Virtuální ODISka -

Příloha s technickým popisem pro odbavovací zařízení

Verze 1.08

1 Obsah

1 Obsah 2
2 Informace o dokumentu 5
2.1 Historie změn 5
2.2 Seznam použitých zkratek a pojmů 6
3 Virtuální ODISka – Identifikátor 6
3.1 Struktura dat předávaná odbavovacímu zařízení 6
3.1.1 Sekce Prefix a sekce Postfix 7
3.1.1.1 DataID 7
3.1.2 Sekce Hlavička 8
3.1.2.1 Version 8
3.1.2.2 NetworkID 8
3.1.2.3 ProviderID 8
3.1.2.4 StaticDataKeyID 8
3.1.3 Sekce Statická část 8
3.1.3.1 CustomerID 8
3.1.3.2 AppInstanceID 8
3.1.3.3 CardLogicalNo 9
3.1.3.4 StaticDataSign 9
3.1.3.5 Podepsání a ověření dat statické části struktury 9
3.1.4 Sekce Dynamická část 9
3.1.4.1 VisualInspectionKeyCollectionID 9
3.1.4.2 AlphanumericColorHash 9
3.1.4.3 LastServerSyncDateTime 9
3.1.5 QR kód 9
3.1.5.1 Příklad vygenerovaného QR kódu 10
4 Bezpečnostní prvky 10
4.1 Tajné klíče 10
4.2 Výpočet barev a alfanumerických znaků bezpečnostního proužku 10
5 Whitelist karet virtuální ODISka 10
5.1 Variabilní datová část souboru 11
5.1.1 Data obsahující všechny operace nad jedním CustomerID 12
5.1.2 Operace INSERT, DELETE, UPDATE 12
5.1.3 Objekty 12
5.1.3.1 Příklad vložení nového záznamu zákazníka s fotkou a obsahující 2 virtuální Odisky: 13
5.1.3.2 Příklad smazání zákazníka: 13

5.1.3.3 Příklad smazání AppInstanceID u zákazníka: 13
5.1.3.4 Příklad smazání CustomerProfile u zákazníka: 14
5.1.3.5 Příklad update fotky, změny přijmení a přiřazení nového AppInstanceID: 14
6 Webová služba KODIS pro Whitelist karet 14
6.1 REST API 14
6.1.1 POST metoda /whitelistCard/Get 14
6.1.1.1 Vstupní parametry 14
6.1.1.2 Výstupní parametry 15
6.1.1.3 Příklad JSON požadavku na získání informací o souborech Whitelistů karet 15
6.1.1.4 Příklad JSON odpovědi (úspěch) na získání informací o souborech Whitelistů karet 16
6.1.1.5 Příklad JSON odpovědi (neúspěch) na stažení souboru s Whitelistem karet 16
6.1.2 POST metoda /whitelistCard/getFile 16
6.1.2.1 Vstupní parametry 16
6.1.2.2 Výstupní parametry 17
6.1.2.3 Příklad JSON požadavku na stažení souboru s Whitelistem karet 17
6.1.2.4 Příklad JSON odpovědi (neúspěch) na stažení souboru s Whitelistem karet 18
6.1.3 POST metoda /whitelistCard/getData 18
6.1.3.1 Vstupní parametry 18
6.1.3.2 Výstupní parametry 18
7 Whitelist jízdenek 19
7.1 REST API 19
7.1.1 POST metoda /whitelistTicket/Get 19
7.1.1.1 Vstupní parametry 19
7.1.1.2 Výstupní parametry 19
7.1.1.3 Příklad JSON požadavku na získání informací o Whitelistech jízdenek 20
7.1.1.4 Příklad JSON odpovědi (úspěch) na získání informací o Whitelistech jízdenek 20
7.1.2 POST /whitelistTicket/ResetWasPerformed 21
7.1.2.1 Vstupní parametry 21
7.1.2.2 Výstupní parametry 21
7.1.2.3 Příklad JSON požadavku na získání informací zdali došlo k resetu Whitelistu jízdenek 21
7.1.2.4 Příklad JSON odpovědi (úspěch) na získání informací zdali došlo k resetu Whitelistu jízdenek 22
7.1.3 POST metoda /whitelistTicket/getData 22
7.1.3.1 Vstupní parametry 22
7.1.3.2 Výstupní parametry 22

8 Velikosti a frekvence přenášených dat 23

8.1 White list karet 23

Řešení umožňuje i stahování po částech. V samotné funkci getFile, kde si odbavovací zařízení požádá o část souboru přes fileOffset a dataLength. Pro stažení souboru po částech je proces nastaven. Data nejsou nijak komprimována, aby se i velké soubory mohly parsovat po částech i po delší době. V takovém případě může dojít k prodlevě mezi staženými daty a správnou kontrolou příslušných dokladů, kterou je potřeba v případě tohoto ojedinělého jevu tolerovat. 23

8.2 White list kupónů 23

9 Číselník chyb 23

2 Informace o dokumentu

2.1 Historie změn

Datum	Verze	Změny	Autor
19. 5. 2020	0.01 – 0.90	Vytvoření dokumentu	Martin Koštuřík, Ondřej Squerzi, Michael Janošík, Pavel Nenka

17. 12. 2021	1.00	• Struktura dat předávaná odbavovacímu zařízení
		  ○ Vytvoření nové sekce Prefix a Postfix, obě s datovou položkou DataID.
		  ○ Rozšíření sekce Hlavička o dvě nové datové položky NetworkID a ProviderID.
			Martin Koštuřík

4.4.2022	1.01	• V kapitole „3.1 Struktura dat předávaná odbavovacímu zařízení“ změněn název položky z VisualInspectionKeyID na VisualInspectionKeyCollectionID.
		• Přepracována kapitola „4 Bezpečnostní proužek“, kde byly aktualizovány procesy pro:
		  ○ stažení jednotlivých typů klíčů z aplikačního serveru
		  ○ výpočet diverzifikovaných klíčů
		  ○ výpočet odvozených klíčů
		  ○ výpočet barev bezpečnostního proužku a alfanumerických znaků
		• Revize kapitoly 6 a kapitoly 7. Doplnění příkladů volání webové služby.
			Martin Koštuřík, Ondřej Squerzi

