Simulátory pro potřeby QA a testování

Simulators for QA and Testing Needs

Mojmír Goláň

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Mojmír Goláň**
Osobní číslo:	**A21818**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Simulátory pro potřeby QA a testování**
Téma práce anglicky:	**Simulators for QA and Testing Needs**

Zásady pro vypracování

1. Nastudujte a rozepište potřebnou terminologii v kontextu tématu práce.
2. Popište stávající možnosti využití simulátorů v rámci QA.
3. Navrhněte využití simulátoru v rámci testování pro potřeby lokální firmy.
4. Realizujte své řešení.
5. Výsledky vhodně reprezentujte a popište.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. SOKOLOWSKI, John A. a BANKS, Catherine M. *Principles of modeling and simulation: a multidisciplinary approach*. Hoboken: John Wiley, 2009. ISBN 978-0-470-28943-3.
2. PELÁNEK, Radek. *Modelování a simulace komplexních systémů: jak lépe porozumět světu*. Brno: Masarykova univerzita, 2011. ISBN 978-80-210-5318-2.
3. O'SULLIVAN, David a George PERRY. *Spatial Simulation: Exploring Pattern and Process*. Wiley-Blackwell, c2013. ISBN 978-1-119-97079-8.
4. CELLIER, François E. a KOFMAN, Ernesto. *Continuous system simulation*. New York: Springer, c2010. ISBN 978-1-4419--3863-3.
5. WORLDWIDE, ACI a Inc. *Standard POS device message specifications manual ACI* [Online PDF]. Version 10. ACI Worldwide, c2011.
6. WORLDWIDE, ACI a Inc. *BIC ISO Standards Manual: BASE24* [Online PDF]. Version 10. ACI Worldwide, c2017.
7. ISO. *ISO 8583-1 – INTERNATIONAL STANDARD: Financial transaction card originated messages — Interchange message specifications* [Online PDF]. ISO, c2003.
8. ATPCO Ticket Exchange Services (TCN): TCN Steering Committee c/o APTCO, Revenue Systems Department, Dulles International Airport, P.O. Box 17415, Washington, D.C. 20041-0415

Vedoucí bakalářské práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **5. listopadu 2023**
Termín odevzdání bakalářské práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Mojmír Goláň, v.r.
podpis studenta

ABSTRAKT

Hlavním cílem této bakalářské práce byl popis vývoje a využití simulátorů v oblasti finanční komunikace, zejména v kontextu komunikačních protokolů jako je ISO8583. Projekt se zaměřuje na potřeby oddělení kontroly kvality lokální firmy Monet+. Práce obsahuje návod pro úpravy simulátorů v jazycích JavaScript a Python pro potřeby testování různých požadavků. Dále podrobněji popisuje protokol ISO8583, který je v simulátorech využíván, typy transakcí s jejich MTID a jak probíhá komunikace uvnitř finančního procesu.

Klíčová slova: Simulátor, ISO8583, komunikace, JavaScript, Python, BICISO, Pos to Host, Host to Host, Jython, Nashorn, ACI, BASE24, skripty, konfigurace, kontrola kvality, testy, protokol, bitmapa, datová pole, MTID, transakce, testovací scénáře, autorizační host, QA

ABSTRACT

The main objective of this bachelor thesis is the development and utilization of simulators for financial communications, particularly in the context of communication protocols such as ISO8583. The project focuses on the needs of the quality control department of the local company Monet+. It includes a guide for modifying simulators in JavaScript and Python to meet various testing requirements. Furthermore, it provides a detailed description of the ISO8583 protocol used in the simulators, transaction types with their MTID, and the communication process within the financial workflow.

Keywords: Simulator, ISO8583, communication, JavaScript, Python, BICISO, Pos to Host, Host to Host, Jython, Nashorn, ACI, BASE24, scripts, configuration, quality control, tests, protocol, bitmap, data fields, MTID, transactions, test scenarios, authorization host, QA

Poděkování, motto a čestné prohlášení, že odevzdaná verze bakalářské práce a verze elektronická, nahraná do IS/STAG jsou totožné ve znění:

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Simulátory jsou specializované softwarové nástroje používané v oblasti testování k napodobování chování skutečných systémů nebo komponent. V kontextu vývoje a testování softwaru simulátory reprodukují funkce hardwarových zařízení, softwarových aplikací nebo celých systémů v kontrolovaném prostředí. Umožňují vývojářům a testerům simulovat různé scénáře a podmínky, což jim umožňuje posoudit výkon, funkčnost a spolehlivost svého softwaru bez potřeby fyzického hardwaru nebo interakcí v reálném světě.

Na našem oddělení QA potřebujeme použít simulátory k otestování funkcí našeho kódu s maximálním přiblížením reálnému produkčnímu prostředí. Tyto funkce jsou spojeny s příchodem nových zákazníků a jejich specifických požadavků, které nemusí být vždy správně navrženy, pochopeny anebo implementovány. K tomu se pro naše oddělení perfektně hodí využití simulátorů, díky oddělenému prostředí jsme schopni se zákazníkem v momentě, kdy nastanou jakékoliv nesrovnalosti se specifikací nebo požadavky, dané nálezy prodiskutovat a rovnou se domluvit na úpravách, změnách požadavků nebo jen poskytnutí názoru na návrh řešení. Hlavním cílem této diplomové práce bylo vytvořit dokument, který by dopodrobna popisoval využití, tvorbu a následnou úpravu simulátorů pro potřeby testování specifických dat.

Projekt lze tematicky rozdělit do několika částí. První je věnována studiu a popisu potřebné terminologie v kontextu práce, je zde rozepsáno, co obnáší testování a co si pod pojmem představit. Pomocí veřejně dostupné literatury a jiných prostředků jsou popsány typy komunikací použitých pro přenos dat pomocí protokolů, jsou zde charakterizovány dva protokoly pro představení a zdůraznění odlišností, dále je podrobně popsán protokol ISO8583, který je předmětem tvorby simulátorů, protože je jako jeden z mála používaných protokolů veřejně dostupný.

Druhá část se zabývá popisem stávajících možností využití simulátorů na oddělení QA a zároveň se zabývá návrhem využití simulátoru v rámci testování pro potřeby lokální firmy. Jsou zde tedy rozebrány a následně zobrazeny možnosti, jejich využití se zobrazením aktuálních statistik.

Poslední část je věnována realizaci řešení a popisu úprav provedených na simulátorech pro různé scénáře testování. V této části je popsán simulátor a jeho funkce, jsou zde zobrazeny a ukázány možné editace pro možnosti testování.

Cílem je vytvořit takový dokument, který by dopodrobna popisoval výhody využití, tvorbu a následnou manipulaci se simulátory pro potřeby testování specifických dat nebo scénářů tak, aby našel využití jako zaškolovací dokument pro nové pracovníky.

I. TEORETICKÁ ČÁST

1 QA A TESTOVANI

1.1 Co je testování

Testování je klíčový proces při vývoji softwaru zaměřený na hodnocení kvality softwaru a snižování rizika selhání v provozu. Zahrnuje činnosti k odhalení defektů a hodnocení kvality softwarových artefaktů, nazývaných testované objekty. Na rozdíl od běžných nedorozumění testování přesahuje pouhé provádění testů; zahrnuje plánování, řízení, odhadování, monitorování a kontrolu po celý životní cyklus vývoje softwaru.

Existují dvě hlavní kategorie testování: dynamické a statické. Dynamické testování vyžaduje spuštění softwaru a zahrnuje techniky k definování testovacích případů, zatímco statické testování nevyžaduje spuštění softwaru a zahrnuje revize a statickou analýzu. [1]

Typické cíle testování zahrnují hodnocení pracovních produktů, nalezení defektů, zajištění požadovaného pokrytí, snížení rizika, ověření požadavků, validaci proti smluvním a regulačním požadavkům, poskytování informací pro rozhodování, budování důvěry v kvalitu softwaru a ověření kompletnosti a očekávané funkčnosti. Cíle testování se mohou lišit v závislosti na faktorech, jako je povaha produktu, úroveň testování, rizika, životní cyklus vývoje softwaru a byznysový kontext, včetně organizační struktury, konkurenčních aspektů a doby uvedení na trh.

Testování má za cíl vyvolat selhání způsobené defekty. Potvrzovací testování ověřuje řešení problémů, ideálně prováděné stejnou osobou, která prováděla původní testování. Regresní testování zajišťuje, že opravy nezpůsobují selhání v jiných částech testovaného objektu. [2]

1.2 Význam QA testovaní

Samotné testování je klíčovou součástí zajištění kvality softwaru, avšak ještě důležitější je celkový proces zajištění kvality (QA – quality assurance). QA představuje preventivní přístup zaměřený na procesy a jejich neustálé zlepšování. Narozdíl od kvality produktu, kterou kontroluje Quality Control (QC), QA se soustředí na procesy a jejich implementaci.

Zajištění kvality se neomezují pouze na testovací tým; všichni zainteresovaní členové projektu mohou využít své testovací dovednosti k dosažení úspěchu projektu. Identifikace defektů v softwaru, testování komponent a systémů a související dokumentace jsou klíčové pro zajištění kvality produktu.

Dále QA umožňuje přímé hodnocení kvality testovaného objektu v různých fázích životního cyklu vývoje softwaru (SDLC), což usnadňuje rozhodování o postupu projektu. QA je založená na preventivním přístupu, který je zaměřen na procesy a jejich kontinuální zlepšování. [3]

1.2.1 Potřeby QA spojené se simulátory

V rámci oddělení zajištění kvality (QA) se objevuje potřeba pro samostatné simulátory pro naše testovací prostředí, kde je tento požadavek podmíněn všeobecnými požadavky pro testování (pod pojmem všeobecné požadavky na testování jsou zde myšleny každodenní potřeby na testování – testování nových implementací podle požadavků zákazníka, testování bugů a fixů s nimi spojených), integračním testováním, nastavením a přípravou regresních testů.

Téměř každý den jsou vyvíjeny a implementovány nové funkce a jejich aplikace – tyto nové funkcionality je také potřeba otestovat a díky simulátorům na našem samostatném prostředí jsme schopni si vše nastavit a otestovat sami bez nutnosti zásahu třetích stran. Mnohdy nová funkcionalita může znamenat fix předchozího problému, který je potřeba opět vyvolat a otestovat. To je další z plusů, které nám simulátory přivádějí, díky jejich nastavení je mnohonásobně snazší nastavení předmětu testování – tedy v případě, že chceme otestovat ošetření chyby nebo bugu, už nebude nutné pokaždé žádat zákazníka o vyvolání daného problému (bugu) a poskytovat kroky, které je potřeba podniknout, ale nyní si potřebné kroky budeme schopni nasimulovat sami a následně při potvrzení správnosti řešení požádat zákazníka o test z jejich strany (namísto kontinuálního testování, stačí jeden test).

V průběhu testování je kladen velký důraz na automatizaci regresních testů, které byly dosud prováděny ručně. Na našem oddělení využíváme několik testovacích prostředí, od Integrace až po Produkci, kde s každou vrstvou dochází k častým změnám, což pro nastavení a odladění regresních testů není vhodné, navíc na těchto prostředích mohou testovat i jiní spolupracovníci z jiných oddělení a může tedy často docházet k interferenci nastavení některých vlastností nebo jiných dat.

Při nastavení automatických regresních testů potřebujeme prostředí, které zůstává neměnné, případně má možnost změn v důsledku požadavků automatických testů (např. bude potřeba validovat částku, kterou si chce zákazník poslat). Pro tyto účely bylo vytvořeno

samostatné prostředí pro automatické testy, kde vzniká potřeba udržet nastavení daných klientů – simulátory, které budou simulovat příslušné issuery (procesory).

1.2.2 Regresní testy

Regresní testování je klíčovým procesem při změnách v softwarovém systému, který zajišťuje, že nové úpravy nezpůsobují nežádoucí dopady na existující funkcionality. Tento typ testování se skládá z konfirmačního a regresního testování.

Konfirmační testování slouží k ověření, zda byl původní defekt úspěšně opraven. Obvykle zahrnuje provedení testovacích případů, které dříve selhaly kvůli defektu, nebo přidání nových testů pokrývajících změny potřebné k opravě. [2]

Naopak, regresní testování zjišťuje, zda nedošlo k nežádoucím dopadům na již stávající fce po realizovaných změnách, včetně již potvrzených konfirmačními testy. Tato testování nemusí být omezena pouze na samotný testovaný objekt, ale mohou se rozšířit i na další související komponenty nebo prostředí. Jejich cílem je zabezpečit, aby žádná změna neovlivnila negativně funkčnost systému.

Regresní testy jsou často spouštěny opakovaně a mohou být automatizovány, což usnadňuje jejich opakované provádění a zvyšuje efektivitu testovacího procesu. Důležité je provádět regresní testování ve všech úrovních testování, kde došlo k opravám defektů nebo změnám, a případně optimalizovat jejich rozsah pomocí analýzy dopadu. [1]

1.2.3 Automatizace testů

Výhody automatizace testů:

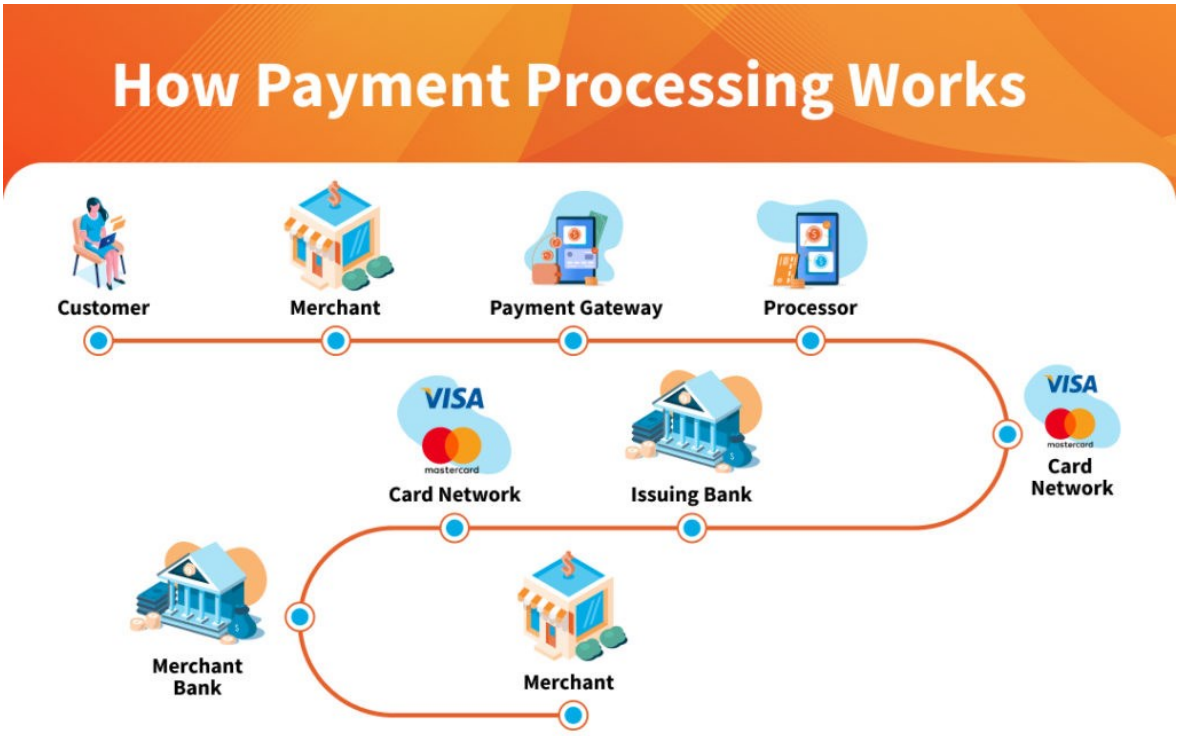
- Úspora času díky snížení manuální práce při provádění regresních testů.
- Prevence lidských chyb díky vyšší konzistenci a opakovatelnosti testů.
- Objektivnější posouzení testů a provádění složitých opatření.
- Snadnější přístup k informacím o testování pro podporu managementu.
- Zkrácení doby provádění testů a rychlejší detekce defektů.
- Úspora času testerů pro návrh nových a efektivnějších testů.
- Statistiky
- Historie vývoje a chybovosti

Rizika automatizace testů:

- Nerealistická očekávání od nástroje a nepřesné odhady času a nákladů na jeho zavedení a údržbu.
- Použití nástroje v situacích, kdy je manuální testování vhodnější.
- Přílišné spoléhání se na nástroj a ignorování kritického myšlení člověka. [3]

2 TYPY KOMUNIKACI

Komunikace hraje klíčovou roli v procesu transakcí, zajišťující spojení mezi různými systémy a umožňující efektivní zpracování platebních operací. Celý proces zpracování TRN probíhá přibližně nějak takto (viz. Obrázek 1). (poznámka – tento obrázek neznázorňuje přesný proces zpracování transakce, ale pouze slouží k představení) [4]



Obrázek 1 Hrubá ukázka procesu zpracování TRN [5]

Zde jsou dva hlavní typy komunikace probíhající na pozadí procesu zpracování TRN:

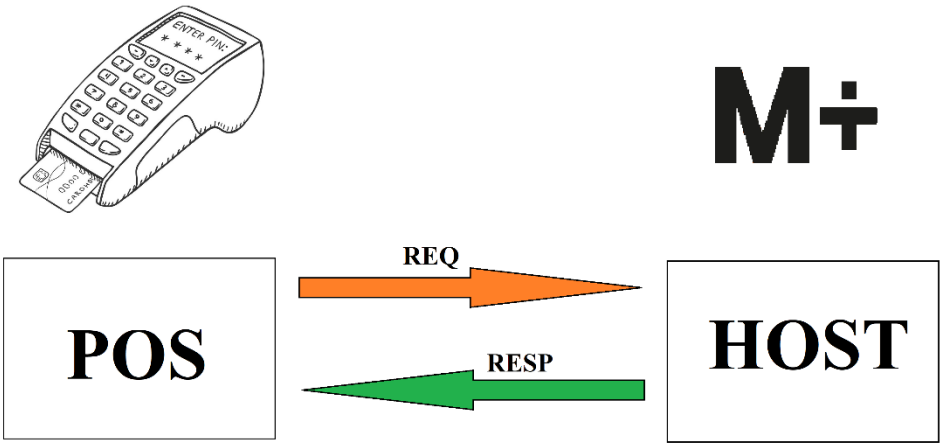
2.1 Pos to Host

P2H představuje typ komunikace, který označuje interakci mezi bodem prodeje (POS) - terminálem a autorizačním hostem (Monet+). Tento typ komunikace probíhá formou požadavku a odpovědi, umožňující POS terminálu zasílat informace o transakcích autorizačnímu serveru hostitele a obdržet odpovědi na tyto transakce (viz. Obrázek 2). [6]

P2H se odlišuje od H2H těmito dvěma hlavními aspekty:

1) P2H neudržuje stále navázané spojení oproti H2H kde je spojení navázáno neustále

2) Při komunikaci P2H je bezprostředním aspektem šifrování komunikace (u H2H se nejčastěji využívá VPN)



