, 3-step APDU flow: SELECT →

GET NEXT DATA (returns a signed certificate bundle up to ~2 KB) → INTERNAL AUTHENTICATE (32-byte SHA-256 challenge, ECDSA P-256 signature response).
- **EMV Kernel 5 SL** (Stockholm public transport, SL): closed-loop contactless applet, AID A000000838101010, standard EMV APDU commands: SELECT, GET PROCESSING OPTIONS, READ RECORD, GENERATE AC.

The deployment will also use AID conflict resolution to route both AIDs exclusively to the transit app, enable screen-off NFC via android:requireDeviceScreenOn="false" (Android 12+), and bind credentials to the device using Android Keystore hardware-backed key attestation.

Questions:

- Is our understanding of how HostApduService handles AID routing, APDU buffer sizes, screen-off behaviour, and Keystore-backed credential binding technically accurate for supporting an account-based travel identifier at a transit gate?
- Are there platform behaviours, OEM deviations, or Android version-specific constraints not fully reflected in the public Android developer documentation that public transport authorities in Sweden should account for in this architecture, particularly in high-throughput validation environments?
- What are the expected APDU response time constraints for reliable operation at transit gates, and are there known latency variations across devices or OEMs that could affect this?

3. **Validating our understanding of MTS1-over-NFC on Android (HostApduService):**

Public transport authorities in Sweden are evaluating a NFC presentation model called MTS1-over-NFC. Part of the BoB standard, MTS1 defines a Mobile Ticket Bundle (MTB): a CBOR-encoded, ECDSA P-256-signed (ES256) ticket payload currently delivered as an Aztec 2D barcode. The proposal is to deliver the same pre-signed MTB over NFC using a push model — the phone holds the MTB in memory and transmits it at tap time; no reader-to-phone challenge is issued.

The proposed Android implementation:

- **Framework**: same HostApduService mechanism as MTS7; a new sub-AID in the A000000781 namespace (e.g. A000000781010200) registered alongside the MTS7 AID in the app's apduservice.xml
- **APDU flow**: 2-step — reader sends SELECT by AID; phone returns the full MTB in the SELECT response (single response if ≤ 256 bytes with Le=00; ISO 7816-4 61 xx / GET RESPONSE chaining for larger payloads)
- **Payload size**: typically 300–600 bytes (ZLIB-compressed); up to ~1500–2000 bytes for large multi-product bundles
- **Screen-off**: same android:requireDeviceScreenOn="false" configuration as MTS7 (Android 12+)
- **Device binding**: optional HMAC-SHA256 Device Signature using a key stored in

Android Keystore; no TSM required

Questions:

- Is registering multiple sub-AIDs within the same A000000781 namespace in a single HostApduService declaration technically supported and correct?
- Is GET RESPONSE chaining (61 xx status word → reader issues 00 C0 00 00 <Le>) handled automatically by the Android HCE framework, or must the app implement the chaining logic explicitly?
- Are there APDU buffer size limits or response length constraints in HostApduService that would affect delivery of MTBs larger than 256 bytes that public transport authorities in Sweden should be aware of?

7.3 Android HCE — Confirming the Recommended Approach

4. **Commercial model — no fees, no gating:** Android HCE (HostApduService) is documented as a standard Android framework API with no associated commercial agreement or Google approval process. Can Google formally confirm there are no current or planned fees, certification requirements, or approval gates for transit operators using HostApduService? Is this API considered stable and long-term supported?

5. **Screen-off NFC in the Swedish Android market:** Android 12+ supports tap-and-go with the screen fully off — directly equivalent to a physical card. What share of Android devices currently in use in Sweden run Android 12 or later? Is this data something Google can provide or point to, so public transport authorities in Sweden can estimate how many users would have the full screen-off experience at launch?

6. **Screen-off user experience — haptics, feedback, notifications:** When a screen-off NFC tap succeeds, what does the user experience? Is there haptic feedback? Does the screen activate to show a confirmation? We want to understand the end-to-end passenger experience before committing to this as the primary Android UX for:
(a) digital ticket use cases and
(b) account-based travel identifier

   use cases, and whether this behaviour is consistent across major Android OEMs sold in Sweden (Samsung, Google Pixel, Xiaomi, OnePlus).

7. **Secure NFC — prevalence and operator impact:** Android 10+ includes a user setting called “Secure NFC” that, when enabled, blocks all card emulation with the screen off. How widely is this setting enabled by default on Android devices sold in Sweden? Can transit operators do anything to guide users who have this setting enabled, or detect it programmatically and prompt them to disable it? Are there OEM-specific differences in how this setting is implemented or exposed to users?

8. **Data privacy — Google’s access to transit transaction data:** In the HCE architecture, NFC APDU exchanges happen directly between the reader and the operator’s app. Google Play Services and Google servers are not involved in the transaction. Can Google formally confirm this data flow, and provide documentation public transport

authorities in Sweden can use in their GDPR data processing assessments?

9. **Reference operators — experience with Android HCE transit deployments:** Which European or Nordic transit operators have deployed production services using Android HostApduService for ticket validation? What were the key operational challenges — validator compatibility, user adoption, device fragmentation? Would Google facilitate introductions so public transport authorities in Sweden can learn from their experience?
10. Background execution and power management:

    Are there Android power management features (e.g. Doze mode, app standby, OEM battery optimisations) that could affect the reliability of HCE services in real-world use? Are there best practices or requirements to ensure consistent behaviour across devices?

7.4 Google Wallet — Evaluating Native Integration

10. **Google Wallet native transit — current state of compatibility:** The two card standards used by public transport authorities in Sweden — MTS7 (BoB-based open standard using an ECDSA-based applet) and EMV Kernel 5 SL (closed-loop transit using the JCB-specific EMVCo kernel, 19-digit PAN, non-payment BIN with no tokenization infrastructure) — do not fit Google Wallet's current MIFARE or EMVCo PLC tracks. Does Google have a roadmap for supporting proprietary transit protocols in Google Wallet, or a "bring your own protocol" track under consideration? Is there a path for a national or jointly coordinated transit authority structure to pursue native Google Wallet integration without a full card specification redesign?
11. **Conditions for Google Wallet transit partnership:** If public transport authorities in Sweden were willing to invest in adapting their card specifications to meet Google Wallet's requirements, what exactly would that entail? What are the minimum technical, commercial, and operational requirements for a national transit operator to become a Google Wallet transit programme partner?
12. **Strategic fit — HCE now, Google Wallet later:** If public transport authorities in Sweden launch direct HCE first, is there a migration path to add Google Wallet integration for the same credentials in a later phase? Does this differ between:

    (a) digital ticket use cases and

    (b) account-based travel identifier

    use cases? Or would native Google Wallet integration require a separate, parallel implementation that runs alongside HCE rather than replacing it?
13. **Google's recommended path for a Swedish national transit operator:** Public transport authorities in Sweden are choosing between two Android paths — direct HCE (HostApduService, available now, full protocol freedom) and native Google Wallet integration (gated partnership, requires card specification redesign for MIFARE or EMVCo PLC compatibility). From Google's perspective, which path offers the better

passenger experience and operator value proposition for:

(a) digital ticket and

(b) account-based travel identifier

in a national public transport context such as Sweden? If HCE is the realistic near-term path, what would public transport authorities in Sweden need to invest in or demonstrate — for each of these use cases — to open a Google Wallet integration discussion in a later phase?

7.5 Security and Long-Term Commitment

14. **Android security recommendation for transit credentials:** Public transport authorities in Sweden plan to use Android Keystore hardware attestation to bind transit credentials to individual devices. Does Google recommend additional security measures — such as Play Integrity API checks or Device Trust signals — that transit operators should layer on top of Keystore attestation to defend against credential cloning or provisioning to compromised devices, for both digital ticket and account-based travel identifier use cases?
15. **Long-term Android NFC roadmap:** Is there anything in Google's NFC or Android roadmap that could affect a transit HCE deployment built on HostApduService — for example, changes to AID routing, new default payment app policies, or new security models? Public transport authorities in Sweden are making a multi-year investment and need confidence in platform stability.

8 References

8.1 Standards and Protocols

- ISO 14443 – Contactless smart card standard (Parts 1–4)
- ISO 7816-4 – Interindustry commands for smart card APDUs
- ISO 7816-5 – Registration of application providers (AID / RID)
- EMVCo: EMV Contactless Specifications for Payment Systems – overview and specifications portal (full Book C-5 PDF requires EMVCo membership)
- EMVCo: EMV Contactless Chip technology overview (describes the Book A / B / C-x kernel structure)
- EMVCo: Kernel C-5 Implementation Conformance Statement (publicly available conformance statement for Kernel 5 / JCB)
- EMVCo: EMV Contactless Quick Reference Guide (2023) (publicly available overview of contactless kernel architecture)
- ISO/IEC 24014 – Intelligent transport systems: public transport interoperable fare management system (ISO standard; link omitted — ISO catalogue pages are paywalled

and the URL previously recorded was incorrect)

8.2 Apple Developer Documentation

- Apple: HCE-based NFC Transactions for EEA Apps
- Apple: NFC & SE Platform for Secure Contactless Transactions
- Apple: DMA and Apps in the EU
- Apple Wallet Transit Programme
- CoreNFC — CardSession Documentation
- NFCPresentmentIntentAssertion Documentation
- com.apple.developer.nfc.hce.iso7816.select-identifier-prefixes Entitlement
- com.apple.developer.nfc.hce.default-contactless-app Entitlement
- Apple Security Guide – Adding transit and eMoney cards
- Apple Support – Use Express Mode in Apple Wallet
- Apple Newsroom: Developers can offer in-app NFC transactions using the Secure Element (Aug 2024)
- Apple Newsroom: Navigo for iPhone and Apple Watch (May 2024)
- Apple Support: How to use your Navigo pass with iPhone
- Île-de-France Mobilités: Transit card on iPhone
- Calypso Networks Association: Look inside your Apple Wallet
- Calypso Networks Association: Calypso for mobile
- Apple Security Guide: NFC & SE Platform security
- Apple Developer Forums – HCE EEA Possibilities (Thread 759442)
- Apple Developer Forums – CardSession (Thread 764008)
- Apple Developer Forums – HCE CredentialSession (Thread 762092)