6.4.2022	1.02	• V kapitole „3.1 Struktura dat předávaná odbavovacímu zařízení“ změněn název položky z AlfanumericColorHash na AlphanumericColorHash. • V kapitole „4.1.3 Diverzifikace klíčů“ doplněna definice chybějící pomocné metody convertGuidToByteArray().	Martin Koštuřík
16.8.2022	1.03	• Aktualizace print screenu	Pavel Nenka
25.10.2022	1.04	• Rozhraní pro on-line dotaz a číselník chyb	Pavel Nenka
13.12.2022	1.05	• Doplněn profil do WL	Pavel Nenka

19.12.2022	1.06	• Doplněna maximální velikost WL v cílovém stavu	Michal Pastrňák
26.1.2023	1.07	• Doplněna maximální velikost WL a velikost fotografie pro přenosná zařízení železniční dopravy, rozdělení WL pro různé druhy zařízení.	Michal Pastrňák, Pavel Nenka
6.2.2023	1.08	• Odstraněno jméno a příjmení z WL pro přenosná zařízení železniční dopravy	Michael Janošík

2.2 Seznam použitých zkratek a pojmů

Pojem	Popis
VO	Virtuální ODISka
CC MSK	Clearingové centrum Moravskoslezského kraje
WL	White list
White list karet/kupónů	Seznam platných karet/kupónů

3 Virtuální ODISka – Identifikátor

3.1 Struktura dat předávaná odbavovacímu zařízení

Následující tabulka zobrazuje datovou strukturu (s jednotlivými sekcemi a jejich datovými položkami), která bude uložena v mobilní aplikaci „ODISapka“. Struktura dat bude rozdělena na hlavičku, statickou a dynamickou část. Podepsaná statická část bude trvale uložena v mobilní aplikaci „ODISapka“ (bude přenesena ze serveru po registraci/přihlášení uživatele). Hlavička a dynamická část se bude sestavovat až při odeslání struktury dat do odbavovacího zařízení.

	Název sekce	Velikost sekce (B)	Název položky	Velikost položky (B)	Popis položky	Poznámka
	Prefix	8	DataID	8	Identifikace datové struktury	"ODISVC01", uloženo jako string v ASCII kódování
Kódováno pomocí BASE64	**Hlavička**	10	Version	1	Číslo verze struktury	Aktuální verze je 1
			NetworkID	4	Číslo sítě, do které patří vydavatel virtuální karty	203811

			ProviderID	4	Číslo dopravce, který je vydavatelem virtuální karty	134
			StaticDataKeyID	1	ID klíče pro podepsání/ověření statické části dat	Aktuální číslo klíče je 1
	Statická část	90	CustomerID	16	ID zákazníka eShopu	.Net GUID uložený v bytovém poli
			AppInstanceID	16	ID instance virtuální ODISky	.Net GUID uložený v bytovém poli
			CardLogicalNo	10	Logické číslo virtuální ODISky	Uloženo jako string v ASCII kódování
			StaticDataSign	48	Podpis statické části dat (modře podbarveno) eliptickou křivkou	Použit ECDSA algoritmus secp192r1
	Dynamická část	7	VisualInspectionKeyCollectionID	1	ID sady klíče pro výpočet alfanumerických znaků grafického prvku	Aktuální číslo klíče je 1
			AlphanumericColorHash	2	Alfanumerické znaky grafického prvku	
			LastServerSyncDateTime	4	Datum a čas poslední synchronizace virtuální ODISky se serverem	Počet sekund od 1.1.2020 00:00:00
	Postfix	8	DataID	8	Identifikace datové struktury (koncové opakování)	"ODISVC01", uloženo jako string v ASCII kódování
	Celkem (B)	**123**				

3.1.1 Sekce Prefix a sekce Postfix

3.1.1.1 DataID

Obě sekce Prefix/Postfix obsahují položku DataID s identifikací typu datové struktury. Pro Virtuální ODISku je použit identifikátor "ODISVC01", který je uložen jako ASCII řetězec o délce 8 znaků.

Řetězec se používá pro identifikaci typu datové struktury v případě, že se v dopravním systému mezi mobilní aplikací a odbavovacím zařízením přenáší více různých typů datových struktur, které je

potřeba od sebe při načtení odlišit. Tzn., v případě, že odbavovací zařízení přijme pomocí komunikačního kanálu (QR kód, NFC) datovou strukturu, a bude si chtít ověřit, že se jedná o data nesoucí informace o Virtuální ODISce, tak provede následující kontrolu, kde obě podmínky musí být splněny:

- Vzájemně porovná jednotlivé byty řetězce Prefix.DataID (prvních 8B datové struktury) a řetězce Postfix.DataID (posledních 8B datové struktury), zda jsou navzájem shodné. Pokud není shodný Prefix.DataID a Postfix.DataID, považuje se, že datová struktura není validní.
- Vzájemně porovná jednotlivé byty řetězce Prefix.DataID s řetězcem "ODISVC01", zda jsou navzájem shodné.

3.1.2 Sekce Hlavička

Sekce obsahuje základní údaje o přenášené struktuře.

3.1.2.1 Version

Číslo verze datové struktury. Číslováno od 1.

3.1.2.2 NetworkID

Číslo sítě, do které patří vydavatel virtuální karty. V případě Virtuální ODISky je použito číslo 203811 (síť ODIS), které se ukládá jako *BigEndian*.

3.1.2.3 ProviderID

Číslo dopravce, který je vydavatelem virtuální karty. V případě Virtuální ODISky je použito číslo 134 (KODIS), které se ukládá jako *BigEndian*.

3.1.2.4 StaticDataKeyID

ID klíče použitého pro podepsání/ověření statické části dat. Číslováno od 1.

3.1.3 Sekce Statická část

Statická část obsahuje informace ohledně virtuální ODISky, které se nemění (jen při registraci virtuální ODISky).

