SimplePay 2.x (API v2)

Fejlesztési dokumentáció

Fizetési folyamat és fejlesztés

2024. 04. 12.

TARTALOM

Dokumentum történet

Dátum	Verzió	Változás
2018. 09. 30.	180930	Eredeti kiadás
2018. 10. 01.	181001	Query funkció és mintakód hozzáadása
2018. 10. 02.	181002	Finish funkció és mintakód hozzáadása
2018. 10. 04.	181004	Refund funkció és mintakód hozzáadása
2018. 10. 17.	181017	OneClick fizetések és mintakódok hozzáadása
2018. 10. 30.	181030	Hash kalkulálás és API hívás mintakódok hozzáadása
2019. 03. 01.	190301	OneClick fizetések kiemelése a dokumentációból SDK IPN függvények hozzáadása
2019. 06. 03.	190603	Pontosítások - forráskód részletek - kétlépcsős fizetés - SDK
2020. 01. 07.	200107	Erős ügyfélhitelesítés (SCA, 3DS) hozzáadása Start funkció további, opcionális változók hozzáadása
2020. 06. 03.	200603	Tranzakció azonosító méretének növelése Utalási napok szabályzása kereskedői admin felületen Simple applikációból kereskedői applikációba navigálás Pontosítások
2020. 08. 15.	200815	Pontosítások 3DS adatmezők esetén
2020. 09. 03.	200903	Hibakód lista frissítés
2020. 09. 06.	200906	Sandbox fizetőoldali 3DS Challenge teszt kártya
2021. 01. 18.	210118	Hibakódok bővítése
2021. 06. 15.	210615	Új IT support bejelentő rendszer alkalmazása (1.3) IPN üzenetek a sikertelen végstátuszok esetére (3.15) Mobil applikációval kapcsolatos mellékletek (I. és II.) revíziója Hibakódok frissítése (6)
2021. 10. 21.	211021	Mintakódok frissítése (3.2)
2022. 10. 12.	221012	Hibakódok bővítése: 2077, 3500-3508, 5043 (6.) Elírás javítása „description" → „desc" (3.8) Új fizetőoldali nyelvek: AR, ZN (3.1) Mobil applikáció deeplink funkcionalitás bővítése (II. melléklet)
2023. 01. 31.	230131	Service desk ideiglenes szüneteltetése, itsupport@otpmobil.com használata
2023. 07. 10	230710	Hibakódok bővítése: 6xxx (6.)
2024. 04. 12	240412	Adattkezelési tájékoztatók eléréseének változása az adattovábbítási nyilatkozatban (8.)

1 Rövid összefoglalás

1.1 A SimplePay online fizetés v2

A SimplePay az OTP Mobil Kft. online fizetési megoldása. Jelen dokumentáció a SimplePay fizetéshez használható kereskedői REST API és tranzakció kezelés v2 verziójához készült.

Ez az API a szolgáltatásait a korábbi v1 API-tól eltérő interface használatával kínálja. Az interface minden végponton JSON formátumú adatokkal kommunikál. Ezért a dokumentáció a technikai háttér megértését minden témakör esetén kétféle módon is illusztrálja.

Egyrészt az adott ponton szükséges **JSON példákkal**, ami alapján **bármilyen programnyelven megvalósítható** az online fizetés implementációja.
Másrészt a kereskedői admin felületről **letölthető PHP SDK mintakódjaival**. Az SDK egy kész fizetési mintakód, ami a fent említett JSON formátumú adatokat előállítja, kommunikál az API-val és kezeli az API-tól kapott válaszokat.

Az API a kereskedői tranzakcionális kérések befogadására szolgál, használata korlátozódik a tranzakciók kezelésére. Ebből adódóan az elszámolások, pénzügyi kérdések tekintetében nem okoz változást azon partnerek számára, akik már alkalmazzák a SimplePay fizetési rendszert a v1 API használatával.

A dokumentációnak nem célja a v1 és v2 interface különbségeinek a tárgyalása, bár helyenként utalhat erre, ha ez a megértést elősegítheti.

1.2 Fogalmak

A dokumentációban gyakran használt fogalmak.

API v1: SimplePay interface, aminek a használatával indítják tranzakcióikat a SimplePay szolgáltatást használó kereskedői rendszerek. A v1 interface továbbra is párhuzamosan elérhető azoknak a kereskedői rendszereknek, amelyek ennek használatával implementálták a SimplePay fizetést.
API v2: a 2018-tól használható új SimplePay interface, aminek a működését jelen dokumentáció írja le.
Authorizáció: a fizetőoldalon a megadott bankkártya adatok ellenőrzése, valamint a kártya terhelhetősége van vizsgálva. Ezek alapján történik meg a konkrét fizetés engedélyezése, vagy elutasítása a kártyát kibocsátó bank által.
Egy lépcsős fizetés: a fizetés elindítása után megtörténik a kártyaterhelés.
Két lépcsős fizetés: a fizetés után a megadott összeg be lesz foglalva a vásárló kártyáján, azonban a terhelés ekkor még nem történik meg. Ez után 21 napja van a kereskedőnek, hogy elindítsa a kártyaterhelést, vagy felszabadítsa a befoglalt összeget. Ha a 21. napig nem történik meg a terhelés, akkor a befoglalt összeg automatikusan felszabadul és a tranzakcióra terhelés már nem indítható.
Sandbox: teszt rendszer az éles rendszert szimuláló funkciókkal. A rajta végrehajtott tranzakciók esetén nem történik valós fizetés. Ezen a rendszeren csak a fejlesztéshez

szükséges tesztelésekhez lehet tranzakciót indítani. A sandbox rendszeren nem történik meg pénzügyi elszámolás.
Éles rendszer: valós tranzakciók kezelésére használható rendszer. Csak a teszt rendszeren elvégzett fejlesztés és sikeres tesztelés után érhető el.
SDK (Software Development Kit): fejlesztéshez, saját rendszerbe implementáláshoz felhasználható, a fizetést megvalósító mintakód.
Tranzakció: a kereskedő weboldaláról elindított fizetés, aminek az eredményéről a SimplePay rendszere visszajelez a kereskedő felé. A tranzakció tartalma (a kifizetendő termék) bármi lehet a szerződött kereteken belül. A fizetési tranzakció a kereskedői rendszerből indul és a fizetésről történő IPN (Instant Payment Notification) visszajelzéssel zárul le. A kereskedő oldali vásárlási folyamatnak a fizetési tranzakció csak egy része.
Fizetőoldal: a SimplePay rendszerének eleme. Itt adja meg a kártyaadatait a vásárló, illetve utalás esetén itt van tájékoztatva a szükséges adatokról.
Kereskedői fiók: a sandboxban és az éles rendszerben is létező egyedi kereskedői adminisztrációs felület. Ennek egyedi azonosítóit felhasználva lehet fizetési tranzakciót indítani.
Devizanem: a tranzakció devizaneme, ami Forint (HUF), Euro (EUR) és Dollár (USD) lehet. A devizanem a kereskedői fiókhoz kötött.
Fizetési mód: a SimplePay rendszerében bankkártya, vagy átutalás lehetséges.
Logo: a kereskedő oldalán megjelenített SimplePay logo.
Adattovábbítási nyilatkozat: a vásárlói adatok SimplaPay felé továbbításának a vásárlói elfogadása. Megtörténhet a kereskedői weboldalon regisztráláskor, vagy a fizetési tranzakció indítása előtt.
Kereskedői tesztek: a kereskedő által a fejlesztés közben végzett tesztek a dokumentációnak megfelelő működés ellenőrzésére.
SimplePay élesítés előtti tesztek: a tesztelés kereskedői jelzésre SimplePay IT support által történik. Csak azok a weboldalak, vagy fizetési rendszerek kerülhetnek élesítésre, ahol a SimplePay tesztek sikeresek és a szükséges technikai, illetve tájékoztató elemek a dokumentációnak megfelelők.
Teszt bankkártya: a sandbox rendszer fizetőoldalán lehet kiválasztani kártyát attól függően, hogy sikeres, vagy sikertelen tranzakciót szeretne a fejlesztő indítani.
Élesítés: a sikeres tesztek után a kereskedő számára aktiválva lesz az éles rendszerben is a fiókja. Ezen a fiókon tud online fizetéseket fogadni.
start: fizetési tranzakció generálása az API-n keresztül.
back: tájékoztató oldal a kereskedő rendszerében. Fizetés után ide érkezik vissza a vásárló a SimplePay fizetőoldaláról.
ipn: háttérben lezajló kommunikáció a SimplePay és a kereskedő rendszere között. A sikeres fizetésről tájékoztat. A kereskedőnek ennek fogadása után kezdheti meg a teljesítést.
query: tranzakció státusz lekérdezése.
finish: két lépcsős fizetés esetén a második lépcső a tranzakció lezárásának (terhelés) elindítása a kereskedő által
refund: visszatérítés indítása a kereskedő által
PSD2: az Európai Unió második pénzforgalmi irányelve a digitális pénzügyi szolgáltatásokról
SCA: (Strong Customer Autentication) erős ügyfélhitelesítés, vagy 2-faktoros ügyfél-hitelesítés a PSD2 irányelv alapján
3DS: kártyatársasági szabvány a kétfaktoros hitelesítés (SCA) megvalósítására

1.3 Az online fizetés fejlesztési és élesítési folyamata

Az alábbi feladatokat kell végrehajtani a fejlesztőnek a fejlesztés során.

- Tranzakció elindítása a vásárlói- és a kosár adatokkal.
- A fizetési oldalról visszatérő vásárló tájékoztatása a tranzakció eredményéről
- Az IPN üzenet fogadása
- SimplePay logo elhelyezése a weboldalon
- SimplePay adattovábbítási nyilatkozat elhelyezése a weboldalon

A fenti feladatok végrehajtása után jelezhető a fizetés fejlesztésének elkészülése az alábbi email címen:
itsupport@otpmobil.com

Ezután kollégáink is ellenőrzik az online fizetést. Ha a tesztek sikeresek, akkor elindul a fizetés élesítése.

A konkrét SimplePay oldali tesztek ugyanebben a dokumentációban, a „Tesztelés" c. fejezetben leírtak szerint történnek meg.

1.4 A fejlesztés időzítése

A SimplePay fizetés implementálásakor az alábbiakra való tekintettel kell tervezni:

- A fejlesztés Ön, vagy fejlesztője számára szükséges ideje
- A weboldal/fizetési rendszer SimplePay általi tesztjének ideje
- Az élesítés technikai átfutásának ideje

Ha teljesen elkészült a fejlesztés és a **8. fejezet** alapján minden kötelező pont ellenőrizve van a fejlesztő által, akkor az itsupport@otpmobil.com címen lehet kérni az weboldal SimplePay tesztelését.

A tesztelés a bejelentés után **1-3** munkanapon belül megtörténik. A tesztek eredményéről kollégáink minden esetben visszajeleznek.
Ha a SimplePay technikai tesztek is sikeresek, akkor az oldalon a fizetés élesíthető. Ebben az esetben **a sikeres tesztek után következő 1-5 munkanap alatt** történik meg az éles fiók aktiválása.

FONTOS: A technikai megfelelőségen túl az élesítés feltétele a szerződés aláírása, illetve a csatlakozási díj befizetése is.

A fentiekből adódóan akkor lehet biztos a bankkártyás fizetés Ön által tervezett indulási határideje tarthatóságában, ha **5-8 munkanappal** korábban jelzi felénk a fejlesztés elkészültét.

1.5 Az élesítés kereskedőt érintő lépései

A fejlesztés a teszt rendszer (sandbox) használatával történik. A SimplePay által a sandbox használatával végzett sikeres tesztek után lesz aktív az éles rendszer. Az éles és a sandbox rendszerben megegyeznek a tranzakció indításához szükséges kereskedői azonosító adatok. Ebből adódóan az éles rendszer aktiválása után egy

paraméter megváltoztatásával (URL) szabályozható, hogy melyik rendszer irányába van indítva a tranzakció.

A teszt rendszer és az éles rendszer között nincs kapcsolat, így a fizetést érintő saját beállításokat az éles rendszerben is szükséges megtenni. Ezek közül a legfontosabb az IPN URL abban az esetben, ha a kereskedőinek más elérési úttal rendelkező teszt és éles rendszere van (IPN leírás jelen dokumentációban később).
További tennivaló nincs, abban az esetben, ha a fejlesztéshez mellékelt mintakódot építi be a saját rendszerébe.

1.6 A fizetési tranzakciók folyamata

1.6.1 Bankkártyás fizetés

a. A vásárló a kereskedő weboldalán összeválogatja a termékeket, ami eredményeképpen kialakul a fizetendő végösszeg.
b. A tranzakciós adatokat a kereskedő átadja a SimplePay felé az API-n keresztül (**start**). Ezen a ponton létrejön a fizetési tranzakció a SimplePay rendszerében. A kapott adatokra válaszként a rendszer egy URL-t ad vissza. A kereskedő erre az URL-re kell átirányítsa a vásárlót a böngészőben.
c. A megadott URL-en a SimplePay fizetőoldalra érkezik a vásárló, ahol a tranzakció korábban megadott adatai jelennek meg. A fizetőoldalon tudja megadni a vásárló a kártyájának adatait. Ha van a Simple applikációban regisztrált kártája, akkor azt választva is elvégezheti a fizetést.
d. A kártyaadatok megadása után megtörténik a fizetés banki hitelesítése (authorizáció).
e. Az authorizáció után a vásárló vissza van irányítva a kereskedő weboldalára (**back**). Itt a visszaadott adatok alapján szükséges tájékoztatni a vásárlót az authorizáció eredményéről.
f. Ezután a háttérben lefut a csalás megfigyelő és megelőző folyamat. Ha ennek során nem észlel a rendszer semmilyen problémát, akkor a háttérben visszajelzést küld a weboldal felé (**ipn**). Ez a fizetési tranzakció vége. Miután az IPN üzenet megérkezik, teljesítheti a megrendelést a kereskedő.