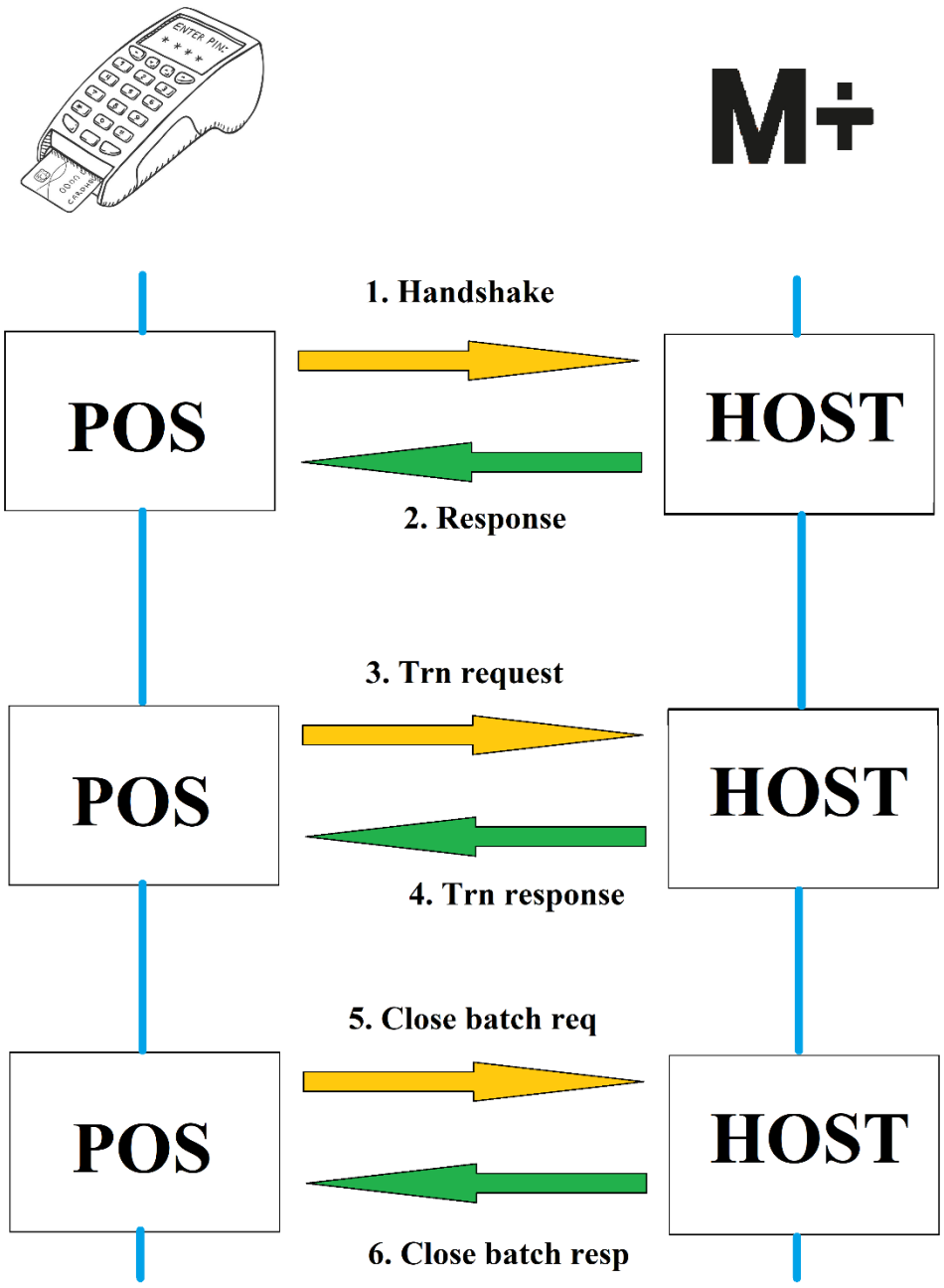
Obrázek 2 Způsob komunikace mezi POS a AH [7; 8]

- Handshake (Navázání spojení): POS terminál navazuje komunikaci se serverem hostitele po zapnutí POS. To zahrnuje navázání spojení přes síť, jako je internet nebo dedikovaná linka. Handshake zajišťuje, že jak POS terminál, tak autorizační hostitel jsou připraveni vyměňovat data. [9]
- Response (Odpověď): Po obdržení požadavku o navázání spojení server hostitele reaguje, aby potvrdil spojení a připravenost pokračovat v transakci. Tato odpověď potvrzuje, že komunikační kanál je otevřený a funkční. [10]
- TRN Request (Požadavek o transakci): POS terminál zasílá autorizačnímu serveru hostitele požadavek o transakci obsahující informace o nákupu, jako je částka transakce, údaje o kartě a identifikace obchodníka. Tato žádost je zaslána zabezpečeně serveru hostitele k autorizaci a zpracování. [11]
- TRN Response (Odpověď na transakci): Server hostitele zpracovává obdržený požadavek o transakci od POS terminálu. Ověřuje údaje o kartě, kontroluje dostupné prostředky a provádí další nezbytné kontroly pro autorizaci. Server hostitele pak odesílá

odpověď zpět na POS terminál, indikující, zda byla transakce schválena nebo odmítnuta. [12]

- Close Batch Request (Požadavek o uzavření dávky): Na konci pracovního dne nebo určitého časového období může POS terminál poslat požadavek o uzavření dávky serveru hostitele. Tato žádost informuje server hostitele, aby vyrovnal všechny schválené transakce a připravil se na další dávku transakcí. [10]
- Close Batch Response (Odpověď na uzavření dávky): Po obdržení požadavku o uzavření dávky server hostitele zpracovává požadavek a vyrovná dávku transakcí. Generuje odpověď na uzavření dávky, potvrzující úspěšné vyrovnání transakcí a poskytující případné relevantní informace o souhrnu dávky. [9]

Tyto body představují typický průběh komunikace z POS terminálu ke serveru hostitele během procesu transakce. Každý krok je nezbytný pro zajištění bezpečného a efektivního zpracování platebních transakcí (viz. Obrázek 3). [13]



Obrázek 3 Znázornění komunikace mezi POS a AH [7; 8]

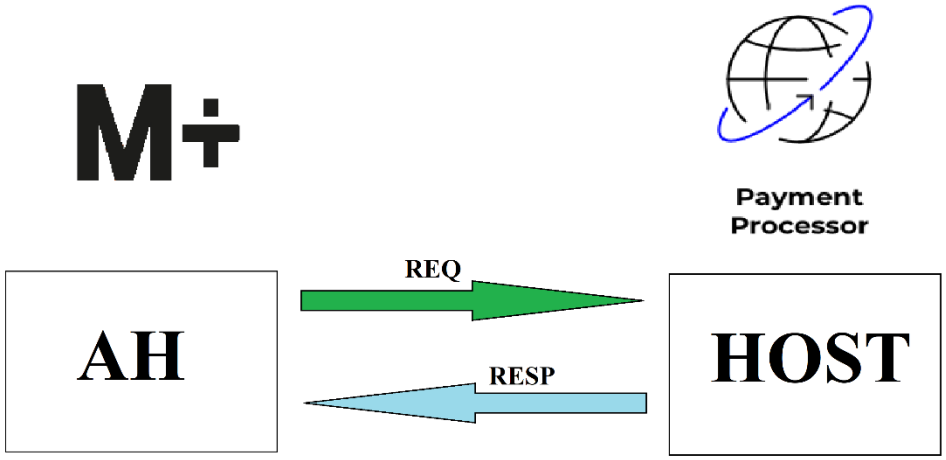
Účel handshaku je povolit na terminálu verifikaci těchto dat:

a) Status komunikačního spojení.
b) Komunikační klíč.
c) Požádat si o aktuální konfiguraci klíče pro terminál.

P2H komunikace probíhá pomocí protokolů, které umožňují zabezpečenou komunikaci mezi POS terminály a autorizačními hostiteli, zajišťují důvěryhodnost, integritu a autentičnost platebních údajů.

2.2 Host to Host

Host to Host je druh komunikace, který se odehrává mezi dvěma hostitelskými systémy, například mezi AH a Processorem. [14] Tato forma komunikace umožňuje přímý přenos dat a informací mezi těmito systémy, což je klíčové pro spolehlivou a efektivní správu platebních transakcí. [15]Tento typ komunikace probíhá podobně jako u P2H, formou požadavku a odpovědi (viz. Obrázek 4). [16]



Obrázek 4 Způsob komunikace u H2H [8; 17]

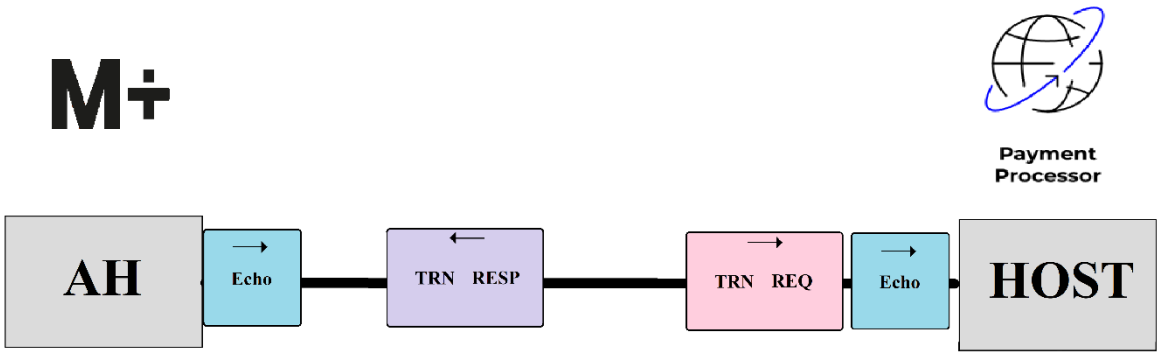
Ačkoliv, obecný průběh komunikace mezi hosty (host-to-host) a mezi bodem prodeje s hostem (POS-to-host) sdílí podobnosti, existují rozdíly v jejich implementaci a konkrétních krocích. [11]

a) V případě komunikace POS-to-host, terminál (POS) inicioval komunikaci s autorizačním hostitelem k zpracování transakce. Terminál posílá hostiteli podrobnosti o transakci k autorizaci a zpracování.
b) V komunikaci host-to-host, dva hostitelé komunikují přímo mezi sebou, přes neustále navázané spojení. Komunikace mezi hosty může zahrnovat výměnu většího objemu dat nebo provádění konkrétních operací na pozadí. [9]

Například v obou případech (jak h2h tak p2h) dochází k procesování transakcí pomocí REQ a RESP TRN zpráv. Nicméně, u H2H již není potřeba Handshaku, neboť je připojení mezi H2H neustále navázané, využívá se zde namísto navázaní spojení např. echo, které zasílá periodické zprávy o existujícím spojení po uplynutí časových intervalů. [18]

- Transaction Request (Požadavek o transakci): Iniciační hostitel posílá přijímajícímu hostiteli požadavek o transakci obsahující informace o transakci, jako je částka transakce, údaje o kartě a identifikace obchodníka. Tato žádost je bezpečně odeslána k autorizaci a zpracování.
- Transaction Response (Odpověď na transakci): Přijímající hostitel zpracovává obdržený požadavek o transakci. Ověřuje údaje o kartě, kontroluje dostupné prostředky a provádí další nezbytné kontroly pro autorizaci. Přijímající hostitel poté odesílá odpověď zpět iniciačnímu hostiteli, indikující, zda byla transakce schválena nebo odmítnuta. [10]
- Echo (Ozvěna): Iniciační host neustále posílá požadavky na ověření spojení přijímajícímu hostiteli. Tento proces většinou bývá zasílán pomocí Network Management Request (NMR) zprávy, a to většinou automaticky po uplynutí časového intervalu.

Tyto body představují typické zprávy v komunikaci od jednoho hostitele k druhému. (viz. Obrázek 5). [13]



Obrázek 5 Ukázka průběhu komunikace u H2H [8; 17]

Host to Host komunikace zahrnuje přímou komunikaci mezi dvěma hostiteli nebo počítači bez prostředního serveru. Tato komunikace je stále navázaná a udržuje se „živá“ pomocí Echo zpráv. Komunikace od Host to Host je běžná v peer-to-peer sítích a může zahrnovat různé protokoly jako TCP/IP, UDP, ISO8583, BICISO atd.

3 PROTOKOLY

Komunikační protokoly jsou standardizované sady pravidel a konvencí, které umožňují různým systémům nebo zařízením komunikovat mezi sebou přes síť. Definují formát, pořadí a mechanismy pro řízení chyb při výměně dat, zajistí tak spolehlivou přenositelnost informací, které mohou být spolehlivě přenášeny a porozuměny jak odesílateli, tak příjemci.

Zde je přehled komunikačních protokolů, včetně několika uvedených příkladů:

ISO 8583: ISO 8583 je široce používaný standard pro zprávy finančních transakcí. Definuje formát zpráv vyměňovaných mezi terminály pro prodej zboží, bankomaty a finančními institucemi během elektronických transakcí.

BICISO: BICISO je proprietární komunikační protokol používaný v bankovním průmyslu. Je často používán pro mezibankovní komunikaci, jako je převod finančních prostředků mezi finančními institucemi.

REST (Representational State Transfer): REST je architektonický styl pro návrh síťových aplikací. Používá standardní metody HTTP jako GET, POST, PUT a DELETE k provádění operací na zdrojích, což ho činí vhodným pro tvorbu webových služeb a API.

Typy komunikačních protokolů:

Binární protokoly: Tyto protokoly kódují data v binárním formátu, což je účinné pro rychlé přenášení velkého množství dat. Například ISO 8583.

Textové protokoly: Tyto protokoly používají textové formáty, které jsou čitelné pro člověka, což je usnadňuje porozumění a ladění. REST je příkladem textového protokolu, protože obvykle používá JSON nebo XML pro výměnu dat.

Standardizované protokoly: Tyto protokoly jsou definovány a udržovány standardizačními organizacemi, což umožňuje jejich široké přijetí a interoperabilitu mezi různými systémy a dodavateli. ISO 8583 je příkladem standardizovaného protokolu.

Personalizace protokolů: Mnoho komunikačních protokolů umožňuje jejich přizpůsobení nebo personalizaci pro splnění konkrétních potřeb různých aplikací nebo organizací. To může zahrnovat definování dalších polí pro zprávy, úpravu formátu zpráv nebo rozšíření funkcionalit protokolu. Personalizace umožňuje organizacím přizpůsobit komunikační protokol jejich jedinečným požadavkům, což zajišťuje efektivní a účinnou komunikaci v jejich systémech.

V kontextu komunikačních protokolů ISO 8583 (které je předmětem této práce) a BICISO, je lepší si nejprve vysvětlit kde se tyto protokoly uplatnily – ACI.

3.1 ACI

ACI Worldwide, poskytuje sofistikované platební řešení a infrastrukturu pro zpracování finančních transakcí. ISO 8583 je mezinárodním standardem pro formát zpráv a komunikační protokoly v elektronických platebních transakcích a BICISO je upravenou verzí ISO8583 protokolu a moc se od něj tedy neliší. [19]

ACI integruje tyto protokoly do svých platebních řešení, což umožňuje finančním institucím, obchodníkům a vystavovatelům faktur zabezpečeně a efektivně zpracovávat transakce v reálném čase. Její software a služby podporují komunikaci pomocí zmíněných protokolů, což umožňuje bezpečnou výměnu platebních dat mezi různými systémy a zařízeními. [20]

Díky ACI Worldwide mohou organizace využívat standardizované protokoly jako ISO 8583 k usnadnění platebních operací, což přispívá k bezpečnosti, spolehlivosti a efektivitě jejich platební infrastruktury. [20]

ACI Worldwide je uznávána pro svá průkopnická platební řešení a inovace. Její komplexní sada produktů a služeb ji činí preferovanou volbou pro organizace hledající zlepšení svých platebních schopností a poskytování prvotřídních zážitků zákazníkům. [19]

3.1.1 BASE24

BASE24 je řada aplikací vytvořených společností ACI Worldwide, Inc., které podporují platební systémy používané velkými finančními institucemi. [21]

„BASE24-atm je integrovaný systém zpracování a přepínání EFT, který poskytuje ovládání zařízení ATM; směrování a autorizaci transakcí; rozhraní pro hosty a výměny; vyúčtování; správní hlášení; síťovou kontrolu; a funkce pro uchovávání hodnoty. Systém pracuje na serveru HP NonStop a poskytuje organizacím robustní, vysokovýkonné, odolné řešení pro poskytování služeb prostřednictvím bankomatů, samoobslužných zařízení a kiosků.“ [22]

„BASE24-eps je integrovaný platební motor pro získávání, ověřování, směrování, přepínání a autorizaci finančních transakcí přes více kanálů. Představuje implementaci další generace platební platformy ACI ...

BASE24-eps poskytuje plnou škálu funkcí pro podporu platebních transakcí... Produkt je kompatibilní s Triple DES a podporuje všechny přední platební karty a nabízí standardní rozhraní pro přední zařízení, přepínače a hostitelské systémy ...“ [23]

„BASE24-infobase sbírá, distribuuje a zpracovává různé typy dat ve finančních a maloobchodních organizacích, pracuje na platformě HP NonStop a je schopný přijímat data z široké škály bankomatů.

Jedná se o systém navržený k uspokojení poptávky po kontrolovaném sběru transakcí, včetně plateb EFT a distribuci provozních dat, jako je stahování softwaru do platebních terminálů, bankomatů a řadičů poboček.“ [24]

„BASE24-pos poskytuje organizacím rychlý a výkonný systém autorizace ... S provozem na serverech HP NonStop™ nabízí BASE24-pos integritu dat pro tyto misijně kritické aplikace a umožňuje firmám využít své investice do infrastruktury a budovat základ pro budoucí rozvoj. BASE24-pos neustále podléhá vylepšování, aby poskytoval novou funkcionalitu a zároveň zachovával dodržování regulací. Proto potřebují společnosti jediné řešení pro dlouhodobé používání.“ [25]

3.2 ISO8583

ISO 8583 je mezinárodní standard pro finanční zprávy, který hraje klíčovou roli při usnadňování elektronických transakcí přes různé platební kanály. Vyvinutá Mezinárodní organizací pro normalizaci (International Organization for Standardization, zkráceně ISO), ISO 8583 definuje formát zpráv a komunikační protokol používaný finančními institucemi, platebními procesory a dalšími subjekty zapojenými do zpracování platebních transakcí. [9]

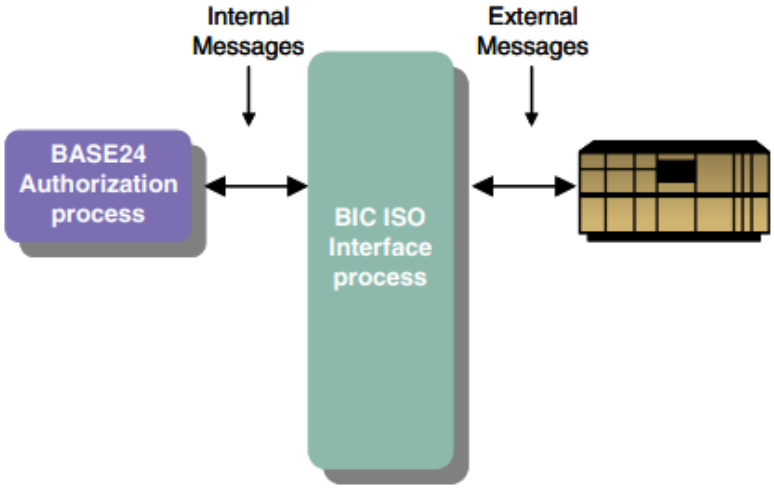
Základem je to, že ISO 8583 poskytuje standardizovaný rámec pro strukturování a výměnu dat týkajících se finančních transakcí, které zahájili držitelé karet prostřednictvím zařízení, jako jsou bankomaty, terminály pro platby v maloobchodě a online platební brány. Standard specifikuje formát a obsah zpráv vyměňovaných během autorizace, vyúčtování a urovnávání transakcí. [10]

Rozsáhlé využívání protokolu a dodržování průmyslových standardů zajišťuje kompatibilitu mezi různými platebními systémy a zúčastněnými stranami. Poskytuje společný jazyk pro komunikaci dat transakcí, což umožňuje bezproblémovou integraci v různých platebních prostředích. [26]

3.3 BICISO

Protokol BICISO rozšiřuje protokol ISO8583, přičemž přebírá jeho základní strukturu a funkčnost. BICISO zachovává hlavní charakteristiky ISO8583, včetně formátu zpráv, organizace datových prvků a schopnosti zpracovávat transakce. [26]

„Proces BIC ISO interface je zodpovědný za překlad příchozích a odchozích externích zpráv BIC ISO do a z interních formátů zpráv BASE24-atm a BASE24-pos. Procesy BIC ISO interface vytvářejí a interpretují externí zprávy podle specifikací uvedených v manuálu. Externí zpráva BIC ISO umožňuje individuální konfiguraci příchozích a odchozích zpráv pro každou co-network v závislosti na informacích, které co-network zvolí poslat a přijmout.“ [27, s. 22]



Obrázek 6 Ilustrace převodu formátu mezi Internal a External [28]

„Externí zpráva BIC ISO se od standardů ISO8583 liší tím, že nepoužívá binární datové pole. Několik prvků v externí zprávě ISO bylo standardizováno jako binární pole; externí zpráva BIC ISO je však odesílána zcela ve formátu zobrazení.“ [27, s. 33]

Důvodem je, že systém BASE24 není schopen provádět transparentní komunikaci se svými co-sítěmi a použití binárních dat by mohlo způsobit nechtěné vložení řídicích znaků do datového přenosového proudu. Navíc většina co-sítí očekává svá data v kódování EBCDIC, zatímco nativní kód Tandemu je ASCII. Použitím zpráv ve formátu zobrazení lze provést překlad z ASCII na EBCDIC a zpět pomocí řadiče komunikace. Pokud by některá datová pole ve zprávě byla binární, překlad by musel provádět proces rozhraní BIC ISO, což by vyžadovalo využití prostředků CPU namísto prostředků řadiče. [21]

Produkt BASE24 nabízí možnost, která umožňuje určitá proměnlivá pole změnit na pevnou délku. V rámci této možnosti jsou proměnlivé prvky jednoduše považovány za pevnou délku procesem rozhraní BIC ISO, přičemž velikost prvku je nastavena na maximální délku stanovenou pro proměnlivá data. [27] Tato volba pevné délky nemění způsob identifikace prvků. Ty stále zůstávají definovány jako prvky s proměnlivou délkou a stále vyžadují předpony pro označení jejich délky. Nicméně data v prvku jsou vždy zarovnána na levou stranu a doplněna na maximální povolenou délku datového prvku. [27]

Proces rozhraní BIC ISO má schopnost přenášet a přijímat tokeny dat v externí zprávě. Při tvorbě externí zprávy musí proces rozhraní BIC ISO určit, které tokeny v interní zprávě jsou zasílány do sítě a v jakém pořadí jsou tyto tokeny umístěny v externí zprávě. [27] Pro zaslání tokenů postupuje proces rozhraní BIC ISO následovně:

1. Hledá záznam tokenu odpovídající typu odesílané zprávy. Pokud není žádný záznam tokenu nalezen, žádné tokeny nejsou zaslány v externí zprávě.
2. Seřadí tokeny v interní zprávě podle pořadí stanoveného v záznamu tokenu. Seřazené tokeny jsou uloženy do dočasného bufferu (bez změny pořadí).
3. Převede každý token z binárního formátu na formát ASCII.
4. Přidá každý token, který je konfigurován k odeslání, do datového prvku. Při přidání prvního tokenu je vytvořen hlavičkový token a je přidán do datového prvku, okamžitě za indikátorem délky pole. Tento hlavičkový token je aktualizován novým počtem tokenů a celkovou délkou dat tokenů, jakmile jsou přidávány následující tokeny do datového prvku. Každý token přidaný do zprávy má vlastní hlavičku tokenu, která identifikuje individuální token a určuje jeho délku.
5. Po přidání všech tokenů konfigurovaných k odeslání do datového prvku zprávy, proces rozhraní BIC ISO aktualizuje indikátor délky pole součtem délky hlavičkového tokenu a délky každého tokenu zprávy (včetně délky každé hlavičky tokenu).