3.1.3.1 CustomerID

Jedná se o ID zákazníka eShopu, které jednoznačné identifikuje přihlášeného zákazníka.

Hodnota ID se v bytovém poli ukládá ve formátu vráceném pomocí .Net metody Guid.ToByteArray(), kdy pořadí počáteční skupiny se čtyřmi bajty a dalších 2 2 – skupin bajtů je obráceno, zatímco pořadí poslední skupiny dvou bajtů a uzavírací šest bajtů je stejné. Např., Guid v řetězcové podobě "35918bc9-196d-40ea-9779-889d79b753f0" bude v bytovém poli zapsán jako následující posloupnost bytů { 0xC9, 0x8B, 0x91, 0x35, 0x6D, 0x19, 0xEA, 0x40, 0x97, 0x79, 0x88, 0x9D, 0x79, 0xB7, 0x53, 0xF0 }.

3.1.3.2 AppInstanceID

Jedná se o ID instance virtuální ODISky, tzn. nainstalované instance mobilní aplikace „ODISapka“, které je generované na straně aplikačního serveru KODIS, toto číslo lze přirovnat v tokenu používanému při odbavení bankovních karet.

Hodnota ID se v bytovém poli ukládá ve formátu vráceném pomocí .Net metody Guid.ToByteArray(), kdy pořadí počáteční skupiny se čtyřmi bajty a dalších 2 2 – skupin bajtů je obráceno, zatímco pořadí poslední skupiny dvou bajtů a uzavírací šest bajtů je stejné. Např., Guid v řetězcové podobě "35918bc9-196d-40ea-9779-889d79b753f0" bude v bytovém poli zapsán jako následující posloupnost bytů { 0xC9, 0x8B, 0x91, 0x35, 0x6D, 0x19, 0xEA, 0x40, 0x97, 0x79, 0x88, 0x9D, 0x79, 0xB7, 0x53, 0xF0 }.

3.1.3.3 CardLogicalNo

Desetimístné logické číslo virtuální ODISky, které je uloženo jako řetězec v ASCII kódování.

Logické číslo karty je v databázi clearingu generováno jako znaménkový INTEGER (4B), jehož hodnota začíná od 1, a postupně se zvyšuje. Toto číslo je poté převedeno na řetězec a doplněno zleva nulami na celkových deset číslic.

3.1.3.4 StaticDataSign

Podpis statické části dat (modře podbarveno) eliptickou křivkou. Je použit ECDSA algoritmus secp192r1 s hashovací funkcí SHA1.

3.1.3.5 Podepsání a ověření dat statické části struktury

Podpis statické části struktury dat privátním klíčem serveru bude prováděn pomocí eliptické křivky (ECDSA algoritmus secp192r1). Podpis bude předem vypočítán na aplikačním serveru KODIS a předán mobilnímu zařízení. Během procesu odbavení pak předá mobilní aplikace „ODISapka“ zvoleným komunikačním kanálem (NFC nebo zobrazením/načtením QR kódu) strukturu dat odbavovacímu zařízení, která podpis statické části dat ověří veřejným klíčem serveru (ten musí být v odbavovacím zařízení předem uložen).

3.1.4 Sekce Dynamická část

Sekce Dynamická část dat obsahuje data, které se mění (např. při každém odbavení na zařízení apod.) nebo volitelná data, které není nutné vždy předávat odbavovacímu zařízení (např. při NFC komunikaci).

3.1.4.1 VisualInspectionKeyCollectionID

ID diverzifikovaného klíče použitého pro výpočet alfanumerických znaků grafického prvku. Číslováno od 1.

Na serveru může existovat více sad klíčů. V případě kompromitace některého klíče, je pak možné vytvořit novou sadu klíčů.

3.1.4.2 AlphanumericColorHash

Jedná se o první dva bajty výsledného hashe při výpočtu alfanumerických znaků grafického prvku, viz **Chyba! Nenalezen zdroj odkazů.**.

3.1.4.3 LastServerSyncDateTime

Datum a čas poslední synchronizace virtuální ODISky se serverem. Počet sekund od 1.1.2020 00:00:00. Více bytová INTEGER hodnota se ukládá jako *BigEndian.*

3.1.5 QR kód

Mobilní aplikace „ODISapka“ bude při použití tohoto komunikačního kanálu zobrazovat strukturu dat jako QR kód. Odbavovací zařízení pak načte pomocí vlastní kamery tento QR kód zobrazený na displeji mobilního zařízení.

Protože struktura dat obsahuje binární data, bude před vygenerováním QR kódu bytové pole se strukturou dat převedeno BASE64 kódování. Konkrétně jsou do jednoho BASE64 bloku zakódovány tyto následující sekce (žlutě podbarveno):

- Hlavička
- Statická část
- Dynamická část

Sekce Prefix a Postfix do BASE64 zakódovány nejsou kvůli rychlé identifikaci typu datové struktury.

3.1.5.1 Příklad vygenerovaného QR kódu

4 Bezpečnostní prvky

V prvé řadě je třeba zdůraznit, že bezpečnost Virtuální Odisky je postavena na fotografii uživatele, protože jakékoliv doposud známé zabezpečení na mobilním telefonu, které není vázáno na konkrétní HW, tedy které by bylo možno použít napříč spektrem různých modelů mobilních telefonů od různých výrobců, je překonatelné.

Níže uvedené zabezpečení bezpečnostním proužkem je pouze doplňkové bránící těm nejjednodušším způsobům možného zneužít, ale základní bezpečnost je postavena na zobrazení fotografie cestujícího.

Bezpečnostní proužek, který je zobrazován na mobilní aplikaci a zároveň na odbavovacím zařízení, je, svým charakterem, dynamický grafický prvek, jehož vzhled se pseudonáhodně mění v závislosti na serverovém času a tajných klíčích.

4.1 Tajné klíče

Kapitola obsahuje tajné informace, které jsou Dopravci poskytnuty v režimu dle „Dohody o mlčenlivosti při výměně důvěrných informací o systému Virtuální ODISka“.