8.3 Android / Google Documentation

- Android: Host-based Card Emulation Overview
- CardEmulation API Reference
- Android Keystore System
- Android Key Attestation
- Google Wallet Smart Tap — Overview
- Google Wallet Smart Tap — Communication Flow
- Google Wallet Transit Passes
- Use Google Wallet for transportation
- Google Wallet: Closed Loop Transit Overview
- Google Wallet: Closed Loop Transit Prerequisites
- Google Wallet: Open Loop Basic Requirements

8.4 Samtrafiken / BoB

- BoB API Documentation
- MTS1 v1.7.2 – Mobile Ticket Specifications: Mobile Ticket Bundle (Samtrafiken BoB specification; defines MTB structure, Issuer Signature ES256/RS256, Device Signature

HS256, and permitted raw transports including NFC)

- MTS7 v1.1.0 – Mobile Ticket Specifications: Travel Card Application (Samtrafiken, 2019-09-05)
- BoB Token API – ID-based travelling
- BoB Standard Introduction
- Samtrafiken BoB Overview

8.5 Regulatory

- EU Digital Markets Act (DMA)
- European Commission: DMA Interoperability Q&A
- Apple EC Commitments document AT 40452 (July 2024)

8.6 Third-Party Analysis

- AtaDistance: iOS 17.4 HCE Analysis
- AtaDistance: iOS 18.1 NFC & SE Platform Analysis
- TechCrunch: Apple opens NFC chip to developers (Aug 2024)
- 9to5Mac: Apple NFC chip open to third-party apps (Aug 2024)
- NFCW: Skånetrafiken enables Apple Pay Express Mode for contactless bus ticketing in southern Sweden (Oct 2021)
- CardWerk: ISO 7816-4 APDU Reference

9 Glossary

AID – Application Identifier. A byte string (5–16 bytes) that uniquely identifies a card application on a smart card or HCE host. NFC readers use a SELECT (by AID) command to activate a specific application (ISO/IEC 7816-5).
APDU – Application Protocol Data Unit. The command/response message format used between an NFC reader and a card application, defined in ISO/IEC 7816-4. Commands carry class byte (CLA), instruction (INS), parameters (P1, P2), and optional data; responses end with a two-byte status word (SW).

BIN – Bank Identification Number. The first 6–8 digits of a PAN that identify the card issuer. In closed-loop transit, a dedicated BIN may be assigned to a transit operator to distinguish its card numbers from payment cards.

CBOR – Concise Binary Object Representation. A compact binary data serialisation format (RFC 8949) used by MTS1 to encode the Mobile Ticket Bundle payload. CBOR is schema-flexible and produces smaller payloads than JSON, making it well-suited for NFC transport where byte count matters.

BoB – Biljett och Betalsystem (“Ticket and Payment System”). Samtrafiken’s national interoperability standard for e-ticketing in Swedish public transport.

DMA – Digital Markets Act. EU Regulation 2022/1925, which among other provisions compelled Apple to open NFC card emulation to third-party apps in the EEA from iOS 17.4 (March 2024) under FRAND terms.

ECP – Enhanced Contactless Polling. Apple’s proprietary NFC polling extension that allows an Express Transit reader to wake the iPhone Secure Element without user authentication. Requires validators to be ECP-certified.

ECDSA – Elliptic Curve Digital Signature Algorithm. A digital signature scheme based on elliptic curve cryptography. MTS7 uses ECDSA P-256 (NIST curve) for the INTERNAL AUTHENTICATE step.

EEA – European Economic Area. The trade area comprising EU member states plus Iceland, Liechtenstein, and Norway. Apple’s DMA-mandated NFC HCE opening (CardSession API) applies to EEA-based developers and users.

EMV – Europay, Mastercard, Visa. The global interoperability standard for chip-based payment cards and contactless transactions. EMV Kernel 5 is the contactless kernel specification defined by JCB, used here by SL for closed-loop transit.

FRAND – Fair, Reasonable, and Non-Discriminatory. Licensing terms mandated under the DMA for Apple’s NFC access APIs in the EEA, prohibiting Apple from imposing unjustified fees or conditions on third-party developers.

GPO – GET PROCESSING OPTIONS. An EMV command issued by the terminal at the start of a transaction to retrieve the card’s Application Interchange Profile (AIP) and Application File

Locator (AFL).

HCE – Host Card Emulation. A software approach to NFC card emulation where APDU commands from an NFC reader are routed to an app running on the device's main CPU, without a dedicated hardware Secure Element.

ISO-DEP – ISO Dependent. The ISO/IEC 14443-4 protocol layer providing a reliable, half-duplex block-transfer channel over an ISO 14443-3 Type A or Type B physical link.

JCB – Japan Credit Bureau. A Japanese payment network. EMV Kernel 5 is the JCB-defined contactless EMV kernel specification; SL uses it for its closed-loop transit card under a separate, proprietary AID.

MIFARE – A family of contactless smart card ICs and protocols by NXP Semiconductors (the name derives from MIkron FARE-collection System), widely used in transit and access control. Google Wallet native transit provisioning supports MIFARE-family cards.

MTB – Mobile Ticket Bundle. A CBOR-encoded, cryptographically signed data structure carrying one or more travel entitlements. In MTS1, the MTB comprises an Issuer Signature (ECDSA P-256 / ES256) over the ticket payload and an optional Device Signature (HMAC-SHA256 / HS256) binding the MTB to a specific device. In MTS7 HCE, the MTB is the signed data structure returned in the GET NEXT DATA response, containing travel entitlements and the cardholder's public key.

MTS1 – Mobile Travel Standard, version 1. Samtrafiken's BoB specification defining the Mobile Ticket Bundle (MTB) format and transport interfaces. An MTB is a CBOR-encoded, ECDSA-signed ticket payload currently issued as an Aztec 2D barcode; MTS1 §3.1 explicitly permits NFC, Bluetooth, and IP as raw transport alternatives.

MTS7 – Mobile Travel Standard, version 7. Samtrafiken's proprietary NFC card application for mobile travel credentials, identified by AID A000000781010101 and authenticated with ECDSA P-256 signatures.

NFC – Near Field Communication. A short-range (≤ 4 cm) wireless communication standard based on ISO/IEC 18000-3 and ISO/IEC 14443, enabling contactless data exchange between a reader and a card or card-emulating device.

NFC-A / NFC-B – The two physical layer variants of ISO/IEC 14443. NFC-A uses Miller encoding and Manchester load modulation; NFC-B uses BPSK modulation. Both are in common use in transit systems.

NDEF – NFC Data Exchange Format. A lightweight binary message format defined by the NFC Forum for encapsulating typed records on NFC tags and in peer-to-peer exchanges. An NDEF message contains one or more records, each with a type (e.g. MIME type, URI, or proprietary), length, and payload.

PAN – Primary Account Number. The card number (typically 16–19 digits) that identifies a payment or transit card account. SL's EMV Kernel 5 card uses a 19-digit PAN prefixed with BIN 97523124.

PCI DSS – Payment Card Industry Data Security Standard. A security standard for organisations that store, process, or transmit payment card data, maintained by the PCI Security Standards Council. Compliance is required for the Apple NFC & SE Platform integration path.

PICC – Proximity Integrated Circuit Card. ISO 14443 terminology for the card or card-emulating device (as opposed to PCD — Proximity Coupling Device — which is the reader).

PLC – Private Label Card (EMVCo). Google Wallet's account-based token track for closed-loop transit, requiring the transit agency to participate in EMVCo's Private Label Card scheme. Not open-loop payment — a proprietary closed-loop scheme is not automatically eligible; EMVCo PLC scheme participation is required.

RID – Registered Application Provider Identifier. The first 5 bytes of an AID, allocated by ISO/IEC 7816-5 to identify the application provider. The MTS7 RID is A0 00 00 07 81.

SE – Secure Element. A tamper-resistant hardware chip used to store credentials and execute cryptographic operations in isolation from the main CPU. In iPhones, the SE is managed exclusively by Apple and is not directly accessible to third-party apps via HCE.

TEE – Trusted Execution Environment. A secure, isolated processing environment within the device's main application processor. Android Keystore uses the TEE (or the stronger StrongBox variant) to protect private keys so they never leave secure hardware.

T4T – Type 4 Tag. An NFC Forum tag platform (NFC Forum Type 4 Tag Technical Specification) based on ISO/IEC 14443-4 and ISO/IEC 7816-4. A T4T emulates a smart card and exposes a Capability Container file and an NDEF data file accessible via READ BINARY APDUs. AID D2760000850101 is used to select the NDEF application.

TSM – Trusted Service Manager. An intermediary party that manages the over-the-air secure personalisation of applets onto a hardware Secure Element, acting on behalf of a service provider using secure channels.

Responses to NFC feasibility study questions

This document brings together Apple's public information in response to questions in chapter 6 and Google's response to questions in chapter 7 of the NFC feasibility study. The source material is reproduced without editorial changes.

Apple response

Samtrafiken Questions for Apple – June 2026

Prepared for inclusion in publicly available reporting as required

1. Viable iOS NFC Options for Transit Card Validation

> *What NFC mechanisms does Apple make available on iPhone that could support (a) digital ticket presentation, and (b) account-based travel identifier presentation, in a public transport validation scenario such as Sweden?*
>
> *There are two transit card standards that validators in Swedish buses, trams, and stations need to read via NFC. Both are currently deployed on physical smartcards; the goal is to enable passengers to use iPhone as a direct replacement.*
>
> *The two card standards are:*
> *• MTS7 (used by 7 PTAs including Skånetrafiken, and chosen by Västtrafik for future implementation): open standard (BoB) for challenge-response protocol over ISO 14443-4 NFC-A, AID prefix A000000781. The validator initiates a 3-step APDU exchange — SELECT, GET NEXT DATA, INTERNAL AUTHENTICATE. The card design does not involve Apple Wallet.*
> *• EMV Kernel 5 SL (Stockholm public transport, SL, and chosen by UL for future implementation): closed-loop transit contactless card, AID A0000008381010, using the JCB-defined EMV Kernel 5 APDU flow. Used for transit access control, not payment. The card design does not involve Apple Wallet.*
>
> *What iPhone NFC mechanisms does Apple make available that could support (a) digital ticket validation, and (b) account-based travel identifier validation, at a transit gate?*

Apple Wallet:

NFC Options for Transit Cards

Transit Cards in Apple Wallet are underpinned by two card methodologies: **Truth on Card (ToC)** and **Truth on Server (ToS)**.