4 ISO8583

ISO 8583 se skládá z:

1. Struktura zpráv: Zprávy ISO 8583 jsou strukturovány do řady polí, přičemž každé pole obsahuje konkrétní informace o transakci, jako je typ transakce, údaje o držiteli karty, částka transakce a další podrobnosti potřebné pro zpracování. [10]
2. Definice polí: Každé pole v rámci zprávy ISO 8583 je definováno svou pozicí, délkou, typem dat a účelem. Například Pole 3 obvykle představuje zpracovací kód, který označuje typ transakce (např. prodej, refundace, autorizace). [10]
3. Formát zprávy: Zprávy ISO 8583 mohou následovat buď pevnou nebo proměnnou délkou, v závislosti na konkrétních požadavcích prostředí pro zpracování transakcí. Formáty pevné délky přidělují předem určenou délku pro každé pole, zatímco formáty proměnné délky umožňují flexibilitu v délkách polí. [10]

Zprávy ISO 8583 jsou obvykle přenášeny přes komunikační sítě pomocí standardních protokolů, jako je TCP/IP. To umožňuje bezpečnou a spolehlivou komunikaci mezi různými systémy zapojenými do zpracování finančních transakcí. [10]

ISO 8583 definuje posloupnost kroků zapojených do zpracování transakce, včetně autorizace, vyúčtování a urovnávání. Každá fáze zahrnuje výměnu zpráv ISO 8583 mezi různými stranami, jako jsou držitel karty (card holder), obchodník (merchant), přijímající banka (acquirer) a vydavatelská banka(issuer) (viz. Obrázek 1). [13]

4.1 Podrobnější popis

4.1.1 Struktura Zprávy

Struktura zprávy ISO 8583 zahrnuje následující hlavní části:

a. MTID (Message Type Identifier) - Identifikátor typu zprávy
b. BITMAP – Bitová mapa, která označuje přítomnost nebo nepřítomnost datových polí v zprávě
c. DATA FIELDS – Datová pole, která obsahují konkrétní informace o transakci

Bitmapa funguje jako indikátor, který určuje, která datová pole jsou obsažena v konkrétní zprávě. Data Fields (datová pole) pak obsahují konkrétní informace o transakci, jako jsou např. číslo karty, částka transakce, typ transakce atd.

Obrázek 7 Struktura zprávy [29]

4.1.2 MTID

MTID je čtyřmístné numerické pole, které označuje verzi protokolu, třídu zprávy, funkci zprávy a iniciátora komunikace. Všechny zprávy vyžadují MTID. [10]

4.1.2.1 První pozice MTID

První číslice MTID označuje verzi ISO 8583, ve které je zpráva zakódována.

Code	Význam
0xxx	ISO 8583:1987
1xxx	ISO 8583:1993
2xxx	ISO 8583:2003
3xxx	Reserved by ISO
4xxx	
5xxx	
6xxx	
7xxx	
8xxx	Národní využití
9xxx	Privátní využití

Tab. 1. MTID – 1. pozice [29]

4.1.2.2 Druhá pozice MTID

Druhá číslice MTID určuje celkový účel zprávy.

Code	Význam	Využití
x**0**xx	Reserved by ISO	
x**1**xx	Autorizační zpráva	Určuje, zda jsou dostupné finanční prostředky, získává schválení, ale nepřidává je na účet pro rekonciliaci.
x**2**xx	Finanční zpráva	Určuje, zda jsou dostupné finanční prostředky, získává schválení a přidává je přímo na účet.
x**3**xx	Zpráva o akcích se soubory	Používáno pro blokovanou kartu, TMS a další výměny.
x**4**xx	Zprávy o stornu a vracení transakcí	Reversal – Vrací akci předchozí autorizace. Refund – Vrací již vyúčtovanou finanční zprávu.
x**5**xx	Zpráva o rekonciliaci	Přenáší informace o settlementu.
x**6**xx	Administrativní zpráva	Přenáší administrativní radu. Často používána pro chybové zprávy
x**7**xx	Zprávy o vybírání poplatků	
x**8**xx	Zpráva o správě sítě	Používáno pro bezpečnou výměnu klíčů, přihlášení, testování odezvy a další síťové funkce.
x**9**xx	Reserved by ISO	

Tab. 2. MTID – 2. pozice [29]

4.1.2.3 Třetí pozice MTID

Třetí číslice MTID určuje funkci zprávy, která definuje, jak by měla zpráva proudit v systému. Např. od acquirera k issuerovi a zpět s nastavenými časovými limity a automatickými storny.

Code	Význam	Využití
xx0x	REQ	REQ od acquirer pro issuer k provedení akce; issuer může přijmout nebo odmítnout.
xx1x	RESP na REQ	RESP na REQ
xx2x	Advice (REQ)	Rada, že k akci došlo; příjemce může pouze přijmout, ne odmítnout.
xx3x	Advice RESP	RESP na advice REQ
xx4x	Notifikace	Oznámení, že událost nastala; příjemce může pouze přijmout, ne odmítnout.
xx5x	Obeznámení o obdržení Notifikace	RESP na notifikaci
xx6x	Instrukce	
xx7x	Obeznámení o obdržení Instrukcí	
xx8x	Reserved for ISO use	Některé implementace (například MC) používají pro pozitivní potvrzení.
xx9x		Některé implementace (například MC) používají pro negativní potvrzení.

Tab. 3. MTID – 3. pozice [29]

4.1.2.4 Čtvrtá pozice MTID

Čtvrtá číslice MTID definuje umístění zdroje zprávy v platebním řetězci.

Code	Význam
xxx0	Acquirer
xxx1	Reapeat acquirer
xxx2	Issuer
xxx3	Repeat issuer
xxx4	Jiné
xxx5	Opakování Jiné
xxx6	Reserved by ISO

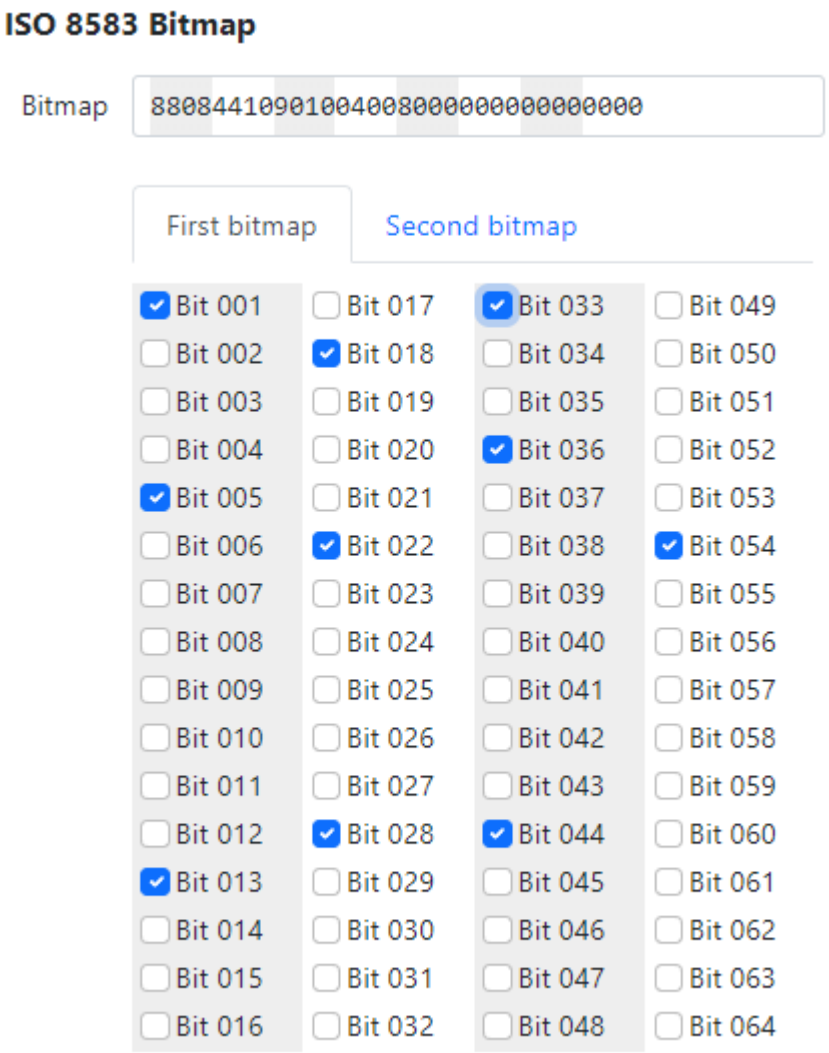
Tab. 4. MTID – 4. pozice [29]

4.1.3 BITMAP

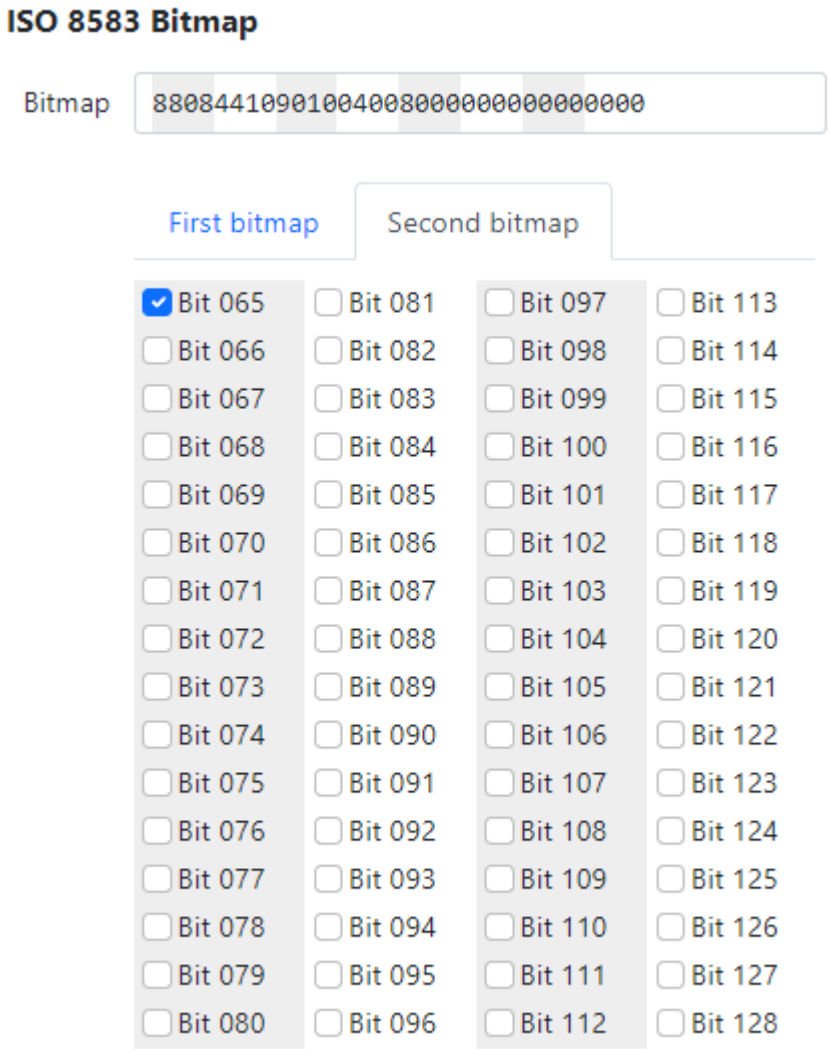
„Každá bitová mapa se skládá z 64 bitů, přičemž každá pozice bitu označuje přítomnost nebo nepřítomnost datového pole ve zprávě. Hodnota bitu 1 indikuje přítomnost pole, zatímco hodnota 0 označuje jeho nepřítomnost.“ [10, s. 44]

Číselná hodnota každé pozice bitu v bitové mapě určuje identifikátory polí, které jsou přiřazeny na základě počítání od první pozice bitu, začínající zleva. Například pozice bitu 15 odpovídá Poli 15. Pokud má první pozice bitu hodnotu 1, znamená to přítomnost následující souvislé bitové mapy, která je nazývána sekundární bitová mapa (viz. Obrázek 8). Tato sekundární bitmapa reprezentuje pole 65 až 128 (viz. Obrázek 9). [10]

„Pozice 65 ve druhé bitové mapě indikuje přítomnost třetí bitové mapy.“ [10, s. 45]



Obrázek 8 Ukázka práce s 1. bitmapou [30]



Obrázek 9 Ukázka práce se sekundární bitmapou [30]

Bitová mapa může být reprezentována jako 8 bytů binárních dat nebo jako 16 hexadecimálních znaků (0–9, A–F) v ASCII nebo EBCDIC znakových sadách.

Příklad z obrázků:

Při bitmapové hodnotě 88 08 44 10 90 10 04 00 80:

0x88 = 1000 1000 (počítáno zleva, první a pátý bit jsou 1, což značí, že pole 1 a 5 jsou přítomna)

0x08 = 0000 1000 (první bit odpovídá poli 9, takže pátý bit zde indikuje, že pole 13 je přítomno)

0x44 = 0100 0100 (pole 18 a 22 jsou přítomna)

0x10 = 0001 0000 (pole 28 je přítomno)

0x90 = 1001 0000 (pole 33 a 36 jsou přítomna)

0x10 = 0001 0000 (pole 44 je přítomno)

0x04 = 0000 0100 (pole 54 je přítomno)

0x00 = 0000 0000 (žádné pole není přítomno)

0x80 = 1000 0000 (pole 65 je přítomno)

n bit	0	10	20	30
	1234567890	1234567890	1234567890	1234567890
Bitmap	1000100000	0010000100	0100000100	0010010000

Tab. 5. Ukázka převodu bitmapy [30]

4.1.4 DATA FIELDS

Datové pole jsou jednotlivá pole nesoucí informace o transakci. V původní normě ISO 8583:1987 jsou specifikovány až 128 datových elementů a v pozdějších verzích až 192 datových elementů.

Každý datový prvek má specifikovaný význam a formát, norma také zahrnuje obecné datové elementy a datové elementy specifické pro systém nebo zemi, které se velmi liší v použití a formě od implementace k implementaci.

Každý datový prvek je popsán ve standardním formátu, který definuje povolený obsah pole (číselný, binární atd.) a délku pole (proměnná nebo pevná) podle následující tabulky:

Zkratka	Význam
a	Písmena, včetně mezer
n	Pouze číselné hodnoty
x+n	Číselné hodnoty, kde první byte je buď 'C' pro označení kladné nebo kreditní hodnoty, nebo 'D' pro označení záporné nebo debetní hodnoty, následované číselnou hodnotou (používá n číslic)
s	Speciální charaktery
an	Alfanumerické znaky
as	Písmena a speciální znaky

ns	Čísla a speciální znaky
ans	Alfanumerické znaky se speciálními
anp	Alfanumerické znaky s vyplňovacími znaky (padding)
b	Binarní data
p	Vyplňovací znaky (padding), mezera
z	Nastavení stopy 2 a 3, jak je definováno v ISO/IEC 7813 a ISO/IEC 4909
. or .. or ...	Indikátor proměnné délky pole, každé . indikuje jednu číslici.
M	Mandatory Data (povinná)
C	Conditional Data (podmíněná)
O	Optional Data (volitelná)

Tab. 6. Tabulka pojmů a kódovacích zkr. [10]

Kromě toho může být každé pole buď pevné nebo proměnné délky. Pokud je proměnné, délka pole bude předcházena indikátorem délky.

Typ	Význam
Fixed	Žádná délka pole není použita
LVAR	Kde 0 < L < 10, první číslo L určují délku pole VAR
LLVAR	Kde 0 < LL < 100, první dvě čísla LL určují délku pole VAR
LLLVAR	Kde 0 < LLL < 1000, první tři čísla LLL určují délku pole VAR
LL a LLL jsou hex nebo ASCII. Pole VAR může být komprimováno nebo ASCII v závislosti na typu datového prvku.	LL může být jeden nebo dva bajty. Například, pokud je komprimováno jako jeden hexadecimální byte, '27x znamená, že následuje 27 bajtů VAR. Pokud je to ASCII, dva byty '32x, '37x znamenají, že následuje 27 bajtů. Tři číslice délky pole LLL používají dva bajty s představou '0' poloviny, pokud je komprimováno, nebo tři bajty, pokud je ASCII. Formát VAR datového prvku závisí na typu datového prvku. Pokud je číselný, bude komprimován, např. 87456 bude reprezentován třemi hexadecimálními byty '087456x. Pokud je to ASCII, pak je použit jeden byte pro každou číslici nebo znak, např. '38x, '37x, '34x, '35x, '36x.