1.6.2 Átutalás

a. A vásárló a kereskedő weboldalán összeválogatja a termékeket, ami eredményeképpen kialakul a fizetendő végösszeg.
b. A tranzakciós adatokat a kereskedő átadja a SimplePay felé az API-n keresztül (**start**). Ezen a ponton létrejön a fizetési tranzakció a SimplePay rendszerében. A kapott adatokra válaszként a rendszer egy URL-t ad vissza. A kereskedő erre az URL-re kell irányítsa a vásárlót a böngészőben
c. A megadott URL-en a SimplePay utalási tájékoztató oldalára érkezik a vásárló. Itt találja meg az utaláshoz szükséges adatokat, mint pl. bankszámla száma, és a szükséges közlemény tartalma.
d. A vásárló opcionálisan visszatérhet a kereskedő weboldalára (**back**), ahol a visszaadott adatok alapján szükséges tájékoztatni a vásárlót a tranzakció eredményéről.
e. Amint beérkezik az utalt összeg a SimplePay számlájára, a rendszer a háttérben visszajelzést küld a weboldal felé (**ipn**). Ez a fizetési tranzakció vége. Miután az IPN üzenet megérkezik, teljesítheti a megrendelést a kereskedő.

1.7 SANDBOX és éles fizetési rendszer

A SimplePay két egymástól teljesen elkülönülő fizetési rendszerből épül föl. Az egyik rendszer a fejlesztési tranzakciós tesztekhez használható, míg a másik az éles tranzakciókhoz. **A két rendszer nincs egymással semmilyen kapcsolatban.**

A fentiek miatt kiemelten fontos szem előtt tartani azt, hogy **minden beállítás**, amit egyik, vagy másik rendszeren elvégez, **csak abban a rendszerben érvényesül**. Ha szeretné alkalmazni a másik rendszerben is, akkor ott is szükséges a beállítása.

A sandbox rendszer használatával végzett sikeres fejlesztői és SimplePay oldali tesztek után lesz aktiválva az éles rendszerben is a kereskedő fiókja. Ha korábban próbál az éles rendszeren tranzakciót indítani, akkor hibát fog kapni.

A két rendszer között az alábbi lényeges különbségek vannak:

Sandbox rendszer

Kizárólag **csak teszt tranzakciók** indítására alkalmas, amikkel a rendszerhez való technikai csatlakozás tesztelhető.
Csak a fizetőoldalon található tesztkártyákkal lehet rajta tranzakciót indítani.

Tranzakciók azonosítói jelenleg 9 számjegyűek és jelenleg „5"-ös értékkel kezdődnek (5xxxxxxxx), emiatt a kereskedő rendszernek legalább 9 számjegyű tranzakció azonosító tárolására kell alkalmasnak legyen, illetve hosszabb távon gondolkodva érdemesebb 10 számjeggyel kalkulálni.

Kereskedői admin felület: https://sandbox.simplepay.hu/admin/
Tranzakciós kéréseket befogadó API: https://sandbox.simplepay.hu/payment/v2/
A fenti tranzakciós URL-t kell kiegészíteni a konkrét szolgáltatás megnevezésével. Például a „start" funkció esetén https://sandbox.simplepay.hu/payment/v2/start az URL.

Mivel a sandbox a tranzakciókat szimulálja, így **az alábbi funkciók tesztjére nem alkalmazható:**

- valódi bankkártya terhelésére
- fizetőoldali Simple accounton alapuló szolgáltatások
- fizetőoldali Simple applikációra alapuló szolgáltatások
- pénzügyi folyamatok és elszámolások
- tranzakciós analitikák generálására

Éles rendszer

Kizárólag **csak valós pénzforgalommal járó tranzakciók** indítására alkalmas.
Csak érvényes, bank által kiadott bankkártyákkal lehet rajta tranzakciót indítani.

Tranzakciók azonosítói jelenleg 9 számjegyűek és jelenleg „5"-ös értékkel kezdődnek (1xxxxxxxx), emiatt a kereskedő rendszernek legalább 9 számjegyű tranzakció

azonosító tárolására kell alkalmasnak legyen, illetve hosszabb távon gondolkodva érdemesebb 10 számjeggyel kalkulálni.

Kereskedői admin felület: https://admin.simplepay.hu/admin/
Tranzakciós kéréseket befogadó API: https://secure.simplepay.hu/payment/v2
A fenti tranzakciós URL-t kell kiegészíteni a konkrét szolgáltatás megnevezésével. Például a „start" funkció esetén https://secure.simplepay.hu/payment/v2/start az URL.

A továbbiakban minden példát a sandbox rendszer használatával mutatunk be.

1.8 Letölthető segédletek

A sandbox és az éles rendszer kereskedői adminisztrációs felületéről egyformán letölthető a fejlesztéshez szükséges segédanyagok.

Mindkét rendszerben a bal oldali menüben a „**Letöltések**" menüpont alatt találhatók meg az alábbiak

- technikai dokumentáció
- mintakód
- logo csomag
- pénzügyi segédanyag

2 Tranzakció és státuszai

A **fizetési tranzakciót** meg kell különböztetni a kereskedő oldali teljes **vásárlási folyamattól**, mivel annak csak egy részét képezi. A vásárlás folyamatába tartozik a termékek kiválasztása, a vásárlói kosár összeállítása, majd opcionálisan a teljesítéshez szükséges vásárlói adatok bekérése.

Amint a vásárláshoz szükséges minden adat rendelkezésre áll, elindítható a SimplePay fizetés. Ennek kimenetétől függően teljesíthető a megrendelés és befejezhető a vásárlási folyamat.

A fizetés állapota, vagy státusza alapvetően a sikeres/sikertelen viszonylatban fontos a kereskedő vásárlási folyamata szempontjából, azonban ennél jóval összetettebb és több állapottal rendelkezik.
Az alábbi státuszok bármikor lekérdezhetők a "**query**" API hívással. Az query hívás részleteit a folyamat többi lépésével együtt a későbbiekben bontjuk ki részletesen.

A SimplePay fizetés folyamata az alábbi státuszokkal rendelkezik.

Státusz	Esemény
INIT	Létrejött tranzakció a SimplePay rendszerében
TIMEOUT	Időtúllépés INIT státuszban
CANCELLED	A fizetőoldalon megszakított fizetés, vagy a vásárló elnavigál a fizető oldalról, vagy bezárja a böngészőt.
NOTAUTHORIZED	Sikertelen authorizáció
INPAYMENT	Fizetés alatt, a „Fizetek" gomb megnyomása után
INFRAUD	Vizsgálat alatt, csalásszűrés futása idejére
AUTHORIZED	Sikeres authorizáció a kártyaadatok megadása után
FRAUD	Csalás gyanú
REVERSED	Zárolt összeg visszafordítva (kétlépcsős)
REFUND	Visszatérítés (részleges, vagy teljes) Negatív összegű tranzakció
FINISHED	Sikeres, befejezett tranzakció

A tranzakció státuszai mellett a fizetőoldalról a kereskedői rendszerbe vissza irányítás esetén négy lehetséges esemény lehetséges. A későbbiekben a „**back**" visszairányítás témakörénél lesznek ezek részletezve. **Ezek az események** összetettebb folyamatok eredményei lehetnek és **nem feltétlenül egyeznek meg az aktuális tranzakció státusszal.**

Esemény	Esemény
SUCCESS	Sikeres kártya authorizáció
FAIL	Sikertelen kártya authorizáció, vagy sikertelen 3DS ellenőrzés
CANCEL	A fizetőoldalon megszakítja a vásárló a fizetést
TIMEOUT	Sikertelen authorizáció

3 Implementáció

Röviden összefoglalva a fizetés a következő módon zajlik le. A vásárlás adatait összeállítva a kereskedő rendszere létrehoz egy tranzakciót a SimplePay rendezerében. Ezt a "**start**" hívással tudja megtenni. A hívásra kapott válaszban kap egy URL-t, amire át kell irányítsa a vásárlót. Ha az átirányítás megtörténik, akkor érkezik meg a vásárló a fizetőoldalra.

A fizetőoldalon megadja a szükséges adatokat, vagy ha van regisztrált kártyája a Simple applikációban, akkor a fizetőoldalon azt is kiválaszthatja.

A választott módon elindítja a fizetést, ami után vissza van irányítva a böngészőben a kereskedő weboldalára, ahol a vásárló tájékoztatást kap az authorizáció eredményéről (**back**).

Eközben lezajlik a SimplePay csalásszűrési folyamata is (fraud monitoring), amiről a kereskedő felé a háttérben egy POST üzenetben (tehát nem a böngészőben) jelzést küld a tranzakció sikerességéről (**ipn**).

A kereskedő az „**ipn**" üzenet fogadása és megfelelő megválaszolása után teljesítheti a megrendelést.

Az alap tranzakciós kommunikációk mellé opcionálisan fejleszthető a visszatérítés (**refund**), vagy kétlépcsős tranzakciók esetére a lezárás (**finish**), illetve a tranzakció adatainak lekérdezése (**query**).

FONTOS: Mielőtt a fejlesztést elkezdi, készítsen biztonsági mentést webáruházáról, adatairól!

A fejlesztés ideje alatt fokozott figyelmet fordítson arra, hogy a fejlesztés alatti, vagy teszt alatt álló bankkártyás fizetést éles vásárláshoz, éles fizetési tranzakcióhoz ne lehessen használni! Ez főleg akkor fontos a fejlesztő részéről, ha már működő webáruházba illeszti be a SimplePay rendszert új fizetési módként.
Abban az esetben, ha nem megoldható különálló fejlesztői rendszer és az üzemelő webáruházban végzi a fejlesztést, akkor **jelezze a vásárlónak, hogy** ne válassza a bankkártyás fizetést, mert **még fejlesztés/tesztelés alatt van**.

3.1 Általános üzenetformátum

A v2 API és fizetési folyamat technikailag a következő alapokon nyugszik. Ez minden hívásnál és válasznál azonos. A továbbiakban minden leírás és mintakód ezen alapul.

Karakterkódolás: **UTF-8**
Az API üzenetek metódusa: **POST**
Az API hívásokon kívül a böngészőben történő visszairányítás metódusa **GET**.
Az aláírások (**Signature**) **HMAC HASH** metódusa: **SHA384**
Az API minden kommunikációban a **header**-ben várja és küldi az aláírásokat.
Az API minden kommunikációban a **http body**-ban várja és küldi a tranzakció adatait.
Az adatok formátuma: **JSON**
A **Content-Type** minden esetben **application/json**
A SimplePay tömör (felesleges white-space-ektől mentes) JSON-t ad vissza a válaszokban és ad át az IPN üzenetben. A kereskedői rendszertől nem elvárt a tömörség.
Minden időpontot **ISO 8601** szabvány szerinti stringként (2018-09-15T11:25:37+02:00) kell átadni és az API is ebben a formában adja vissza az idő értékeket.
A pénznemet **ISO 4217** szabvány szerint (HUF, EUR, USD, stb.) kell átadni.
Az országokat **ISO 3166-1 alpha-2** szabvány szerint (HU, GB, DE, stb.) kell átadni.
A nyelveket **ISO 639-1 alpha-2** szabvány szerint kell átadni, jelenleg a következők lehetnek: **AR, BG, CS, DE, EN, ES, FR, IT, HR, HU, PL, RO, RU, SK, TR, ZH**
Minden kérés és válasz tartalmaz egy "**salt**" elnevezésű mezőt, aminek a tartalma 32 véletlenszerű karakter. Ennek célja, hogy az üzenet varianciáját és ezzel az aláírás biztonságát növelje.
Az **összeg** minden esetben nullánál nagyobb számérték. HUF devizanem esetén egész szám, EUR és USD esetén két tizedesjegy használható. A tizedes elválasztó pont (.). Ezres elválasztó nem értelmezhető.
Az API hívások sikertelensége esetén a válasz kiegészül az „errorCodes" tömbbel, ami tartalmazza a hiba beazonosításához szükséges hibakódokat.

A **SimplePay** által generált **tranzakció azonosító** jelenleg **9** számjegyű. Emiatt a tranzakció adatmezőjének a kereskedői adatbázisban **legalább 9 számjegyű** érték tárolására kell alkalmasnak legyen, illetve hosszabb távon gondolkodva érdemesebb **10** számjeggyel kalkulálni.

Az API aktív fejlesztés alatt áll. A folyamatos funkcióbővülés miatt a végpontokon a kereskedő által kapott válaszok a jövőben további mezőkkel bővülhetnek.
Emiatt **a kereskedő** oldali rendszernek **akkor is fel kell tudja dolgozni a** SimplePay felől kapott, **már ismert mezők tartalmát, ha az üzenetben** olyan **új elemek jelennek meg,** amit még a kereskedői rendszerben nem implementáltak.
Ilyen esetben ezeket az elemeket egyszerűen ki kell hagyja és nem kell figyelembe vennie.

3.2 Üzenetek validálása

A body-ban küldött üzenet nem tartalmazza az aláírást. Az aláírást az üzenet header-ben szükséges küldeni "**Signature**" néven.

A "Signature" alapja az üzenet body (azaz a teljes JSON string). Az aláírás számításhoz a body-nak az SHA384 HMAC HASH értéke szükséges.
A "Signature" a kapott HASH-nek a Base64 enkódolt kimenete lesz.