Truth on Card allows the interchange of both a Transit Card identifier (card number) and fare products / stored value between the Apple device and the reader. The reader determines what product is the most correct to use. Truth on Card requires a specific applet technology to be used such as DESFire or Calypso.

Truth on Server allows the issuer to provision a token used as an identifier with an account-based travel backend handling end-of-day settlement. This also requires a specific applet such as DESFire or Calypso; however, for ToS we also accept white label EMV applets to recreate the credential which is presented to the reader.

In both instances the credential is a tokenised representation of existing card media, stored in the Secure Element (SE) of our devices, with the same UI applying to both in terms of representation of products / value on the card as well as tap / journey / payment history. ToC displays this UI from data on the card using an Apple-developed decoder, whereas ToS draws from a server-driven data stream.

Transit Cards in Apple Wallet support **Express Mode**, which, if enabled by the user, ensures that the user's preferred card is presented at the transit reader. iPhone Power Reserve saves a small amount of battery when the main charge runs out, which allows Express Mode to function up to 5 hours.

In terms of other Apple Wallet NFC technologies, we have a standard called **VAS (Value Added Services)**, this however is not approved for use in Transit.

HCE Contactless-based NFC Transactions for Apps in the EEA

HCE is handled by a separate team at Apple. Please direct your questions to our Worldwide Developer Relations team, who will be able to provide information on the requirements and use cases surrounding HCE for transit.

Developer link for support: https://developer.apple.com/support/hce-transactions-in-apps/

NFC & SE Platform for Secure Contactless Transactions

The Apple NFC & SE Platform for third-party apps is available in specific territories. Inquiries regarding this solution can be submitted through the NFC & SE Platform contact form on the developer link below.

Developer link for support: https://developer.apple.com/support/nfc-se-platform/

2. Recommended Mechanism for Each Use Case

> *Which of those mechanisms would Apple recommend for each of these use cases — (a) digital ticket and (b) account-based travel identifier — in a national public transport context such as Sweden?*

Apple Wallet:

Our Transit Cards in Wallet solution addresses both parts (a) and (b) of your question. We propose that Sweden looks to adopt a **Truth on Server Transit Card in Apple Wallet**, which works as an identifier, behind which an account-based solution holds the records of the user's tickets.

These tickets — once purchased, updated, or removed — will be reflected directly in Apple Wallet UI as soon as the card issuer's server records a change. The readers need to determine if the credential is good to travel and has attributed value (SEK eMoney or tickets).

Given existing EMV deployment in the Stockholm area, we would propose utilising **white label EMV** as the applet technology, replicating EMV Kernel 5 in Apple Wallet and expanding EMV acceptance to readers across Sweden. Open Loop payment acceptance nationally is not a requirement for this approach, but this architecture could pave the way to implement that capability in the future.

Interoperability across Sweden — at both reader presentment and inspection — would benefit from a common standard for EMV Apple Wallet Transit Card issuance shared across the PTAs issuing credentials.

3. Integration Paths and Platform Capabilities

> *Are there integration paths or platform capabilities public transport authorities in Sweden should be aware of that are not covered in Apple's public developer documentation?*

Apple Wallet:

Apple's primary integration path for transit authorities is through the **Integration Partner model**. Further technical documentation relevant to a specific integration can be made available through the appropriate partnership channels. We recommend Samtrafiken engage through this process to receive documentation tailored to their chosen integration approach.

4. Express Transit — Evaluating the iOS Gap

> *The most significant iOS limitation under the recommended iOS HCE (EEA) + Android HCE approach is that HCE apps require authentication per tap, even when set as the default contactless app. For both digital ticket and account-based travel identifier use cases, the main usability gap on iPhone compared to physical cards and Android is the lack of passive 'tap-and-go' (Express Mode equivalent). Is it realistic to achieve a similar user experience for either of these use cases on iPhone, and if so, through which Apple-supported path?*

Apple Wallet:

As described above, Transit Cards in Apple Wallet support **Express Mode**. A credential provisioned to Apple Wallet can be automatically selected at the reader without the need to unlock the device with Face ID / Touch ID — and this also works when the device runs out of power. It also enables the user to choose whether they would prefer to use an Open Loop payment card or a Closed Loop Transit Card for automatic selection, for a transit authority that supports both systems.

Express Mode support page: https://support.apple.com/en-gb/105123

5. Requirements to Become an Apple Wallet Transit Partner

> *This initiative represents public transport authorities in Sweden acting jointly at national scale. What are Apple's minimum requirements for a national transit body — such as a jointly coordinated Swedish initiative — to apply for a bilateral Apple Wallet transit card partnership? What evidence of ridership, technical readiness, or standard maturity does Apple need to see?*

Apple Wallet:

Any transit authority wanting to enable a transit credential in Apple Wallet can do so through our **Integration Partner model**. The key requirements for the transit authority (Issuer) include:

- Reader readiness and any required updates
- Establishing a project to integrate with an Integration Partner
- Back-office and app development work
- Accepting Apple's commercial terms

If the request is to enable a standard not already supported on our Transit Partner Service Platform and development work is required, further discussion between the integration partner, transit authority and our Product and Engineering teams would be required.

6. Timeline and Cost of a Wallet Partnership

> *If public transport authorities in Sweden, acting through a jointly coordinated national initiative, expressed interest in pursuing an Apple Wallet native transit card, what is the realistic end-to-end timeline from initial expression of interest to a passenger-facing deployment? What are the cost categories involved — certification, TSM, commercial licence, ongoing fees?*

Apple Wallet:

To enable a transit credential in Apple Wallet through Apple Pay Transit Partner Services, the timeline depends on the integration model preferred and the partner's development timelines. Indicative timelines vary by integration approach; further detail will be provided through the Integration Partner process based on Samtrafiken's specific technical and commercial circumstances.

Cost and commercial terms are discussed as part of the Integration Partner engagement and are specific to each partnership.

Google response

Hi Johan,

Sorry for the delay in getting back to you. I think it is easier to give you some high level steer here on how best to work with Android transit ticketing, based on our experience globally.

- I highly recommend going directly to native Google Wallet support. This is the path nearly every other transit operator, and bank, globally is using. Some important points to consider here :
    - Wallet is the best option for providing a compelling mobile customer experience. Uppermost in this is automatic card selection at transit terminals, screen off NFC, our full set of UX for transit across Wallet, and integrations with Maps, AI search, Gemini etc.
    - Making use of free and mature Wallet APIs will save a lot of time and development costs, as well as ongoing support costs after launch
    - Several large banks started out with HCE apps and have since retired them and moved to native Wallet support for the above reasons (examples would be Barclays in the UK and American Express globally). There is one large transit agency in Europe using their own HCE implementation today that is in the process of switching to Wallet for the reasons outlined above.
    - You mention that EMV open loop is out of scope, but an integration with Google Wallet would allow you to offer the full mobile experience for both EMV closed loop and open loop with a single API. We highly recommend serving all users with a great mobile experience, not just closed loop users.
- Further to the above, transit agencies can of course develop their own app using the HCE documentation for Android that you have identified, but this is likely to cause customer issues; specifically the default app for NFC when a customer is using Google Wallet for all non-transit use cases. We have not seen a successful implementation where agencies have developed a standalone NFC capability.
- For ABT platforms we support EMV private label and Mifare tokens. I highly recommend you engage with a kernel agnostic TSP such as Thales to discuss the details of which to select, and I can introduce you to the correct team if this is helpful. Our existing SDKs via Thales support JCB kernel, C2, PURE, DPAS, Wise. Thales is our most advanced partner for delivering closed loop EMV tokens into Wallet for transit.
- It is highly unlikely we will be able to prioritise support for the MTS7 any time in the near future
- For single use tickets we recommend open-loop EMV cards and/or barcode tickets using our Transit Passes API
- I confirm our commercial model : Wallet is a free to use Google service; we do not charge issuers, merchants, PSP, networks or users to utilise the Google Wallet APIs

- A screen off Wallet tap wakes the screen to show a success checkmark animation and the card presented on screen, as well as providing haptic feedback.
- Users must enable the ability to skip device unlock for transit. This is a Google Wallet setting, with the user prompted during onboarding and after a (locked) transit tap is made

I hope this is helpful. I think answering the questions from the perspective of delivering the ideal intended customer experience is a better approach than letting NFC protocols define the project approach.

Let me know how you'd like to proceed from here!