Tab. 7. Tabulka indikátorů délky pole [10]

Příklad:

Definice polí	Význam
n 6	Fixní délka pole o 6 číslech
n.6	LVAR číselné pole až do 6-ti čísel
a..11	LLVAR alfabetické pole do 11 znaků
b...999	LLLVAR binární pole do délky 999

Tab. 8. Příklad práce se zkr. [31]

4.1.4.1 ISO-8583 DATA FIELDS (ver 1987):

Z důvodu lepší čitelnosti jsem tabulku nepřekládal do češtiny (s většinou pojmů se pracuje v Anglickém jazyce)

Data field	Typ	Využití
1	b 16	Bitmap
2	n..19	Primary account number (PAN)
3	n 6	Processing Code
4	n 12	Amount Transaction
5	n 12	Amount, settlement
6	n 12	Amount, cardholder billing
7	n 10	Transmission date & time
8	n 8	Amount, cardholder billing fee
9	n 8	Conversion rate, settlement
10	n 8	Conversion rate, cardholder billing
11	n 6	System trace audit number (STAN)
12	n 6	Local transaction time (hhmmss)
13	n 4	Local transaction date (MMDD)
14	n 4	Expiration date (YYMM)
15	n 4	Settlement date
16	n 4	Currency conversion date
17	n 4	Capture date
18	n 4	Merchant type, or merchant category code
19	n 3	Acquiring institution (country code)
20	n 3	PAN extended (country code)
21	n 3	Forwarding institution (country code)

22	n 3	Point of service entry mode
23	n 3	Application PAN sequence number
24	n 3	Function code (ISO 8583:1993), or network international identifier (NII)
25	n 2	Point of service condition code
26	n 2	Point of service capture code
27	n 1	Authorizing identification response length
28	x+n 8	Amount, transaction fee
29	x+n 8	Amount, settlement fee
30	x+n 8	Amount, transaction processing fee
31	x+n 8	Amount, settlement processing fee
32	n ..11	Acquiring institution identification code
33	n ..11	Forwarding institution identification code
34	ns ..28	Primary account number, extended
35	z ..37	Track 2 data
36	n ...104	Track 3 data
37	an 12	Retrieval reference number
38	an 6	Authorization identification response
39	an 2	Response code
40	an 3	Service restriction code
41	ans 8	Card acceptor terminal identification
42	ans 15	Card acceptor identification code
43	ans 40	Card acceptor name/location (1–25 card acceptor name or automated teller machine (ATM) location, 26-28 city name, 39-40 country code)
44	an ..25	Additional response data
45	an ..76	Track 1 data

46	an ...999	Additional data (ISO)
47	an ...999	Additional data (national)
48	an ...999	Additional data (private)
49	a or n 3	Currency code, transaction
50	a or n 3	Currency code, settlement
51	a or n 3	Currency code, cardholder billing
52	b 64	Personal identification number data
53	n 16	Security related control information
54	an ...120	Additional amounts
55	ans ...999	ICC data – EMV having multiple tags
56	ans ...999	Reserved (ISO)
57	ans ...999	Reserved (national)
58	ans ...999	
59	ans ...999	
60	ans ...999	Reserved (national) (e.g. settlement request: batch number, advice transactions: original transaction amount, batch upload: original MTI plus original RRN plus original STAN, etc.)
61	ans ...999	Reserved (private) (e.g. CVV2/service code transactions)
62	ans ...999	Reserved (private) (e.g. transactions: invoice number, key exchange transactions: TPK key, etc.)
63	ans ...999	Reserved (private)
64	b 64	Message authentication code (MAC)
65	b 1	Extended bitmap indicator
66	n 1	Settlement code
67	n 2	Extended payment code
68	n 3	Receiving institution country code
69	n 3	Settlement institution country code

70	n 3	Network management information code
71	n 4	Message number
72	n 4	Last message's number
73	n 6	Action date (YYMMDD)
74	n 10	Number of credits
75	n 10	Credits, reversal number
76	n 10	Number of debits
77	n 10	Debits, reversal number
78	n 10	Transfer number
79	n 10	Transfer, reversal number
80	n 10	Number of inquiries
81	n 10	Number of authorizations
82	n 12	Credits, processing fee amount
83	n 12	Credits, transaction fee amount
84	n 12	Debits, processing fee amount
85	n 12	Debits, transaction fee amount
86	n 16	Total amount of credits
87	n 16	Credits, reversal amount
88	n 16	Total amount of debits
89	n 16	Debits, reversal amount
90	n 42	Original data elements
91	an 1	File update code
92	an 2	File security code
93	an 5	Response indicator
94	an 7	Service indicator
95	an 42	Replacement amounts

96	b 64	Message security code
97	x+n 16	Net settlement amount
98	ans 25	Payee
99	n ..11	Settlement institution identification code
100	n ..11	Receiving institution identification code
101	ans ..17	File name
102	ans ..28	Account identification 1
103	ans ..28	Account identification 2
104	ans ...100	Transaction description
105	ans ...999	Reserved for ISO use
106	ans ...999	
107	ans ...999	
108	ans ...999	
109	ans ...999	
110	ans ...999	
111	ans ...999	
112	ans ...999	Reserved for national use
113	ans ...999	
114	ans ...999	
115	ans ...999	
116	ans ...999	
117	ans ...999	
118	ans ...999	
119	ans ...999	
120	ans ...999	Reserved for private use
121	ans ...999	

122	ans ...999	
123	ans ...999	
124	ans ...999	
125	ans ...999	
126	ans ...999	
127	ans ...999	
128	b 64	Message authentication code

Tab. 9. Přehled datových polí v ISO8583 a jejich využití [10]

Typy TRN

Transakce v rámci ISO 8583 zahrnují širokou škálu finančních aktivit, z nichž každá slouží konkrétním účelům v elektronickém platebním ekosystému. Tyto transakce lze široce rozdělit do dvou hlavních typů: PCI (finanční) a NON-PCI (nefinanční) transakce. [32]

PCI transakce zahrnují převody finančních prostředků mezi účty a jsou obvykle spojeny s peněžní hodnotou. Příklady finančních transakcí zahrnují dotazy na zůstatek, výběry hotovosti, vklady, převody finančních prostředků a platby za zboží nebo služby.

Na druhou stranu nefinanční transakce nezahrnují přímý převod finančních prostředků, ale jsou důležité pro správu informací o účtu a zajištění bezpečnosti transakce. Tyto transakce často podporují administrativní funkce, jako jsou ověřování držitele karty, změny PIN kódu, dotazy na stav účtu a aktivace karty.

Porozumění různým typům transakcí v rámci protokolu ISO 8583 je základní pro vývoj robustních a spolehlivých finančních systémů.

Typ transakce se definuje právě v již zmíněném poli MTID, následovně:

	MTID	Význam
Autorizační	1100	REQ
	1110	RESP
	1120	Advice REQ
	1130	Advice RESP

Finanční	1200	REQ
	1210	RESP
	1220	Advice REQ
	1230	Advice RESP
Storno	1420	REQ
	1430	RESP
Rekonciliace	1500	REQ
	1510	RESP
Administrativní	1800	REQ
	1810	RESP

Tab. 10. Typy TRN a kdy se použijí s jakým MTID [10]

Advice požadavek je posílán v případě, když držitel karty spustí transakci, jako je nákup pomocí kreditní nebo debetní karty. Jsou podrobnosti o transakci odeslány vydavateli karty ke schválení. Pokud vydavatel transakci schválí, poskytne autorizační kód, který umožní pokračování transakce.

Avšak v některých případech, z důvodu různých faktorů, jako jsou síťové problémy nebo zpoždění, může být odpověď na autorizaci obchodníkovi doručena se zpožděním. V těchto situacích může být odeslána zpráva "Authorization Advice Request" obchodníkem nebo akviziční bankou vydavatele karty s cílem požádat o radu nebo pokyny ohledně dalšího postupu s transakcí.

Tato zpráva obvykle obsahuje podrobnosti o původní transakci, jako je výše transakce, informace o obchodníkovi a držiteli karty, spolu s žádostí o radu k autorizaci. Vydavatel karty pak odpoví s radou ohledně toho, zda schválit, zamítnout nebo přijmout jiná opatření v souvislosti s transakcí. [10]

Každý typ transakce má konkrétní účel v rámci platebního zpracování, přispívá k plynulé a bezpečné výměně finančních informací mezi stranami. Porozumění tomu, kdy a jak používat jaký typ transakce, je klíčové pro jejich využití v simulátorech. Zde je jejich souhrn:

4.1.5 Authorizace:

- Používá se k žádosti o schválení od vydavatele karty pro transakci.
- Spouští se, když držitel karty usiluje o provedení nákupu nebo finanční transakce.
- Ověřuje, zda má držitel karty dostatečné finanční prostředky nebo kredit pro transakci.
- Pokud je schváleno, generuje autorizační kód, který umožňuje pokračování transakce.

4.1.6 Finanční TRN:

- Používá se k zahájení finanční transakce, jako jsou výběry hotovosti, dotazy na zůstatek nebo převody finančních prostředků.
- Představuje skutečné pohyby finančních prostředků mezi účty.
- Vyžaduje autorizaci před dokončením transakce.

4.1.7 Reversal (Storno):

- Používá se k zrušení nebo zpětnému zrušení dříve autorizované transakce.
- Spouští se v případech, kdy je zapotřebí zrušit autorizovanou transakci, například kvůli chybě, podezření z podvodu nebo požadavku zákazníka.
- Pomáhá udržovat přesné účetní záznamy a zabránit dvojímu účtování.

4.1.8 Rekonciliace:

- Používá se k srovnání (rekonciliaci) transakcí mezi různými systémy nebo subjekty.
- Pomáhá zajistit, aby záznamy o transakcích byly konzistentní a přesné u všech zúčastněných stran.
- Obvykle zahrnuje porovnávání dat transakcí, identifikaci nesrovnalostí a řešení případných odlišností.

4.1.9 Administrativní:

- Používá se k testování komunikace a propojení mezi systémy nebo zařízeními.
- Obvykle zahrnuje odesílání předdefinované zprávy nebo žádosti a přijetí odpovědi k potvrzení správného fungování komunikace.
- Pomáhá diagnostikovat a řešit možné problémy s přenosem nebo zpracováním zpráv.

4.1.10 Přehled polí a zpráv použitých v protokolu

Autorirace: MTID = 1100, 1110

DATA FIELD		Format	Message types			
			1100	1110	1120	1130
P-1	Secondary bitmap	AN16	M	M	M	M
P-2	PAN	AN.. 19	M	C	M	C
P-3	Processing code	AN6	M	C	M	C
P-4	Transaction amount	N12	M	C	M	C
P-7	Transmission date and time	N12	M	C	M	C
P-11	Systems trace audit number	N6	M	M	M	M
P-12	Local transaction time	N12	M	C	M	C
P-14	Card expiration date	N4	M	C	M	C
P-15	Settlement date	N6	C	C	C	C
P-17	Capture date	N4	C	C	C	C
P-22	Point of service entry mode	AN12	M	C	M	C
P-23	Card sequence number	N3	C	C	C	C
P-26	Card acceptor business code	N4	M	C	M	C
P-32	Acquiring institution ID code	N.. 11	M	C	M	C
P-35	Track 2 data	ANS.. 37	C	C	C	C
P-37	Retrieval reference number	ANS12	M	M	M	M
P-38	Approval code	AN.. 8	O	M	O	M
P-39	Response code	N3	O	M	O	M
P-41	Card acceptor terminal ID	ANS8	C	C	C	C
P-42	Card acceptor ID code	ANS15	C	C	C	C
P-43	Card acceptor name/location	ANS.. 99	C	C	C	C

P-45	Track 1 data	ANS.. 76	C	C	C	C
P-47	Industry	ANS..... 99999	C	C	C	C
P-48	Special processing	ANS..... 99999	C	C	C	C
P-49	Transaction currency code	N3	M	C	M	C
P-52	PIN Data (block)	B16	C	C	C	C
P-53	Security related control information	AN16	C	C	C	C
P-54	Additional amounts	AN.. 20	C	C	C	C
P-55	Smart card data	ANS... 999	C	C	C	C
P-64	Primary MAC	AN16	C	C	C	C
S-70	Network management information code	N3	C	C	C	C
S-90	Original data elements	ANS.. 999	C	C	C	C
S-128	Secondary MAC	AN	C	C	C	C

Finanční TRN: MTID = 1200,1210

DATA FIELD		Format	Message types			
			1200	1210	1220	1230
P-1	Secondary bitmap	AN16	M	M	M	M
P-2	PAN	AN.. 19	M	C	M	C
P-3	Processing code	AN6	M	C	M	C
P-4	Transaction amount	N12	M	C	M	C
P-7	Transmission date and time	N12	M	C	M	C
P-11	Systems trace audit number	N6	M	C	M	C
P-12	Local transaction time	N12	M	C	M	C

P-14	Card expiration date	N4	M	C	M	C
P-15	Settlement date	N6	C	C	C	C
P-17	Capture date	N4	C	C	C	C
P-22	Point of service entry mode	AN12	M	C	M	C
P-23	Card sequence number	N3	C	C	C	C
P-26	Card acceptor business code	N4	M	C	M	C
P-32	Acquiring institution ID code	N.. 11	M	C	M	C
P-35	Track 2 data	ANS.. 37	C	C	C	C
P-37	Retrieval reference number	ANS12	M	M	M	M
P-38	Approval code	AN.. 8	O	C	C	C
P-39	Response code	N3	O	M	O	M
P-41	Card acceptor terminal ID	ANS8	C	C	C	C
P-42	Card acceptor ID code	ANS15	C	C	C	C
P-43	Card acceptor name/location	ANS.. 99	C	C	C	C
P-45	Track 1 data	ANS.. 76	C	C	C	C
P-47	Industry	ANS..... 99999	C	C	C	C
P-48	Special processing	ANS..... 99999	C	C	C	C
P-49	Transaction currency code	N3	M	C	M	C
P-52	PIN Data (block)	B16	C	C	C	C
P-53	Security related control information	AN16	C	C	C	C
P-54	Additional amounts	AN.. 20	C	C	C	C
P-55	Smart card data	ANS... 999	C	C	C	C
P-64	Primary MAC	AN16	C	C	C	C

S-70	Network management information code	N3	C	C	C	C
S-90	Original data elements	ANS.. 999	C	C	C	C
S-128	Secondary MAC	AN	C	C	C	C

Reversal (storno): MTID = 1420,1430

DATA FIELD		Format	Message types	
			1420	1430
P-1	Secondary bitmap	AN16	M	M
P-2	PAN	AN.. 19	M	C
P-3	Processing code	AN6	M	C
P-4	Transaction amount	N12	M	C
P-7	Transmission date and time	N12	M	C
P-11	Systems trace audit number	N6	M	C
P-12	Local transaction time	N12	M	C
P-14	Card expiration date	N4	C	C
P-15	Settlement date	N6	C	C
P-17	Capture date	N4	C	C
P-22	Point of service entry mode	AN12	M	C
P-23	Card sequence number	N3	C	C
P-26	Card acceptor business code	N4	M	C
P-32	Acquiring institution ID code	N.. 11	M	C
P-35	Track 2 data	ANS.. 37	C	C
P-37	Retrieval reference number	ANS12	M	M
P-38	Approval code	AN.. 8	C	C

P-39	Response code	N3	C	M
P-41	Card acceptor terminal ID	ANS8	M	M
P-42	Card acceptor ID code	ANS15	C	C
P-43	Card acceptor name/location	ANS.. 99	C	C
P-45	Track 1 data	ANS.. 76	C	C
P-47	Industry	ANS..... 99999	C	C
P-48	Special processing	ANS..... 99999	C	C
P-49	Transaction currency code	N3	M	M
P-52	PIN Data (block)	B16	C	C
P-53	Security related control information	AN16	C	C
P-54	Additional amounts	AN.. 20	C	C
P-55	Smart card data	ANS... 999	C	C
P-64	Primary MAC	AN16	C	C
S-70	Network management information code	N3	C	C
S-90	Original data elements	ANS.. 999	C	C
S-128	Secondary MAC	AN	C	C

Administrativní: MTID = 1800,1810

DATA FIELD		Format	Message types	
			1800	1810
P-1	Secondary bitmap	AN16	M	M
P-2	PAN	AN.. 19	C	C
P-3	Processing code	AN6	C	C
P-4	Transaction amount	N12	C	C

P-7	Transmission date and time	N12	M	M
P-11	Systems trace audit number	N6	M	M
P-12	Local transaction time	N12	C	C
P-14	Card expiration date	N4	C	C
P-15	Settlement date	N6	C	C
P-17	Capture date	N4	C	C
P-22	Point of service entry mode	AN12	C	C
P-23	Card sequence number	N3	C	C
P-26	Card acceptor business code	N4	C	C
P-32	Acquiring institution ID code	N.. 11	C	C
P-35	Track 2 data	ANS.. 37	C	C
P-37	Retrieval reference number	ANS12	C	C
P-38	Approval code	AN.. 8	C	C
P-39	Response code	N3	C	M
P-41	Card acceptor terminal ID	ANS8	C	C
P-42	Card acceptor ID code	ANS15	C	C
P-43	Card acceptor name/location	ANS.. 99	C	C
P-45	Track 1 data	ANS.. 76	C	C
P-47	Industry	ANS..... 99999	C	C
P-48	Special processing	ANS..... 99999	C	C
P-49	Transaction currency code	N3	C	C
P-52	PIN Data (block)	B16	C	C
P-53	Security related control information	AN16	C	C
P-54	Additional amounts	AN.. 20	C	C

P-55	Smart card data	ANS... 999	C	C
P-64	Primary MAC	AN16	C	C
S-70	Network management information code	N3	C	C
S-90	Original data elements	ANS.. 999	C	C
S-128	Secondary MAC	AN	C	C

Tab. 11. Tabulka s přehledem mandatorních polí u všech MTID

4.2 Rozdílnost od jiných protokolů (BICISO)

Mezi množstvím protokolů využívaných v tomto odvětví patří BICISO a ISO 8583 mezi ty významnější. Jedním z hlavních rozdílů mezi těmito dvěma protokoly je způsob kódování dat. BICISO obvykle posílá data ve formě tokenů, což umožňuje efektivní a bezpečný přenos dat při bankovních operacích. Naopak ISO 8583 se častěji používá pro přenos dat v ASCII nebo EBCDIC formátu, což poskytuje větší flexibilitu, ale vyžaduje dodatečné úsilí při zpracování dat.

Přenos binárních dat: BICISO nepoužívá binární datová pole, na rozdíl od ISO standardu, který obsahuje standardizované binární pole.

Volba délky datových prvků: BASE24 produkt, který je spojen s protokolem BICISO, umožňuje nastavit určité proměnlivé délky polí jako pevné. Toto neovlivňuje identifikaci prvků, ale data jsou vždy zarovnána vlevo a vyplněna na maximální povolenou délku.

Zamítnuté zprávy: BICISO označuje zamítnuté zprávy změnou první pozice typu zprávy na 9 a vložením čísla bitu datového prvku, který způsobil problém, nebo hodnoty indikující selhání související se zabezpečením, do pole stavu záhlaví externí zprávy BICISO. Tyto zprávy jsou poté vráceny do sítě.

Ukázka zprávy BICISO:

ISO	MESSAGE HEADER	MTID	BITMAP 1
ISO	026000070	0200	B238C48128E1841E

BITMAP 2	DATA ELEMENTS
0000000000000108	0000000000000001080000000000000000133031610462..

Tab. 12. Ukázka práce s čistými daty u BICISO p. [27]

Ukázka RAW DATA:

IS00260000700200B238C48128E1841E0000000000000108000000000000000013303161046240003051146240316031654440710004010029920312*****0603-**** ********00000000030551KP0026

TESTEETKB

TEST EET

PRAHA 10

000000002030120000000200000160000000 0+0000000019

388 0000700388! 0400020

CZ027TESTEETKB

000

01010000

Y ! B200158

7EE90000800480000000D2ABD80AFA1F0748000000000133000000000000198000782032031803160 0

7F5FCBF40014A501A000032400000000000000000000000000000000000000000000000000000000000!

B300080

4F0000000000E00808000000000000002200021F00020007A000000679401000000000000000000!

B400020 0710 0001F0002

040000000083029TEST EET

0

Ukázka zprávy ISO8583:

MTID	BITMAP 1	DATA ELEMENTS
1200	169203070000000009	00000000000000117920071511544200284 1..

Tab. 13. Ukázka práce s čistými daty u ISO8583 p. [10]

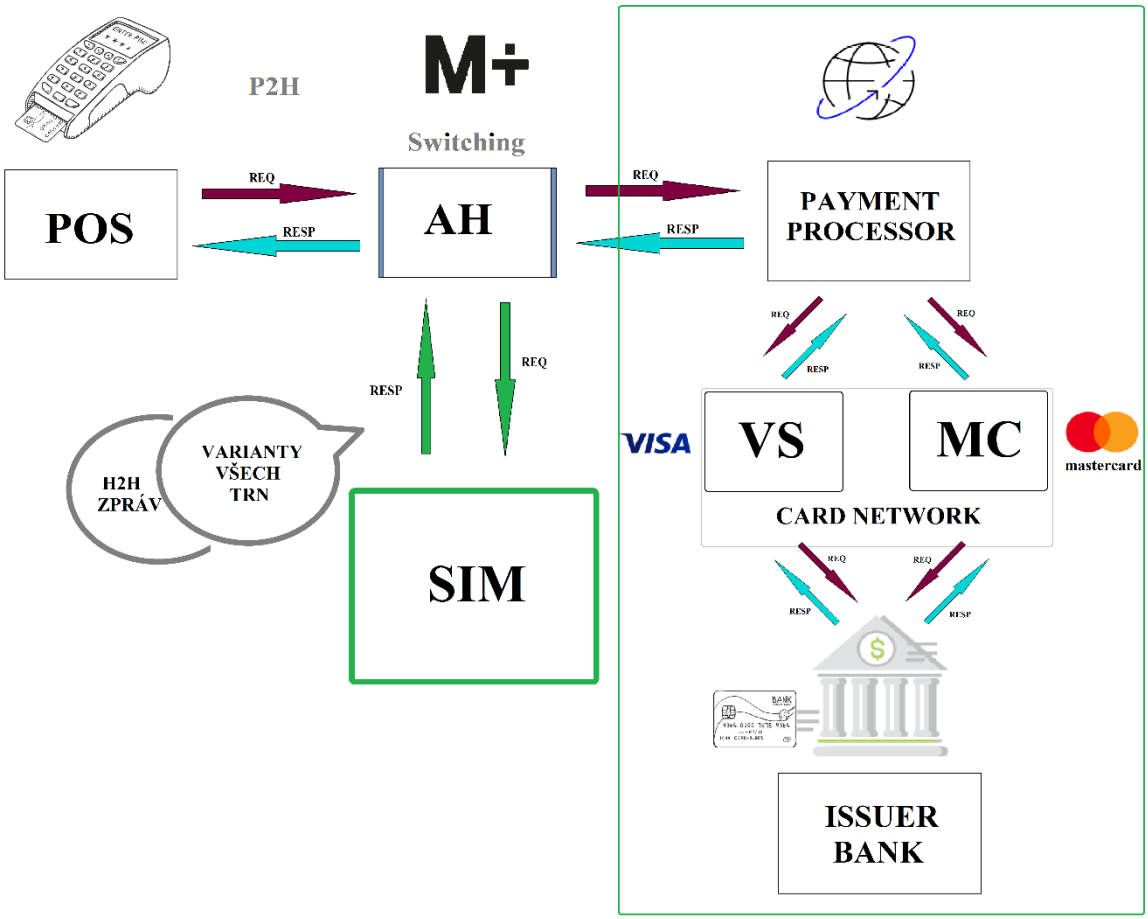
Ukázka RAW DATA:

313230307234064128E092003136353333343737303630303030303039393030303030303
030303030303030313137393230303731353131353434323030323834313230303731353131
313534343232323131303731 4D313031303030344330303035353432303931323334353 6
373839333339323033303730303030303030303030393D323231313230313031373931363 6
343930313937313130303238343154303140 D303030314F553031314D3031202020202020
2039305465726D696E616C205430314D303030312074657374202020202020202020202
020205A612064766F72656D203530352020202020205A6C696E2D537469706120202
020202020202020373633203134202020204 35A20435A3230335EAA7216A12A57C9313
13982189C010050322009F35370416B05D71407A0000000031010950500000000009A03
150510120110A04009F34030000

5 VYUŽITÍ SIMULATORŮ

5.1 Proč simulátor potřebujeme?

Simulátory jsou neocenitelným nástrojem v oddělení zajišťování kvality (QA) pro testování platebních systémů. Ale proč jsou tak důležité? Jedním z hlavních důvodů je jejich schopnost nahrazovat třetí strany, jako jsou platební procesory, karetní sítě a vydavatelské banky, v procesu testování komunikace a dalších aspektů platebního toku (viz. Obrázek 10).