A számítás fent leírt logikája az alábbiakban foglalható össze (a függvények és a szintaxis minden programnyelven mások lehetnek)

```
signature = codeBase64(hmacWithSha384(merchantKey, message))
```

Az alábbi tranzakció adatait használjuk mintaként.

```
{
"salt":"c1ca1d0e9fc2323b3dda7cf145e36f5e",
"merchant":"PUBLICTESTHUF",
"orderRef":"101010516348232058105",
"currency":"HUF",
"customerEmail":"sdk_test@otpmobil.com",
"language":"HU",
"sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e",
"methods":["CARD"],
"total":"25",
"timeout":"2021-10-30T12:30:11+00:00",
"url":"https:\/\/sdk.simplepay.hu\/back.php"
}
```

A lenti minta JSON string (**message**) és minta kereskedői kulcs (**merchantKey**) segítségével az alábbi eredmény (**signature**) kapható bármilyen programnyelven.
A **merchantKey** változóhoz a fiókonként egyedi érték a kereskedői vezérlőpulton a kereskedői fiók technikai beállításai között található meg "SECRET_KEY" néven.

Az alábbi adatokkal való API tesztelésre használatos eszközök esetén (pl. postman.com) legyen figyelemmel az alábbiakra:

- URL: https://sandbox.simplepay.hu/payment/v2/start
- metódus: POST
- az alábbi, enterek és egyéb formázások nélküli JSON stringet használja a hívás body-ban
- ügyeljen arra, hogy egy sorba legyen bemásolva a JSON sortörések nélkül
- a header tartalmazza a „Signature" elemet a lenti értékkel
- header „Content-Type" értéke „application/json" legyen
- a tranzakció már sikeresen ki lett fizetve a sandbox rendszeren, ezért a válaszban hibaüzenet fog megjelenni

body


```
{"salt":"c1ca1d0e9fc2323b3dda7cf145e36f5e","merchant":"PUBLICTESTHUF","orderRef":"101010516348
232058105","currency":"HUF","customerEmail":"sdk_test@otpmobil.com","language":"HU","sdkVersio
n":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e","methods":["C
ARD"],"total":"25","timeout":"2021-10-30T12:30:11+00:00","url":"https:\/\/sdk.simplepay.hu\/ba
ck.php"}
```


merchantKey


```
FxDa5w314kLlNseq2sKuVwaqZshZT5d6
```


Signature


```
gcDJ8J7TyT1rC/Ygj/8CihXaLwniMWRav09QSEMQUnv5TbYaEDvQAuBE1mW3plvZ
```


Minden API hívásra adott válasz is tartalmaz aláírást, amit kereskedői oldalon szükséges ellenőrizni a válasz hitelesítése végett.

3.3 start – bankkártyás fizetési tranzakció létrehozása

A start hívást az API az alábbi URL-en várja:
https://sandbox.simplepay.hu/payment/v2/start

A **start** a fizetési tranzakció kezdete. Ezen a ponton történik meg a kereskedői rendszerben összegyűjtött tranzakciós adatok továbbítása a SimplePay felé.
Az adatok között megtalálható a megrendelés globális adatai, a kosár tartalom, számlázási adatok, szállítási adatok, SimplePay rendszer-specifikus adatok, URL-ek, stb.
A tranzakciós adatokat az "**Általános üzenetformátum**" fejezetben leírt módon egy JSON stringben szükséges küldeni, POST metódussal a fent megadott URL-re. A tranzakció indításához az alábbi adatok szükségesek.

salt: 32 karakter hosszú random string,
merchant: a kereskedői fiók egyedi azonosítója a SimplePay rendszerben.
orderRef: a kereskedői rendszerben egyedi tranzakció azonosító
currency: a tranzakció devizaneme
customerEmail: a vásárló email címe
language: a fizetőoldal nyelve
sdkVersion: a kereskedői rendszerben a fizetés verzió száma
methods: fizetési mód tömbje
total: a tranzakció összege
timeout: a tranzakció érvényességi ideje, amíg a fizetést meg lehet kezdeni
url: redirect URL, ahova a kereskedő szeretné irányítani a vásárlót fizetés után
invoice: tomb a számlázási adatoknak

```
{
   "salt":"126dac8a12693a6475c7c24143024ef8",
   "merchant":"PUBLICTESTHUF",
   "orderRef":"101010515680292482600",
   "currency":"HUF",
   "customerEmail":"sdk_test@otpmobil.com",
   "language":"HU",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e",
   "methods":[
      "CARD"
   ],
   "total":"25",
   "timeout":"2019-09-11T19:14:08+00:00",
   "url":"https:\/\/sdk.simplepay.hu\/back.php",
   "invoice":{
      "name":"SimplePay V2 Tester",
      "company":"",
      "country":"hu",
      "state":"Budapest",
      "city":"Budapest",
      "zip":"1111",
      "address":"Address 1",
      "address2":"Address 2",
      "phone":"06203164978"
   }
}
```


Az adatokkal feltöltött JSON stringhez az "**Üzenetek validálása**" fejezetben leírt módon szükséges az aláírást kiszámolni. A fenti adatokhoz az alábbi Base64 enkódolt HASH tartozik:


```
rV2AffURYaUFMDhZgwN7fYZha0XGFCqsvBlRotCWg4MZ5e/EBZIVU3Vn8yypimPy
```


A kiszámolt aláírást az üzenet fejlécéhez kell adni a "Signature" értékeként.

```
Content-type: application/json
Signature: rV2AffURYaUFMDhZgwN7fYZha0XGFCqsvBlRotCWg4MZ5e/EBZIVU3Vn8yypimPy
```

Az üzenet body és header fenti minta szerinti létrehozása után indítható el a tranzakció POST metódussal a SimplePay „**start"** interface felé:
https://sandbox.simplepay.hu/payment/v2/start

Minden végpont hívása azonos alapokon nyugszik, azaz az üzenet adatait tartalmazó JSON string összeállításából és az ehhez tartozó Signature kiszámításából.
A Signature számítása és elküldése minden esetben megegyező módon történik, ezért a további API hívások esetén ezt külön nem tárgyalja a dokumentáció.

3.4 start – utalási tranzakció (Instant Transfer) létrehozása

A SimplePay rendszerben jelenleg bankkártyás fizetés, illetve utalás fizetési módok alkalmazhatók. A funkcióról a https://simplepay.hu/instant-transfer/ oldalon található bővebb, nem technikai információ.

A két fizetési típus között a start hívásban az egyetlen különbség a methods értéke.

Utalás esetén ez az érték „WIRE" az alábbiak szerint.

```
"methods":["WIRE"]
```

Instant Transfer esetén a „start" hívásban visszaadott URL-en az utalás adatainak a tájékoztató oldalára érkezik a vásárló.

A tranzakciót a vásárló a saját bankjának az online banki felületén, vagy a banki mobil applikációjában hagyhatja jóvá. Az utalás megtörténte után az összeg az OTP Mobil Kft. bankszámlájára kerül. Erről a SimplePay rendszere a bankkártyás fizetésnél is alkalmazott – később részeletesen tárgyalt – „**ipn**" üzenetben értesíti a kereskedői rendszert. Ennek alternatívája lehet a szintén későbbiekben leírásra kerülő „**query**" tranzakciótátusz lekérdezés használata.

Akár az „ipn", akár a „query" használata a megfelelőbb technikai megoldás a kereskedői rendszer számára a tranzakció **FINISHED** státusza az a pont, ahol teljesíthető a kifizetett megrendelés.

Instant transfer esetén is értelmezhető a refund, azaz ha a kereskedő az utalt összeget, vagy annak részösszegét vissza szeretné forgatni, akkor a „**refund**" API hívás erre is igénybe vehető.

Instant transzfer esetén nem értelmezhetők az alábbi bankkártyás fizetésekre jellemző folyamatok:

- kétlépcsős fizetés
- bankkártya regisztráció
- SCA/3DS

3.4.1 Utalás timeout beállítások

Átutalás esetén jelenleg a „timeout" változót nem szükséges küldeni a „start" hívásban. Abban az esetben, ha küldve van, akkor az értékét felülírja a kereskedői admin rendszerben a „**Technikai adatok**" fülön az „**Átutalásra várakozás banki napokban**" mező értéke. Ez a mező szabadon szerkeszthető, a kereskedői igénynek megfelelően.

Az azonnali utalás lehetőségeinek jobb kihasználása végett a fenti időtúllépési szabályok 2021 harmadik negyedévben az alábbi módon fognak változni.
Amikor a „timeout" változó nem lesz küldve a start hívásban, akkor továbbra is a kereskedői fiókban beállított banki napokban megadott érték lesz figyelembe véve a tranzakció indításakor.
Abban az esetben viszont, amikor a kereskedői rendszer küldi a „timeout" értékét, akkor a fiók szinten beállított, banki napokban értelmezett idő helyett az API-n kapott értékkel fog számolni a rendszer.

Mivel a „start" hívásban a timeout lehet akár 180 nap is, így az a leginkább rugalmas megoldás, ha minden esetben küldve van a lejárati idő, még akkor is, ha ez több nap, vagy több hét.

3.5 Fizetési előválasztó oldal

A fizetési előválasztó oldal arra szolgál, hogy kereskedői rendszerben fejlesztés nélkül legyen elérhető a bankkártyás fizetés mellett az utalásos fizetés (Instant transfer) is.
Ebben az esetben a korábban kereskedői oldalon lefejlesztett bankkártyás fizetést elindítva egy választó oldal töltődik be, ahol a vásárló meghatározhatja, hogy bankkártyával, vagy átutalással szeretne fizetni.

Mivel az utalás során a vásárló lassabban adhat meg adatokat, vagy lassabban léphet be banki felületére, ezért a fizetési előválasztó oldal csak abban az esetben jelenik meg, ha 5 percnél nagyobb timeout idő lett megadva a kereskedő által a tranzakció elindításakor.

Ha a vásárló kiválasztotta valamelyik fizetési módot az előválasztó oldalon, akkor a folyamat már a bankkártyás fizetésnek, vagy az utalásos fizetésnek megfelelően fog folytatódni.

A fizetési előválasztó oldal alkalmazásához kereskedő oldali fejlesztés nem szükséges.

3.6 Tranzakció indítás erős ügyfélhitelesítéssel, (SCA, 3DS)

A 2020 szeptember 14-én érvénybe lépő erős ügyfél hitelesítés megköveteli a kibővített tranzakciós adatok küldését. Abban az esetben, ha az alábbi adatok küldve vannak a „**start**" kommunikációban, akkor a kereskedő eleget tesz ezen szabályoknak.

Az erős ügyfélhitelesítésről a **2.** mellékletben olvasható további információ.

customerEmail: a vásárló e-mail címe
invoice (számlázási adatok) tömb, ami az alábbi elemeket tartalmazza
name számlázási név
country ország szövegesen megadva
state: megye szövegesen megadva
city: város
zip: irányítószám
address: cím

threeDSReqAuthMethod: (opcionális) a vásárló regisztrációs módja a kereskedői rendszerben
lehetséges értékek:
01: vendég
02: kereskedőnél regisztrált
05: harmadik feles azonosítóval regisztrált (Google, Facebook, account stb.)
address2: **(opcionális)** második címsor
phone: **(opcionális)** telefonszám

A fenti adatokat a SimplePay rendszere továbbítja az fizetés elindításakor. A bank ezeket figyelembe veszi a tranzakció engedélyezésénél, emiatt a tranzakció sikeressége érdekében kiemelten fontos a küldésük és az adatok valódisága!

Abban az esetben, **ha a kereskedői rendszerben nem ismert a vásárló e-mail adata**, akkor a „**maySelectEmail**" változó hatására a fizetőoldalon is meg tudja adni ezt a vásárló.

```
"maySelectEmail":true,
```

Abban az esetben, **ha a kereskedői rendszerben nem ismertek a vásárló számlázási adatai**, akkor a „**maySelectInvoice**" változó hatására a fizetőoldalon is meg tudja adni ezt a vásárló.

```
"maySelectInvoice":true,
```

3.7 3DS challenge

A 3DS challenge során a kártyakibocsátó bank interaktívan szeretné beazonosítani a kártya birtokosát az adott tranzakció során.
A folyamatban éles működés esetén SMS-ben küld a bank a kártyatulajdonosnak egy egyszer használható azonosítót, amit a banki felületen kell megadjon. Ha az azonosító kód megfelelő, akkor folytatódik a fizetési folyamat.

A sandbox fizetőoldalon szimuláva van a 3DS challenge folyamat. Annak érdekében, hogy a tranzakció során 3DS challenge történjen a sandbox legördülő kártyalistájából a 3DS folyamathoz megjelölt kártyát kell kiválasztani a teszt fizetéskor.

Ebben az esetben nem történik meg azonnal a fizetés hanem előbb átirányításra kerülünk a banki kódbekérő oldal szimulációjára. Az **éles** működés esetén itt (vagy ennek megfelelő kártyakibocsátó banki felületen) lehet megadni az SMS-ben kapott kódot.
A **sandbox** 3DS Challenge szimuláció esetén **NEM küld** a rendszer **SMS-ben kódot**, hanem két konstans érték megadására van lehetőség:

- **1234** a sikeres teszt folyamathoz
- **1111** a sikertelen teszt folyamathoz

3.8 További változók

A „start" hívás további elemekkel egészíthető ki. Ezek küldése opcionális.

items: kifizetendő tételek listája. A tételek összeadódnak és ebből számítja ki a rendszer a kifizetendő összeget.

- **ref:** a termék azonosítója a kereskedő rendszerében
- **title:** a termék neve
- **desc:** a termék leírása
- **amount:** rendelt mennyiség, egész szám, kötelező
- **price:** egységár, nullánál nagyobb számérték, kötelező
- **tax:** ÁFA százalék egész számmal megadva, pl. 27. Ha nulla (0) értékkel van küldve, akkor nincs ÁFA számítás, azaz ebben az esetben ez bruttó ár. Ha a mező nincs küldve, akkor alapértelmezett 0 értékkel kerül beállításra.