4.2 Výpočet barev a alfanumerických znaků bezpečnostního proužku

Kapitola obsahuje tajné informace, které jsou Dopravci poskytnuty v režimu dle „Dohody o mlčenlivosti při výměně důvěrných informací o systému Virtuální ODISka“.

5 Whitelist karet virtuální ODISka

Tento dokument popisuje formát souboru whitelistu použitého pro data virtuálních ODIS karet pro odbavovací zařízení.

WL karet bude generován ve dvou typech, jeden se základní velikostí fotografie a vyplněným jménem a příjmením a druhý s komprimovanou fotografií v průměru do 4kB a s nevyplněnými údaji jména a příjmení. Tento druhý typ je určen pouze pro přenosná zařízení železniční dopravy.

To, který typ WL bude dále uvedenou službou vrácen, bude rozlišeno podle přihlašovacího loginu při volání WS pro vrácení WL karet. Struktura obou typů WL karet bude naprosto stejná, rozdíl bude pouze ve velikosti fotografie a ve vyplnění či nevyplnění jména a příjmení ve WL.

Použité zkratky:

Zkratka	Popis
RFU	Reserved for Future Use (Rezervováno pro budoucí použití)
Whitelist	Seznam dat o virtuálních Odiskách
TLV	TLV (Tag-length-value) je obecný název pro formát dat

Soubor se skládá z následujících bloků:

- Záhlaví souboru – pevná velikost 16B
- Datová část souboru – variabilní, obsahující jednotlivé data o Odiskách ve formátu TLV

Soubor používá následující kódování:

- Little-endian – pro pořadí bytů číselných datových typů
- UTF-8 – kódování řetězce znaků

Datová část souboru může být komprimovaná

Proměnná	Velikost [B]	Popis
FileVersion	1	Verze souboru (0x02, zbytek RFU)
FileGenTime	5	Čas vytvoření souboru (1B hodiny, 1B minuty, 1B sekundy, 2B milisekundy), UTC čas
FileGenDate	4	Datum vytvoření souboru (1B den, 1B měsíc, 2B rok)
DataCompressType	1	Typ komprimace datové části (0x00 – bez komprimace, zbytek RFU)
DataLength	5	Délka datové části souboru
VariableData	var	Variabilní datová část souboru v TLV

5.1 Variabilní datová část souboru

Tagy:

0x01 – Data obsahující všechny operace nad jedním CustomerID – ID zákazníka eShopu (převedeného do bajtového pole)

0x11 – INSERT

0x12 – DELETE

0x13 – UPDATE

0x21 – Photo – pro přenos binárních dat fotografie svázaného s daným CustomerID

0x22 – AppInstanceID – pro přenos ID instance virtuální ODISky (převedeného do bajtového pole) svázaného s daným CustomerID

0x23 – CustomerProfile – Číslo platného zákaznického profilu. V ODIS v jednu chvíli mohou být platné až 2 zákaznické profily.

0x24 – Firstname – Křestní jméno držitele VO

0x25 – Lastname – Příjmení držitele VO

Vždy se začíná tagem 0x01 který začíná CustomerID a dále obsahuje operace INSERT, DELETE, UPDATE. V těchto operacích jsou dalšími tagy označeny objekty, se kterými se pracuje (Photo, AppInstanceID). Pokud při operaci DELETE nejsou vyplněna žádná data, maže se celý profil.

5.1.1 Data obsahující všechny operace nad jedním CustomerID

Proměnná	Velikost [B]	Popis
Tag	1	0x01 – Operace nad jedním CustomerID
Length	2	Délka všech následně vnořených dat
Value	var	Data – zde se pokračuje operacemi ve formátu TLV (0x11 – INSERT, 0x12 – DELETE, 0x13 – UPDATE)

5.1.2 Operace INSERT, DELETE, UPDATE

Proměnná	Velikost [B]	Popis
Tag	1	0x11 – INSERT, 0x12 – DELETE, 0x13 - UPDATE
Length	2	Délka všech následně vnořených dat
Value	var	Data – zde se pokračuje objekty s kterými se pracuje ve formátu TLV (0x21 – Photo, 0x22 – AppInstanceID, 0x23 – CustomerProfile, 0x24 – Firstname, 0x25 – Lastname, RFU)

5.1.3 Objekty

Proměnná	Velikost [B]	Popis
Tag	1	0x21 – Photo, 0x22 – AppInstanceID, 0x23 – CustomerProfile, 0x24 – Firstname, 0x25 – Lastname, RFU
Length	2	Délka dat
Value	var	Data

Bude provedena optimalizace fotografií pro WL tak, aby bylo zaručeno nepřekročení maximální velikosti WL dle kapitoly 8.1.

Průměrná datová velikost fotografie ve WL přenosných zařízení železniční dopravy bude 4 kB.

Dopravce zodpovídá za bezpečné uložení dat v zařízení (bezpečnost uložení může být řešena kryptograficky).

5.1.3.1 Příklad vložení nového záznamu zákazníka s fotkou a obsahující 2 virtuální Odisky:

Tag

Length

CustomerID - 15bc279b-dda6-4a96-8a32-c83d798ab01c

Photo – 0x00 0x01 0x00 0x01 0x00 0x01

CustomerProfile 1 – 1 (Dospělý 15+)

CustomerProfile 2 – 9 (Zaměstnanec)

AppInstanceID 1 - e917e5e3-f912-4c90-9a32-94dd25bd0c0e

AppInstanceID 2 - ae4567ef-e5fb-4285-a04f-7259add186bd

Firstname – Petr

Lastname - Novák

Fotografie je vždy jenom jedna, i když struktura podporuje více fotografií u jednoho uživatele.