Obrázek 10 Proč jsou pro nás sim výhodné [7; 8; 17; 33; 34; 35]

Zatímco v reálném světě je integrace s těmito třetími stranami nepostradatelná, představte si, že můžeme simulovat jejich chování a odezvu. To nám umožní provádět testy a ověřovat funkčnost našich systémů bez závislosti na reálných třetích stranách, což vede k větší flexibilitě a kontrolním možnostem v procesu testování.

Simulátory nám poskytují prostředí, kde můžeme provádět širokou škálu scénářů, včetně testování chování systému při různých typů transakcí, selhání síťových komponent

nebo neočekávaných událostí. Tím nám umožňují identifikovat potenciální problémy a chyby v systému ještě předtím, než jsou nasazeny do produkčního prostředí.

Dalším klíčovým faktorem je, že simulátory umožňují opakovatelné a konzistentní testování. Můžeme snadno reprodukovat určité scénáře a ověřit, zda se systém chová přesně tak, jak očekáváme. Tato opakovatelnost je zásadní pro důkladné testování a zajištění kvality produktu.

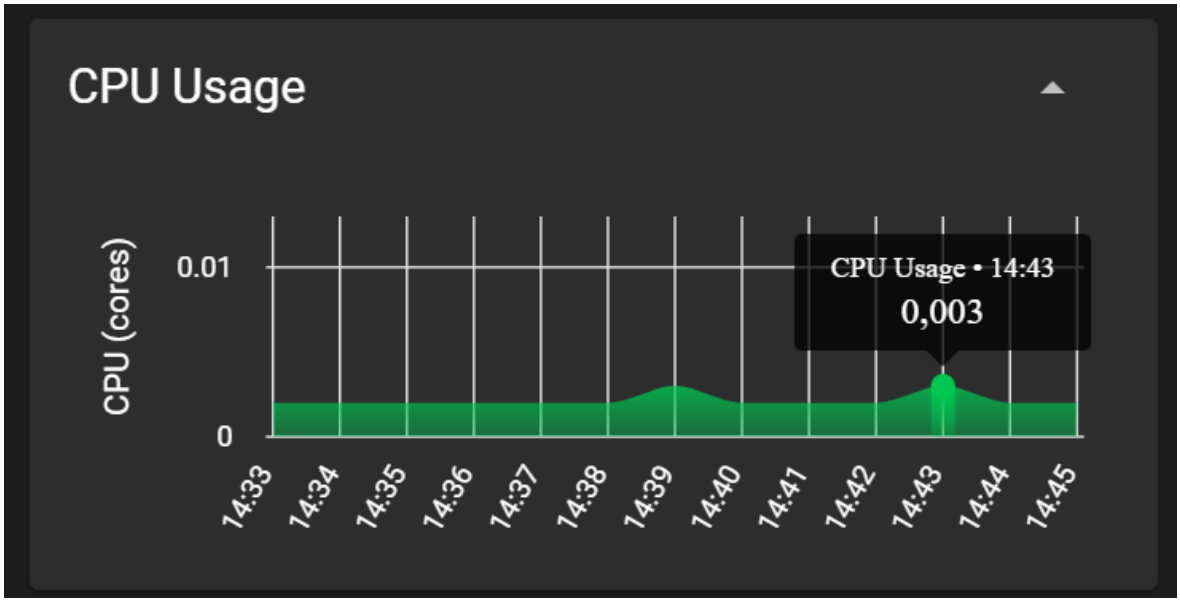
V souhrnu, jsou simulátory nejen nepostradatelným nástrojem pro testování, ale také přinášejí řadu výhod, které přispívají k úspěchu platebních systémů.

5.2 Jak to funguje?

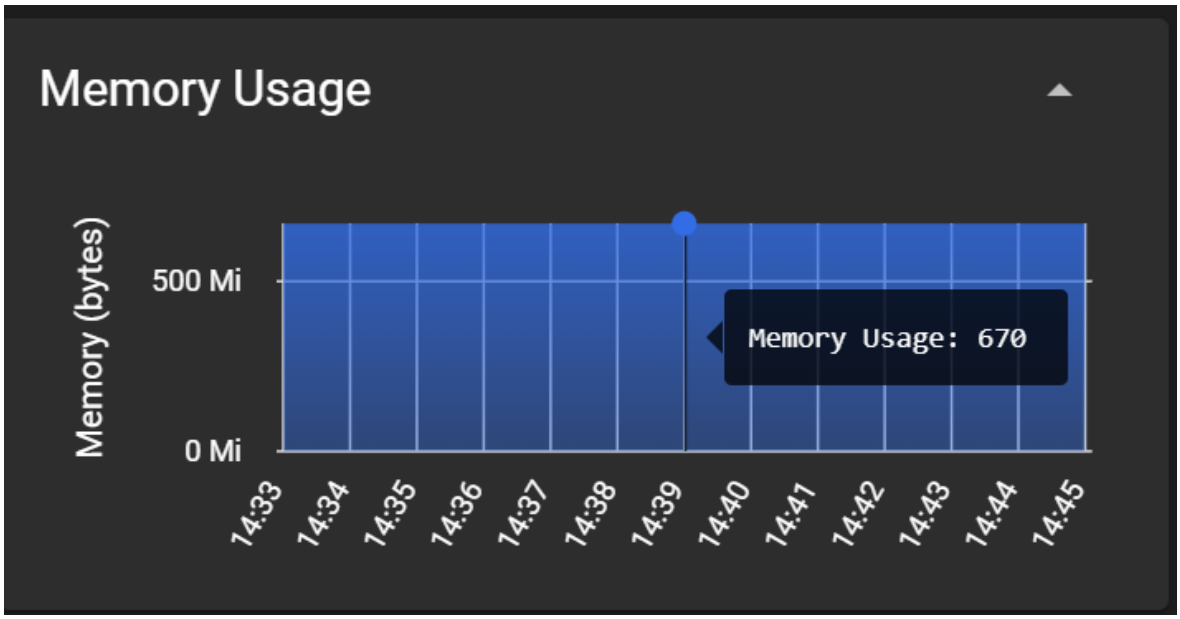
TRN si simulujeme z aplikace Postman nebo posíláme z reálného terminálu, TRN jsou zasílány do AH (Monet+), kde na základě routovacích pravidel (switchingu) dojde k zaslání TRN do simulátoru, který následně odpovídá na REQ poslaný z POS nebo Postmanu. Odpověd na Požadavek se specifikuje v simulátoru, pošle se zpět do AH, který RESP nasměruje zpět na POS.

Simulátory využívají jazyků JavaScript a Python. S přechodem ze starší verze JAVY v aplikaci (Monetu+) na novější bylo třeba změnit zastaralý JS engine Nashorn za Pythonovský Jython z důvodu kompatibility embedování do JAVY programů. Jython podporuje pouze verzi Pythonu 2.7.

Ke spuštění simulátorů dojde při startu nebo restartem aplikace. Celková velikost jednoho simulátoru a všech jejich konfiguračních souborů nepřesahuje 55,0 kB. Z toho největší velikost má JSON definiční soubory (do 30,0 kB) a python definiční soubor obsahující funkce, které využívají všechny simulátory (do 20 kB). Jelikož nezabírají moc místa, jsou tedy snadno udržovatelné a nenáročné na chod (viz. Obrázek 11,12).

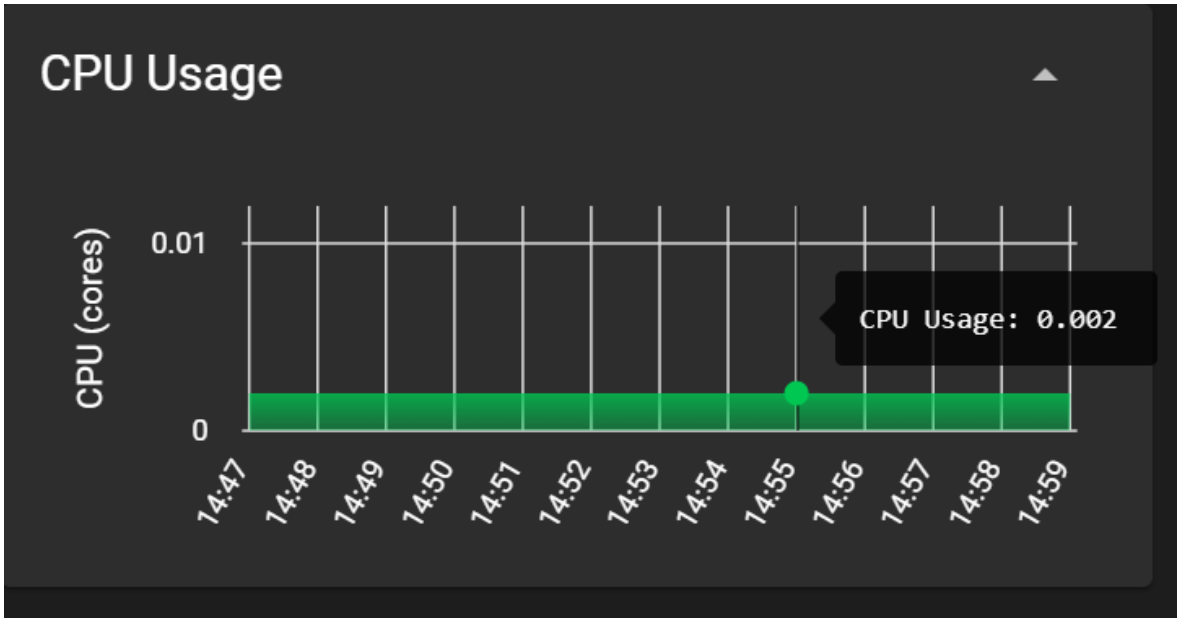


Obrázek 11 Využití CPU (při komunikaci bez zátěže)

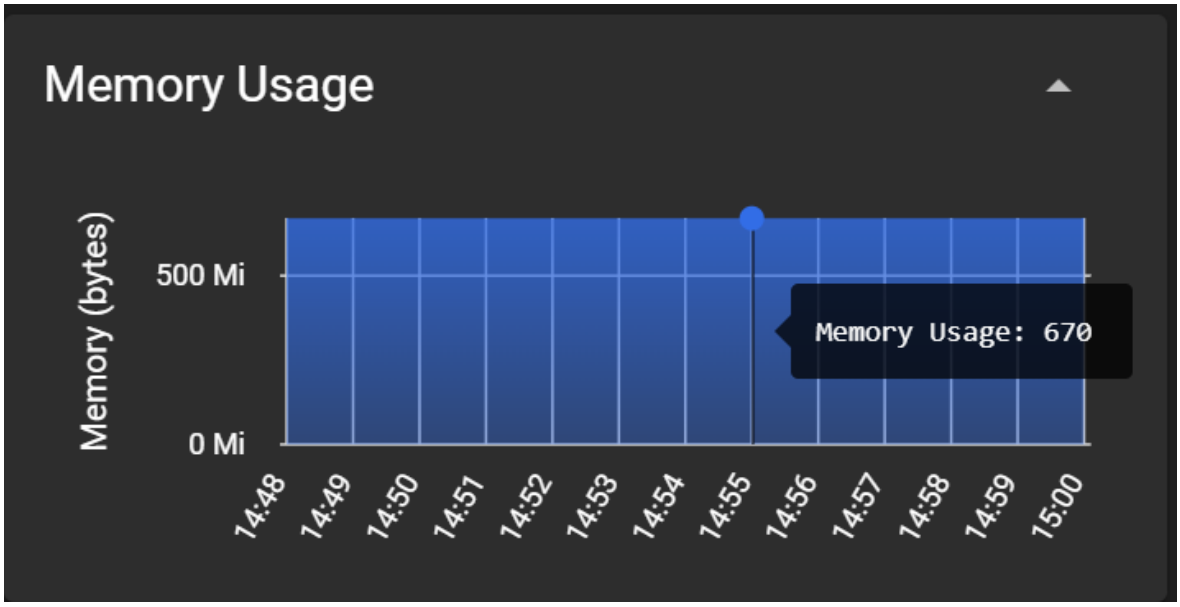


Obrázek 12 Využití RAM (bez zátěže)

Obrázky ukazují zátěž CPU a využití RAM, dvou simulátorů (těch, které budou později dále rozebrány) na prostředí k8s pro znázornění potřeb simulátorů pro chod. Statistiky byly vytvořeny v době, kdy nebyla žádná běžící komunikace kromě automatické echo komunikace.



Obrázek 13 Využití CPU (při komunikaci se zátěží)



Obrázek 14 Využití RAM (se zátěží)

4d1cb5fa...	2024-05-11 14:55:49	2024-05-11 14:55:49
91230cab...	2024-05-11 14:55:47	2024-05-11 14:55:47
d820b613...	2024-05-11 14:55:46	2024-05-11 14:55:46
ce5afa07...	2024-05-11 14:55:44	2024-05-11 14:55:44
3e3ead45...	2024-05-11 14:55:43	2024-05-11 14:55:43
8ac40ba5...	2024-05-11 14:55:42	2024-05-11 14:55:42
27593256...	2024-05-11 14:55:40	2024-05-11 14:55:40

Obrázek 15 Podložení zatížení komunikace (poslané TRN)

Obrázky (13,14) ukazují zátěž CPU a využití RAM, dvou simulátorů na prostředí k8s pro znázornění potřeb simulátorů pro chod. Statistiky byly vytvořeny v době, kdy byla provozována komunikace s POS (viz. Obrázek 15), spolu s automaticky generovanou echo zprávou. Jak je z Obrázků patrné, statistiky jsou prakticky stejné.

5.2.1 Jazyky

5.2.1.1 Jython

Jython je implementace jazyka Python v Javě, umožňující spouštění Pythonovského kódu na Java Virtual Machine (JVM) a přístup k Javovým třídám. V současnosti podporuje Jython verze 2.7.x Pythonu 2, přičemž probíhá práce na kompatibilitě s Pythonem 3. [36]

Licencován pod licencí PSF verze 2, Jython je k dispozici jak pro komerční, tak nekomerční použití, distribuován s volně dostupným zdrojovým kódem. [37]

Jython slouží k různým účelům, včetně:

a. Embedded Scripting:

   Jython umožňuje vývojářům v Javě začlenit jeho knihovny do svých systémů, což umožňuje koncovým uživatelům vytvářet skripty různé složitosti a zvyšovat tak funkčnost aplikace. [36]

b. Interactive Experimentation:

   „Jython poskytuje interaktivní interpret, který lze použít k interakci s Java balíčky nebo s běžícími Java aplikacemi.“ [36]

c. Rapid Application Development:

   "Programy v Pythonu jsou obvykle 2-10x kratší než ekvivalentní programy v Javě. To se přímo promítá do zvýšené produktivity programátora." [36]

5.2.1.2 JavaScript – Nashorn

JavaScriptový engine Nashorn byl zaveden jako součást Javy ve verzi 8 a sloužil k vykonávání JS kódu na platformě JVM. Byl vyvinut jako moderní a výkonná alternativa k původnímu JS enginu Rhino. Nashorn přinesl vylepšenou výkonost díky použití jazyka Java 7 a novým technologiím, jako je například just-in-time (JIT) kompilace, která umožňuje rychlejší interpretaci a vykonávání JS kódu přímo do Java programů. [38]

Díky své integraci s JVM mohl Nashorn snadno komunikovat s existujícím Java kódem a využívat knihovny a frameworky dostupné v ekosystému Javy. Nashorn byl oblíbeným nástrojem pro tvorbu skriptů, automatizaci a vývoj různých typů aplikací, včetně serverových, webových a desktopových aplikací. Nicméně, od verze Java 11 byl Nashorn označen jako zastaralý a odstraněn z distribuce JDK, a doporučuje se používat modernější alternativy, jako například GraalVM. [39]

5.2.2 Postman

Postman slouží jako klíčový nástroj pro vývojáře a testery, usnadňující manipulaci s rozhraními API. Rozhraní API fungují jako prostředníci, umožňující komunikaci mezi různými softwarovými systémy. Hlavní přínos Postmanu spočívá v jeho schopnosti zjednodušit složitosti průzkumu, testování a spolupráce s API. [40]

Klíčové funkce:

- Odesílání požadavků: Postman umožňuje vytváření a odesílání různých typů požadavků (např. GET, POST) s možností nastavení parametrů a autentizačních metod.
- Kontrola odpovědí: Po odeslání požadavků Postman představuje odpovědi API ve srozumitelném formátu, což uživatelům umožňuje snadno ověřit kódy stavu, hlavičky a těla odpovědí.
- Organizace a sdílení: Uživatelé mohou kategorizovat požadavky do kolekcí, usnadňující jejich správu a sdílení mezi členy týmu, s přidanou dokumentací a štítky pro větší srozumitelnost.
- Spolupráce: Postman podporuje spolupráci prostřednictvím sdílených kolekcí, spolupráce na požadavcích a správy verzí, podporující týmovou práci.

5.2.3 Použité Vývojové Prostředí

5.2.3.1 Visual Studio Code

Microsoft Visual Studio Code (VS Code) je robustní editor pro psaní kódu navržený tak, aby podporoval učení i profesionály v jejich kódovacích dovednostech. Nabízí bezproblémové a intuitivní uživatelské rozhraní, které umožňuje rychlé a efektivní psaní kód u v různých programovacích jazycích, aniž by bylo nutné přepínat editory. Jeho rozsáhlá podpora jazyků zahrnuje populární jazyky jako PY, Java, C++, JS a další. [41]

Hlavní funkce:

- Snadnost použití: VS Code poskytuje uživatelsky přívětivé rozhraní a podporu pro širokou škálu programovacích jazyků, což umožňuje uživatelům rychle začít s programováním.
- Spolupráce: Díky rozšíření Live Share mohou uživatelé spolupracovat na dálku s kolegy a instruktory v reálném čase, což zvyšuje produktivitu a usnadňuje týmovou práci.
- Korekce chyb: VS Code nabízí návrhy pro dokončení kódu a rychlé opravy běžných chyb, zatímco jeho debugger umožňuje uživatelům postupně procházet kód a identifikovat problémy efektivně.
- Přizpůsobení: Uživatelé si mohou přizpůsobit své prostředí pro psaní kódu výběrem vlastních motivů, písem, ikon a barevných schémat podle svých preferencí a pracovních postupů.

- Správa verzí: Integrované funkce správy verzí umožňují uživatelům sledovat změny ve svém kódu v čase, zajistit ukládání pokroku a usnadnit spolupráci v týmových projektech.

V podstatě je Microsoft VS Code všestranným a funkcemi bohatým editorem pro psaní kódu, který odpovídá potřebám jak začátečníků se, tak profesionálů, a vytváří prostředí příznivé pro produktivitu, spolupráci a neustálé učení. [41]

5.2.3.2 InteliJ IDEA

IntelliJ IDEA je sofistikované integrované vývojové prostředí (IDE), které uspokojí potřeby jak začátečníků, tak zkušených profesionálů ve sféře softwarového vývoje. Nabízí komplexní sadu funkcí a nástrojů, které usnadňují efektivní psaní kódu v různých programovacích jazycích a frameworkách, poskytují uživatelům plynulý vývojový zážitek. [42]

Hlavní rysy:

- Komplexní nástrojová sada: IntelliJ IDEA disponuje širokou paletou nástrojů a funkcí navržených pro zefektivnění procesu softwarového vývoje. Od inteligentního doplňování kódu po pokročilé nástroje pro ladění kódu, IntelliJ IDEA vybavuje vývojáře potřebnými nástroji pro snadné psaní kvalitního kódu.
- Podpora jazyků a frameworků: S podporou několika programovacích jazyků a frameworků, včetně Java, Kotlin, JS a dalších, IntelliJ IDEA splňuje různorodé potřeby vývojářů pracujících na široké škále projektů.
- Pokročilá analýza kódu: IntelliJ IDEA sahá dál než základní kontrola syntaxe a nabízí pokročilé nástroje pro analýzu kódu, které pomáhají identifikovat potenciální problémy, optimalizovat výkon kódu a zlepšit jeho kvalitu.
- Nástroje pro zvýšení produktivity: Vestavěné funkce, jako je refaktoring kódu, integrace správy verzí a automatické generování kódu, pomáhají vývojářům zvýšit svou produktivitu a udržovat konzistenci kódu po celý životní cyklus vývoje.
- Nástroje pro spolupráci: IntelliJ IDEA nabízí funkce pro spolupráci, jako je integrace správy verzí a nástroje pro revizi kódu, které usnadňují týmovou práci a umožňují vývojářům účinně spolupracovat na projektech.