Thanks,

Jon

Trafikförvaltningen
Kommunikation och försäljning

PM
2026-07-21
Version 1

Ärende/Dok. id.
3573126

Infosäk. klass
K1 (Öppen)

Handläggare
Babak Keshmiri
+46 762 715 373
babak.keshmiri@regionstockholm.se

Utredning - Införande av NFC-validering i mobila plattformar

Trafikförvaltningen
Kommunikation och försäljning

PM
2026-07-21
Version 1

Ärende/Dok. id.
TN 2026-0641

Infosäk. klass
K1 (Öppen)

Innehållsförteckning

1	Bakgrund	3
2	Mål och utvärderingsparametrar	3
2.1	Mål	3
2.2	Utvärderingsparametrar	5
2.3	Övergripande mål	5
2.4	Tidplan	5
2.5	Arbetsgrupp	6
3	Beskrivning av tjänst	6
3.1	Nuvarande mobila biljettsystem och validering (Aztec-kod)	6
3.2	Teknisk funktion och valideringsarkitektur	6
3.3	Identifierade begränsningar i nuvarande lösning	7
4	Nulägesbeskrivning	7
4.1	Nuvarande arbetssätt och utmaningar med optisk validering	7
4.2	Förväntningar från Trafiknämnden	8
5	Möjliga val för framtida anskaffning av resursobservationer	9
5.1	Beskrivning av alternativ 1: *NFC-validering via SL-appen*	10
5.2	Beskrivning av alternativ 2: *NFC-validering via Wallet*	11
5.3	Beskrivning av alternativ 4: *NFC-validering via Transit Card-plattformar*	13
5.4	Beskrivning av övriga metoder som undersökts	16
5.5	Erfarenheter och förväntningar från nuvarande förvaltning	16
6	Samlad bedömning och rekommendation	17
6.1	Bakgrund	17
6.2	Samlad rekommendation från utredningen	17
6.3	Överväganden och motivering	18
6.4	Ekonomiska konsekvenser	18
6.5	Riskbedömning	19
6.6	Konsekvenser för miljön	19
6.7	Sociala konsekvenser	19

7 Bilagor 19

1 Bakgrund

Denna utredning syftar till att ta fram ett beslutsunderlag för kommande vägval gällande virtuella kort inom området. Virtuella kort är digitala representationer av fysiska kort eller biljetter som lagras i en mobil plattform eller plånbok (t.ex. Apple Wallet eller Google Wallet). De används för att möjliggöra kontaktlös biljettvalidering via NFC-teknik. För kollektivtrafik innebär detta att resenärer kan använda sina smartphones eller wearables för att validera sin biljett vid spärrar och validatorer i stället för att uppvisa en aztec-kod eller ett fysiskt plastkort.

2 Mål och utvärderingsparametrar

Denna utredning syftar till att identifiera och rekommendera den mest optimala lösningen för ett virtuellt SL-kort baserat på NFC-teknik. Rapporten utgör ett strategiskt beslutsunderlag för trafikförvaltningens fortsatta arbete med att modernisera biljettsystemet och förbättra användarupplevelsen för trafikförvaltningens resenärer. Analysen grundas på en samlad bedömning där föreslagna tekniska alternativ ställs mot fastställda effektmål samt kritiska aspekter såsom användarvänlighet, ekonomisk bärighet och långsiktig förvaltningsbarhet. Målet är att fastställa en väg framåt som säkerställer en sömlös, robust och kostnadseffektiv valideringsupplevelse för resenärerna.

2.1 Mål

Det primära syftet med införandet av NFC-baserad biljettvalidering är att modernisera trafikförvaltningens biljettsystem för att uppnå ökad driftsäkerhet, förbättrad kundupplevelse och långsiktig kostnadseffektivitet. Nedan specificeras de strategiska effektmålen för projektet:

1. **Digital migrering och kundövergång**
   Mål: Migrera minst 70 % av resenärerna som idag använder optiska biljetter till NFC-baserad validering.

   Beskrivning: Genom att utnyttja den breda spridningen av NFC-kapabla enheter (för Apple - iPhone XR och framåt) ska trafikförvaltningen styra resenärer mot en mer sömlös valideringsmetod. Detta minskar beroendet av fysiska biljetter och optiska lösningar. För de 5–10 % av resenärer som är låsta till tredjepartsåterförsäljare (t.ex.

UL och Movingo) ska strategier för inkluderande teknikval utvärderas för att säkerställa en enhetlig kundresa.

2. **Optimering av teknisk infrastruktur**
   Mål: Minimera framtida investeringsbehov i optisk läsarutrustning.

   Beskrivning: Genom att etablera en robust NFC-lösning kan trafikförvaltningen undvika behovet av att installera optiska läsare i cirka 600 spärrar. Detta frigör resurser och minskar komplexiteten i den fysiska infrastrukturen, vilket skapar en mer enhetlig teknisk miljö för framtida biljettlösningar.

3. **Ekonomisk effektivisering**
   Mål: Reducera kapitalutgifter (CapEx) och optimera driftskostnader (OpEx).

   Beskrivning:
   - **Investeringsbesparingar:** Genom att undvika ombyggnad av spärrarnas fysiska konstruktion ("plåt") beräknas en potentiell besparing om 30 miljoner kronor. Vid ett scenario där hårdvaruinstallation krävs utan omfattande ombyggnad bedöms besparingen uppgå till 5–10 miljoner kronor (ca 10 000 kr per spärr).
   - **Förvaltningsvinster:** Implementeringen förväntas sänka långsiktiga underhålls- och förvaltningskostnader. Detta förutsätter att NFC-lösningen är fullt operativ senast våren 2028, vilket utgör en kritisk milstolpe för att avveckla befintlig optisk infrastruktur.

4. **Intäktssäkring och operativ robusthet**
   Mål: Stärka intäktssäkringen och optimera trafikflöden.

   Beskrivning:
   - **Operativ effektivitet:** NFC-validering ger högre läspålitlighet och snabbare transaktionstider jämfört med optisk avläsning. Detta eliminerar behovet av att manuellt öppna spärrar eller bussdörrar för hela köer vid läsfel, vilket drastiskt förbättrar genomströmningen i högtrafik.
   - **Skydd mot manipulation:** Teknikskiftet minskar riskexponeringen för fusk och biljettmanipulation, såsom otillåten kopiering eller "livestreaming" av statiska Aztec-koder.

2.2 Utvärderingsparametrar

Aspekter som behöver stå för grunden till utredningen:

- Användarvänlighet
- Kostnad (intern och extern)
- Förvaltning (drift av lösningen)

Användarvänlighet:
Hur enkelt det är för resenären att provisionera ett virtuellt kort och därefter använda det.

- Kan man utgå från SL-appen för administration av det virtuella kortet, såsom provisionering och biljettköp?
- Kompabilitet med wearables
- Hur stor skillnad är det mellan olika operativsystem? Känner man igen sig om man går från iOS till Android eller tvärtom?
- Möjlighet till nationell interoperabilitet.

Införandekostnad:

- Interna kostnader för trafikförvaltningen för att utveckla lösningen eller integrationer mot en eventuell tredje-part.
- Externa kostnader för trafikförvaltningen vid tillhandahållande av lösningen från tredje-part.

Löpande förvaltning:

- Redundans och tillgänglighet
- Beroenden till övriga biljettsystemet (ex. appen) eller andra parter
- Kostnader för att underhålla och vidareutveckla lösningen.

2.3 Övergripande mål

Utredningens övergripande mål är att utreda vilka alternativ som finns för att uppnå NFC-validering via mobil plattform och vara ett underlag inför kommande beslutsfattning. Rapportens syfte är även att rekommendera hur trafikförvaltningen bör jobba framåt i denna fråga. Utredningen kommer inte gå ner på djupet i många av de förutsättningar som finns för lösningar utan på en högre nivå sammanställa beroenden och kostnader samt tekniska förutsättningar för möjligheter.

2.4 Tidplan

Utredningar rörande NFC-validering har genomförts tidigare, men har inte kunnat slutföras på grund av bristande förutsättningar att klargöra vissa

beroenden. Sedan slutet av 2025, då EU ändrade regelverket vilket innebar att Apple blev tvungna att öppna upp NFC för tredjepartsapplikationer, har en ny utredning påbörjats. Tidplanen för att arbeta med denna utredning är satt till det första halvåret av 2026.

2.5 Arbetsgrupp

Arbetet har skett utifrån både en projektgrupp och en arbetsgrupp. Projektgruppens syfte har framför allt varit att sätta mål och utvärderingsparametrar samt löpande rapportera status till Beställare av rapporten på Kommunikation och Försäljning. Arbetsgruppen har bestått av olika grupperingar från främst IT som kommer vara delaktiga i en eventuell framtida implementering.

3 Beskrivning av tjänst

3.1 Nuvarande mobila biljettsystem och validering (Aztec-kod)

Trafikförvaltningens nuvarande mobila biljettlösning i applikationen SL-Reseplanerare och biljetter (framåt benämnt SL-appen) baseras på optisk avläsning av en tvådimensionell streckkod av typen Aztec-kod. Denna teknologi valdes ursprungligen för att uppfylla den gemensamma svenska branschstandarden BoB (Biljett- och Betalstandard) i syfte att säkra interoperabilitet med andra regionala trafikhuvudmän. Validering via Aztec-kod står idag för ca 35% av valideringar som sker inom SL-trafiken. Majoriteten av valideringar sker idag med ett fysiskt SL-kort, vilket sannolikt kan bero på pålitligheten jämfört med en Aztec-kod. Det är därför ett strategiskt mål att se över alternativ på de mobila plattformarna för att ersätta Aztec-koden med en snabbare och mer sömlös NFC-validering.

3.2 Teknisk funktion och valideringsarkitektur

Valideringen av mobilbiljetten sker genom att resenären håller mobilskärmen mot den optiska läsare som är integrerade i spärrlinjer/spärrkiosk (tunnelbana och pendeltåg), installerade ombord på bussar samt på fristående stolpar vid lokalbanor. Samma optiska läsare hanterar även analoga pappersbiljetter (kvitton med statisk Aztec-kod) utställda av ombud eller i spärrkiosk.

För att minimera risken för biljettfusk och skärmdelning är Aztec-koden i mobilapplikationen dynamisk. Koden uppdateras var femte sekund, vilket kräver att appen är aktiv vid valideringstillfället.

3.3 Identifierade begränsningar i nuvarande lösning

Den optiska avläsningen av Aztec-koder kan ha tekniska och användarmässiga utmaningar som negativt kan påverka användarvänligheten i samband med validering:

- *Känslighet mot användarspecifika faktorer*
  Den optiska avläsningen är till viss del beroende av resenärens lokala skärminställningar, skärmens fysiska skick (repor, sprickor eller smuts) samt att mobilen hålls på ett optimalt avstånd från den optiska läsaren. Dessa variabler ökar frekvensen av misslyckade valideringsförsök vid första interaktion, vilket under högtrafik kan leda till fördröjningar.