0x01 0x5A 0x00 0x9B 0x27 0xBC 0x15 0xA6 0xDD 0x96 0x4A 0x8A 0x32 0xC8 0x3D 0x79 0x8A 0xB0 0x1C 0x11 0x47 0x00 0x21 0x06 0x00 0x00 0x01 0x00 0x01 0x00 0x01 0x22 0x10 0x00 0xE3 0xE5 0x17 0xE9 0x12 0xF9 0x90 0x4C 0x9A 0x32 0x94 0xDD 0x25 0xBD 0x0C 0x0E 0x22 0x10 0x00 0xEF 0x67 0x45 0xAE 0xFB 0xE5 0x85 0x42 0xA0 0x4F 0x72 0x59 0xAD 0xD1 0x86 0xBD 0x23 0x01 0x00 0x01 0x23 0x01 0x00 0x09 0x24 0x04 0x00 0x50 0x65 0x74 0x72 0x25 0x06 0x00 0x4e 0x6f 0x76 0xc3 0xa1 0x6b

5.1.3.2 Příklad smazání zákazníka:

Tag

Length

CustomerID - 15bc279b-dda6-4a96-8a32-c83d798ab01c

0x01 0x13 0x00 0x9B 0x27 0xBC 0x15 0xA6 0xDD 0x96 0x4A 0x8A 0x32 0xC8 0x3D 0x79 0x8A 0xB0 0x1C 0x12 0x00 0x00

5.1.3.3 Příklad smazání AppInstanceID u zákazníka:

Tag

Length

CustomerID - 15bc279b-dda6-4a96-8a32-c83d798ab01c

AppInstanceID - e917e5e3-f912-4c90-9a32-94dd25bd0c0e

0x01 0x26 0x00 0x9B 0x27 0xBC 0x15 0xA6 0xDD 0x96 0x4A 0x8A 0x32 0xC8 0x3D 0x79 0x8A 0xB0 0x1C 0x12 0x13 0x00 0x22 0x10 0x00 0xE3 0xE5 0x17 0xE9 0x12 0xF9 0x90 0x4C 0x9A 0x32 0x94 0xDD 0x25 0xBD 0x0C 0x0E

5.1.3.4 Příklad smazání CustomerProfile u zákazníka:

Tag

Length

CustomerID - 15bc279b-dda6-4a96-8a32-c83d798ab01c

CustomerProfile – 9 (Zaměstnanec)

0x01 0x17 0x00 0x9B 0x27 0xBC 0x15 0xA6 0xDD 0x96 0x4A 0x8A 0x32 0xC8 0x3D 0x79 0x8A 0xB0 0x1C 0x12 0x13 0x00 0x23 0x01 0x00 0x09

5.1.3.5 Příklad update fotky, změny přijmení a přiřazení nového AppInstanceID:

Tag

Length

CustomerID - 15bc279b-dda6-4a96-8a32-c83d798ab01c

Photo – 0x01 0x02 0x03 0x04 0x05 0x06

AppInstanceID - ae4567ef-e5fb-4285-a04f-7259add186bd

Lastname - Nováková

0x01 0x3F 0x00 0x9B 0x27 0xBC 0x15 0xA6 0xDD 0x96 0x4A 0x8A 0x32 0xC8 0x3D 0x79 0x8A 0xB0 0x1C 0x13 0x16 0x00 0x21 0x06 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x25 0x0A 0x00 0x4e 0x6f 0x76 0xc3 0xa1 0x6b 0x6f 0x76 0xc3 0xa1 0x11 0x13 0x00 0x22 0x10 0x00 0xEF 0x67 0x45 0xAE 0xFB 0xE5 0x85 0x42 0xA0 0x4F 0x72 0x59 0xAD 0xD1 0x86 0xBD

6 Webová služba KODIS pro Whitelist karet

6.1 REST API

6.1.1 POST metoda /whitelistCard/Get

Metoda pro získání informací o předgenerovaných souborech whitelistů karet pro Virtuální ODISku, které je potřeba stáhnout do zařízení ze serveru KODIS, aby zařízení získalo aktualizovaný whitelist karet.

6.1.1.1 Vstupní parametry

- Parametry budou předány jako HTTP *Content-Type: application/json*.

- *object* ***credentials*** – Obsahuje informace pro autentizaci a autorizaci. [povinná položka]
  - *string* **userLogin** – Uživatelské jméno v clearingu KODIS. [povinná položka]
  - *string* **password** – Heslo uživatele v clearingu KODIS. [povinná položka]
- *long* **deviceNo** – Číslo zařízení v clearingu KODIS. [povinná položka], pro dopravce, který komunikuje se serverem KODIS prostřednictvím jednoho serveru, bude dostačující jedno deviceNo, dohledání případně sporných případů u konkrétních zařízení pak bude na straně tohoto dopravce
- *enum* **type** – Typ whitelistu. [povinná položka]
  - 1 = "FULL" (Plný)
  - 2 = "INC" (Inkrementální), inkrementy budou generovány minimálně jednou za 15 minut
- *int* **sequenceNo** – Počáteční pořadové číslo whitelistu, který se má stáhnout. [povinná položka pro inkrementální whitelist]

  Řešení umožňuje i stahování po částech. V samotné funkci getFile, kde si odbavovací zařízení požádá o část souboru přes fileOffset a dataLength. Pro stažení souboru po částech je proces nastaven. Data nejsou nijak komprimována, aby se i velké soubory mohly parsovat po částech i po delší době. V takovém případě může dojít k prodlevě mezi staženými daty a správnou kontrolou příslušných dokladů, kterou je potřeba v případě tohoto ojedinělého jevu tolerovat.