V podstatě je IntelliJ IDEA silným a všestranným IDE, které umožňuje vývojářům psát, ladit a nasazovat kód s jistotou. S intuitivním rozhraním, rozsáhlou sadou funkcí a robustním ekosystémem pluginů a rozšíření zůstává IntelliJ IDEA přední volbou pro profesionály ve

softwarovém vývoji. [42] Právě InteliJ je hlavním vývojovým prostředím požívaným u nás ve firmě Monet+.

5.3 Kde je využijeme?

Simulátory jsou využívány tam, kde je potřeba autorizovat transakce. To zahrnuje širokou škálu prostředí a situací, kde je nezbytné ověřit správnost a spolehlivost platebních systémů.

Jedním z hlavních míst, kde jsou simulátory používány, jsou testovací laboratoře a oddělení zajišťování kvality. Zde jsou simulátory využívány k testování nových softwarových aktualizací, konfigurací a platebních scénářů před jejich nasazením do produkčního prostředí.

Dalším důležitým místem je vývojové prostředí, kde jsou simulátory integrovány do procesu vývoje softwaru. Vývojáři je využívají k testování nových funkcí, ladění a ověřování integrace s různými platebními partnery.

Simulátory jsou také využívány ve školení a vzdělávacích programech pro zaměstnance, kteří pracují s platebními systémy. Poskytují jim možnost se seznámit s různými platebními scénáři a procedurami bez rizika ovlivnění reálných transakcí.

Jednoduše řečeno, simulátory jsou široce využívány ve všech fázích vývoje, testování a provozu platebních systémů, kde hrají klíčovou roli při zajištění kvality, spolehlivosti a bezpečnosti platebních operací.

5.4 Co je možné testovat?

Simulátory umožňují testovat širokou škálu funkcionalit platebních systémů v různých situacích a prostředích. Mezi hlavní oblasti, které je možné testovat pomocí simulátorů, patří autorizace (P2H), přepínání (switching) a komunikace mezi systémy (H2H).

Autorizace (P2H):

- Autorizace neboli proces ověření platby, je klíčovou částí platebního toku. Simulátory umožňují testovat autorizaci transakcí z různých perspektiv, včetně ověřování platby pomocí různých platebních karet nebo bankovních účtů.

Switching (Přepínání):

- Switching se týká směrování a přenosu platebních transakcí mezi různými platebními sítěmi a institucemi. Simulátory umožňují testovat tento proces v různých scénářích, včetně testování směrování transakcí, zpracování chyb, a řízení toku platebních dat.

Komunikace mezi systémy (H2H):

- Komunikace mezi systémy (H2H) je důležitým aspektem platební infrastruktury, kde různé systémy komunikují a výměnou dat provádějí platební operace. Simulátory umožňují testovat tuto komunikaci v různých scénářích, včetně testování integrace mezi platebními partnery, synchronizaci dat a zpracování různých typů platebních zpráv.

Díky simulátorům jsou organizace schopny lépe porozumět a ověřit funkčnost svých platebních systémů v různých scénářích a prostředích. Takto mohou lépe připravit své systémy na reálné situace a minimalizovat riziko vzniku chyb nebo selhání během provozu.

II. PRAKTICKÁ ČÁST

6 REALIZACE SIMULATORU PRO ISO8583 PROTOKOL

V této praktické části budou provedeny programové realizace simulátoru v programovacích jazycích Jython a JavaScript s uvedenými popisy úprav při používání scriptů.

Při realizaci simulátoru jsem vycházel ze vzoru, který se již využíval při tvorbě simulátorů na jiném oddělení

6.1 Základ simulátoru

Simulátory se skládají minimálně ze dvou konfiguračních souborů (počet souborů se liší podle toho o jaký programovací jazyk se jedná), které tvoří základ simulátoru a application.properties, které obsahuje cestu ke konfiguračním souborům. Jádro simulátoru se tedy skládá ze tří základních souborů:

- Definiční soubor s koncovou příponou – JSON
- Modifikační soubor – Handler
- Application properties

A v případě, že se jedná o python (Jython) simulátory, tak ze čtyř:

- Definiční soubor s koncovou příponou – JSON
- Modifikační soubor – Handler
- Application properties
- Globální python definiční soubor pro daný komunikační protokol

6.1.1 Addresářová struktura simulátorů

a) V případě INT, ACC, PROD:
   Všechny zmíněné soubory musejí být ve stejné složce, kromě application.properties, kterou si systém hledá ve složce conf (a podle hodnoty nastavené property iso.server.def.path bude systém hledat konfigurační soubory simulátorů v uvedeném adresáři)
b) V případě k8s:
   Se očekává, že v defaultním nastavení budou všechny soubory na jednom místě včetně application.properties a to konkrétně ve složce configs
c) V případě local host:
   Je potřeba mít nejdříve nainstalovaný (nadefinovaný systém) daný protokol (jak toho dosáhnout je mimo scope této práce, ale ve zdrojích je uveden tutorial jak toho

dosáhnout) [29] a následně v application.properties definovat cestu ke konfiguračním souborům simulátoru (většinou C://test např.)

6.1.1.1 Application properties

Tento soubor slouží k definování cesty, kde se bude kontrolovat přítomnost konfiguračních souborů simulátorů (tedy kde je bude script očekávat). Další funkcionalitou lze mezi JavaScriptem a Pythonem volně přepínat pomocí property use.python (defaultně true) definované v application.properties.

```
### Adresar kde se vyhledavaji definice pro ISO8583 porty
iso.server.def.path=/tohle/je/cesta/k/systemovym/definicim/protokolu

### Definition to specify if python should be used
use.python=false
```

Obrázek 16 Ukázka obsahu souboru application.properties

6.1.1.2 JSON definiční soubor

Tento soubor bývá z pravidla pojmenován podle jména třetí strany, kterou má daný simulátor simulovat, pro lepší přehlednost (např. Monet+.json). Soubor je využíván pro definování polí (Data Fields) a filtrů, které definují, jaká pole se budou posílat, a to, jak nutná bude jejich přítomnost ve zprávě při posílání konkrétního typu TRN. Soubor se dělí na hlavičku, která definuje:

- Port na, kterém simulátor poběží (bude naslouchat)
  Je nutné jej vždy definovat, a to v maximálním rozmezí 0 až 64 000 z důvodu velikosti pole
- headType a headLen jsou pole, která spolu souvisejí
  headType určuje délku celé zprávy (může být nastaven jako ASCII, BCD nebo BYTE)
  headLen udává kolik znaků (headType znaků), je ta délka
  (bez těchto hodnot by nebylo možné z paketu data dobře přečíst)
- Handler, který se bude využívat pro modifikaci odpovědí

Je nutné jej vždy definovat. Bývá standardem, že nese stejné pojmenování jako definiční soubor s příponou json a handler za jménem (např. Monet+handler), ale není to pravidlem.

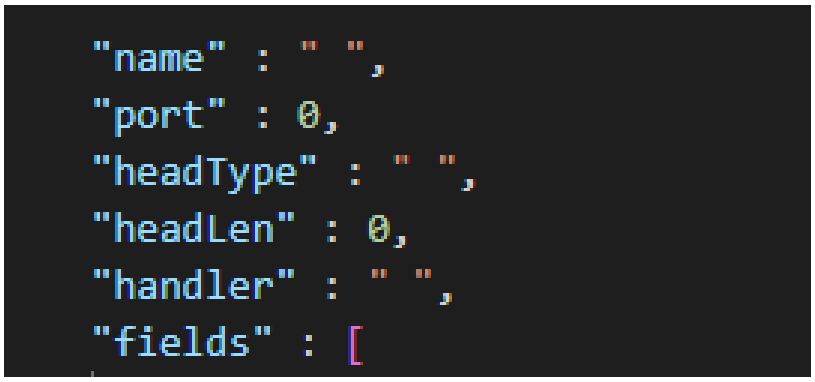
Důležité ovšem je, aby handlery PY a JS měli stejné pojmenování a byly správně pojmenovány v hlavičce definičního souboru.

- Pinkey pro šifrování pinu

  Pole musí být vždy vyplněno a jeho hodnota se nastavuje pomocí šifrovaní DES

- Mackey slouží jako verifikační hodnota nad celou zprávou

  Nutnost nastavení Mackey záleží na třetí straně, zda využívá macování (ve většině případů, ale není použit). Využívá se šifrování také pomocí DES



Obrázek 17 Ukázka hlavičky def. souboru

A tělo, které definuje vlastnosti pole:

- ID
- Jméno
- Délku pole (maximální) a typ délky pole

  Tyto dvě pole spolu navzájem souvisí (viz. Tab. 8)

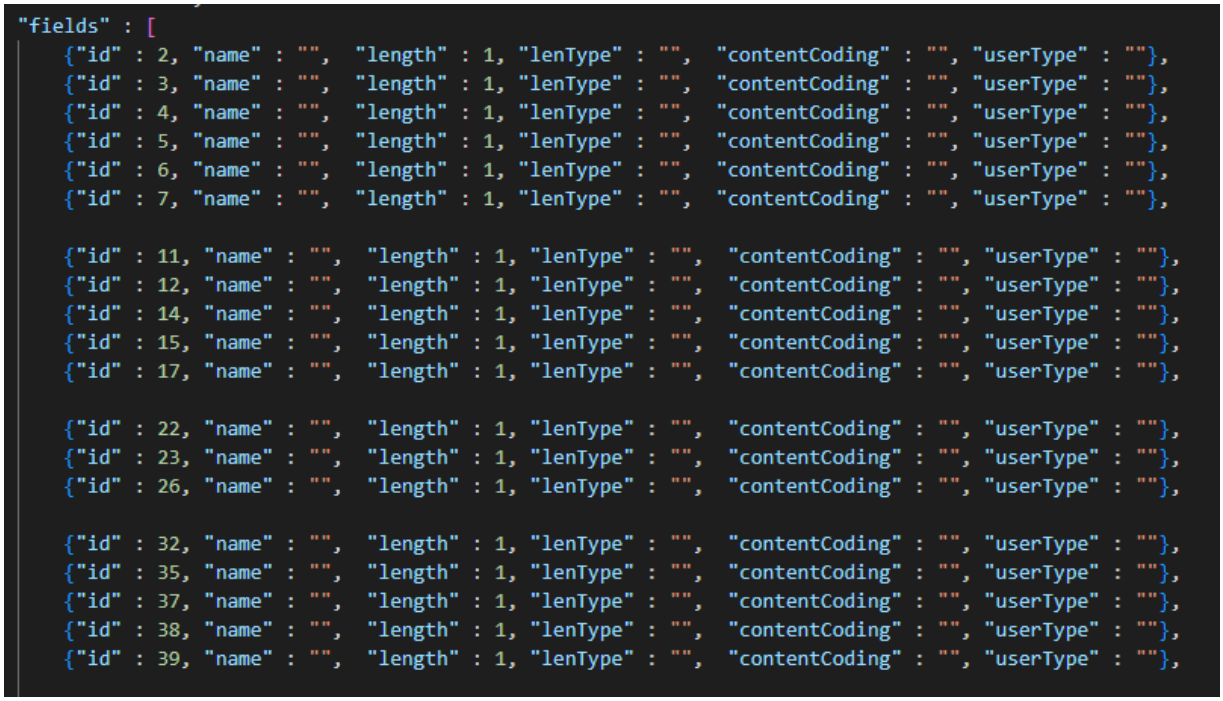
- Kódování pole a typ pole

  Tyto dvě pole jsou na sobě také závislé, jedná se o definování datového typu pole a následného kódování pole

  UserType může nabývat hodnot definovaných v Tab. 6.

  ContentCoding může mít hodnoty ASCII, EBCDIC, BCD, BIN tyto hodnoty definují, do jakého datového formátu se mají data serializovat.

Všechny tyto data jsou nastavovány podle specifikací dodaných třetí stranou. Specifikace většinou mývají podobu tabulky podobnou viz. Tab. 11.



Obrázek 18 Ukázka těla def. Souboru

Filtry pro nastavení mandatorních polí se nastavují pod tělem v definičním souboru a specifikujeme u nich tyto hodnoty:

- MTID
- Data fields (datová pole)

  A u datových polí ještě uvádíme:
  - Přítomnost Datového pole

    To může nabývat hodnot:
    - M
    - C
    - O (případně nemusí být vyplněno nebo se používá –)

      Hodnoty přítomnosti dat. pole jsou vysvětleny v Tab. 6

Obrázek 19 Ukázka filtru def. souboru

6.1.1.3 Handler soubor

Soubor Handler definuje chování simulátoru. Skládá se z funkcí napsaných v programovacím jazyce (JavaScript nebo Jython). Jeho jádrem je funkce 'processMessage' u JS a „process_message“ u PY, která funguje podobně jako funkce main, sloužící jako vstupní bod.

```python
def process_message(msg, spec, req_context):
    """ Main function for message processing """
    # Retyping to Python Java object wrappers
    msg = Iso8583(msg, spec)
    result = msg
    mtid = msg.get_mtid()

    if mtid == '1800':
        print('IS NMR')
        result = proc_nmm(msg)
    else:
        print('Unknown MTID: ' + mtid)

    return result.to_java()
```

Obrázek 20 Ukázka hlavní fce v PY

```javascript
var processMessage = function(msg,spec) {
    if (msg.getMtid() == "1800") {
        print("IS NMM");
        return NetworkEcho(msg,spec);
    } else {
        print("Unknown MTID : "+msg.getMtid());
        return msg;
    }
};
```

Obrázek 21 Ukázka hlavní fce v JS

V této funkci lze definovat akce, které mají být provedeny, když simulátor obdrží požadavek s konkrétním identifikátorem typu zprávy (MTID), pomocí samostatných volání funkcí konfigurovaných pro odpovídající MTID transakce.

```python
def proc_nmm(msg_in):
    msg_in = check_type(msg_in)
    return msg_in.with_mtid('1810').with_field(39, '000')
```

Obrázek 22 Ukázka samostatné fce volané uvnitř hlavní fce v PY

```
var NetworkEcho = function(msgIn,spec) {
    return msgIn
        .withMtid("1810")
        .withField(spec.fKey(39), "0000");
};
```

Obrázek 23 Ukázka samostatné fce volané uvnitř hlavní fce v JS

Tento soubor může být buď typu 'py' nebo 'js', jeho pojmenování je nepodstatné. Ovšem je důležité, aby handlery PY a JS měli stejné pojmenování a byly správně pojmenovány v hlavičce definičního souboru.

6.1.1.4 Konfigurační soubor

Konfigurační soubor má definovány všechny základní objekty a funkce pro správné fungování simulačních skriptů, pomáhá udržovat přehlednost kódu v handlerech a vytváří sjednocené fce, které bude možné volat z jakéhokoliv pythnovského simulátoru. Tento soubor obsahuje funkce, které zastávají stejnou funkcionalitu jako JsUtils v JS handlerech.

```
"""Config file for Iso8583 worker - version 1.0"""
# Do not delete or modify this file!
# !/usr/bin/env python2.7
from cz.monetplus.iso8583 import JsUtils


class Iso8583(object):
    """Python wrapper for Iso8583 Java class"""

    def __init__(self, iso8583Java, iso8583SpecJava):
        """Converts Java object to Python object"""
        self.iso8583Java = iso8583Java
        self.iso8583Spec = Iso8583Spec(iso8583SpecJava)

    def get_mtid(self):
        """
        Gets MTID from Iso8583 object.

        :returns: [str] Iso8583 MTID

        Java code:
            Iso8583.getMtid();
        """
        return self.iso8583Java.getMtid()
```

Obrázek 24 Ukázka konfiguračního souboru pro PY sim

6.1.1.5 Psaní scriptů

Stejně jako u JavaScriptu se veškeré Pythonovské skripty načítají ze složky definované v property iso.server.def.path v soubrou application.properties. Tato složka musí obsahovat soubor iso8583_config.py, kde jsou definovány všechny základní objekty a funkce pro správné fungování simulačních skriptů. Tato složka rovněž obsahuje jednotlivé skripty k handlerům pojmenované namehandler.py a popřípadě dodatečné Pythonovské soubory, obsahující funkce navíc k jednotlivým serverům. Tyto dodatečné soubory lze importovat do Python enginu tak, že v JSONu definující server doplníme key "imports", který obsahuje seznam názvů všech souborů, které chceme importovat do enginu před spuštěním samotného skriptu (např. "imports" : ["example_file.py", "another_file.py"],).

6.1.1.5.1 Pro JavaScript

Skript využívá objekty a funkce z interního souboru JsUtils. Zpráva se začne zpracovávat ve funkci processMessage = function(msg,spec). Ta přejímá přijaté objekty msg a spec, které dovolují přímo pracovat s jejich obsahem, např. přes metody v JsUtils nebo přes vytvořené metody v Java objektu.

Interní metody a fce v Java objektu a JsUtils jsou mimo scope této práce ovšem podobné fce můžeme najít u python simulátorů (níže), které obsahují vysvětlení volaných fcí.

Funkce processMessage vrací zapouzdřený Java objekt, který je možné vracet přímo z funkcí volaných uvnitř fce processMessage.

Skript napsaný pro jednotlivý server by měl dodržovat následnou strukturu:

```
var NetworkEcho = function(msgIn,spec) {
    return msgIn
        .withMtid("1810")
        .withField(spec.fKey(39), "0000");
};

var processMessage = function(msg,spec) {
    if (msg.getMtid() == "1800") {
        print("IS NMM");
        return NetworkEcho(msg,spec);
    } else {
        print("Unknown MTID : "+msg.getMtid());
        return msg;
    }
};
```

6.1.1.5.2 Pro Python

Jython podporuje pouze verzi Pythonu 2.7! Skript napsaný pro jednotlivý server by měl dodržovat následnou strukturu:

```
# !!! JYTHON SUPPORTS PYTHON 2.7 ONLY !!! #
# !/usr/bin/env python2.7

# Safe import
try:
    from iso8583_config import *
except ImportError:
    pass


def proc_example(msg_in):
    # Function check_type retypes Any type to Python java object wrapper
    # Python 2.7 do not support this style of typing: proc_example(msg_in:
Iso8583) -> Iso8583
    msg_in = check_type(msg_in)

    # Process in function using Iso8583 and Utils methods

    return msg_in


def process_message(msg, spec, req_context):
    """ Main function for message processing """
    # Retyping to Python Java object wrapper
    msg = Iso8583(msg, spec)
    req_context = RequestContext(req_context)

    print('In message : ' + msg.to_string())
    result = msg

    # Process msg
    result = proc_example(msg)

    return result.to_java()
```

Skript využívá objekty a funkce z iso8583_config.py. Pokud budeme chtít použít ještě další Pythonovský soubor, tak jej přidáme do try-except bloku a jeho název do konfiguračního JSONu jak je popsáno výše. Zpráva se začne zpracovávat ve funkci process_message(msg, spec, req_context). Ta přetypuje přijaté objekty msg a spec na Pythonovský wrapper objekt Iso8583. Ten poskytuje metody, které provolávají metody v Java objektu.

Dále objekt RequestContext, který obsahuje proměnné mimo samotnou zprávu (např. pinKey). K těmto proměnným lze přistupovat pomocí metody RequestContext.get('key'), která vrátí hodnotu pro daný klíč, jestli existuje (např. RequestContext.get('pinKey')).

Jelikož Python 2 nepodporuje typování v hlavičce funkce, tak je typování nahrazeno funkcemi z config souboru - check_type(msg) a check_types(msg, req_context). Tyto funkce vezmou vstupní objekt typu Any a vrátí ty samé objekty, ale se správnými typy. Pokud budou vstupní objekty jiného typu, tak funkce vyhodí TypeError. Používat tyto funkce ke kontrole typů není nutné, nicméně je to doporučeno a usnadňuje to psaní skriptů.

Funkce process_message vrací zapouzdřený Java objekt, proto je nutné na Pythonovském wrapper objektu zavolat funkci to_java() během returnu.

Metody wrapper objektů

Wrapper objekty nám umožňují přístup k metodám zapouzdřených Java objektů, aniž bychom museli volat Java kód přímo ze skriptu. Všechny metody jsou podrobně popsány přímo v kódu.