Abban az esetben, ha a „**total**" és az „**items**" is küldve van egy tranzakcióban, akkor a „**total**" nem lesz figyelembe véve.

```
"items":[
   {
      "ref":"Product ID 2",
      "title":"Product name 2",
      "desc":"Product description 2",
      "amount":"2",
      "price":"5",
      "tax":"0"
   }
],
```

shippingCost: szállítási költség összege a fiók devizanemében, ami hozzáadódik a kifizetendő összeghez. 12 HUF esetén az alábbi módon lehet megadni.

```
"shippingCost":12,
```

discount: árengedmény összege a fiók devizanemében, ami kivonásra kerül a kifizetendő összegből. 5 HUF esetén az alábbi módon lehet megadni.

```
"discount":"5",
```

customer: a vásárló neve, ha eltér az „invoice" / „name" értékétől

```
"customer":"V2 Test User",
```

urls: ha nem egy közös URL-en, hanem az eseménynek megfelelő külön helyen dolgozza fel a kereskedő rendszere az authorizáció eredményét, akkor itt minden eseményhez külön URL adható meg.
A lehetséges események:

- **success**: sikeres authorizáció
- **fail**: sikertelen authorizáció
- **cancel**: megszakított tranzakció
- **timeout**: időtúllépés

```
"urls":{
   "success":"https://sdk.simplepay.hu/success.php",
   "fail":"https://sdk.simplepay.hu/fail.php",
   "cancel":"https://sdk.simplepay.hu/cancel.php",
   "timeout":"https://sdk.simplepay.hu/timeout.php"
},
```

Abban az esetben, ha az „**url**" és az „**urls**" is küldve van egy tranzakcióban, akkor a „**url**" nem lesz figyelembe véve.

twoStep: kétlépcsős tranzakciók esetén küldhető. Ha false az értéke, akkor a tranzakció nem vár a „finish" hívásra, hanem azonnali teljes összegű terhelést vált ki. Ezzel a kétlépcsős fiókokon is lehet azonnali terhelést indítani.

```
"twoStep":false,
```

delivery: szállítási adatok.

```
"delivery":{
   "name":"SimplePay V2 Tester",
   "company":"",
   "country":"hu",
   "state":"Budapest",
   "city":"Budapest",
   "zip":"1111",
   "address":"Delivery address",
   "address2":"",
   "phone":"06203164978"
},
```

maySelectEmail: abban az esetben, **ha a kereskedői rendszerben nem ismert a vásárló e-mail adata**, akkor a „**maySelectEmail**" változó hatására a fizetőoldalon is meg tudja adni ezt a vásárló.

```
"maySelectEmail":true,
```

maySelectInvoice: bban az esetben, **ha a kereskedői rendszerben nem ismertek a vásárló számlázási adatai**, akkor a „**maySelectInvoice**" változó hatására a fizetőoldalon is meg tudja adni ezt a vásárló.

```
"maySelectInvoice":true,
```

maySelectDelivery: ha nem áll rendelkezésére a kereskedőnek a vásárló szállítási adatai, akkor a SimplePay fizetőoldala is be tudja kérni. Ehhez a változó küldése esetén fel kell sorolni azon országokat, ahova a kereskedő szállít. A fizetőoldalon csak a megadott országokból választhat a vásárló.

```
"maySelectDelivery":[
   "HU",
   "AT",
   "DE"
],
```

3.9 start – response

A start hívásra szinkron választ ad vissza a SimplePay rendszere. A válasz header is tartalmaz Signature értéket. Kereskedői oldalon ezt minden esetben ajánlott ellenőrizni. A válasz body egy JSON string, ami az alábbiakat tartalmazza:

salt: 32 karakter hosszú random string
merchant: a kereskedő SimplePay fiókja, amiben a tranzakció létrejött
orderRef: a start hívásban megadott orderRef értéke
currency: a start hívásban megadott currency értéke
transactionId: a létrejött SimplePay tranzakció azonosítója
timeout: az időhatár, amíg elindítható a fizetés

total: a tranzakció összege
paymentUrl: az az URL, ahova a kereskedő rendszerének a vásárlót át kell irányítani, hogy a fizetést elvégezhesse.


```
{
   "salt":"KAC6ZRUacmQit98nFKOpjXgkwdC0Grzl",
   "merchant":"PUBLICTESTHUF",
   "orderRef":"101010515680292482600",
   "currency":"HUF",
   "transactionId":99844942,
   "timeout":"2019-09-11T21:14:08+02:00",
   "total":25.0,
   "paymentUrl":"https://sandbox.simplepay.hu/pay/pay/pspHU/8f4oKRec5R1B696xlxbOcj1jRhhABA2pwS
LQDPW60zoGSDWzDU"
}
```


A **paymentUrl** értékét több módon lehet arra használni, hogy a vásárlót a fizetőoldalra irányítsuk a kereskedői oldalról. Ezek közül a legegyszerűbb egy gomb megjelenítése. ha a gombot megnyomja a vásárló, akkor megtörténik az átirányítás a fizetőoldalra.

```
<form action="https://sandbox.simplepay.hu/pay/pay/pspHU/8f4oKRec5R1B696xlxbOcj1jRhhABA2pwSLQD
PW60zoGSDWzDU" method="POST" id="SimplePayForm" accept-charset="UTF-8">
          <button type="submit">Start SimplePay Payment</button>
</form>
```

3.10 Sikertelen eredményű API hívás

Abban az esetben ha az API a hívást valamilyen okból nem tudja végrehajtani, akkor a válaszban megjelenik az „**errorCodes**" tömb, ami tartalmazza a hibakódokat.


```
{
   "errorCodes":[
      5321
   ]
}
```


Az „**errorCodes**" az összes API végponton megegyező módon adja vissza az aktuális hibakódot, így a továbbiakban a dokumentáció nem tárgyalja az egyes funkciók megvalósításánál.

3.11 Fizetőoldalra átirányítás előtti kereskedő oldali tennivalók

Az fizetés elindítása előtt a kereskedői weboldalon **meg kell jeleníteni a SimplePay logót**. A logo elhelyezését jelen dokumentáció „**Logók és tájékoztatások**" fejezete tárgyalja.

Továbbá a fizetés elindítása előtt **meg kell jeleníteni és el kell fogadtatni a vásárlóval az adattovábbítási nyilatkozatot**. A nyilatkozat tartalmi és megjelenítési kritériumait jelen dokumentáció „**Adatovábbítási nyilatkozat**" c. fejezete tárgyalja.

3.12 SimplePay fizetőoldal

Eddig a "**start**" funkcióval létre lett hozva a SimplePay rendszerében a fizetési tranzakció. A kereskedői rendszer a SimplePay API-tól kapott egy URL-t, amire át kell irányítani a vásárlót. Az URL-en a SimplePay fizetőoldal töltődik be a létrejött tranzakció adataival.

A fizetőoldal megjeleníti a korábban megadott fizetési tranzakció adatait, a kereskedőt, akinek a vásárló fizet, illetve az összes szükséges tájékoztatást. A vásárlói adatok és a termékek adatait megjelenítő panel a kereskedői fiókban a technikai beállítások között ki- és bekapcsolható.

A sandbox rendszeren a kártyaszám mezője valójában egy legördülő menü, ahol ki lehet választani teszt kártyákat a sikeres és sikertelen fizetések szimulálásához. A fizetés kimenete a választott kártyától függ.
Az éles rendszeren itt a kártyaszám adható meg.

A "FIZETEK" gombra kattintva indul el a tranzakció. Ekkor megtörténik a kártya authorizáció, ami után a vásárló vissza lesz irányítva a kereskedő weboldalára.
Az éles fizetőoldalon lehet használni a korábban a Simple alkalmazásban regisztrált bankkártyákat is, illetve el lehet végezni a regisztrációt, valamint további funkciók is elérhetők.

A **sandbox fizetőoldalon** azok a **funkciók nem érhetők el**, amikhez valós banki adatok, vagy olyan külső rendszerek szükségesek, amik a SimplePay számára csak éles üzemben érhetők el.
Ilyenek a

- Simple appban regisztrált kártyák
- Simple mobil applikáció
- QR kódos fizetés
- Push payment
- 3DS SMS ellenőrző kódok
- Installment

A fenti funkciók a kereskedői rendszer által nem befolyásolhatók, emiatt fejlesztést nem igényelnek.

A fizetőoldal szolgáltatásait teljeskörűen a következő dokumentáció mutatja be:
http://simplepartner.hu/download.php?target=paymentflowhu

3.13 back - tájékoztatás a kereskedő weboldalán

A fizetőoldalról a böngészőben vissza van irányítva a vásárló a kereskedői weboldalra. A visszairányítás a start hívásban megadott URL-re történik. Ebből adódóan a "back" nem az API funkciója, hanem a kereskedői rendszer része, mivel a böngészőben történt tranzakciónak **a vásárló által látható része a kereskedő weboldalán ér véget**.

Ez a hívás a böngészőben történik GET metódussal. Az indításkor megadott URL-hez két változót fűz hozzá a SimplePay rendszere.

“r” változó
Az “r” változó (response) a fizetés eredményét és részleteit tartalmazza. A tartalom egy Base64 enkódolt JSON string.

“s” változó
Az “s” változó (signature) a JSON string aláírása. Az aláírás validálása a böngészőben történő híváskor ajánlott.

A korábban szemléltetett minta tranzakció esetén az alábbi adatokkal kiegészítve van vissza irányítva a vásárló a kereskedő oldalán található URL-re. Az URL korábban, a „**start**” kommunikáció során lett megadva az „**url**” változóban, vagy az „**urls**” egyik értékében.

```
https://sdk.simplepay.hu/back.php?r=eyJyIjowLCJ0Ijo5OTg0NDk0MiwiZSI6IlNVQ0NFU1MiLCJtIjoiUFVCTElDVEVTVEhVRiIsIm8iOiIxMDEwMTA1MTU2ODAyOTI0ODI2MDAifQ%3D%3D&s=El%2Fnvex9TjgjuORI63gEu5I5miGo4CSAD5lmEpKIxp7WuVRq6bBeh1QdyEvVGSsi
```

Az **“r”** változó tartalma Base64 dekódolás után az alábbi JSON string:

```
{
   "r":0,
   "t":99844942,
   "e":"SUCCESS",
   "m":"PUBLICTESTHUF",
   "o":"101010515680292482600"
}
```

A JSON elemei a következők:
r: válaszkód (response code)
t: tranzakció SimplePay azonosítója (transaction id)
e: esemény (event)
m: kereskedői fiók azonosítója (merchant)
o: kereskedői tranzakció azonosító (order id)

A fizetés négyféle eseménnyel érkezhet vissza a kereskedői oldalra. Ezek nem feltétlenül azonosak a tranzakció aktuális státuszával:
success: sikeres authorizáció
fail: sikertelen authorizáció
timeout: időtúllépés, ha nem lett a megadott ideig elindítva a fizetés
cancel: megszakított fizetés, bezárt böngésző, elnavigálás a fizetőoldalról

Abban az esetben, ha nem sikeres a fizetés, akkor a válaszkód tartalmazza a probléma hibakódját.

3.14 A tranzakció eredményétől függő tájékoztatások

A böngészőben a vásárlónak megfelelő tájékoztatást kell nyújtani a tranzakció eredményéről.

3.14.1 Megszakított fizetés

Ha valami miatt a fizetőoldalról a vásárló vissza szeretne lépni, akkor megnyomhatja a "**vissza a kereskedő oldalára**" gombot. A gomb megnyomásával a SimplePay fizetőoldalról vissza lesz irányítva a tranzakció indításakor erre a célra megadott kereskedői weboldalra. A visszairányítás az "**url**", változóban, vagy az "**urls**" tömb "**cancel**" elemében megadott URL-re történik.

Nagyon fontos figyelembe venni a tájékoztatásnál, hogy ebben az esetben nem történt fizetés, hiszen még a kártyaadatok megadása és a fizetés elindítása előtt lett megszakítva a fizetés. **Ebből adódóan nem lehet a vásárlót itt sikertelen fizetésről tájékoztatni.**

Minta tájékoztatás megszakított fizetés esetére
A vásárlót megfelelően tájékoztatni kell, hogy miért került vissza a fizetőoldalról a kereskedői weboldalra, például az alábbi lehetőségek közül valamelyikkel.

Ön megszakította a fizetést

vagy

Megszakított fizetés

3.14.2 Időtúllépés

Időtúllépés esetén a tranzakció elindítása után a vásárló megérkezik a SimplePay fizetőoldalra, azonban nem ad meg kártyaadatot és nem indítja el a fizetést a „**start**" hívásban megadott határidőig.
A fizetésre felhasználható időt (a fizetés határidejét) a kereskedő rendszere adja át a tranzakció létrehozásakor a "**timeout**" változó értékeként.

Ilyen esetben az idő túllépésekor a fizetőoldal automatikusan visszairányítja a vásárlót a tranzakció indításakor megadott kereskedői weboldalra. A visszairányítás az "**url**", változóban, vagy az "**urls**" tömb "**timeout**" elemében megadott URL-re történik.

Nagyon fontos figyelembe venni a tájékoztatásnál, hogy ebben az esetben nem történt fizetés, hiszen még a kártyaadatok megadása és a fizetés elindítása előtt lett túllépve a tranzakció elindítására megadott idő. **Ebből adódóan nem lehet a vásárlót itt sikertelen fizetésről tájékoztatni.**

Minta tájékoztatás időtúllépés esetére
A vásárlót megfelelően tájékoztatni kell, hogy miért került vissza a fizetőoldalról a kereskedői weboldalra, például az alábbi lehetőségek közül valamelyikkel.