- *Känslighet för yttre faktorer*
  Den optiska avläsningen kan påverkas av bländande ljus som träffar avläsningsytan vilket kan leda till misslyckade valideringsförsök.

Dessa begränsningar utgör den primära drivkraften till att utreda en övergång från optisk Aztec-kod till en renodlad NFC-baserad validering för mobila plattformar, i syfte att uppnå ett snabbare och mer sömlöst resenärsflöde.

4 Nulägesbeskrivning

Syftet med denna del av utredningen är att visa hur NFC-validering som alternativ valideringsmetod gynnar trafikförvaltningen och SL:s resenärer, samt att lista de alternativ som finns om trafikförvaltningen skulle välja att implementera NFC-validering.

4.1 Nuvarande arbetssätt och utmaningar med optisk validering

I dagsläget förekommer återkommande felläsningar vid optisk läsning av biljetter i valideringsutrustningen, vilket har störst påverkan på buss och spärrlinjer där tidsaspekten är viktig. Dessa problem är starkt kopplade till yttre påverkan, vilket leder till ökade underhålls- och reparationskostnader. Problemen uppenbarar sig sällan i isolerade testmiljöer.

Orsaker till felläsningar:

- **Slitage:** Glaset till avläsningskameran blir repigt, vilket försvårar för avläsningskameran att läsa av Aztec-koden.

- **Värme- och ljuspåverkan:** Optiska läsaren utsätts för hetta i form av direkt solljus och kan påverka avläsningsförmågan eller i värsta fall innebära att komponenter överhettas.

- **Slutanvändarfel:** Ytterligare komplikationer kan vara kopplade till slutanvändare där riktning av telefon, skärmskydd, lokala inställningar, spruckna eller smutsiga displayer kan försvåra avläsning eller tolkning av biljetten.

4.2 Förväntningar från Trafiknämnden

Det finns inga direkta förväntningar från Trafiknämnden kopplat till problematiken kring Aztec baserade biljetter. Det framgår dock i Region Stockholms Budget 2026 följande punkter och ekonomiska förutsättningar från Trafiknämnden som syftar till att förbättra biljettsystemet för resenärerna:

- **Utredning av framtida biljettutbud**
  Arbetet med framtidens biljettutbud har fortsatt under 2025 med en utredning av flera aspekter av biljettsortimentet. Utredningen resulterade bland annat i ett storskaligt test av en ny evenemangsbiljett och lansering av prenumerationsbiljett för företag under hösten.

- **Pilotförsök med Capping**
  För att öka användarvänligheten och attraktiviteten ska Trafiknämnden ta fram förslag på ett pilotförsök med "capping" (pristak) i kollektivtrafiken.

- **Kontaktlösa betalkort**
  Dagens moderna biljettlösningar inkluderar möjligheten att använda kontaktlösa betalkort, vilket innebär en valfrihet för resenärerna att kunna resa på sitt sätt.

Stort fokus ligger därmed på att öka användarvänligheten och attraktiviteten för SL:s resenärer. Det är därmed viktig att nuvarande och framtida biljettutbud kan valideras på ett sömlöst och lättförståeligt tillvägagångsätt för våra resenärer. Det finns även fördelar som är intressant för både Trafiknämnden och trafikförvaltningen rörande de ekonomiska aspekterna i form av att säkerställa mängden valideringar samt ett mer driftsäkert och pålitlig lösning för trafikförvaltningens biljettbärare.

5 Möjliga val för framtida anskaffning av resursobservationer

Tre alternativa metoder har identifierats för implementering av NFC-validering i trafikförvaltningens system. Dessa alternativ utgör underlaget för kommande teknisk analys i stycke 5.1–5.3:

- **Alternativ 1: NFC-validering via SL-appen**
  Lösningen bygger på att NFC validering i den mobila plattformen sker direkt genom SL-appen. Väldigt likvärdig användarupplevelse som speglar dagens optiska biljetter (beroende på hur implementering och gränssnitt utförs). Majoriteten av utveckling sker direkt i SL-appen utan större beroenden till andra leverantörer/aktörer.

- **Alternativ 2: NFC-validering via Wallet**
  Lösningen bygger på att provisionerna ett digitalt EMV-kort (samma infrastruktur som gröna SL-kortet) till en mobil plånbok (Google-/Apple Wallet). Det digitala EMV-kort agerar som identifierare och fungerar enligt plånbokens standardmekanismer.

- **Alternativ 4: NFC-validering via OS-specifika Transit Card plattformar**
  Lösningen bygger på en digital infrastruktur framtagna av Apple och Google för hantering av ”kollektivtrafikkort” i deras respektive mobila plånböcker (Google-/Apple Wallet). Dessa färdiga lösningar erbjuder funktionalitet som finns på plats för att både validera och köpa biljetter i de digitala plånböckerna.

Sammanfattning av alternativen:

Område	**Alternativ 1**	**Alternativ 2**	**Alternativ 4**
Beskrivning	NFC i SL-appen	EMV-kort i Wallet	Transitkort i Wallet
Användarvänlighet	Medel	Medel	Hög
Införandekostnad	Låg–medel	Hög	Hög
Förvaltning	Hög kontroll	Delvis kontroll	Låg kontroll
Beroenden	Delvis begränsade	Native wallets	Native wallets

5.1 Beskrivning av alternativ 1: *NFC-validering via SL-appen*

5.1.1 Teknisk princip

Alternativ 1 innebär att NFC-validering sker direkt i SL-appen. Ett virtuellt kort emuleras i mobilen med hjälp av Host Card Emulation (HCE). Det emulerade kortet fungerar som en bärare för identifiering mot trafikförvaltningens centrala backendsystem (TAP, trafikförvaltningens biljettplattform), där biljetter och dess giltighet hanteras enligt en account-based ticketing-modell. Vid validering via NFC används bäraren för att slå upp den aktiva biljetten i TAP, som därefter avgör biljettens giltighet i samband med validering.

5.1.2 Pågående utredningsspår

Trafikförvaltningen har ett pågående projekt för att utreda lösningen genom ett proof of concept.

Två tekniska vägval utvärderas inom ramen för lösningen:

1. Payload-baserad validering
    - NFC används för att överföra biljettinformation direkt mellan mobil och läsare.
    - Liknar dagens Aztec-lösning men använder NFC som bärare.

Innebär en icke kontobaserad lösning som efterliknar nuvarande optiska lösning, istället för en aztec så skickas en payload via NFC mellan enhet och läsare.

2. Account-Based Ticketing (ABT)
    - Det virtuella kortet används som identifierare.
    - Biljettens giltighet kontrolleras i TAP vid validering.
    - Motsvarar den modell som används för dagens gröna SL-kort.

Innebär ett account-based ticket lösning likt det gröna SL-kortet. För användaren blir det samma biljettpresentation som den optiska lösningen.

Detta alternativ är den som kräver minst beroende till Apple och Google. Beroendet är främst mot Apple (iOS), gällande tillgång till vissa API:er som ger appen behörighet att prata NFC genom Host Card Emulation. Gällande Google (Android) är det mindre restriktivt.

5.2 Beskrivning av alternativ 2: *NFC-validering via Wallet*

Alternativ 2 innebär att ett digitalt SL-kort provisioneras till Apple Wallet eller Google Wallet och används för NFC-validering i trafikförvaltningens befintliga infrastruktur.

Lösningen bygger på samma grundprincip som dagens gröna SL-kort, där ett EMV-baserat kort används som identifierare vid validering. Kortet lagras digitalt i resenärens mobila plånbok och kan därefter användas för kontaktlös validering via NFC.

Till skillnad från Alternativ 4 används inte plattformarnas särskilda transitlösningar. Kortet hanteras i stället på motsvarande sätt som ett traditionellt betalkort i Wallet.

5.2.1 Teknisk princip

Lösningen baseras på att trafikförvaltningen utfärdar ett digitalt EMV-kort som provisioneras till Apple Wallet eller Google Wallet.

Kortet fungerar som en identifierare mot trafikförvaltningens befintliga biljettsystem. Wallet ansvarar för lagring av kortet medan biljettlogiken och kontroll av biljettgiltighet fortsatt hanteras i trafikförvaltningens bakomliggande system.

Efter provisionering skapas en token som lagras i enhetens Secure Element för säker användning vid validering.

5.2.2 Provisionering

För att möjliggöra provisionering behöver trafikförvaltningen agera kortutgivare och tillhandahålla ett EMV-baserat kort.

På Apple-sidan innebär detta att kortet provisioneras genom Apple Pay-flödet, antingen via SL-appen eller direkt från Apple Wallet.

Eftersom kortet hanteras som ett betalkort kräver Apple att trafikförvaltningen samarbetar med en BIN-sponsor (Bank Identification Number Sponsor). BIN-sponsorn möjliggör att kortet kan utfärdas inom ett etablerat kortnätverk och hanteras enligt Apples krav för Apple Pay.
För Google gäller motsvarande process för provisionering, men utan krav på BIN-sponsor.

5.2.3 Tokenisering

När kortet provisioneras genomförs en tokenisering av kortet.

En Token Service Provider (TSP), exempelvis VISA, Mastercard ellr JCB White Label Alliance, skapar då ett digitalt kortnummer (DPAN/token) som lagras i enhetens Secure Element.

Apple Wallet eller Google Wallet ansvarar enbart för lagring och presentation av token medan biljett- och valideringslogik fortsatt ligger hos trafikförvaltningen.

5.2.4 Validering

Efter provisionering och tokenisering kan det digitala kortet användas för NFC-validering i trafikförvaltningens befintliga validatorer och spärrar.

Vid validering används kortets identifierare för att slå upp biljettinformation i trafikförvaltningens biljettsystem, som därefter avgör biljettens giltighet.

Wallet hanterar inte själva biljettlogiken utan fungerar endast som bärare av kortet.

5.2.5 Förutsättningar och beroenden

För att realisera lösningen på Apple-plattformen krävs normalt samarbete med en Qualified Integration Partner (QIP).