Iso8583 (BicIso)

Jedná se o hlavní objekt nesoucí v sobě zprávu a a specifikaci polí zprávy. Ten obsahuje metody jako get_mtid(), get(key), get_by_index(index), get_or_else(key, or_else_value), with_mtid(mtid), atd. Pokud název funkce končí slovem by_index, tak se data dohledávají podle čísla indexu. pokud ne, tak se data dohledávají pomocí klíče pole (ekvivalent v Javě např. Iso8583.get(Iso8583Spec.fKey(key)).get()). Specifikace polí je součástí přímo wrapperu, takže jako vstupní parametr key stačí zadat pouze číslo pole. Použití takové metody může vypadat následovně:

```
# Getting response code from message
response_code = msg_in.get_or_else(39, '000')
```

Tento kód uloží hodnotu pole 39 do proměnné response_code. Pokud je toto pole null, tak použije hodnotu '000'. Pokud chceme zprávu modifikovat, tak použijeme metody začínající slovem with/without - with_mtid(mtid), with_field(key, value), without_field(key), atd. Objekt v těchto metodách vrací sám sebe, akorát s modifikovaným polem. Speciálním případem je metoda with_fields(key_val_pairs). Ta na vstupu přijímá seznam dvojic klíč-hodnota - seznam všech polí, které modifikuje:

```
# Modifying fields 39, 47 and 54
msg_res = msg_in.with_fields([
        (39, '000'),
        (47, 'Transakce schvalena'),
        (54, None)
    ])
```

V tomto kódu se uloží modifikovaný objekt do proměnné msg_res. Oproti původnímu se změní pole 39 a 47 a odstraní pole 54.

Utils

Statická třída Utils zajišťuje přístup k metodám statické třídy v Javě JsUtils. Obsahuje metody jako num_approval(), now_to_string(format), hex(to_hex), unhex(str_hex), validate_pin(pin_block, control, pan, key), atd. Metoda now_to_string(format), nám vrátí aktuální datum a čas ve formátu, který si specifikujeme proměnnou format, to může vypadat následovně:

```
# Getting date in format: 20230725
now = Utils.now_to_string('yyyyMMdd')
```

Pro pokročilejší práci s datumem a časem můžeme použít knihovnu datetime v Pythonu.

RequestContext

Objekt RequestContext obsahuje jedinou metodu get(key). Ta vrátí hodnotu podle klíče z kontextu, pokud kontext hodnotu pro tento klíč obsahuje (např. pinKey).

```
# Getting pinKey from request context
pin_key = req_context.get('pinKey')
```

Tipy k psaní skriptů

- Doporučené vývojové prostředí pro psaní skriptů je PyCharm. Lze také použít VS Code s rozšířením pro Python.
- Pokud importujeme do skriptu handleru jiný Python soubor, ohraničme tenhle import blokem try-except a přidejme název souboru do seznamu imports v konfiguračním JSONu. Jinak by mohlo docházet k chybám při spouštění v Jython enginu.
- Podrobnosti o metodách jednotlivých wrapper objektů můžeme zjistit najetím kurzoru na název metody nebo prokliknutím se přes metodu přímo do konfiguračního souboru.

6.2 Hotová struktura simulátoru

V této kapitole je rozebrán ISO8583 simulátor, který byl vytvořen na základě specifikací ze 4. kapitoly o ISO8583. Simulátory jsem ovšem pro potřeby práce vytvořil dva, druhý bude rozebrán v 7. kapitole úprava simulátoru pro validaci dat.

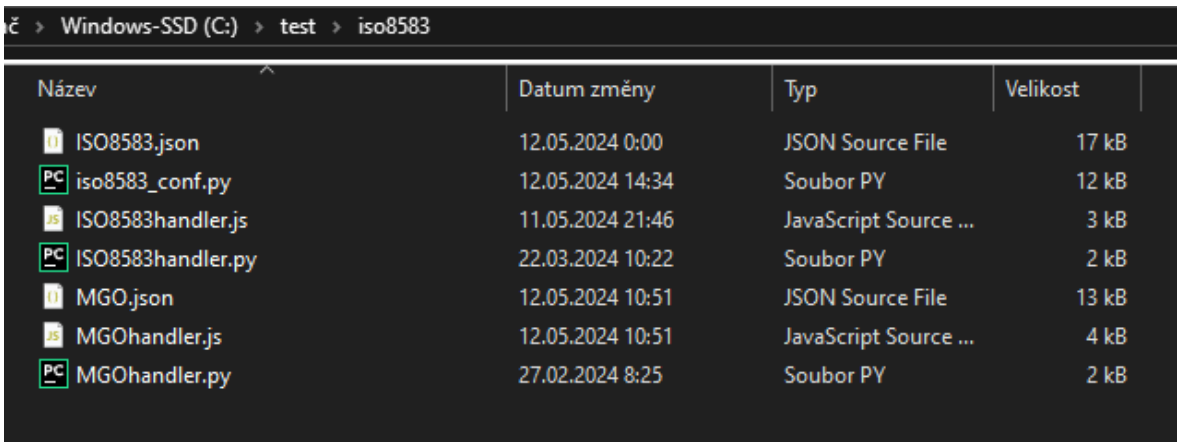
Simulátor má připraveny veškeré soubory potřebné pro jeho správný chod. Simulátor ISO8583 slouží jako ukázka vyplnění polí v konfiguračních souborech.

- Application.properties
- ISO8583.json
- ISO8583handler.js
- ISO8583handler.py
- Iso8583_conf.py.

6.2.1 Soubor Application Properties

```
### Port na kterem je interni WEB server
server.port=451

### Adresar kde se vyhledavaji definice pro ISO8583 porty
iso.server.def.path=C://test/iso8583

### Definition to specify if python should be used
use.python=false
```

Zde jsme nastavili správnou cestu k definičním souborům



Obrázek 25 Screen adresáře s uloženými soubory

6.2.2 Soubor ISO8583.json

Hlavička:

```
{
    "name" : "ISO8583",
    "port" : 1984,
    "headType" : "BYTE",
    "headLen" : 2,
    "handler" : "ISO8583handler",
```

V hlavičce jsme nastavili správné jméno souboru, volitelný port (který již není obsazený – v tom případě by se spustil simulátor, který byl dříve definovaný), headType na hodnotu BYTE a headLen na hodnotu 2.

Tělo (Data Fields):

```
    "fields" : [
       {"id" :    2, "name" : "pan",                      "length"
:    19, "lenType" : "LLVAR",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    3, "name" : "processingCode",           "length"
:     6, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" :
"AN"},
       {"id" :    4, "name" : "amount",                   "length"
:    16, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    5, "name" : "amountReconciliation",     "length"
:    16, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    6, "name" : "amountCardholderBilling",  "length"
:    16, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    7, "name" : "dateTimeTransmission",     "length"
:    10, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},

       {"id" :    11, "name" : "stan",                    "length"
:    12, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    12, "name" : "timeCreate",              "length"
:    14, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    14, "name" : "dateExpire",              "length"
:     4, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    15, "name" : "dateSettlement",          "length"
:     8, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
       {"id" :    17, "name" : "dateCapture",             "length"
:     4, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},

       {"id" :    22, "name" : "posEntryMode",            "length"
:    16, "lenType" : "FIXED",  "contentCoding" : "BIN", "userType" : "BIN"},
       {"id" :    23, "name" : "cardSequenceNumber",      "length"
:     3, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},
```

```
        {"id" :    26, "name" : "merchantCategoryCode",     "length"
:     4, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},

        {"id" :    32, "name" : "acquiringInstCode",        "length"
:    11, "lenType" : "LLVAR",  "contentCoding" : "ASCII", "userType" : "N"},
        {"id" :    35, "name" : "track2",                   "length"
:    37, "lenType" : "LLVAR",  "contentCoding" : "ASCII", "userType" :
"ANS"},
        {"id" :    37, "name" : "rrn",                      "length"
:    12, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" :
"AN"},
        {"id" :    38, "name" : "approvalCode",             "length"
:     6, "lenType" : "LLVAR",  "contentCoding" : "ASCII", "userType" : "N"},
        {"id" :    39, "name" : "actionCode",               "length"
:     4, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},

        {"id" :    41, "name" : "cardAcceptorTerminalId",   "length"
:    16, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" :
"ANS"},
        {"id" :    42, "name" : "cardAcceptorIdCode",       "length"
:    35, "lenType" : "LLVAR",  "contentCoding" : "ASCII", "userType" :
"ANS"},
        {"id" :    43, "name" : "cardAcceptorName",         "length" :
99999, "lenType" : "LLLLVAR", "contentCoding" : "ASCII", "userType" :
"ANS"},
        {"id" :    45, "name" : "track1",                   "length"
:    76, "lenType" : "LLVAR",  "contentCoding" : "ASCII", "userType" :
"ANS"},
        {"id" :    47, "name" : "industry",                 "length" :
99999, "lenType" : "LLLVAR", "contentCoding" : "ASCII", "userType" : "ANS"},
        {"id" :    48, "name" : "additionalDataPrivate",    "length" :
99999, "lenType" : "LLLVAR", "contentCoding" : "ASCII", "userType" : "ANS"},
        {"id" :    49, "name" : "currency",                 "length"
:     3, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},

        {"id" :    52, "name" : "pinData",                  "length"
:     8, "lenType" : "FIXED",  "contentCoding" : "BIN", "userType" : "BIN"},
        {"id" :    53, "name" : "secInfo",                  "length"
:    48, "lenType" : "LLVAR",  "contentCoding" : "BIN", "userType" : "BIN"},
        {"id" :    54, "name" : "additionalAmounts",        "length"
:   126, "lenType" : "LLLVAR", "contentCoding" : "ASCII", "userType" :
"ANS"},
        {"id" :    55, "name" : "iccSystemRelatedData",     "length"
:   999, "lenType" :"LLLLVAR", "contentCoding" : "BIN", "userType" : "BIN"},

        {"id" :    64, "name" : "mac",                      "length"
:     4, "lenType" : "FIXED",  "contentCoding" : "BIN", "userType" : "BIN"},
```

```
        {"id" :    70, "name" : "fileTransfDescriptData",   "length"
:    18, "lenType" : "FIXED",  "contentCoding" : "ASCII", "userType" : "N"},

        {"id" :    90, "name" : "reeserved",                "length"
:   999, "lenType" :"LLLLVAR", "contentCoding" : "ASCII", "userType" :
"ANS"},

        {"id" :    128, "name" : "secondaryMac",            "length"
:     4, "lenType" : "FIXED",  "contentCoding" : "BIN", "userType" : "BIN"}

    ],
```

Pole simuáltoru byla definována na základě specifikací protokolu ISO8583

Filtry:

```
"filters" : [
        {
            "mtid":"1100",
            "fields" : [
                {"index": 2, "presence":"M"},
                {"index": 3, "presence":"M"},
                {"index": 4, "presence":"M"},
                {"index": 7, "presence":"M"},
                {"index": 11, "presence":"M"},
                {"index": 12, "presence":"M"},
                {"index": 14, "presence":"M"},
                {"index": 15, "presence":"C"},
                {"index": 17, "presence":"C"},
                {"index": 22, "presence":"M"},
                {"index": 23, "presence":"C"},
                {"index": 26, "presence":"M"},
                {"index": 32, "presence":"M"},
                {"index": 35, "presence":"C"},
                {"index": 37, "presence":"M"},
                {"index": 41, "presence":"M"},
                {"index": 42, "presence":"C"},
                {"index": 43, "presence":"C"},
                {"index": 45, "presence":"C"},
                {"index": 47, "presence":"C"},
                {"index": 48, "presence":"C"},
                {"index": 49, "presence":"M"},
                {"index": 52, "presence":"C"},
                {"index": 53, "presence":"C"},
                {"index": 54, "presence":"C"},
                {"index": 55, "presence":"C"},
                {"index": 64, "presence":"C"},
                {"index": 70, "presence":"C"},
```

```
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1110",
        "fields" : [
            {"index": 2, "presence":"C"},
            {"index": 3, "presence":"C"},
            {"index": 4, "presence":"C"},
            {"index": 7, "presence":"C"},
            {"index": 11, "presence":"M"},
            {"index": 12, "presence":"C"},
            {"index": 14, "presence":"C"},
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"C"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"C"},
            {"index": 32, "presence":"C"},
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"M"},
            {"index": 38, "presence":"M"},
            {"index": 39, "presence":"M"},
            {"index": 41, "presence":"C"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"C"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1200",
        "fields" : [
            {"index": 2, "presence":"M"},
            {"index": 3, "presence":"M"},
            {"index": 4, "presence":"M"},
            {"index": 7, "presence":"M"},
            {"index": 11, "presence":"M"},
            {"index": 12, "presence":"M"},
            {"index": 14, "presence":"M"},
```

```
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"M"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"M"},
            {"index": 32, "presence":"M"},
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"M"},
            {"index": 41, "presence":"M"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"M"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1210",
        "fields" : [
            {"index": 2, "presence":"C"},
            {"index": 3, "presence":"C"},
            {"index": 4, "presence":"C"},
            {"index": 7, "presence":"C"},
            {"index": 11, "presence":"C"},
            {"index": 12, "presence":"C"},
            {"index": 14, "presence":"C"},
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"C"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"C"},
            {"index": 32, "presence":"C"},
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"M"},
            {"index": 38, "presence":"C"},
            {"index": 39, "presence":"M"},
            {"index": 41, "presence":"C"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
```

```
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"C"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1220",
        "fields" : [
            {"index": 2, "presence":"M"},
            {"index": 3, "presence":"M"},
            {"index": 4, "presence":"M"},
            {"index": 7, "presence":"M"},
            {"index": 11, "presence":"M"},
            {"index": 12, "presence":"M"},
            {"index": 14, "presence":"M"},
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"M"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"M"},
            {"index": 32, "presence":"M"},
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"M"},
            {"index": 38, "presence":"C"},
            {"index": 41, "presence":"M"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"M"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1230",
        "fields" : [
```

```
            {"index": 2, "presence":"C"},
            {"index": 3, "presence":"C"},
            {"index": 4, "presence":"C"},
            {"index": 7, "presence":"C"},
            {"index": 11, "presence":"C"},
            {"index": 12, "presence":"C"},
            {"index": 14, "presence":"C"},
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"C"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"C"},
            {"index": 32, "presence":"C"},
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"M"},
            {"index": 38, "presence":"C"},
            {"index": 39, "presence":"M"},
            {"index": 41, "presence":"C"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"C"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1420",
        "fields" : [
            {"index": 2, "presence":"M"},
            {"index": 3, "presence":"M"},
            {"index": 4, "presence":"M"},
            {"index": 7, "presence":"M"},
            {"index": 11, "presence":"M"},
            {"index": 12, "presence":"M"},
            {"index": 14, "presence":"C"},
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"M"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"M"},
            {"index": 32, "presence":"M"},
```

```
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"M"},
            {"index": 38, "presence":"C"},
            {"index": 39, "presence":"C"},
            {"index": 41, "presence":"M"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"M"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1430",
        "fields" : [
            {"index": 2, "presence":"C"},
            {"index": 3, "presence":"C"},
            {"index": 4, "presence":"C"},
            {"index": 7, "presence":"C"},
            {"index": 11, "presence":"C"},
            {"index": 12, "presence":"C"},
            {"index": 14, "presence":"C"},
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"C"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"C"},
            {"index": 32, "presence":"M"},
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"M"},
            {"index": 38, "presence":"C"},
            {"index": 39, "presence":"M"},
            {"index": 41, "presence":"M"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"M"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
```

```
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1800",
        "fields" : [
            {"index": 2, "presence":"C"},
            {"index": 3, "presence":"C"},
            {"index": 4, "presence":"C"},
            {"index": 7, "presence":"M"},
            {"index": 11, "presence":"M"},
            {"index": 12, "presence":"C"},
            {"index": 14, "presence":"C"},
            {"index": 15, "presence":"C"},
            {"index": 17, "presence":"C"},
            {"index": 22, "presence":"C"},
            {"index": 23, "presence":"C"},
            {"index": 26, "presence":"C"},
            {"index": 32, "presence":"C"},
            {"index": 35, "presence":"C"},
            {"index": 37, "presence":"C"},
            {"index": 38, "presence":"C"},
            {"index": 39, "presence":"C"},
            {"index": 41, "presence":"C"},
            {"index": 42, "presence":"C"},
            {"index": 43, "presence":"C"},
            {"index": 45, "presence":"C"},
            {"index": 47, "presence":"C"},
            {"index": 48, "presence":"C"},
            {"index": 49, "presence":"C"},
            {"index": 52, "presence":"C"},
            {"index": 53, "presence":"C"},
            {"index": 54, "presence":"C"},
            {"index": 55, "presence":"C"},
            {"index": 64, "presence":"C"},
            {"index": 70, "presence":"C"},
            {"index": 90, "presence":"C"},
            {"index": 128, "presence":"C"}
        ]
    },{
        "mtid":"1810",
        "fields" : [
            {"index": 2, "presence":"C"},
            {"index": 3, "presence":"C"},
            {"index": 4, "presence":"C"},
```

```
                {"index": 7, "presence":"M"},
                {"index": 11, "presence":"M"},
                {"index": 12, "presence":"C"},
                {"index": 14, "presence":"C"},
                {"index": 15, "presence":"C"},
                {"index": 17, "presence":"C"},
                {"index": 22, "presence":"C"},
                {"index": 23, "presence":"C"},
                {"index": 26, "presence":"C"},
                {"index": 32, "presence":"C"},
                {"index": 35, "presence":"C"},
                {"index": 37, "presence":"C"},
                {"index": 38, "presence":"C"},
                {"index": 39, "presence":"M"},
                {"index": 41, "presence":"C"},
                {"index": 42, "presence":"C"},
                {"index": 43, "presence":"C"},
                {"index": 45, "presence":"C"},
                {"index": 47, "presence":"C"},
                {"index": 48, "presence":"C"},
                {"index": 49, "presence":"C"},
                {"index": 52, "presence":"C"},
                {"index": 53, "presence":"C"},
                {"index": 54, "presence":"C"},
                {"index": 55, "presence":"C"},
                {"index": 64, "presence":"C"},
                {"index": 70, "presence":"C"},
                {"index": 90, "presence":"C"},
                {"index": 128, "presence":"C"}
            ]
        }
    ]
}
```

Filtry byly definovány podle Tab. 9

6.2.3 Soubor ISO8583handler.js

```
var PreAuthorization = function(msgIn,spec) {
    var Utils = Java.type('cz.monetplus.iso8583.JsUtils');
    var respCode = processResponseCodeByAmount(msgIn.get(4));
    var msgBase = respCode == "0000" ? msgIn.withField(spec.fKey(38),
Utils.numApproval()) : msgIn;
    return msgBase
        .withMtid("1110")
        .withoutFieldIndex(55)
        .withField(spec.fKey(39), respCode);
};
```

```
var BaseTransaction = function(msgIn,spec) {
    var Utils = Java.type('cz.monetplus.iso8583.JsUtils');
    var respCode = processResponseCodeByAmount(msgIn.get(4));
    var msgBase = respCode == "0000" ? msgIn.withField(spec.fKey(38),
Utils.numApproval()) : msgIn;
    return msgBase
        .withMtid("1210")
        .withoutFieldIndex(55)
        .withField(spec.fKey(39), respCode);
};

var AdviceTransaction = function(msgIn,spec) {
    return msgIn
        .withMtid("1230")
        .withoutFieldIndex(55)
        .withField(spec.fKey(39), "0000");
};

var ReversalTransaction = function(msgIn,spec) {
    return msgIn
        .withMtid("1430")
        .withoutFieldIndex(55)
        .withField(spec.fKey(39), "0000");
};

var NetworkEcho = function(msgIn,spec) {
    return msgIn
        .withMtid("1810")
        .withField(spec.fKey(39), "0000");
};

var processMessage = function(msg,spec) {
    print('In message : ' + msg);
    if (msg.getMtid() == "1100") {
        //predautorizace
        print("IS Preauthorization");
        return PreAuthorization(msg,spec);
    } else if (msg.getMtid() == "1200") {
        //normalni trn
        print("IS BaseFIn");
        return BaseTransaction(msg,spec);
    } else if (msg.getMtid() == "1220") {
        //normalni trn
        print("IS AdviceFIn");
        return AdviceTransaction(msg,spec);
    } else if (msg.getMtid() == "1420") {
        //reverzni trn
        print("IS Reversal");
        return ReversalTransaction(msg,spec);
```

```
    } else if (msg.getMtid() == "1800") {
        //echo (network managemet)
        print("IS NMM");
        return NetworkEcho(msg,spec);
    } else {
        print("Unknown MTID : "+msg.getMtid());
        return msg;
    }
};

var processResponseCodeByAmount = function(amount) {
    if (amount == null || amount == 0)
        return "0000";

    if (amount > 10000 && amount < 19999) {
        var rc = amount - 10000;
        return ""+rc;
    }
    return "0000";
};
```

Handler je nastavený tak, aby měl definovanou funkci na všechny možné typy TRN, kde se kromě funkcí na zpracování autorizace a finanční TRN vrací odpověď s MTID, které odpovídá RESP pro dané MTID requestu. Odpověď se vrací bez pole s id 55 téměř ve všech případech kromě MTID 1800 což je echo a pole s id 55 se zde ani neposílá. Téměř ve všech případech je zde posíláno pole s id 39 a hodnotou 0000, což znamená potvrzeno.