Ön túllépte a tranzakció elindításának lehetséges maximális idejét.

vagy

Időtúllépés

3.14.3 Sikertelen bankkártyás fizetés

Ebben az esetben a vásárló a kártyaadatai megadása után a „Fizetek" gombra kattintva elindítja a tranzakciót, amit a kártyáját kibocsátó bank elutasít, így sikertelen lesz a fizetés. A sikertelen fizetés után a kereskedőnek tájékoztatnia kell a vásárlót.

A sikertelenség alapvető oka lehet, hogy nem megfelelő adatokat adott meg a vásárló a fizetőoldalon, pl. a CVC/CVV értékét elgépelte.

Helyes adatok megadása esetén is lehetséges, hogy nincs megfelelő fedezet a kártyán az adott tranzakcióhoz. Továbbá lehet, hogy van fedezet, de a vásárló elérte az általa beállított napi limitet, illetve megtörténhet, hogy a vásárló kártyája már nem létezik, lejárt, vagy le lett tiltva.

Ezeket a problémákat a kereskedő nem tudja kezelni, mivel a kártyával és a fizetéssel kapcsolatos információt csak a kártya tulajdonosának ad a kártyát kibocsátó bank.

Csak a kártya tulajdonosa tudja megoldani, vagy kezdeményezni a megoldását a kártyáját kibocsátó banknál, pl. internetbankján keresztül meg tudja emelni kártyájának a napi limitjét és ismét meg tudja próbálni a fizetést.

Ha sikertelen fizetésnél a vásárló nem tanácstalan, hogy mi történhetett, hanem a megfelelő tájékoztatás hatására megpróbálja megoldani a problémát, akkor jóval kevesebb sikertelen tranzakció történik.

A fentiek miatt kiemelten fontos, hogy ilyen esetben megfelelő tájékoztatást adjon a weboldal.
A tájékoztatásnak két lényeges eleme van:

- a tranzakció SimplePay azonosítója, ami alapján ügyfélszolgálat is tud segíteni konkrét tranzakcióval kapcsolatos kérdések esetén
- az egyértelmű tájékoztató, ami a vásárlót a probléma megoldása felé irányítja

Minta tájékoztatás sikertelen fizetés esetére

Sikertelen tranzakció. SimplePay tranzakció azonosító: 1xxxxxxxx Kérjük, ellenőrizze a tranzakció során megadott adatok helyességét. Amennyiben minden adatot helyesen adott meg, a visszautasítás okának kivizsgálása érdekében kérjük, szíveskedjen kapcsolatba lépni kártyakibocsátó bankjával.

3.14.4 Sikeres bankkártyás fizetés

Sikeresség esetén a kártyaadatok megadása után megtörténik az authorizáció, majd a vásárló vissza van irányítva a kereskedő weboldalára.

Ebben az esetben tájékoztatásként elegendő a SimplePay tranzakció azonosító megjelenítése.

Minta tájékoztatás sikeres fizetés esetére

Sikeres tranzakció. SimplePay tranzakció azonosító: 1xxxxxxxx

3.15 ipn - értesítés a tranzakció státuszokról és teljesíthetőségéről

IPN küldés az alábbi SimplePay IP címről történik:
94.199.53.96

Az Instant Payment Notification (**ipn**) során a fizetési tranzakció végstátuszáról tájékoztatja a kereskedői rendszert a SimplePay. A SimplePay szervere POST metódussal küldi az üzenetet a kereskedő részére.

Korábban csak a sikeres fizetésről, a refundról, valamint a kétlépcsős fizetések esetén a preauthorizációról küldött a rendszer értesítést, azaz olyan esetekről, ahol pénzügyileg is értelmezhető státuszváltás történt.

2021.06.15. után a **sikertelen végstátuszok esetén** is küldhet a rendszer IPN üzenetet. Ez a funkció **opcionális**, ami csak kereskedői beállításra történik meg. Azon kereskedői rendszerek esetén, ahol korábban implementálva volt a funkció a sikeres státuszokra, **nem igényel kötelező jellegű változtatást**.
A beállítást a kereskedői admin felületen a „Technikai adatok" oldalon a „Rendszer értesítések" panelen lehet megtenni.

IPN üzenet az alábbi végstátuszok esetén értelmezhető

Státusz	**Esemény**
FINISHED	Sikeres terhelés esetén a tranzakció lezárása. Ez alapján értesülhet a kereskedői rendszer arról, hogy teljesítheti a megrendelést
AUTHORIZED	Kétlépcsős fizetés esetén az összeg befoglalásakor, vagy preauthorizáció
REFUND	Terhelt összeg visszatérítésekor
REVERSED	Kétlépcsős fizetés esetén az összeg feloldásakor
CANCELLED	Megszakított fizetés esetén
TIMEOUT	Időtúllépés esetén

Az IPN üzenetet **fogadni kell a kereskedőnek és megfelelően válaszolni kell rá**. A válasz alapján értesül a SimplePay rendszere arról, hogy **a kereskedő** fogadta az üzenetet, azaz a kereskedő **értesült a tranzakció állapotában bekövetkezett változásról.**

Az IPN üzenet fogadásához meg kell határozni azt az URL-t, ahol a kereskedő rendszere várja a hívást. Az URL az az online elérhető cím, ahol a kereskedő fogadja a SimplePay rendszer által küldött IPN üzenetet.

Az IPN URL beállítását a kereskedői vezérlőpanelen lehet elvégezni.
https://sandbox.simplepay.hu/admin/

A címet a „**Technikai adatok**" menüpont alatt lehet beállítani. Az IPN beállítását minden kereskedői fiókon el kell végezni. Ha a kereskedő több fiókot használ egy domainen (pl. HUF és EUR devizanemben is lehet fizetni), akkor minden esetben (fiókban) meg kell adni az URL-t.

Az IPN URL esetén az alábbiakra kell figyelmet fordítani
- publikusan elérhetőnek kell legyen
- ne legyen .htpassword, vagy bármilyen védelem rajta
- átirányítási szabályok, SSL beállítások ne akadályozzák az elérését

Ha nem volt sikeres az IPN üzenet fogadása vagy a visszajelzés nem megfelelő a megadott URL-en, akkor a SimplePay úgy veszi, hogy a kereskedő **NEM** értesült a státuszváltásról. Ebben az esetben automatikusan ismétli a rendszer az üzenet kiküldését.
Az első sikertelen küldésről e-mail útján automatikusan értesíti a rendszer a kereskedőt.

A sikertelen fogadás oka lehet, ha nem lehetett elérni az IPN üzenettel a kereskedő rendszerét (HTTP 200-tól eltérő állapotkód), vagy nem megfelelő választ adott a kereskedő.

Az ipn timeout ideje 20 másodperc. Ennek túllépése akkor fordulhat elő, ha a válasz a kereskedő felől az IPN üzenetre ennyi ideig nem történik meg a hívás fogadása után.

Ha bármilyen okból **sikertelen** az **első IPN**, az **újra küldés** a következő időpontokban történik **az első küldés után**:
- 5, 10, 15, 30, 45 perccel
- 1, 2, 3, 6, 9, 12, 18 órával
- 1, 1.5, 2, 2.5, 3 nappal

Ezek után a szerver a továbbiakban nem küldi ki az IPN üzenetet. Ugyanakkor a kereskedő az admin felületen manuálisan ezután is bármikor elindíthatja a küldést.

Az alábbi minta a rendszer által kiküldött IPN üzenet tartalmát szemlélteti. Az üzenet headerben megtalálható a Signature változó, aminek értéke alapján hitelesíteni kell a kapott üzenetet.


```
{
   "salt":"223G0O18VAqdLhQYbJz73adT36YzLtak",
   "orderRef":"101010515680292482600",
   "method":"CARD",
   "merchant":"PUBLICTESTHUF",
   "finishDate":"2019-09-09T14:46:18+0200",
   "paymentDate":"2019-09-09T14:41:13+0200",
   "transactionId":99844942,
   "status":"FINISHED"
}
```


A válaszban a fenti adatokat kell visszaküldeni, kiegészítve az IPN üzenet fogadásának időpontjával. Az időpontot a "**receiveDate**" mezőnek kell tartalmazza.

A válasz fejlécében a korábban leírt módon itt is szükséges küldeni a Signature változót. A Signature értékét a "**receiveDate**" értékkel kiegészített válasz JSON stringre kell számítani.


```
{
   "salt":"223G0O18VAqdLhQYbJz73adT36YzLtak",
   "orderRef":"101010515680292482600",
   "method":"CARD",
   "merchant":"PUBLICTESTHUF",
   "finishDate":"2019-09-09T14:46:18+0200",
   "paymentDate":"2019-09-09T14:41:13+0200",
   "transactionId":99844942,
   "status":"FINISHED",
   "receiveDate":"2019-09-09T14:46:20+0200"
}
```


3.16 finish - kétlépcsős tranzakciók lezárása

A finish hívást az API az alábbi URL-en várja:
https://sandbox.simplepay.hu/payment/v2/finish

A korábban szemléltetett tranzakciók esetén **egylépcsős** fizetés történt. Egylépcsős fizetésnél amikor a vásárló a fizetőoldalon megadja a bankkártya adatait, akkor elindul az authorizáció. Abban az esetben, ha ez sikeres (adatok helyesen vannak megadva, a kártyán van megfelelő fedezet a fizetési tranzakció végrehajtásához), akkor a **terhelés** azonnal megtörténik.

A **kétlépcsős** tranzakció esetén viszont az **authorizáció után** a megadott összeg csak be van foglalva, de **a terhelés nem történik** meg. Ez az első lépcső.
A terhelés elindításához egy külön lezáró hívásra van szükség, amit a finish funkcióval lehet kezdeményezni. Ez a második lépcső.

A **finish** hívással lehet lezárni azokat a kétlépcsős tranzakciókat, amik esetén még csak az összeg befoglalása (azaz az első lépcső) történt meg. A lezárásra 21 naptári napja van a kereskedőnek. Ha addig nem történik meg a terhelés, vagy a feloldás, akkor a SimplePay rendszere automatikusan feloldja a vásárló kártyáján befoglalt összeget.

Ennek a fizetési folyamatnak ott van létjogosultsága, ahol meg kell bizonyosodni a vásárló kártyájának adott összegű terhelhetőségéről (első lépcső), de a megrendelés végrehajtásához a kereskedőnek még további lépéseket kell tegyen, aminek eredményét ekkor még nem látja, emiatt nem akar azonnal terhelni.
Ennek oka lehet például raktárkészlet ellenőrzés, beszállítókkal való egyeztetés, vagy más online rendszerekben szükséges kereskedői rendelések feladása stb.

A finish hívással zárható le a tranzakció. A lezárás megtörténhet a befoglalt teljes összeggel, vagy annál kisebb összeggel is, ha csak részlegesen teljesít. Abban az esetben, ha nem teljesít, akkor nulla összeggel is lezárható, azaz ilyenkor feloldja a teljes befoglalt összeget a vásárló kártyáján.

A kétlépcsős fizetés lehetőségét a SimplePay IT support tudja bekapcsolni a sandbox rendszeren. A szolgáltatás aktiválásától kezdve a kereskedő tud egy- és kétlépcsős tranzakciókat is indítani.

A korábban a start funkció leírásának megfelelő indítás esetén kétlépcsős fizetések fognak indulni, azaz az authorizáció után a lezárásra fog várni a tranzakció.

Kétlépcsős fiók esetén kiegészíthető a „start" hívás a „**twoStep**" paraméterrel.

Kétlépcsős fiókon abban az esetben, ha a twoStep értéke **true**, vagy nincs twoStep átadva, akkor kétlépcsős tranzakció indul. Ha át van adva és az értéke **false**, akkor a sikeres authorizáció (első lépcső) után automatikusan el fog indulni a terhelés (második lépcső) is a „finish" hívás nélkül.
A twoStep változó segítségével egy kereskedői fiókon lehet párhuzamosan normál kétlépcsős, illetve olyan tranzakciókat is indítani, ahol azonnal megtörténik a terhelés.


```
{
   "salt":"67b253f7f7ee8c264f01b16e3bcb9611",
   "merchant":"PUBLICTESTHUF",
   "orderRef":"101010515680496082852",
   "currency":"HUF",
   "customerEmail":"sdk_test@otpmobil.com",
   "language":"HU",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e",
   "methods":[
      "CARD"
   ],
   "total":"25",
   "twoStep":true,
   "timeout":"2019-09-12T00:53:28+00:00",
   "url":"https:\/\/sdk.simplepay.hu\/back.php",
   "invoice":{
      "name":"SimplePay V2 Tester",
```

```
        "company":"",
        "country":"hu",
        "state":"Budapest",
        "city":"Budapest",
        "zip":"1111",
        "address":"Address 1",
        "address2":"",
        "phone":"06203164978"
    }
}
```


A tranzakció a sikeres authorizáció után **AUTHORIZED** státuszban marad. Ekkor a fizetendő összeg be van foglalva a vásárló kártyáján, de terhelés még nem történt meg. A terhelés elindításához a finish hívást az alábbi adatokkal kell elindítani.

A hívás kulcs adatai az **originalTotal** és az **approveTotal** értékei.
originalTotal: a tranzakció során eredetileg befoglalt összeg
approveTotal: a valójában terhelni kívánt összeg

Az **originalTotal** értéke csak pontosan a befoglalt összeg lehet.
Az **approveTotal** értéke három módon adható meg

1. Teljes befoglalt összeg. Ekkor az originalTotal összege terhelődik.
2. A teljes összegnél kisebb, de nullánál nagyobb összeg. Ekkor az itt megadott összeg terhelődik, a fennmaradó pedig felszabadul a vásárló kártyáján.
3. Nulla érték. Ekkor a teljes befoglalt összeg felszabadul a vásárló kártyáján.


```
{
   "salt":"a182f12e696d483985133e299c245b83",
   "merchant":"PUBLICTESTHUF",
   "orderRef":"101010515680496082852",
   "originalTotal":"25",
   "approveTotal":"15",
   "currency":"HUF",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e"
}
```


A **transactionId** és az **orderRef** közül legalább az egyiket meg kell adni, hogy a tranzakció azonosítható legyen.