En sådan implementation omfattar provisionering, tokenisering och integrationsarbete mot Apple Pay-ekosystemet. Ett införande med QIP uppskattas ta cirka tio månader.

Det kan även finnas möjlighet att genomföra integrationen direkt via Apples officiella integrationsguide. Detta kan minska beroendet av externa leverantörer och reducera både projekt- och förvaltningskostnader.

Lösningen medför dock ett större beroende av kortnätverk, tokeniseringstjänster och Wallet-plattformarna än Alternativ 1.

5.2.6 Kostnader och konsekvenser

Alternativet innebär kostnader för:

- integration mot Apple Pay och Google Wallet,
- eventuell BIN-sponsor,
- Token Service Provider,

- extern integrationspartner (QIP),
- löpande administration av utfärdade kort.

För Apple tillkommer dessutom en avgift per utfärdat kort kopplat till ett iCloud-konto. Kostnaden uppgår till cirka 10 cent per transaktion med ett tak om 1 euro per kort.

Alternativet erbjuder en mer sömlös användarupplevelse än Alternativ 1, men innebär samtidigt större beroenden till externa aktörer och högre förvaltningskomplexitet.

5.3 Beskrivning av alternativ 4: *NFC-validering via Transit Card-plattformar*

Alternativ 4 innebär att SL:s biljettbärare implementeras som ett digitalt kollektivtrafikkort direkt i Apple Wallet och Google Wallet genom respektive plattforms transitlösning.

Till skillnad från Alternativ 2, där kortet hanteras som ett traditionellt EMV-kort, används här plattformarnas särskilda stöd för kollektivtrafik. Lösningen ger tillgång till funktioner som är utvecklade specifikt för resande med kollektivtrafik, vilket innebär den mest sömlösa användarupplevelsen av de alternativ som har analyserats.
Särskilt på iOS möjliggör lösningen funktioner som Express Mode, vilket gör att resenären kan validera sitt kort utan att låsa upp telefonen eller använda Face ID/Touch ID. Validering kan även ske när telefonens batteri är urladdat, förutsatt att enheten stödjer funktionen.

5.3.1 Teknisk princip

Lösningen bygger på att ett digitalt resekort provisioneras till Apple Wallet eller Google Wallet och därefter lagras i enhetens Secure Element.

Biljettsystemet ansvarar fortsatt för produkter, priser, konton och biljettgiltighet. De mobila plattformarna tillhandahåller infrastrukturen för lagring av kortet samt kommunikation mellan mobiltelefonen och trafikförvaltningens NFC-läsare.

Arkitekturen innebär en tydlig ansvarsfördelning där:

- Trafikförvaltningens biljettsystem hanterar biljetter, produkter och kundkonton.
- Integrationspartner eller interna integrationer tillhandahåller API:er och SDK:er mot de mobila plattformarna.

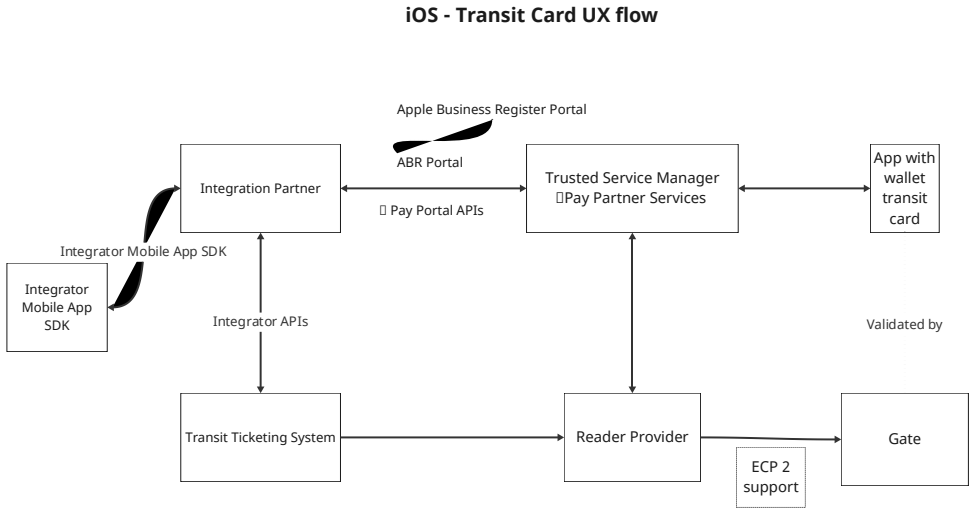
- Apple och Google tillhandahåller den tekniska infrastrukturen för hantering av de digitala korten.
- Kortet lagras i enhetens Secure Element och används vid NFC-validering.

5.3.2 Provisionering

Provisionering kan initieras antingen från SL-appen eller direkt från Apple Wallet respektive Google Wallet.

För Apple sker provisioneringen genom Apple Pay for Transit. Användaren lägger till ett kort via SL-appen eller Apple Wallet, begäran går via en integrationspartner och Apple Pay-portals-API:er till en trusted service manager som samordnar utgivning och livscykel mot biljettsystemet. Information skickas därefter via integrationsgränssnitt till de bakomliggande systemen som ansvarar för kortets utgivning och livscykelhantering.

För Google tillhandahålls i stället en mer plattformsbaserad lösning där trafikförvaltningen i större utsträckning själv ansvarar för kortutfärdande och kortets livscykel. Google Wallet fungerar främst som behållare för kortet och som gränssnitt mot NFC-läsarna.



1 Figur. Flöde över kortprovisionering till Apple wallet och validering.

5.3.3 Validering

Efter provisionering lagras kortet i telefonens Secure Element och kan användas för NFC-validering i trafikförvaltningens befintliga läsare.

Valideringen sker genom att det digitala kortet presenteras mot läsaren på motsvarande sätt som ett fysiskt SL-kort. Biljettens giltighet kontrolleras av trafikförvaltningens biljettsystem.

En viktig fördel med transitplattformarna är att kortet kan användas utan att resenären behöver öppna en app inför varje validering. På iOS möjliggör Express Mode att validering sker utan autentisering, vilket minskar friktionen i resenärsflödet.

5.3.4 Förutsättningar och beroenden

Införandet av lösningen innebär ett större beroende av Apple och Google än övriga alternativ.

För Apple krävs affärsavtal samt integration mot Apples transitplattform. Lösningen bygger normalt på samverkan med en integrationspartner och kan även inkludera en Trusted Service Manager (TSM) för hantering av kort i Secure Element.

För Google finns motsvarande integrationsbehov, men plattformen ger generellt större flexibilitet och ställer inte samma krav på en Trusted Service Manager.

För båda plattformarna kan en Qualified Integration Partner (QIP) användas för att realisera provisionering och tokenisering. Ett sådant införande uppskattas ta omkring tio månader. Möjlighet finns även att genomföra integrationerna med stöd av plattformarnas officiella integrationsguider, vilket kan minska beroendet av externa leverantörer och reducera införandekostnaderna.

Införandet kan även medföra behov av:

- nya integrationer mot Apple, Google och eventuella tredjepartsleverantörer,
- ny eller förändrad lösning för digital kortfabrik,
- utökad hantering av kortens livscykel,
- avtal för användning av plattformarnas transitfunktioner.

5.3.5 Kostnader och konsekvenser

Alternativet innebär högre införande- och förvaltningskostnader än Alternativ 1.

Utöver kostnader för utveckling och integration tillkommer löpande avgifter kopplade till de mobila plattformarna. För Apple tillkommer en avgiftsmodell

per utfärdat kort kopplat till ett iCloud-konto, där kostnaden uppgår till cirka 10 cent per transaktion med ett tak om 1 euro per kort.

Eventuella kostnader för integrationspartner, Trusted Service Manager, digital kortfabrik samt löpande licenser behöver också beaktas vid ett införande. Samtidigt erbjuder lösningen den mest utvecklade användarupplevelsen och tillgång till plattformsspecifika funktioner som inte är tillgängliga i övriga alternativ.

5.4 Beskrivning av övriga metoder som undersökts

Alternativ 3:
Lösningen innebär att enskilda biljetter provisionernas till en mobil plånbok istället för ett digitalt kort. Detta alternativ ströks under utredningen då det konstaterades att Apple, efter att ha infört Apple Pay for Transit, inte tillåter PassKit VAS med NFC för kollektivtrafik då de konkurrerar med varandra.

5.5 Erfarenheter och förväntningar från nuvarande förvaltning

En workshop utfördes tillsammans med nuvarande IT förvaltning för att förstå de olika alternativen utifrån de aspekter som ska utvärderas. Sammanvägning av resultatet från workshopen identifierar en tydlig målkonflikt mellan förvaltningsbarhet och användarupplevelsen. Utifrån förvaltningens perspektiv framgår följande strategiska slutsatser gällande de undersökta alternativen:

- **Alternativ 1** är det mest förvaltningssäkra spåret. Detta alternativ ger trafikförvaltningen full rådighet över biljettarkitekturen, god nationell interoperabilitet via BoB-standarden, och minimerar beroendet till respektive OS-plattforms och deras villkor. Lösningen kan ha begränsningar i vissa mobilanpassade funktioner som underlättar sömlös validering samt att appen måste vara i förgrunden vid validering på iOS.

- **Alternativ 2** bör avföras eller nedprioriteras. Förvaltningsrisken på iOS-sidan bedöms som oacceptabelt hög (-3). Att förlita sig på en icke-sanktionerad miljö där Apple när som helst kan stänga av funktionaliteten skapar en kritisk sårbarhet i valideringsinfrastrukturen.

- **Alternativ 4** erbjuder den bästa användarupplevelsen men kan medföra höga kostnader och förvaltningskomplexitet. Detta alternativ kräver affärsavtal med Apple och Google, löpande licenskostnader per provisionerat kort (cirka 1 € per kort) och en sannolik ny upphandling av

en digital kortfabrik. Dessutom tappar trafikförvaltningen den direkta kundkontakten i appen.