U zprávy s MTID 1100 a 1200 je navíc využita nová funkce, která na základě posílané částky v poli s id 4, vrátí číslo, které je následně přiřazeno proměnné respCode. Ta je poté použita při definování proměnné msgBase jako ternary operand kde pokud bude hodnota respCode nastavena na 0000 tak se má proměnné msgBase přiřadit msgIn a pokud ne tak se proměnné msgBase nastaví hodnota msgIn s polem id 38 na hodnotu vygenerovanou pomocí Java object metody. MsgBase je poté vracen v odpovědi s hodntou respCode pro id 39.

7 ÚPRAVA SIMULÁTORU

V této kapitole je ukázka úpravy simulátoru pro validaci response kódu na základě posílané částky. Je zde ukázka přetypování částky na pevnou hodnotu pro validaci funkcionality. A další možnosti úprav simulátorů.

7.1 Ukázky modifikace simulátoru

```
var PreAuthorization = function(msgIn,spec) {
    var Utils = Java.type('cz.monetplus.iso8583.JsUtils');
    var LongZ = Java.type('java.lang.Long');
    return msgIn
        .withMtid("1110")
        .withField(spec.fKey(4), LongZ.parseLong("54654"))
        .withField(spec.fKey(38), Utils.numApproval())
        .withField(spec.fKey(39), "000")
        ;
};

var BaseTransaction = function(msgIn,spec) {
    var Utils = Java.type('cz.monetplus.iso8583.JsUtils');
    var respCode = processResponseCode-
ByAmount(msgIn.get(spec.fKey(4)).get());
    var msgBase = respCode == "000" ? msgIn.withField(spec.fKey(38),
Utils.numApproval()) : msgIn;
    var LongZ = Java.type('java.lang.Long');
    var newAmount = changeAmount(msgIn.get(spec.fKey(4)).get());
    return msgBase
        .withMtid("1210")
        .withoutFieldIndex(55)
        .withField(spec.fKey(4), LongZ.parseLong(newAmount))
        .withField(spec.fKey(6), msgIn.get(4)-1)
        .withField(spec.fKey(12), msgIn.get(12))
        .withField(spec.fKey(28), msgIn.get(28))
        .withField(spec.fKey(39), respCode)
        .withField(spec.fKey(51), msgIn.get(49))
        .withField(spec.fKey(60),
Utils.unhex("002602010022010F3030303030303030333135353731020430303030130247310F0156FF024646"))
        ;
};

var ReversalTransaction = function(msgIn,spec) {
    return msgIn
        .withMtid("1430")
        .withField(spec.fKey(39), "000");
};
```

```
var NetworkEcho = function(msgIn,spec) {
    return msgIn
        .withMtid("1814")
        .withField(spec.fKey(3), msgIn.get(3))
        .withField(spec.fKey(11), msgIn.get(11))
        .withField(spec.fKey(12), msgIn.get(12))
};

var processMessage = function(msg,spec) {
    print('In message : ' + msg);
    if (msg.getMtid() == "1804") {
        print();
        print(" IS NMM ");
        print("- CHECK WHETHER THIS SIMULATOR STILL LIVES - AUTOMATIC RE-
SPONSE -");
        return NetworkEcho(msg,spec);
    } else if (msg.getMtid() == "1200") {
        print();
        print("* IS BaseFIn *");
        return BaseTransaction(msg,spec);
    } else if (msg.getMtid() == "1100" || msg.getMtid() == "1110") {
        print();
        print("~ IS Pre-Auth ~");
        return PreAuthorization(msg,spec);
    } else if (msg.getMtid() == "1420" || msg.getMtid() == "1421") {
        print();
        print("xX IS Reversal Xx");
        return ReversalTransaction(msg,spec);

    } else {
        print();
        print('!-! Unknown MTID: ' + msg.getMtid() +  ' !-!');
        print('!---------------------------!');
        return msg;
    }
};
var processResponseCodeByAmount = function(amount) {
    var Utils = Java.type('cz.monetplus.iso8583.JsUtils');
    if (amount == null || amount == 0)
        return "000";
    else if (amount == 1)
        return "001";

    else if (amount == 10000)
        return "100";
    else if (amount == 10100)
        return "101";
    else if (amount == 10600)
        return "106";
```

```
    else if (amount == 10700)
        return "107";
    else if (amount == 10800)
        return "108";
    else if (amount == 10900)
        return "109";

    else if (amount == 14200)
        return "142";
    else if (amount == 14300)
        return "143";
    else if (amount == 14400)
        return "144";
    else if (amount == 14500)
        return "145";
    else if (amount == 14600)
        return "146";
    else if (amount == 14700)
        return "147";

    else if (amount == 19000)
        return "190";

    else if (amount == 30100)
        return "301";
    else if (amount == 30600)
        return "306";

    else if (amount == 40000)
        return "400";

    else if (amount == 100010)
        return "001";

    else if (amount == 2222) {
        print("Simulating response delay");
        Utils.sleep(60000);
        return "909";
    }
    else if (amount == 2223 || amount == 2224 || amount == 2225)
        return "902"; // Testovani reverzalu

    return "000";
};

var changeAmount = function(amount) {
    if (amount == 100010){
        amount = 11210;
        return amount = 11210;
```

```
    }
    if (amount == 25000){
        amount = 4443;
        return amount;
    }
        if (amount == 31100){
            amount = 6100;
        return amount;
        }
    else{
        return amount;
    }
};
```

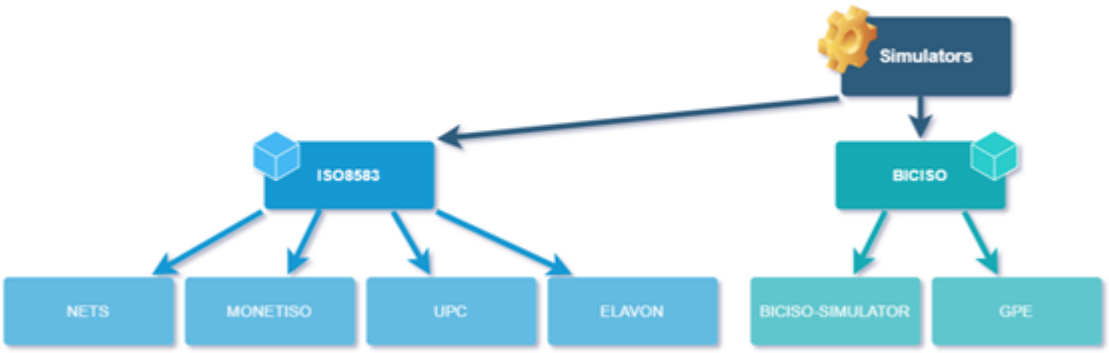
```
# Safe import
try:
    from iso8583_config import *
except ImportError:
    pass


def proc_base_fin(msg_in):
    msg_in = check_type(msg_in)
    rc = proc_rc_by_amount(msg_in.get_by_index(4))

    if rc == '00':
        msg_in = msg_in.with_field(38, Utils.num_approval())
        f47 = '0000048CHRDTransakce schvalena'
    else:
        f47 = "0002FU097CHRDTransakce zamitnuta"

    return msg_in.with_mtid('0110').with_fields([
        (39, rc),
        (47, f47)
    ])


def proc_online_fin(msg_in):
    msg_in = check_type(msg_in)
    rc = proc_rc_by_amount(msg_in.get_by_index(4))

    if rc == '00':
        msg_in = msg_in.with_field(38, Utils.num_approval())
        f47 = '0000048CHRDTransakce schvalena'
    else:
        f47 = "0002FU097CHRDTransakce zamitnuta"
```

```
    return msg_in.with_mtid('0110').with_fields([
        (39, rc),
        (47, f47)
    ])


def proc_reversal(msg_in):
    msg_in = check_type(msg_in)
    return msg_in.with_mtid('1440').with_field(39, '00')


def proc_nmm(msg_in):
    msg_in = check_type(msg_in)
    return msg_in.with_mtid('1830').with_field(39, '00')


def process_message(msg, spec, req_context):
    """ Main function for message processing """
    # Retyping to Python Java object wrappers
    msg = Iso8583(msg, spec)

    print('In message : ' + msg.to_string())
    result = msg
    mtid = msg.get_mtid()

    if mtid == '1100':
        print('IS BaseFIn')
        result = proc_base_fin(msg)
    elif mtid == '1220':
        print('IS OnlineFIn')
        result = proc_online_fin(msg)

    elif mtid == '1440':
        print('IS Reversal')
        result = proc_reversal(msg)
    elif mtid == '1820':
        print('IS NMM')
        result = proc_nmm(msg)
    else:
        print('Unknown MTID: ' + mtid)

    return result.to_java()


def proc_rc_by_amount_ccs(amount):
    if not amount or amount == 0:
        return '00'

    if 1000 < amount < 1100:
```

```
        return str(amount - 1000)

    return '00'
```

ZÁVĚR

Cílem této práce bylo vytvořit dokument, který by byl vhodný pro zaškolení nových pracovníků a představil jim práci se simulátory. Výsledkem této práce byly dva fungující simulátory, které demonstrují možnosti využití. Z podkladů tohoto projektu jsem byl schopný zprovoznit nové simulátory, které hned našly využití při testování fixů a nových funkcí. Vytvořené simulátory aktuálně běží na našem separátním prostředí v Kubernetees, kde se mi doposud podařilo zprovoznit simulátory protokolu ISO8583, BICISO a REST. Simulátory v aktuálním stádiu jsou již využívány QA oddělením k otestování nejrůznějších testovacích scénářů. Využití simulátorů se našlo téměř okamžitě, také pro automatizaci regresních testů. Na obrázku níže je znázorněno, jaké simulátory jsou již na oddělení QA funkční z protokolu ISO8583 a BICISO, protože REST simulátory jsou podstatou úplně jiné než oba zmíněné.



SEZNAM POUŽITÉ LITERATURY

[1] ŽÁČEK, Petr. *Testování Software: Úvod do problematiky*. Online, prezentace. 2023. Dostupné také z: https://moodle.utb.cz/mod/folder/view.php?id=560006.

[2] INTERNATIONAL SOFTWARE TESTING QUALIFICATIONS BOARD (ISTQB), . *Certified Tester Foundation Level (CTFL) Syllabus Version 4.0*. Online. 2023. Dostupné také z: https://castb.org/uploads/downloadables/_files_0/44__files_0.pdf.

[3] INTERNATIONAL SOFTWARE TESTING QUALIFICATIONS BOARD (ISTQB), . *Certified Tester Foundation Level (CTFL) Syllabus Version 3.1*. Online. 2018. Dostupné také z: https://castb.org/uploads/downloadables/_files_0/4__files_0.pdf.

[4] BERNSTEIN, Philip A. a NEWCOMER, Eric. *Principles of Transaction Processing*. online. 2. Morgan Kaufmann, 2009. Dostupné z: https://shorturl.at/asOT8. [cit. 2024-05-09].

[5] Obrazek. online. In: *Payway*. 2024. Dostupné z: https://www.payway.com/wp-content/uploads/how-payment-processing-works-payway-1024x651.jpg. [cit. 2024-02-20].

[6] *How Does a POS System Work?*. online. In: Koronapos. 2024. Dostupné z: https://koronapos.com/blog/how-does-a-pos-system-work/. [cit. 2024-03-28].

[7] Terminal. online. In: *Vectorstock*. 2024. Dostupné z: https://cdn5.vectorstock.com/i/1000x1000/46/69/single-sketch-payment-terminal-with-credit-card-vector-29564669.jpg. [cit. 2024-02-20].

[8] Logo M+. online. In: *Monetplus*. 2024. Dostupné z: https://www.monetplus.cz/sites/default/files/monet-logo/logoshortenweb_0.svg. [cit. 2024-02-20].

[9] *Transaction Processing Rules*. 2023. Dostupné také z: https://www.mastercard.us/content/dam/public/mastercardcom/na/global-site/documents/transaction-processing-rules.pdf.

[10] *ISO 8583 Reference Guide*. online. 2023. Dostupné také z: https://developerengine.fisglobal.com/assets/pdf/Worldpay_ISO_8583_Reference_Guide_V2.46.pdf.

[11] ZDRAVKOVIC, Andrej. *Wireless point of sale terminal for credit and debit payment systems*. online. 1. Waterloo, ON, Canada, 2002. ISBN 0-7803-4314-X. ISSN 0840-7789. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=685641. [cit. 2024-03-09].

[12] *Acquiring: Authorization by Bank Card, Protocols and Exchange Nodes. Part 1, «POS to HOST»*. online. In: Mstcompany. 2020. Dostupné z: https://mstcompany.net/blog/acquiring-authorization-by-bank-card-protocols-and-exchange-nodes-part-1-pos-to-host. [cit. 2024-03-09].

[13] *Choosing the Right Payment Processing Solution for Your Subscription-Based Business*. online. In: Payway. 2024. Dostupné z: https://www.payway.com/blog/choosing-right-payment-processing-solution-subscription-based-business. [cit. 2024-05-10].

[14] *Host-to-Host*. online. In: Sap. 2024. Dostupné z: https://shorturl.at/sFMQ1. [cit. 2024-03-12].

[15] *ISO8583 HOST-TO-HOST OR POS-TO-HOST?*. online. In: Field39. 2023. Dostupné z: https://www.field39.com/articles/iso8583-host-to-host-or-pos-to-host. [cit. 2024-03-09].

[16] *POS Host-to-Host Integration Guide*. online. In: Getneteurope. 2024. Dostupné z: https://docs.getneteurope.com/POSHostToHostIntegrationGuide.html. [cit. 2024-03-09].

[17] Processor. online. In: *Rapyd*. 2021. Dostupné z: https://www.rapyd.net/wp-content/uploads/2021/04/Online-Payment-Process.png. [cit. 2024-02-20].

[17] *What Are Host To Host Payments*. online. In: Gocardless. 2021. Dostupné z: https://gocardless.com/guides/posts/what-are-host-to-host-payments/. [cit. 2024-03-12].

[18] ACI. online. In: *Aciworldwide*. 2021. Dostupné z: https://www.aciworldwide.com/wp-content/uploads/2021/04/ACI-Corporate-Brochure-BR-US-4562-0111.pdf. [cit. 2024-05-09].

[19] *ACIWORLDWIDE*. online. In: Aciworldwide. 2021. Dostupné z: https://www.aciworldwide.com. [cit. 2024-03-12].

[20] *A Guide to the ACI Worldwide BASE24-eps on z/OS*. online. 1. IBM, 2009. Dostupné z: https://www.redbooks.ibm.com/abstracts/sg247684.html. [cit. 2024-03-12].

[21] BASE24-atm. online. In: *Aciworldwide*. 2009. Dostupné z: https://web.archive.org/web/20091004101338/http:/www.aciworldwide.com/igsbase/igstemplate.cfm/SRC%3DDB/SRCN%3D/GnavID%3D16/SnavID%3D48. [cit. 2024-03-10].

[22] BASE24-eps. online. In: *Aciworldwide*. 2009. Dostupné z: https://web.archive.org/web/20090620090322/http://www.aciworldwide.com/igsbase/igstemplate.cfm/SRC%3DDB/SRCN%3D/GnavID%3D16/SnavID%3D49. [cit. 2024-03-10].

[23] BASE24-infobase. online. In: *Aciworldwide*. 2009. Dostupné z: https://web.archive.org/web/20120312013147/http://www.aciworldwide.com/Products-and-services/Products/BASE24-infobase.aspx. [cit. 2024-03-10].

[25] BASE24-pos. online. In: *Aciworldwide*. 2009. Dostupné z: https://web.archive.org/web/20090529005051/http://www.aciworldwide.com/igsbase/igstemplate.cfm/SRC%3DDB/SRCN%3D/GnavID%3D16/SnavID%3D103. [cit. 2024-03-10].

[26] *What is ISO 8583?*. online. In: Github. 2021. Dostupné z: https://github.com/moov-io/iso8583/blob/master/docs/intro.md. [cit. 2024-05-09].

[27] *BASE24 BIC ISO Standards*. online. 2004. Dostupné také z: https://pdfcoffee.com/base24-bic-iso-standards-pdf-2-pdf-free.html.

[28] Obrázek. online. In: *Pdfcoffee*. 2004. Dostupné z: https://pdfcoffee.com/base24-bic-iso-standards-pdf-2-pdf-free.html. [cit. 2024-03-15].

[29] *An Introduction to ISO 8583: What you need to know*. online. In: Medium. 2023. Dostupné z: https://medium.com/@extio/a-introduction-to-iso-8583-what-you-need-to-know-7a78a304c31. [cit. 2024-03-22].

[30] ISO8583 Bitmap. online. In: *Paymentcardtools*. 2024. Dostupné z: https://paymentcardtools.com/iso-8583-bitmap. [cit. 2024-04-23].

[31] ISO 8583. online. In: *Wikipedia: the free encyclopedia*. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: https://en.wikipedia.org/wiki/ISO_8583. [cit. 2024-04-24].

[32] *PCI Compliance: Definition, 12 Requirements, Pros & Cons*. online. In: Investopedia. 2023. Dostupné z: https://www.investopedia.com/terms/p/pci-compliance.asp. [cit. 2024-04-16].

[33] Card brands. online. In: *Clearlypayments*. 2020. Dostupné z: https://www.clearlypayments.com/wp-content/uploads/2020/02/square-card-logos-whitebg-300x252.png. [cit. 2024-04-11].

[34] Issuer Bank. online. In: *Midigator*. 2021. Dostupné z: https://midigator.com/wp-content/uploads/2021/09/issuer-and-acquirier-roles-chart-1024x420.png. [cit. 2024-04-14].

[35] Carta. online. In: *Shutterstock*. 2024. Dostupné z: https://www.shutterstock.com/image-vector/vector-sketch-bank-credit-card-260nw-467164985.jpg. [cit. 2024-04-16].

[36] *Jython*. online. In: Jython. 2024. Dostupné z: https://www.jython.org. [cit. 2024-05-11].

[37] JUNEAU, Josh; BAKER, Jim; WIERZBICKI, Frank; MUOZ, Leo Soto; NG, Victor et al. *The Definitive Guide to Jython: Python for the Java Platform*. online. 1. apress, 2010. Dostupné z: https://shorturl.at/bmuAY. [cit. 2024-05-11].

[38] *One solution for execution of JavaScript in Java EE application servers*. online. 1. IEEE, 2018.

[39] SHARAN, Kishori. *Scripting in Java: Integrating with Groovy and JavaScript*. online. 1. apress, 2012. Dostupné z:

[9] https://books.google.cz/books?hl=en&lr=&id=qF8nCgAAQBAJ&oi=fnd&pg=PP3&dq=JavaScript+engine+Nashorn&ots=bOzbpJKqvU&sig=UMzlE7bWapci2bzqVCMAGKfdZ6s&redir_esc=y#v=onepage&q=JavaScript%20engine%20Nashorn&f=false. [cit. 2024-05-11].

[40] *Postman*. online. In: Postman. 2024. Dostupné z: https://www.postman.com. [cit. 2024-05-11].

[41] *VS Code*. online. In: Visualstudio. 2024. Dostupné z: https://code.visualstudio.com/learn. [cit. 2024-04-28].

[42] *InteliJ IDEA*. online. In: Jetbrains. 2024. Dostupné z: https://www.jetbrains.com/idea/features/. [cit. 2024-04-30].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ACC	Akceptační prostředí
Acquirer	Subjekt, který akceptuje karty / má terminálovou síť
Acquiring	Přijímaní plateb přes kartu
AH	Autorizační host (Monet+)
Approval code	Autorizační kód, vydává Issuer k potvrzení transakcí
DES	Data Encryption Standard – algoritmus pro šifrování elektornických dat
Fce	Funkce
H2H	Host-to-Host komunikace mezi systémy
Host	Nejčastěji referuje banku nebo její clienty
INT	Integrační prostředí
Issuer	Vystavitel platební karty
JS	JavaScript
JSON	Javascript Object Notation – formát pro výměnu dat mezi aplikacemi
JVM	Java Virtual Machine
K8S	Kubernetees
MC	MasterCard
MTID	Message type identifier – Uznačuje typ TRN
NON-PCI	Non-Payment Card Industry – Neplatební karty
P2H	Pos-to-Host komunikace mezi systémy
PAN	Primary account number = číslo platební karty viditelné na kartě
PCI	ayment card industry – Platební karty
POS	Point of sale – místo, kde dochází k použití karty (např. terminál)
Processor	Poskytovatel přímého spojení obchodníků s platebními sítěmi (VS, MC)
PROD	Produkční prostředí
PY	Python
QA	Oddělení zabývající se „zajištěním kvality“ (Quality Assuarence).
REQ	Požadavek (např. o zahájení komunikace)
RESP	Odpověď na REQ
Settlemenet	Pohyb peněz, převod na obchodní účet obchodníka (po určitém čase)
Sim	Simulátor
STAN	System trace audit number – pomáhá sledovat a auditovat transakce
TRN	Transakce

VS	Visa
VS CODE	Visual Studio Code

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA PI: SEZNAM PŘÍLOH NA CD

Prilohy.zip – všechny soubory využívané v práci