3.17 refund - visszatérítések kezelése

A refund hívást az API az alábbi URL-en várja:
https://sandbox.simplepay.hu/payment/v2/refund

A refund alkalmazásával korábbi terhelt tranzakciók összege adható vissza részlegesen, vagy teljesen. Az összeg nullánál nagyobb értékű kell legyen. A maximum érték a korábban terhelt összeg lehet.

Ha részösszeg lett visszaadva, akkor az eredeti tranzakció fentmaradó összegére további refund is indítható mindaddig, amíg a visszaadott részösszegek együtt el nem érik az eredeti terhelés összegét. Minden refund kérés külön tranzakcióként fut, ami az eredeti tranzakcióhoz kapcsolódik.

A hívás elindításakor az alap adatokon túl az eredeti terhelés tranzakció azonosítóját, a visszatérítendő összeget és a devizanemet kell megadni az alábbi módon. A tranzakció azonosítója lehet a kereskedői tranzakció azonosító, vagy a SimplePay tranzakció azonosító is.


```
{
   "salt":"6a85ef475fa491618a94af9bb0b2065d",
   "orderRef":"101010515680496082852",
   "merchant":"PUBLICTESTHUF",
   "currency":"HUF",
   "refundTotal":5,
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e"
}
```


A hívásra adott válasz három kulcs mezője:

- **refundTransactionId**: a visszatérítés tranzakció azonosítója
- **refundTotal**: a visszatérített összeg
- **remainingTotal**: a visszatérítés után az eredeti tranzakció fennmaradó összege (eredeti terhelésből az eddigi visszatérítések kivonva)


```
{
   "salt":"WZ7Ncc0qoDSMYG4twsmeOdBs6PSnsj1Z",
   "merchant":"PUBLICTESTHUF",
   "orderRef":"101010515680496082852",
   "currency":"HUF",
   "transactionId":99826864,
   "refundTransactionId":99826879,
   "refundTotal":5.0,
   "remainingTotal":10.0
}
```


3.18 query - tranzakció adatok lekérdezése

A query hívást az API az alábbi URL-en várja:
https://sandbox.simplepay.hu/payment/v2/query

A **query** segítségével konkrét tranzakció adatai kérdezhetők le a SimplePay rendszeréből. A POST metódussal küldött kérésre szinkron választ ad az API.

A lekérdezés alapja a tranzakció egyedi azonosítója. Megadható az egyedi kereskedői azonosító (**orderRef**), vagy a SimplePay tranzakció azonosító (**transactionId**).

A legegyszerűbb a lekérdezés indítás a SimplePay tranzakció azonosító alapján.


```
{
   "merchant":"PUBLICTESTHUF",
   "transactionIds":[
      "99325412"
   ],
   "salt":"c4de372a465b56aa0187b8482570a2cd",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e"
}
```

A query hívás használatával több tranzakció adatai is kinyerhetők egy lekérdezés során. Ebben az esetben a **transactionIds** listában, több tranzakció azonosítót is meg lehet adni.


```
{
   "merchant":"PUBLICTESTHUF",
   "transactionIds":[
      "99325521",
      "99325516"
   ],
   "salt":"42c938bdaad569154753eb63697f9918",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e"
}
```


A kereskedői azonosító (**orderRef**) és a SimplePay azonosító (**transactionId**) vegyesen is használható egy lekérdezésben. Ekkor a kétféle adat alapján kikeresi a rendszer mindegyik tranzakciót. Egyszerre többet is meg lehet adni mindkét típusú azonosítóból.


```
{
   "merchant":"PUBLICTESTHUF",
   "orderRefs":[
      "101010515383930534733"
   ],
   "transactionIds":[
      "99325521",
      "99325516"
   ],
   "salt":"d2a919eeb8746453777e790c9ab09377",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e"
}
```


A válaszban a transactions lista fogja tartalmazni a lekérdezett tranzakciók adatait.


```
{
   "salt":"sNWsKj5sDsQUSlc4LqjliAH3unCG4Mt3",
   "merchant":"PUBLICTESTHUF",
   "transactions":[
      {
         "salt":"qnoc9TsW1fCSA3ynsM2vCfPpeTcny1eJ",
         "merchant":"PUBLICTESTHUF",
         "orderRef":"101010515383930534733",
         "total":25,
         "transactionId":99325535,
         "status":"FINISHED",
         "resultCode":"OK",
         "remainingTotal":0,
         "paymentDate":"2018-10-01T13:24:14+02:00",
         "finishDate":"2018-10-01T13:24:34+02:00",
         "method":"CARD"
      },
      {
         "salt":"5sOUArueyVvLHi7zELIkgyy6BczwWl5o",
         "merchant":"PUBLICTESTHUF",
         "orderRef":"101010515383924753263",
         "total":25,
         "transactionId":99325521,
         "status":"INIT",
         "remainingTotal":0,
         "paymentDate":"2018-10-01T13:14:36+02:00",
         "method":"CARD"
      },
      {
         "salt":"KwrgINkzKpUzqOgFtwZaDnSTxZExQD7k",
         "merchant":"PUBLICTESTHUF",
         "orderRef":"101010515383923675972",
```

```
            "total":25,
            "transactionId":99325516,
            "status":"FINISHED",
            "resultCode":"OK",
            "remainingTotal":0,
            "paymentDate":"2018-10-01T13:12:47+02:00",
            "finishDate":"2018-10-01T13:14:39+02:00",
            "method":"CARD"
         }
      ],
      "totalCount":3
}
```


A **totalCount** értékeben található meg az eredményben visszaadott tranzakciók darabszámát.

A lekérdezés kiegészíthető a „**detailed**" paraméterrel, aminek hatására a tranzakció adatainak részletei is szerepelnek a válaszban.


```
{
   "merchant":"PUBLICTESTHUF",
   "detailed":true,
   "transactionIds":[
      "99325516"
   ],
   "salt":"94fe1c712565697d732a7bce510b9146",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e"
}
```


Részletes válasz


```
{
   "salt":"O99RQ1cf2K4nZaxWA50v6ySavi6DiwFa",
   "merchant":"PUBLICTESTHUF",
   "transactions":[
      {
         "salt":"FNoR545Lw2kUQdQOXXy7sJWPBGu4h5T9",
         "merchant":"PUBLICTESTHUF",
         "orderRef":"101010515383923675972",
         "currency":"HUF",
         "customer":"v2 START Tester",
         "customerEmail":"sdk_test@otpmobil.com",
         "language":"HU",
         "twoStep":false,
         "total":15.0,
         "shippingCost":0.0,
         "discount":0.0,
         "invoice":{
            "company":"",
            "country":"hu",
            "state":"Budapest",
            "city":"Budapest",
            "zip":"1111",
            "address":"Address 1",
            "address2":"",
            "phone":"06203164978",
            "lname":"SimplePay V2 Tester"
         },
         "delivery":{
            "company":"",
            "country":"hu",
            "state":"Budapest",
            "city":"Budapest",
            "zip":"1111",
            "address":"Address 1",
            "address2":"",
            "phone":"06203164978",
```

```
            "lname":"SimplePay V2 Tester"
         },
         "transactionId":99325516,
         "status":"FINISHED",
         "resultCode":"OK",
         "remainingTotal":0.0,
         "paymentDate":"2018-10-01T13:12:47+02:00",
         "finishDate":"2018-10-01T13:14:39+02:00",
         "method":"CARD"
      }
   ],
   "totalCount":1
}
```


A lekérdezés kiegészíthető a „**refunds**” paraméterrel, aminek hatására a tranzakció adatai mellett megkapható a rájuk indított visszatérítések adatait is.


```
{
   "merchant":"PUBLICTESTHUF",
   "refunds":true,
   "orderRefs":[
      "101010515384686499284"
   ],
   "transactionIds":[
      "99326020"
   ],
   "salt":"ac17ee8c6e33f89cf7505e40decd33ed",
   "sdkVersion":"SimplePayV2.1_Payment_PHP_SDK_2.0.7_190701:dd236896400d7463677a82a47f53e36e"
}
```


A visszatérítések minden esetben különálló tranzakcióként vannak végrehajtva. A visszatérítések az adott tranzakción belül a „**refunds**” listában találhatók meg.


```
{
   "salt":"QxQmqOfkqV9khWU6SHJxlKmYyuN74x1E",
   "merchant":"PUBLICTESTHUF",
   "transactions":[
      {
         "salt":"kKlf2RZJdtn5wHiOGBOqfNKE8yFtGwNW",
         "merchant":"PUBLICTESTHUF",
         "orderRef":"101010515384686499284",
         "total":15,
         "transactionId":99326020,
         "status":"FINISHED",
         "resultCode":"OK",
         "refundStatus":"PARTIAL",
         "refunds":[
            {
               "transactionId":99326030,
               "refundTotal":5.0,
               "refundDate":"2018-10-02T10:29:43+02:00",
               "status":"FINISHED"
            }
         ],
         "remainingTotal":10.0,
         "paymentDate":"2018-10-02T10:24:09+02:00",
         "finishDate":"2018-10-02T10:29:28+02:00",
         "method":"CARD"
      }
   ],
   "totalCount":1
}
```

4 PHP SDK

A fizetési rendszer implementálásához felhasználható PHP mintakód (SDK) ezzel a dokumentációval együtt elérhető a SimplePay rendszert használó, vagy azt fejlesztő kereskedők részére.

Az SDK egy minta megoldás, ami **egy lehetséges módja az implementációnak** és kizárólag csak a fizetési funkció technikai megvalósítására koncentrál. Az SDK a 3. fejezetben leírt API működést valósítja meg. Az SDK mintakódja előállítja a hívásokhoz szükséges JSON tartalmakat, az üzenetek validálásához szükséges aláírásokat. Elvégzi a kommunikációt a SimplePay rendszere felé. Fogadja, validálja és elérhetővé teszi a kapott válaszokat.

A fentieken túlmenően az SDK nem valósítja meg a kereskedői rendszer semmilyen egyéb funkcióját.

A mintakód GNU GPL3 licenc feltételek mellett használható a SimplePay rendszerben való tranzakciók elindításához. A SimplePay rendszer használatának lehetőségét a SimplePay kereskedői szerződés szabályozza.

A mintakód alkalmas a fizetési tranzakció indítására, ezért **leginkább lényeges a saját, gazda rendszer működésének az ismerete, hogy** pontosan átlássa annak logikáját és **be tudja illeszteni** a megfelelő helyre **az online fizetést**.

Az SDK segítségével 15 perc alatt működő teszt rendszert lehet létrehozni. Amint ezt a kódot kicsomagolja az Ön szerverére és az továbbiakban leírt módon beállítja, akkor azonnal kipróbálható a bankkártyás fizetés a SimplePay teszt rendszerében (sandbox). Ha ezt elvégzi, akkor Ön a működő mintakódon pontosan láthatja, hogy mit szükséges megvalósítson a saját rendszerében.

A fejlesztés alatt nagyban megkönnyíti a hibakeresést, ha változtatás nélküli SDK-t saját, különálló teszt rendszerként használja. Ebben az esetben, ha a beépített kódjában problémába ütközik, akkor azonos adatokkal indított tranzakció esetén nyomban össze tudja hasonlítani és ellenőrizni tudja, hogy az adott funkció hogyan működik, vagy mi okozhatja a problémát.

4.1 Az SDK felépítése

Az SDK az alábbi elemekből épül föl.

/src
A mappában található meg a bankkártyás fizetést megvalósító minta forráskód a **SimplePayV21.php** fájlban. A fájl minden szükséges forráskódot tartalmaz az összes megvalósítható funkcióhoz.
Ugyanebben a mappában található meg a **config.php** fájl. Ebben a fájlban szükséges beállítani többek között a kereskedői fiók adatait, ami nélkül nem lehet tranzakciót indítani.

/log
Az SDK alapértelmezett log mappája. Ha Ön máshova szeretné irányítani a logolást, akkor a **config.php** fájlban lehet megadni a saját logolási útvonalat.

php fájlok a főkönyvtárban
Mindegyik fájl egy kereskedő oldali funkció mintakódja. A mintakódok jellemzően az alábbiakat hajtják végre:

- a tranzakcióhoz szükséges adatokat gyűjtik össze
- a megfelelő adatokkal paraméterezve futtatják az **src/SimplePayV21.php** fájlban található class-t, ami a szükséges fizetési funkciót végrehajtja
- a kapott válasz adatokat feldolgozzák, megjelenítik

4.2 Az SDK config beállításai

Csomagolja ki a mintakódot az ön szerverén. Nyissa meg az src/config.php fájlt. A fájlban egy tömb található, aminek az elemei közül az első tesztekhez elegendő a kereskedői adatokat és az URL-t beállítani a következőkben leírt módon. Az első teszthez egyelőre a többi paraméterrel nem szükséges foglalkozni.

4.2.1 Kereskedői fiók adatai

A mintakód alkalmas mindhárom lehetséges devizanem párhuzamos használatára. Ha több devizanemben is értékesít a weboldalán és van HUF, EUR és USD fiókja is, akkor az aktuális tranzakció devizaneme alapján tud automatikusan váltani a mintakód az eltérő devizanemű fiókok között.