Fullständig sammanställning från workshop återfinns i Bilaga – *Sammanställning från Workshop*

6 Samlad bedömning och rekommendation

6.1 Bakgrund

Trafikförvaltningen står inför en nödvändig modernisering av sitt biljettsystem för att möta framtida krav på driftsäkerhet, effektivitet och kundupplevelse. Den nuvarande lösningen med optisk avläsning (Aztec-kod) har uppvisat brister gällande pålitlighet och genomströmning, vilket resulterar i ökade underhållskostnader och driftsstörningar. Syftet med denna utredning har varit att utvärdera vägval för en övergång till NFC-baserad validering, med målsättningen att säkerställa en robust och kostnadseffektiv lösning senast våren 2028.

6.2 Samlad rekommendation från utredningen

Baserat på den tekniska analysen och workshops med IT-förvaltningen rekommenderas trafikförvaltningen att i första hand gå vidare med - **Alternativ 1: NFC-validering via SL-appen**.

Denna lösning innebär att trafikförvaltningen integrerar NFC-teknik direkt i den befintliga SL-appen. För trafikförvaltningen innebär detta en betydande strategisk fördel genom att trafikförvaltningen bibehåller full rådighet över kundrelationen och biljettsystemets arkitektur. Det är en robust och kostnadseffektiv lösning som minimerar beroendet av externa teknikplattformar, vilket ger en trygg grund för att modernisera resenärernas biljettupplevelse med full kontroll över SL:s egna kanaler.

Alternativ 2 (NFC-validering via Wallet) och alternativ 4 (*OS-specifika Transit Card plattformar)* avförs inte, men bedöms i nuläget som högre komplexitet och krav på externa integrationer eventuellt via externa partners.

Långsiktigt kan Alternativ 1 komma att kompletteras eller ersättas av Alternativ 2 eller 4 för att ytterligare höja användarupplevelsen.

Möjligheten att genomföra integrationen för alternativ 2 och 4 direkt via integrationsguider minskar beroendet av externa partners och eliminerar

onödiga kostnader. Sammantaget innebär dessa förutsättningar att 'time-to-market' kortas och att den ekonomiska risken för båda spåren är betydligt lägre än tidigare bedömt.

Därför bedömer projektet att en fördjupad analys bör genomföras där Alternativ 2 och Alternativ 4 ställs mot varandra, för att avgöra vilken av dessa lösningar som bäst matchar trafikförvaltningens framtida behov av användarvänlighet och plattformskapacitet.

6.3 Överväganden och motivering

Rekommendationen grundar sig i en avvägning mellan *förvaltningsbarhet* och *användarupplevelse*:

- **Rådighet och kontroll:** Alternativ 1 ger trafikförvaltningen full kontroll över biljettsystemets logik och arkitektur. Trafikförvaltningen behåller ägarskapet över kundrelationen i SL-appen och säkerställer interoperabilitet genom BoB-standarden.

- **Riskhantering:** Alternativ 4 erbjuder den mest friktionsfria användarupplevelsen, men innebär ett betydande beroende av tredje part och risk för inlåsningseffekter. Alternativ 1 är det mest robusta valet för att säkerställa att trafikförvaltningen når den kritiska tidsplanen till våren 2028 utan att låsa fast trafikförvaltningen i externa licensavtal.

- **Strategisk hållbarhet:** Genom att bygga en intern lösning (Alternativ 1) kan trafikförvaltningen realisera vinsterna med NFC-validering utan att navigera komplexa licensavtal med globala teknikjättar i ett initialt skede.

6.4 Ekonomiska konsekvenser

Implementeringen av Alternativ 1 möjliggör betydande ekonomiska besparingar genom minskade behov av fysisk infrastruktur:

- **Investeringsbesparingar:** Genom att undvika ombyggnad av spärrarnas fysiska konstruktion ("plåt") vid 600 spärrar beräknas en potentiell besparing om 30 miljoner kronor. Vid ett scenario där endast komponentbyte krävs bedöms besparingen till 5–10 miljoner kronor (ca 10 000 kr per spärr).

- **Driftskostnader:** Genom att avveckla optisk utrustning minskar behovet av löpande underhåll och reparationer av kameror och optiska komponenter.

- **Utvecklingskostnader:** Alternativ 1 kräver interna utvecklingsinsatser men medför inga löpande licenskostnader per provisionerat kort, vilket är fallet i Alternativ 4.

6.5 Riskbedömning

Teknisk sårbarhet: Alternativ 1 är mindre känsligt för yttre påverkan än den optiska lösningen, vilket höjer driftsäkerheten.

- **Beroenden:** Genom att välja Alternativ 1 minimeras beroendet av Apple/Google-avtal. Den främsta risken är begränsningar i operativsystemens begränsning av NFC i bakgrunden för iOS, vilket dock är en hanterbar utmaning i gränssnittsdesignen.

- **Förvaltningsrisk:** Den största risken i projektet är att inte nå målet om driftstart våren 2028. Alternativ 1 bedöms vara det spår med lägst projektrisk för att hålla denna tidslinje.

6.6 Konsekvenser för miljön

Förslag till beslut innebär inga/oförändrade miljökonsekvenser.
Beslut kan indirekt leda till en mindre förbrukning av elektronikkomponenter och avfall, vilket stödjer trafikförvaltningens hållbarhetsmål.

6.7 Sociala konsekvenser

Förslag till beslut innebär inga/oförändrade sociala konsekvenser.
Beslut kan ha en förbättrad påverkan på tillförlitlighet och effektivera validering, vilket förbättrar reseupplevelsen och tillgängligheten för alla resenärer.

7 Bilagor

Bilaga 1 – Sammanställning från Workshop

Trafikförvaltningen
Kommunikation och försäljning

PM
2026-06-02
Version 1

Ärende/Dok. id.
3573131TN 2026-0641
Infosäk. klass
K1 (Öppen)

Handläggare
Babak Keshmiri
+46 762 715 373
babak.keshmiri@regionstockholm.se

Sammanställning från workshop

Här följer en sammanställning av underlaget från workshopen den 18 maj 2026. Workshopen gällde att diskutera framtida strategiska vägval för virtuella SL-kort på mobil plattform.

1 Upplägg

Workshopen genomfördes som en del av projektet för att ta fram ett konkret beslutsunderlag gällande införandet av virtuella SL-kort i mobila plattformar. Arbetet strukturerades i ett antal sammanhängande steg för att säkerställa en bred förankring och djupgående analys:

1.1 Inledning

Sessionen startade med en introduktion av målsättningen, följt av en bakgrundsbeskrivning av utredningens syfte, en övergripande presentation av de fyra lösningsalternativen samt en nulägesanalys av de tekniska förutsättningarna.

1.2 Genomgång av alternativ

Deltagarna delades upp i grupper efter vilken plattform deltagarna var specialiserade inom utefter Android, iOS eller BoB/TAP. Arbetet skedde i fokuserade sessioner per alternativ (Alternativ 1, 2 och 4). Varje session bestod av 20–30 minuters brainstorming och dokumentation, följt av 15–20 minuters presentation och gemensam diskussion.

1.3 Utvärdering och motivering

Varje tekniskt alternativ utvärderades och poängsattes utifrån tre av Trafikförvaltningen fastställda och prioriterade förhållningssätt:

1. **Användarvänlighet (Prioritet 1):** Fokus på hur enkelt det är för resenären att administrera, provisionera och resa med det virtuella kortet.
2. **Förvaltningsbarhet (Prioritet 2):** Fokus på långsiktig drift, redundans, livscykelhantering samt beroenden till externa parter.
3. **Införandestorlek (Prioritet 3):** Omfattar interna och externa utvecklingskostnader, integrationer, avtalsbehov samt anpassningar i backend och valideringshårdvara.

Teknisk förutsättning: Som en kritisk parameter i utvärderingen lyftes att Trafikförvaltningen nu har erhållit **HCE Entitlement via Apple™**, vilket förändrar de tekniska spelreglerna för mjukvarubaserad kortemulering på iOS-enheter.

1.3.1 Utvärderingsskala

Lösningsalternativen tilldelades betyg baserat på följande matris:

Betyg	**Användarvänlighet / Förvaltningsbarhet**	**Införandestorlek (Kostnad + Beroenden)**
+++	Jättebra	Jättelåg
++	Mycket bra	Mycket låg
+	Bra	Låg
-	Dålig	Hög
--	Mycket dålig	Mycket hög
---	Jättedålig	Jättehög

2 Sammanställning av resultat

2.1 Sammanställning per gruppering

Nedan redovisas utfallet och de specifika argumenten uppdelat på de tre analyserade perspektiven: Android-plattformen, iOS-plattformen samt BoB/TAP.

2.1.1 Android-perspektivet

Alternativ 1 (NFC-validering via SL-appen):
Betyg: Användarvänlighet +3 | Förvaltningsbarhet +3 | Införandestorlek -2

- *Användarvänlighet:* Mycket hög då allt behålls i SL:s egna applikation, vilket skapar en tydlig och enkel integrationspunkt och är betydligt smidigare för kunden än optisk avläsning.
- *Förvaltningsbarhet:* Optimalt betyg eftersom Trafikförvaltningen slipper långsiktig förvaltning, avtal och API-synkronisering gentemot tredje part.
- *Införandestorlek:* Kräver ett omfattande designarbete för en ny biljettvy samt utveckling av ett nytt protokoll för att överföra biljettdata via NFC.

Alternativ 2 (Betalkort i Wallet):
Betyg: Användarvänlighet +2 | Förvaltningsbarhet +1 | Införandestorlek -2

- *Användarvänlighet:* Google Wallet har ett inbyggt native-stöd för hanteringen, vilket är bättre än dagens optiska lösning, även om det finns en ovana hos resenärer att hantera resekort i den digitala plånboken.
- *Förvaltningsbarhet:* Kräver uppdateringar av ett flertal API:er och innebär att fler externa parter (såsom plattformsägare och specifika kortutgivare) måste hanteras i förvaltningen.
- *Införandestorlek:* Bedöms som svårdefinierad med oklara integrationer i backend.