6.1.1.2 Výstupní parametry

- Metoda vrací návratové parametry jako HTTP *Content-Type: application/json.*
  - *object* **responseStatus** – Obsahuje informace o případné chybě. [povinná položka]
    - *int* **code** [povinná položka]
      - 0 = OK
      - jinak chyba
    - *string* **message** – Chybové hlášení. [povinná položka]
  - *long* **maxDataLength** – Max. délka dat v bajtech z obsahu souboru whitelistu, kterou je možné naráz stáhnout, viz metoda getFile(). [povinná položka, v případě, že nedošlo k chybě]
  - *array* **whitelistFiles[]** – Pole jednotlivých objektů whitelistů (informací o souborech předgenerovaných whitelistů karet). [povinná položka, v případě, že nedošlo k chybě]
    - *string* **fileName** – Název souboru whitelistu ke stažení. [povinná položka]
    - *long* **fileSize** – Celková velikost souboru whitelistu v bajtech. [povinná položka]
    - *enum* **type** – Typ whitelistu. [povinná položka]
      - 1 = "FULL" (Plný)
      - 2 = "INC" (Inkrementální)
    - *int* **sequenceNo** – Pořadové číslo whitelistu. [povinná položka]

6.1.1.3 Příklad JSON požadavku na získání informací o souborech Whitelistů karet

```
{
        "credentials": {
                "userLogin": "string",
                "password": "string"
        },
```

```
        "deviceNo": 1,
        "type": 2,
        "sequenceNo": 11
}
```


6.1.1.4 Příklad JSON odpovědi (úspěch) na získání informací o souborech Whitelistů karet


```
{
        "responseStatus":
        {
                "code": 0,
                "message": "string"
        },
        "maxDataLength": 107374182400,
        "whitelistFiles":
        [
                {
                        "fileName": "VO_WL_CARDS_INC_20200430_0015",
                        "fileSize": 1073741824,
                        "type": "INC",
                        "sequenceNo": 11
                },
                {
                        "fileName": "VO_WL_CARDS_INC_20200430_0030",
                        "type": "INC",
                        "fileSize": 536870912,
                        "sequenceNo": 12
                }
        ]
}
```


6.1.1.5 Příklad JSON odpovědi (neúspěch) na stažení souboru s Whitelistem karet


```
{
        "responseStatus":
        {
                "code": 1,
                "message": "string"
        }
}
```


6.1.2 POST metoda /whitelistCard/getFile

Univerzální metoda pro stažení souboru po částech ze serveru KODIS.

6.1.2.1 Vstupní parametry

- Parametry budou předány jako HTTP *Content-Type: application/json*.
    - *object* ***credentials*** – Obsahuje informace pro autentizaci a autorizaci. [povinná položka]
        - *string* **userLogin** – Uživatelské jméno v clearingu KODIS. [povinná položka]
        - *string* **password** – Heslo uživatele v clearingu KODIS. [povinná položka]
    - *long* **deviceNo** – Číslo zařízení v clearingu KODIS. [povinná položka], pro dopravce, který komunikuje se serverem KODIS prostřednictvím jednoho serveru, bude dostačující jedno deviceNo, dohledání případně sporných případů u konkrétních zařízení pak bude na straně tohoto dopravce
    - *string* **fileName** – Název souboru ke stažení. [povinná položka]

- *long* **fileOffset** – Pořadové číslo bajtu v obsahu souboru (offset), od kterého se má začít stahovat. [povinná položka]
- *long* **dataLength** – Požadovaná délka dat v bajtech z obsahu souboru, která má být stažena. [povinná položka]

6.1.2.2 Výstupní parametry

- V případě úspěchu metoda vrací binární data s obsahem požadovaného souboru jako HTTP *Content-Type: application/octect-stream*.
- V případě neúspěchu vrací metoda návratové parametry jako HTTP *Content-Type: application/json.*
  - *object* **responseStatus** [povinná položka]
    - *int* **code** [povinná položka]
      - číslo chyby, číselník chyb uveden v kap. 9
    - *string* **message** – chybové hlášení [povinná položka]

6.1.2.3 Příklad JSON požadavku na stažení souboru s Whitelistem karet

```
{
	"credentials": {
		"userLogin": "string",
		"password": "string"
	},
	"deviceNo": 1,
	"fileName": "VO_WL_CARDS_FULL_20200430_0000",
	"fileOffset": 0,
	"dataLength": 1024
}
```

6.1.2.4 Příklad JSON odpovědi (neúspěch) na stažení souboru s Whitelistem karet

```
{
	"responseStatus":
	{
		"code": 2,
		"message": "string"
	}
}
```

6.1.3 POST metoda /whitelistCard/getData

Metoda pro stažení aktuálních informací o kartě dle logického čísla pomocí online dotazu ze serveru KODIS.

6.1.3.1 Vstupní parametry

- Parametry budou předány jako HTTP *Content-Type: application/json*.
  - *object* ***credentials*** – Obsahuje informace pro autentizaci a autorizaci. [povinná položka]
    - *string* **userLogin** – Uživatelské jméno v clearingu KODIS. [povinná položka]
    - *string* **password** – Heslo uživatele v clearingu KODIS. [povinná položka]
  - *long* **deviceNo** – Číslo zařízení v clearingu KODIS. [povinná položka], pro dopravce, který komunikuje se serverem KODIS prostřednictvím jednoho serveru, bude dostačující jedno deviceNo, dohledání případně sporných případů u konkrétních zařízení pak bude na straně tohoto dopravce
  - *int* **cardNo** – Logické číslo karty. [povinná položka]

6.1.3.2 Výstupní parametry

- Metoda vrací návratové parametry jako HTTP *Content-Type: application/json.*
  - *object* **responseStatus** – Obsahuje informace o případné chybě. [povinná položka]
    - *int* **code** [povinná položka]
      - 0 = OK
      - jinak chyba
    - *string* **message** – Chybové hlášení. [povinná položka]
  - *object* **card** – informace o kartě [povinná položka, v případě, že nedošlo k chybě]
    - *guid* **customerID** – číslo zákazníka [povinná položka]
    - *byte[]* **photo** – binárních data fotografie svázaného s daným CustomerID [povinná položka]

guid **appInstanceID** – ID instance virtuální ODISky [povinná položka]

7 Whitelist jízdenek

Tento dokument popisuje formát předávání whitelistu jízdenek virtuálních karet ODISka do odbavovacích zařízení pomocí RestApi. Data zahrnují dlouhodobé časové jízdenky.