A fiókok adatai az alábbi mezőkben adhatók meg

```
'HUF_MERCHANT' => "",          //merchant account ID (HUF)
'HUF_SECRET_KEY' => "",        //secret key for account ID (HUF)
'EUR_MERCHANT' => "",          //merchant account ID (EUR)
'EUR_SECRET_KEY' => "",        //secret key for account ID (EUR)
'USD_MERCHANT' => "",          //merchant account ID (USD)
'USD_SECRET_KEY' => "",        //secret key for account ID (USD)
```

Ezekhez a mezőkhöz az adatokat a SimplePay kereskedői admin felületén találja meg a következő URL-en: https://sandbox.simplepay.hu/admin

A „**Fiókkezelő**" oldalon a fejlécben megtalálhatók a fiók alap adatai. Ezek között található meg a Kereskedői azonosító (**MERCHANT**).
Ezen belül a „**Technikai adatok**" aloldalon az „**Alap adatok**" között található meg az aláírás számításához szükséges kulcs (**SECRET_KEY**).
Ezek az adatok fiókonként (devizanemenként) változók.

Az egyedi devizanemű fiókok között a jobb oldali választó menüben (**Másik fiók választása**) tud váltani.

4.2.2 URL-ek

Miután megtörténik a fizetés, a fizetőoldal a megadott kereskedői URL-re fogja visszairányítani a vásárlót. Attól függően, hogy hogyan megfelelőbb a kereskedői rendszernek, kétféle logika szerint lehet ezt az URL-t megadni.

Az egyik esetben egy közös URL-re történik meg a visszairányítás, függetlenül a tranzakció eredményétől. Ebben az esetben az URL meghívásakor a paraméterekben levő értékek alapján lehet kereskedő oldalon eldönteni, hogy mi lett a tranzakció eredménye.

```
'URL' => 'http://' . $_SERVER['HTTP_HOST'] . '/back.php',
```

A másik esetben külön URL-re van irányítva a tranzakció attól függően. hogy mi lett az eredménye.

```
'URLS_SUCCESS' => 'http://' . $_SERVER['HTTP_HOST'] . '/success.php',
'URLS_FAIL' => 'http://' . $_SERVER['HTTP_HOST'] . '/fail.php',
'URLS_CANCEL' => 'http://' . $_SERVER['HTTP_HOST'] . '/cancel.php',
'URLS_TIMEOUT' => 'http://' . $_SERVER['HTTP_HOST'] . '/timeout.php',
```

Az **URLS_SUCCESS** változóan megadott URL-re érkezik vissza a vásárló sikeres fizetés esetén a SimplePay fizetőoldaláról.
Az **URLS_FAIL** változóan megadott URL-re érkezik vissza a vásárló sikertelen fizetés esetén a SimplePay fizetőoldaláról.
Az **URLS_CANCEL** változóan megadott URL-re érkezik vissza a vásárló abban az esetben, ha megszakítja a fizetést a SimplePay fizetőoldalon.
Az **URLS_TIMEOUT** változóan megadott URL-re érkezik vissza a vásárló abban az esetben, ha az elindításnál megadott ideig nem ad meg kártyaadatot a vásárló és nem indítja el a fizetést.

FONTOS: a tranzakciókhoz használható teszt bankkártya adatokat a sandbox fizetőoldalon lehet kiválasztani.

4.2.3 Váltás teszt és éles tranzakció között

Alapértelmezetten a sandbox használata van bekapcsolva az SDK-ban.

```
'SANDBOX' => true,
```

Abban az esetben, ha a sikeres tesztek után az éles rendszerre vált, akkor ezt a változót szükséges **false** értékre állítani. A sandbox fiók az élesítés után is megmarad, így a későbbiekben is lehet tesztelni, ha bármilyen további fejlesztést szükséges végezni.

4.2.4 Logolás

A logolás a fejlesztés során a hibakereséshez jelent fontos segítséget. Alapértelmezetten be van kapcsolva az SDK-ban, csak annyira van szükség, hogy az sdk/log mappa írható legyen.

```
'LOGGER' => true,
'LOG_PATH' => 'log',
```

4.3 Az SDK IPN beállítása

Az Instant Payment Notification (IPN) a fizetési tranzakció vége. Ezen a ponton tájékoztatja a SimplePay rendszere a kereskedő webáruházát a tranzakció sikerességéről. Az IPN egy háttérfolyamat a SimplePay és a kereskedő között. Az IPN-t POST metódussal a kereskedő által megadott URL-re küldi ki a SimplePay rendszere.

Az **IPN** üzenet **csak a sikeres és teljesíthető tranzakcióról van kiküldve**. Ez a tranzakció sikerességének az egyértelmű jelzője. Sikeresség esetén ekkor a kereskedői rendszerben be lehet állítani a fizetést - az áruháznak megfelelően - fizetett státuszba és teljesíteni lehet a megrendelést.

Az IPN URL az az online elérhető cím, ahol a kereskedő **fogadja** a SimplePay rendszer által küldött IPN üzenetet. Erre a címre fogja a SimplePay rendszere küldeni az értesítést. Ezt az adatot kell beregisztrálni a SimplePay rendszerébe, aminek a beállítását a kereskedői vezérlőpanelen lehet elvégezni. Az adat a „**Technikai adatok**" menüpont alatt lehet beállítani.

A mintakód esetén az ipn.php fájl elérési útja: „http://domainnev.hu/ipn.php"
Az IPN beállítását fiókonként el kell végezni. Ha Ön több fiókot használ egy domainen (pl. HUF és EUR devizanemben is lehet fizetni), akkor mindkét esetben (fiókban) meg kell adni az URL-t.

Az IPN URL esetén az alábbiakra kell figyelmet fordítani

- legyen publikusan elérhető
- ne legyen .htpassword védelem rajta
- átirányítási szabályok, SSL beállítások ne akadályozzák az elérését

Az IPN használatával válik teljes értékűvé a teszt rendszere. Az IPN használatának további részleteiről az IPN tesztelése fejezetben talál további lényeges részleteket.

4.4 start - tranzakció létrehozása

Mintakód az SDK-ban: **start.php**

A mintakód az „**Implementáció**" fejezeten belül a „**start**" hívás leírásában található JSON stringet létrehozza, elküldi a SimplePay API felé, majd fogadja és validálja a kapott választ.

Az **start** működési logikája korábban az „**Implementáció**" c. fejezeten belül lett kifejtve. A továbbiakban az ott leírt működési logikának a PHP SDK segítségével történő megvalósítását mutatja be ez a fejezet.

```php
//Import config data and SimplePay class
require_once 'src/config.php';
require_once 'src/SimplePayV21.php';

// new SimplePayStart instance
$trx = new SimplePayStart;

//add config data
$trx->addConfig($config);

//add transaction data
$trx->addData('currency', 'HUF');
$trx->addData('total', '100');
$trx->addData('orderRef', '101010515363456734591');
$trx->addData('customerEmail', 'sdk_test@otpmobil.com');
$trx->addData('language', 'HU');

$timeoutInSec = 600;
$trx->addData('timeout ', date("c", time() + $timeoutInSec));

$trx->addData('methods', array('CARD'));
$trx->addData('url', $config['URL']);

// invoice
$trx->addGroupData('invoice', 'name', 'SimplePay V2 Tester');
$trx->addGroupData('invoice', 'company', '');
$trx->addGroupData('invoice', 'country', 'hu');
$trx->addGroupData('invoice', 'state', 'Budapest');
$trx->addGroupData('invoice', 'city', 'Budapest');
$trx->addGroupData('invoice', 'zip', '1111');
$trx->addGroupData('invoice', 'address', 'Address 1');
$trx->addGroupData('invoice', 'address2', '');
$trx->addGroupData('invoice', 'phone', '06203164978');

//create transaction in SimplePay system
$trx->runStart();
```

A fenti mintának megfelelő, de kicsit bővebb fájl található az SDK-ban **start.php** néven. A fájl elején a két include a konfig adatokat és a fizetést végrehajtó PHP kódot tartalmazza.

Ezután a **SimplePayStart** class példányositása következik. Ebben az osztályban található meg az a mintakód, ami végrehajtja a start kommunikációhoz szükséges fentebb bemutatott JSON string létrehozását, adatokkal feltöltését, a Signature kiszámítását, a header beállítását, majd az üzenet elküldését a SimplePay szervere felé.

A konfig adatok hozzá adásához a **addConfig()** függvény használható.

A továbbiakban hozzáadjuk a tranzakcióhoz a szükséges adatokat. Ehhez két függvényt nyújt az SDK.

addData()
Ezzel a függvénnyel név/érték párokat lehet a tranzakcióhoz adni. Ilyen lehet például a devizanem. A függvény első paramétere a mező neve, a második pedig az értéke.

```
$trx->addData('currency', 'HUF');
```

addGroupData()
Ezzel a függvénnyel egy csoporthoz lehet adatokat adni. Ilyen lehet például a számla adatok csoportján belül a város. A függvény első paramétere a csoport neve, a második a mező neve, majd a harmadik pedig az értéke.

```
$trx->addGroupData('invoice', 'city', 'Budapest');
```

runStart()
Miután minden szükséges adat hozzá van adva a tranzakcióhoz el kell végezni a Signature kiszámítását, a header feltöltését ezzel az adattal, valamint a hívást el kell indítani a SimplePay szervere felé.
Mindezeket egyben elvégzi a runStart() föggvény.

```
//create transaction in SimplePay system
$trx->runStart();
```

4.5 start – response

A mintakód a „start" hívásra kapott válasz JSON stringet dolgozza fel és validálja.

Az **start** működési logikája korábban az „**Implementáció**" c. fejezeten belül lett kifejtve. A továbbiakban az ott leírt működési logikának a PHP SDK segítségével történő megvalósítását mutatja be ez a fejezet.

Az SDK-ban a $trx->getReturnData() függvény kimenete tartalmazza a korábban indított runStart() eredményét. A függvény kimenete egy tömb, amiben megtalálható a fenti JSON válasz, illetve tömb elemeiként a tartalma is.

A runStart() függvény után a **getHtmlForm()** futtatása a **paymentUrl** elem értékét felhasználva létrehoz egy kész formot.

Tranzakciót elindító form elemek meghatározhatók az alábbi módon.

```
$trx->formDetails['element'] = 'button';
```

A lehetőségek az alábbiak:

- **button**: egy gombot hoz létre, ami a vásárló kattintására vár. A kattintás után történik meg a fizetőoldalra átirányítás
- **link**: egy linket hoz létre, ami a vásárló kattintására vár. A kattintás után történik meg a fizetőoldalra átirányítás
- **auto**: azonnali átirányítást végez a fizetőoldalra, anélkül hogy vásárlói interakcióra várna

```
$trx->getHtmlForm();
```

Ezt felhasználva a vásárlót azonnal a fizetőoldalra lehet irányítani.
Ez a tömbnek a **form** mezőjében található meg.

```
Array
(
    [responseBody] => {"salt":"OORbo29VweDZ8rCVW3THCmgDkh7Qtj3H","merchant":"PUBLICTESTHUF","o
rderRef":"101010515363456734591","currency":"HUF","transactionId":99310118,"timeout":"2018-09-
07T20:46:13+02:00","total":100.0,"paymentUrl":"https://sandbox.simplepay.hu/pay/pay/pspHU/aCKK
ahLcGamHCdLSARoOdC2jS5yBqgUSwSt6X4KsgWmod13zDc"}
    [responseSignature] => 3WGqCnWJArhA224xVdUY1fPh91tpd6va6JvBrPNuHK449TZTgsRn3DBu5UBGbcTn
    [responseSignatureValid] => 1
    [salt] => OORbo29VweDZ8rCVW3THCmgDkh7Qtj3H
    [merchant] => PUBLICTESTHUF
    [orderRef] => 101010515363456734591
    [currency] => HUF
    [transactionId] => 99310118
    [timeout] => 2018-09-07T20:46:13+02:00
    [total] => 100
    [paymentUrl] => https://sandbox.simplepay.hu/pay/pay/pspHU/aCKKahLcGamHCdLSARoOdC2jS5yBqgU
SwSt6X4KsgWmod13zDc
    [form] => <form action="https://sandbox.simplepay.hu/pay/pay/pspHU/mueK6aKcDuhbPYpqQHQOTl0
jk11BWgXRwSvwR4rvQ3sEXQbjBw" method="POST" id="SimplePayForm" accept-charset="UTF-8">
          <button type="submit">Start SimplePay Payment</button>
</form>
)
```

A **form** elem megjelenítésével egy gombot, vagy linket kapunk, a korábban megadott $trx->formDetails['element'] értékének megfelelően. Ezt megjelenítve a vásárló kattintásával, vagy automatikusan elindul az átirányítás a fizetőoldalra.

```
print $trx->transaction['form'];
```

4.6 SimplePay fizetőoldal

Eddig a "**start**" funkcióval létre lett hozva a SimplePay rendszerében a fizetési tranzakció. A kereskedői rendszer a SimplePay API-tól kapott egy URL-t, amire át kell irányítani a vásárlót. Az URL-en a SimplePay fizetőoldal töltődik be a létrejött tranzakció adataival.

A fizetőoldal megjeleníti a korábban megadott fizetési tranzakció adatait, a kereskedőt, akinek a vásárló fizet, illetve az összes szükséges tájékoztatást.

Az fizetőoldalon elérhető funkciók korábban az „Implementáció” c. fejezeten belül lettek bemutatva.

Ezen a felületen lehet megadni a fizetésre felhasználandó kártya adatait. A sandbox rendszeren a kártyaszám mezője valójában egy legördülő menü, ahol ki lehet választani teszt kártyákat a sikeres és sikertelen fizetések szimulálásához. A fizetés kimenete a választott kártyától függ.

A “**FIZETEK**” gombra kattintva indul el a tranzakció. Ekkor megtörténik a kártya authorizáció, ami után a vásárló vissza lesz irányítva a kereskedő weboldalára.

4.7 back

A fenti tranzakció után megtörténik a kereskedői weboldalra a visszairányítás, ahol a GET metódussal küldött paraméterek hitelesítése és feldolgozása történik.