Alternativ 4 (OS-specifika Transit Card-plattformar):
Betyg: Användarvänlighet +2 | Förvaltningsbarhet +2 | Införandestorlek -1

- *Användarvänlighet:* Liknar Alternativ 2 men riskerar att skapa förvirring och otydlighet för användaren på grund av att det tillkommer flera parallella flöden för att sätta upp lösningen, köpa biljett samt hantera fel. Det finns även en oklarhet kring om hela flödet är native samt hur biljettutbudet begränsas till specifika bärare.
- *Förvaltningsbarhet:* Samma nivå som Alternativ 2, men med fördelen att det ger ett brett och enhetligt stöd för tillhörande bärbara enheter såsom smartklockor.
- *Införandestorlek:* Innebär höga kostnader då integrationer och backend-system måste anpassas, såvida inte mycket funktionalitet kan levereras "out of the box" via plattformarnas befintliga tjänster.

2.1.2 iOS-perspektivet

Alternativ 1 (NFC-validering via SL-appen):

Betyg: Användarvänlighet +1 | Förvaltningsbarhet -1 | Införandestorlek +1

- *Användarvänlighet:* Valideringen blir avsevärt mer träffsäker än idag, men friktionen i att användaren aktivt måste låsa upp telefonen och öppna SL-appen drar ner betyget. Pålitligheten är också något sämre än ett fysiskt kort eftersom det är helt oanvändbart om telefonens batteri är dött.
- *Förvaltningsbarhet:* Negativt då det skapar ett behov av att parallellt förvalta användargränssnitt för två separata valideringssystem i apparna. Det kräver en utökad livscykelhantering där problem måste testas och undersökas under hela appversionernas livstid, men ger samtidigt fördelen att SL har full rådighet över biljettens utseende.
- *Införandestorlek:* Positivt då det inte medför några dolda kostnader och utvecklingskostnaderna bedöms som relativt låga.

Alternativ 2 (Betalkort i Wallet):
Betyg: Användarvänlighet +2 | Förvaltningsbarhet -3 | Införandestorlek -1

- *Användarvänlighet:* Upplevelsen blir mycket god om resenären kan förvälja biljetten direkt i sin wallet, men en stor nackdel är att walleten agerar som ett externt ombud och kan därmed inte slutföra själva kortköpet i samma applikation.
- *Förvaltningsbarhet:* Mycket kritisk risknivå (-3) då Trafikförvaltningen blir helt beroende av externa integratörer. Lösningen är inte officiellt sanktionerad av Apple, vilket innebär okända risker; exempelvis kan Apple stänga ner lösningen när som helst eller tvinga fram egna spår (som Alternativ 4).
- *Införandestorlek:* Förknippat med okända implementations- och införandekostnader eftersom det inte är en standardiserad lösning för mellanhänder, samt risk för dolda avgifter (provision eller per blipp).

Alternativ 4 (OS-specifika Transit Card-plattformar):
Betyg: Användarvänlighet +3 | Förvaltningsbarhet +3 | Införandestorlek -1

- *Användarvänlighet:* Erbjuder bäst upplevelse och störst flexibilitet för olika resenärstyper. Köpflödet är lika enkelt som idag, och biljetten fungerar i "Express Mode" även när telefonens batteri är dött, även om en viss initial otydlighet mellan appbiljett och digital kortbiljett kan uppstå.
- *Förvaltningsbarhet:* Högsta betyg då flödet är standardiserat och stabilt; det liknar Alternativ 2 men eliminerar de plattformsrisker och nedstängningshot som fanns där.

- *Införandestorlek:* Kräver inga jättestora resurser på klientsidan, utan arbetet hamnar primärt i backend-systemen.

2.1.3 BoB / TAP

Alternativ 1 (NFC-validering via SL-appen):

Betyg: Användarvänlighet +2 | Förvaltningsbarhet +2 | Införandestorlek -2

- *Användarvänlighet:* Positivt att det blir liten skillnad mot befintliga flöden, och lösningen lirar mycket bra med utlåning av biljetter på ett sätt som andra alternativ inte klarar av. Genom att skicka en Mobile Ticket Broker (MTB) istället för bara ett bärar-ID kan resenären själv välja vilken biljett den vill åka på. Negativt är att appen måste vara i förgrunden, vilket kan kännas icke-intuitivt för användare som är vana vid vanliga wallets.
- *Förvaltningsbarhet:* Bygger på etablerad ISO-standard via Host Card Emulation (HCE), vilket tryggar Samtrafikens interoperabilitet. Trafikförvaltningen kan driva detta som en nationell standard (vilket redan görs inom Samtrafiken, då t.ex. Västtrafik saknar möjlighet att slå upp biljetter on-demand för en bärare). HCE bedöms vara tekniskt tillgängligt under lång tid.
- *Införandestorlek:* Kräver nyutveckling i båda apparna (gränssnitt, nyckelhantering, biljetthämtning), uppsättning och distribution av signeringsnycklar samt anpassningar för att biljettkontrollen och Sunmi-enheterna ska få stöd.

Alternativ 2 (Betalkort i Wallet):

Betyg: Användarvänlighet +2 | Förvaltningsbarhet 0 (osäker) | Införandestorlek -1

- *Användarvänlighet:* Resenärer förväntar sig intuitivt att SL-kortet ska kunna leva i en wallet, men provisioneringen innebär ett helt nytt flöde som riskerar att bli komplext (UX-utmaning kring var biljetten hamnar).
- *Förvaltningsbarhet:* Stor osäkerhet kring kontroller och risker. Det krävs nya integrationer med ett flertal tredjepartssystem (Idemia, Google Pay, Apple Pay) och SL saknar kontroll över de plattformar man förlitar sig på, vilket riskerar att bli spretigt. Positivt är att man löpande kan återanvända normal kortvalideringslogik.
- *Införandestorlek:* Innebär en osäkerhet kring kostnader. Digitala SL-kort måste hanteras separat från fysiska och tilldelas ett eget AID. Provisionering kräver en beställning hos Idemia (eventuellt via existerande avtal, annars krävs ny upphandling av en digital kortfabrik) och läsarna måste omkonfigureras för masstransit.

Alternativ 4 (OS-specifika Transit Card-plattformar):
Betyg: Användarvänlighet +2 | Förvaltningsbarhet -1 | Införandestorlek -3

- *Användarvänlighet:* Erbjuder mycket rik funktionalitet i walleten (t.ex. geofencing och hantering vid avstängd telefon), men köp- och provisioneringsflödena passar SL:s affärsmodell dåligt och riskerar att bli förvirrande för kunden.
- *Förvaltningsbarhet:* Samma kortvalideringslogik som för fysiska kort kan återanvändas, men Trafikförvaltningen tappar den direkta kundkontakten, drabbas av en löpande kostnad per provisionerat kort (ca 1 € per kort) och har historiskt dålig erfarenhet av digitala testkort.
- *Införandestorlek:* Bedöms som jättehög (-3). Kräver omfattande integrationer (Idemia, Google, Apple) för API:er gällande köp, provisionering och kortinnehåll, nya affärsavtal med teknikjättarna samt en sannolik ny upphandling av en digital kortfabrik.

2.2 Gemensam sammanställning

När de olika grupperingarnas och systemens parametrar sammanvägs till en helhetsbild framträder en tydlig målkonflikt mellan förvaltningsbarhet och användarupplevelse.

Sammanställd utvärderingsmatris
Följande tabell visar de sammanvägda betygen från workshopen för de tre huvudsakliga alternativen:

Lösningsalternativ	**Plattform / Standard**	**Användarvänlighet (Prio 1)**	**Förvaltningsbarhet (Prio 2)**	**Införandestorlek (Prio 3)**
Alternativ 1: NFC via SL-appen	Android iOS BoB (Standard)	+++ + ++	+++ - ++	-- + --

Alternativ 2: Betalkort i Wallet	Android iOS BoB (Standard)	++ ++ ++	+ --- +	-- - -
Alternativ 4: OS-specifika Transit Card	Android iOS BoB (Standard)	++ +++ ++	++ +++ -	- - ---

3 Strategiska slutsatser för vidare beslut

1. **Alternativ 1 (Egen app via HCE) är det mest förvaltningssäkra spåret:** Genom att Trafikförvaltningen nyligen har säkrat *HCE Entitlement via Apple* faller det största historiska hindret för denna lösning. Det ger SL full rådighet över biljettarkitekturen, god nationell interoperabilitet via BoB-standarden och minimerar beroendet av teknikjättarnas godtyckliga villkor. Den stora utmaningen ligger i införandestorleken (nytt protokoll, app-utveckling och anpassning av kontrollutrustning) samt kravet på att appen måste vara i förgrunden vid validering på iOS.

2. **Alternativ 2 (Betalkort i Wallet) bör avföras eller nedprioriteras:** Det ger förvisso en acceptabel användarupplevelse, men förvaltningsrisken på iOS-sidan är oacceptabelt hög (-3). Att ligga i en icke-sanktionerad miljö där Apple när som helst kan stänga av funktionaliteten skapar en kritisk sårbarhet i SL:s valideringsinfrastruktur.

3. **Alternativ 4 (Transit Card) erbjuder den bästa produkten men till högst kostnad:** För resenären (särskilt på iOS) är detta den

ultimata lösningen eftersom den stöder "Express Mode" (blippa med död telefon). Det är dock det mest komplexa alternativet att införa (-3 i införandestorlek för BoB). Det kräver tunga affärsavtal med Apple och Google, löpande licenskostnader per kort samt en potentiellt ny upphandling av en digital kortfabrik. Man riskerar även att tappa den direkta kundkontakten i appen.

Rekommendation från workshop: I första hand rekommenderas **Alternativ 1**, då erhållandet av Apples HCE-godkännande gör det möjligt att bygga en kostnadseffektiv, standardiserad och helt egenkontrollerad NFC-lösning i SL-appen. Alternativ 4 bör behållas i rapporten som ett långsiktigt, premiumalternativ (eventuellt för parallellkörning), men med ett tydligt medskick om de betydande kommersiella och backend-mässiga trösklar det innebär.