7.1 REST API

7.1.1 POST metoda /whitelistTicket/Get

Metoda vrací whitelist jízdenek virtuálních ODISek. V případě FULL požadavku vrací metoda aktuální data do poslední dávky. V případě INC požadavku vrací data od zadaného sequenceNo až do poslední dávky. Pro zjištění, zdali došlo k resetu whitelistu slouží metoda ResetWasPerformed. Pokud došlo k resetu whitelistu, je doporučeno stáhnout rovnou whitelist pomocí FULL požadavku. Řešení umožňuje i stahování po částech. V samotné funkci getFile, kde si odbavovací zařízení požádá o část souboru přes fileOffset a dataLength. Pro stažení souboru po částech je proces nastaven. Data nejsou nijak komprimována, aby se i velké soubory mohly parsovat po částech i po delší době. V takovém případě může dojít k prodlevě mezi staženými daty a správnou kontrolou příslušných dokladů, kterou je potřeba v případě tohoto ojedinělého jevu tolerovat.

Pokud je však zajištěna konzistence WL na zařízení, není reset nutný.

7.1.1.1 Vstupní parametry

- Parametry budou předány jako HTTP *Content-Type: application/json*.
    - *object* ***credentials*** – Obsahuje informace pro autentizaci a autorizaci. [povinná položka]
        - *string* **userLogin** – Uživatelské jméno v clearingu KODIS. [povinná položka]
        - *string* **password** – Heslo uživatele v clearingu KODIS. [povinná položka]
    - *long* **deviceNo** – Číslo zařízení v clearingu KODIS. [povinná položka], pro dopravce, který komunikuje se serverem KODIS prostřednictvím jednoho serveru, bude dostačující jedno deviceNo, dohledání případně sporných případů u konkrétních zařízení pak bude na straně tohoto dopravce
    - *enum* **type** – Typ whitelistu. [povinná položka]
        - 1 = "FULL" (Plný)
        - 2 = "INC" (Inkrementální)
    - *int* **sequenceNo** – Počáteční pořadové číslo whitelistu, který se má stáhnout. [povinná položka pro inkrementální whitelist]

7.1.1.2 Výstupní parametry

- Metoda vrací návratové parametry jako HTTP *Content-Type: application/json.*
    - *object* **responseStatus** – Obsahuje informace o případné chybě. [povinná položka]
        - *int* **code** [povinná položka]
            - 0 = OK
            - jinak chyba
        - *string* **message** – Chybové hlášení. [povinná položka]
    - *array* **whitelistTickets[]** – Pole jednotlivých objektů whitelistů jízdenek. [povinná položka, v případě, že nedošlo k chybě]
        - *object* **header** – Obsahuje obecné informace o whitelistu. [povinná položka]
            - *string* **fileVersion** – Verze souboru. [povinná položka]
            - *datetime* **issueDateTime** – Datum vytvoření dávky. [povinná položka]
            - *enum* **type** – Typ whitelistu. [povinná položka]
                - 1 = "FULL" (Plný)
                - 2 = "INC" (Inkrementální)

- *int* **sequenceNo** – Pořadové číslo whitelistu. [povinná položka]
- *int* **itemsCount**– Počet záznamů. [povinná položka]
- *bool* **reset** – Příznak, zdali došlo k resetu whitelistu. [povinná položka]

▪ *object* **data** – Obsahuje data jízdenek. [povinná položka]

- *guid* **appInstanceID** – ID instance virtuální ODISky. [povinná položka]
- *int* **cp** – Číslo zákaznického profilu. [povinná položka]
- *int* t**p** – Číslo tarifu. [povinná položka]
- *datetime* **validFrom** – Začátek platnosti jízdenky. [povinná položka]
- *datetime* **validTo** – Konec platnosti jízdenky. [povinná položka]
- *int* ***zones[]*** – Pole obsahující čísla zón. [povinná položka]
- *enum* ***stateCode*** – Stav záznamu. [povinná položka]
  - 1 = "ADD" (přidání záznamu)
  - 2 = "DEL" (odebrání záznamu)

7.1.1.3 Příklad JSON požadavku na získání informací o Whitelistech jízdenek


```
{
	"credentials": {
		"userLogin": "string",
		"password": "string"
	},
	"deviceNo": 1,
	"type": 2,
	"sequenceNo": 11
}
```


7.1.1.4 Příklad JSON odpovědi (úspěch) na získání informací o Whitelistech jízdenek


```
{
	"whitelistTickets": [
	{
		"header":
		{
			"fileVersion": "string",
			"issueDateTime": "2022-04-04T11:45:42.347Z",
			"sequenceNo": 0,
			"type": 1,
			"itemsCount": 0,
			"reset": true
		},
		"data": [
		{
			"appInstanceID": "00000000-0000-0000-0000-000000000000",
			"cp": 0,
			"tp": 0,
			"validFrom": "2022-04-04T11:45:42.347Z",
			"validTo": "2022-04-04T11:45:42.347Z",
			"zones": [ 0 ],
			"stateCode": 1
		} ]
	} ],
	"responseStatus":
	{
		"code": 0,
		"message": "string"
```

```
        }
}
```

7.1.2 POST /whitelistTicket/ResetWasPerformed

Metoda vrací informaci, zdali došlo k resetu whitelistu jízdenek virtuálních ODISek od zadaného sequenceNo. Pokud došlo k resetu whitelistu, je doporučeno stáhnout rovnou whitelist pomocí FULL požadavku. Řešení umožňuje i stahování po částech. V samotné funkci getFile, kde si odbavovací zařízení požádá o část souboru přes fileOffset a dataLength. Pro stažení souboru po částech je proces nastaven. Data nejsou nijak komprimována, aby se i velké soubory mohly parsovat po částech i po delší době. V takovém případě může dojít k prodlevě mezi staženými daty a správnou kontrolou příslušných dokladů, kterou je potřeba v případě tohoto ojedinělého jevu tolerovat.

Pokud je však zajištěna konzistence WL na zařízení, není reset nutný.

7.1.2.1 Vstupní parametry

- Parametry budou předány jako HTTP *Content-Type: application/json*.
    - *object* ***credentials*** – Obsahuje informace pro autentizaci a autorizaci. [povinná položka]
        - *string* **userLogin** – Uživatelské jméno v clearingu KODIS. [povinná položka]
        - *string* **password** – Heslo uživatele v clearingu KODIS. [povinná položka]
    - *long* **deviceNo** – Číslo zařízení v clearingu KODIS. [povinná položka], pro dopravce, který komunikuje se serverem KODIS prostřednictvím jednoho serveru, bude dostačující jedno deviceNo, dohledání případně sporných případů u konkrétních zařízení pak bude na straně tohoto dopravce
    - *int* **sequenceNo** – Pořadové číslo whitelistu, od kterého bude kontrola na reset. [povinná položka pro inkrementální whitelist]

7.1.2.2 Výstupní parametry

- Metoda vrací návratové parametry jako HTTP *Content-Type: application/json.*
    - *object* **responseStatus** – Obsahuje informace o případné chybě. [povinná položka]
        - *int* **code** [povinná položka]
            - 0 = OK
            - jinak chyba
        - *string* **message** – Chybové hlášení. [povinná položka]
    - *enum* **reset** – Příznak, zdali došlo k resetu whitelistu. [povinná položka]
        - 1 = "RESET_WAS_PERFORMED" (Reset byl proveden)
        - 2 = "RESET_NOT_PERFORMED" (Reset nebyl proveden)
    - *int* **sequenceNo** – Poslední aktuální pořadové číslo whitelistu. [povinná položka]

7.1.2.3 Příklad JSON požadavku na získání informací zdali došlo k resetu Whitelistu jízdenek

```
{
        "credentials": {
                "userLogin": "string",
                "password": "string"
        },
        "deviceNo": 1,
        "sequenceNo": 11
}
```

7.1.2.4 Příklad JSON odpovědi (úspěch) na získání informací zdali došlo k resetu Whitelistu jízdenek


```
{
	"reset": 1,
	"sequenceNo": 0,
	"responseStatus":
	{
		"code": 0,
		"message": "string"
	}
}
```


7.1.3 POST metoda /whitelistTicket/getData

Metoda pro stažení aktuálních informací o jízdenkách dle logického čísla pomocí online dotazu ze serveru KODIS.

7.1.3.1 Vstupní parametry

- Parametry budou předány jako HTTP *Content-Type: application/json*.
  - *object* ***credentials*** – Obsahuje informace pro autentizaci a autorizaci. [povinná položka]
    - *string* **userLogin** – Uživatelské jméno v clearingu KODIS. [povinná položka]
    - *string* **password** – Heslo uživatele v clearingu KODIS. [povinná položka]
  - *long* **deviceNo** – Číslo zařízení v clearingu KODIS. [povinná položka], pro dopravce, který komunikuje se serverem KODIS prostřednictvím jednoho serveru, bude dostačující jedno deviceNo, dohledání případně sporných případů u konkrétních zařízení pak bude na straně tohoto dopravce
  - *int* **cardNo** – Logické číslo karty. [povinná položka]

7.1.3.2 Výstupní parametry

- Metoda vrací návratové parametry jako HTTP *Content-Type: application/json.*
  - *object* **responseStatus** – Obsahuje informace o případné chybě. [povinná položka]
    - *int* **code** [povinná položka]
      - 0 = OK
      - jinak chyba
    - *string* **message** – Chybové hlášení. [povinná položka]
  - *array[]* **tickets** – Pole jednotlivých objektů jízdenek. [povinná položka, v případě, že nedošlo k chybě]
    - *int* **cp** – Číslo zákaznického profilu. [povinná položka]
    - *int* t**p** – Číslo tarifu. [povinná položka]
    - *datetime* **validFrom** – Začátek platnosti jízdenky. [povinná položka]
    - *datetime* **validTo** – Konec platnosti jízdenky. [povinná položka]
    - *int* ***zones[]*** – Pole obsahující čísla zón. [povinná položka]

8 Velikosti a frekvence přenášených dat

8.1 White list karet

Maximální velikost absolutního WL karet je předpokládána v prvé etapě do 2 GB, v cílovém stavu i vyšší (až do 15 GB).

Pro přenosná zařízení železniční dopravy bude maximální velikost absolutního WL karet do 2 GB.

Frekvence stahování (přírůstku) – každých 15 minut, s možností on-line dotazu (popis v kap. 6.1.3)

Maximální velikost běžného denního přírůstku 20MB, který bude rozprostřen do půlhodinových přírůstků během dne. Výjimečně může být tato velikost vyšší z technických důvodů hromadných operací apod.

Řešení umožňuje i stahování po částech. V samotné funkci getFile, kde si odbavovací zařízení požádá o část souboru přes fileOffset a dataLength. Pro stažení souboru po částech je proces nastaven. Data nejsou nijak komprimována, aby se i velké soubory mohly parsovat po částech i po delší době.
V takovém případě může dojít k prodlevě mezi staženými daty a správnou kontrolou příslušných dokladů, kterou je potřeba v případě tohoto ojedinělého jevu tolerovat.

8.2 White list kupónů

Maximální velikost absolutního WL kupónů je v první etapě předpokládána do 30 MB, v cílovém stavu až do 250 MB.

Frekvence stahování (přírůstku) – každých 20 minut, s možností on-line dotazu (popis v kap. 7.1.3)

Maximální velikost denního přírůstku bude cca 1,5 MB, který bude rozprostřen do půlhodinových přírůstků během dne.

9 Číselník chyb

0	OK
100	ERROR on checking user credentials: User not found
101	ERROR on checking user credentials: User found more than once
102	ERROR on checking user right: User is not assigned the right
103	ERROR on checking user right: User not found by ID
104	ERROR on checking device: Device not found by number and provider
234	ERROR on getting whitelist ticket: value of parameter sequenceNo must be greater than 0
235	ERROR on getting whitelist ticket: value of parameter sequenceNo must be not null
236	ERROR on getting whitelist card: value of parameter sequenceNo must be greater than 0
237	ERROR on getting whitelist card: value of parameter sequenceNo must be not null
238	ERROR on getting whitelist card: file not found or file is corrupted
239	ERROR on getting whitelist card: file name must be not empty
240	ERROR on getting whitelist card: file offset value must be equal or greater than 0
241	ERROR on getting whitelist card: file data value must be greater than 0