A **back** működési logikája korábban az „**Implementáció**” c. fejezeten belül lett kifejtve. A továbbiakban az ott leírt működési logikának a PHP SDK segítségével történő megvalósítását mutatja be ez a fejezet.

```
//Import config data and SimplePay class
require_once 'src/config.php';
require_once 'src/SimplePayV21.php';

// new SimplePayBack instance
$trx = new SimplePayBack;

//add config data
$trx->addConfig($config);

//result
$result = array();
if (isset($_REQUEST['r']) && isset($_REQUEST['s'])) {
    if ($trx->isBackSignatureCheck($_REQUEST['r'], $_REQUEST['s'])) {
        $result = $trx->getRawNotification();
    }
}
```

A $result változó értéke egy tömb lesz, aminek a tartalma megegyezik a JSON string adataival.

```
Array
(
    [r] => 0
    [t] => 99310118
    [e] => SUCCESS
    [m] => PUBLICTESTHUF
    [o] => 101010515363456734591
)
```

4.8 A tranzakció eredményétől függő tájékoztatások a kereskedői oldalon

A kereskedői weboldalon a tranzakció eredményének megfelelő tájékoztatást szükséges megjeleníteni a vásárló számára.

A szükséges **back** tájékoztatások korábban az „**Implementáció**" c. fejezeten belül lettek kifejtve a sikeres, a sikertelen, a megszakított tranzakciókhoz, valamint időtúllépés esetére. Az SDK használata mellett is ugyanazokat a tájékoztatásokat kell alkalmazni.

4.9 IPN

Az IPN üzenetet a SimplePay rendszere küldi ki a kereskedő webáruháza felé. Az üzenet a sikeres tranzakció végét jelenti, ami alapján a kereskedő teljesítheti a megrendelést.

A mintakód a SimplePay rendszere által kiküldött JSON stringet hitelesíti, feldolgozza, illetve a szükséges választ összeállítja. A választ automatikusan vissza is adhatja a SimplePay rendszere felé, vagy csak az adatokat generálja le, amit a kereskedői rendszernek megfelelő ponton lehet alkalmazni.

Az **ipn** működési logikája korábban az „**Implementáció**" c. fejezeten belül lett kifejtve. A továbbiakban az ott leírt működési logikának a PHP SDK segítségével történő megvalósítását mutatja be ez a fejezet.

```
//Import config data and SimplePay class
require_once 'src/config.php';
require_once 'src/SimplePayV21.php';

//input
$json = file_get_contents('php://input');

// new SimplePayIpn instance
$trx = new SimplePayIpn;

//add config data
$trx->addConfig($config);

// check signature
if ($trx->isIpnSignatureCheck($json)) {
    $trx->runIpnConfirm();
}
```

Az IPN feldolgozás első lépésben a hívás validálását végzi el a **isIpnSignatureCheck()** függvény.

Ha a validálás sikeres, akkor a **runIpnConfirm()** függvény előállítja a válasz üzenetet és a szükséges Signature értéket. Alapértelmezetten a szükséges válasz megjelenítését is elvégzi a függvény, így egyéb fejlesztés a fenti kódon kívül nem szükséges.

Mivel a válasz Signature értékét a header-ben kell visszaadni, ami a header módosításával jár, így fontos, hogy **a fenti kódrészlet futása előtt NE legyen semmilyen egyéb adat megjelenítve** az IPN üzenetet feldolgozó és megválaszoló URL-en.

Mivel ez egy háttérben lezajló folyamat, így ezen a ponton minden más adat megjelenítése fölösleges.

Az IPN választ nem kötelező a **runIpnConfirm()** függvénnyel megjeleníteni.
Ebben az esetben a **getIpnConfirmContent()** függvényt kell meghívni az alábbi módon.

```
$confirm = $trx->getIpnConfirmContent();
```

A függvény létre fogja hozni a szükséges választ és a Signature tartalmát is, de nem fogja azokat megjeleníteni, illetve nem lesz módosítva a header. Az adatok a függvény kimenetében visszaadott **$confirm** tömbben lesznek megtalálhatók, amiből kinyerve a kereskedői rendszer számára megfelelő helyen végezhető el az adatok megjelenítése.

4.10 finish - kétlépcsős tranzakciók kezelése

A **finish** hívás a kétlépcsős tranzakció lezárása.

A mintakód a szükséges JSON stringet létrehozza, elküldi a SimplePay API felé, majd fogadja és validálja a kapott választ.

Az **finish** működési logikája korában az „**Implementáció**" c. fejezeten belül lett kifejtve. A továbbiakban az ott leírt működési logikának a PHP SDK segítségével történő megvalósítását mutatja be ez a fejezet.

Mintakód az SDK-ban: **finish.php**

```
//Import config data
require_once 'src/config.php';

//Import SimplePayment class
require_once 'src/SimplePayV21.php';

// new SimplePayFinish instance
$trx = new SimplePayFinish;

//config
$trx->addConfig($config);

// transaction ID
$trx->addData('transactionId', '99326614');

// original total value
$trx->addData('originalTotal', 15);

// approved total value
$trx->addData('approveTotal', 15);

//currency
$trx->transactionBase['currency'] = 'HUF';

//start finish communication
```

```
$trx->runFinish();

//result
$result = $trx->getReturnData()
```

A választ a $result értéke tartalmazza

```
Array
(
    [responseBody] => {"salt":"liAOP84G3nh0cNBCVijXqhY02PookbUv","merchant":"PUBLICTESTHUF","o
rderRef":"101010515385149346625","currency":"HUF","transactionId":99326614,"approveTotal":15.0
}
    [responseSignature] => Hf3ZHmhtFpTK3HoLjVmYwQgCAFYu9RDFa5nx296pFwpTHw6XBLVq3ePO+VSSejwL
    [responseSignatureValid] => 1
    [salt] => liAOP84G3nh0cNBCVijXqhY02PookbUv
    [merchant] => PUBLICTESTHUF
    [orderRef] => 101010515385149346625
    [currency] => HUF
    [transactionId] => 99326614
    [approveTotal] => 15
)
```

4.11 refund - visszatérítések kezelése

A **refund** hívást az alábbi módon hajtja végre az SDK.

A mintakód a szükséges JSON stringet létrehozza, elküldi a SimplePay API felé, majd fogadja és validálja a kapott választ.

Az **refund** működési logikája korában az „**Implementáció**" c. fejezeten belül lett kifejtve. A továbbiakban az ott leírt működési logikának a PHP SDK segítségével történő megvalósítását mutatja be ez a fejezet.

Mintakód az SDK-ban: **refund.php**

```
// Import config data
require_once 'src/config.php';

// Import SimplePayment class
require_once 'src/SimplePayV21.php';

// new SimplePayRefund instance
$trx = new SimplePayRefund;

//config
$trx->addConfig($config);

// add transaction ID
$trx->addData('transactionId', '99326864');

// amount to refund
$trx->addData('refundTotal', 5);

// add currency
$trx->addData('currency', 'HUF');
```

```
// start refund
$trx->runRefund();

// result
$result = $trx->getReturnData()
```

A hívásra adott válasz egy tömb, ami az alábbi adatokat tartalmazza.

```
Array
(
    [responseBody] => {"salt":"WZ7Ncc0qoDSMYG4twsmeOdBs6PSnsj1Z","merchant":"PUBLICTESTHUF","o
rderRef":"101010515385577564359","currency":"HUF","transactionId":99326864,"refundTransactionI
d":99326874,"refundTotal":5.0,"remainingTotal":10.0}
    [responseSignature] => +vtH3O/4TyiuLuWEqn4qCwyX+8xdLiuOPFczwrLRIFm7m44uJpF/VRmXgoS33pnk
    [responseSignatureValid] => 1
    [salt] => WZ7Ncc0qoDSMYG4twsmeOdBs6PSnsj1Z
    [merchant] => PUBLICTESTHUF
    [orderRef] => 101010515385577564359
    [currency] => HUF
    [transactionId] => 99326864
    [refundTransactionId] => 99326874
    [refundTotal] => 5
    [remainingTotal] => 10
)
```

A hívásra adott válasz három kulcs értéke a tömbben:

- **refundTransactionId**: a visszatérítés tranzakció azonosítója
- **refundTotal**: a visszatérített összeg
- **remainingTotal**: a visszatérítés után az eredeti tranzakció fennmaradó összege (eredeti terhelésből az eddigi visszatérítések kivonva)

4.12 query - tranzakció adatainak lekérdezése

A **query** funkció használatával lehet tranzakciók adatait lekérdezni a SimplePay rendszeréből.

A mintakód a szükséges JSON stringet létrehozza, elküldi a SimplePay API felé, majd fogadja és validálja a kapott választ.

A **query** működési logikája korábban az „**Implementáció**" c. fejezeten belül lett kifejtve. A továbbiakban az ott leírt működési logikának a PHP SDK segítségével történő megvalósítását mutatja be ez a fejezet.

Mintakód az SDK-ban: **query.php**

```
//Import config data and SimplePay class
require_once 'src/config.php';
require_once 'src/SimplePayV21.php';

// new SimplePayQuery instance
$trx = new SimplePayQuery;

//add config data
$trx->addConfig($config);
```

```
//add SimplePay transaction ID
$trx->addSimplePayId('99325516');

//query
$trx->runQuery();

//result data
$result = $trx->getReturnData();
```

SimplePay tranzakció azonosítókat az **addSimplePayId()** függvénnyel lehet a híváshoz adni az alábbi módon.

```
$trx->addSimplePayId('99325516');
```

Kereskedői tranzakció azonosítókat az **addMerchantOrderId()** függvénnyel lehet a híváshoz adni az alábbi módon.

```
$trx->addMerchantOrderId('101010515383930534733');
```

Az az **addSimplePayId()** és az **addMerchantOrderId()** függvények többszöri hívásával több elemet lehet hozzá adni a lekérdezéshez, így egyszerre több tranzakció adatát is megkaphatjuk.

A fenti mintakódban a **$trx->runQuery()** függvény hajtja végre a lekérdezést.

```
$trx->runQuery();
```

A választ a lekérdezésre a **$trx->getReturnData()** függvénnyel nyerhetjük ki, aminek a tartalma a lenti módon épül föl. A válaszon belül a transactions tömb tartalmazza a lekérdezett tranzakció(k) adatait. A **totalCount** értéke tartalmazza a talált tranzakciók darabszámát.

```
Array
(
    [responseBody] => {"salt":"YyUNzD6Qhw9iWGGq9hhpd7gfyIFulL1o","merchant":"PUBLICTESTHUF","t
ransactions":[{"salt":"zgtZuLZZVkuTDIPuv4mkxZQFsx4LNLSY","merchant":"PUBLICTESTHUF","orderRef"
:"101010515383923675972","transactionId":99325516,"status":"FINISHED","resultCode":"OK","remai
ningTotal":0.0,"paymentDate":"2018-10-01T13:12:47+02:00","finishDate":"2018-10-01T13:14:39+02:
00","method":"CARD"}],"totalCount":1}
    [responseSignature] => QJRVV4lCxZOzibXMsre+MEgnW+6T2xLCjNlFnmPEZ74wCnD5jW3r6+SFJAnvzcma
    [responseSignatureValid] => 1
    [salt] => YyUNzD6Qhw9iWGGq9hhpd7gfyIFulL1o
    [merchant] => PUBLICTESTHUF
    [transactions] => Array
        (
            [0] => Array
                (
                    [salt] => zgtZuLZZVkuTDIPuv4mkxZQFsx4LNLSY
                    [merchant] => PUBLICTESTHUF
                    [orderRef] => 101010515383923675972
                    [total] => 25
                    [transactionId] => 99325516
                    [status] => FINISHED
                    [resultCode] => OK
                    [remainingTotal] => 0
                    [paymentDate] => 2018-10-01T13:12:47+02:00
                    [finishDate] => 2018-10-01T13:14:39+02:00
```

```
                    [method] => CARD
                )

        )

    [totalCount] => 1
)
```

A lekérdezés kiegészíthető a „**detailed**" paraméterrel, aminek hatására a tranzakció adatainak részletei is szerepelnek a válaszban.

```
$trx->addData('detailed', true);
```

Részletes válasz

```
Array
(
    [responseBody] => {"salt":"djXXCGhaqbJx0Q9S6HgmaLo2lb2kMjfm","merchant":"PUBLICTESTHUF","t
ransactions":[{"salt":"WG9Uxej8IiU1seKt1ww8piAsRYnqynPj","merchant":"PUBLICTESTHUF","orderRef"
:"101010515383923675972","currency":"HUF","customer":"v2 START Tester","customerEmail":"sdk_te
st@otpmobil.com","language":"HU","twoStep":false,"total":15.0,"shippingCost":0.0,"discount":0.
0,"invoice":{"company":"","country":"hu","state":"Budapest","city":"Budapest","zip":"1111","ad
dress":"Address 1","address2":"","phone":"06203164978","lname":"SimplePay V2 Tester"},"deliver
y":{"company":"","country":"hu","state":"Budapest","city":"Budapest","zip":"1111","address":"A
ddress 1","address2":"","phone":"06203164978","lname":"SimplePay V2 Tester"},"transactionId":9
9325516,"status":"FINISHED","resultCode":"OK","remainingTotal":0.0,"paymentDate":"2018-10-01T1
3:12:47+02:00","finishDate":"2018-10-01T13:14:39+02:00","method":"CARD"}],"totalCount":1}
    [responseSignature] => u5M4bBRe2P8Tr8nAeI/FMa0HziqetlUE87GIgtkbAzCaW9VPbhE+9oXW0fyVKK4Q
    [responseSignatureValid] => 1
    [salt] => djXXCGhaqbJx0Q9S6HgmaLo2lb2kMjfm
    [merchant] => PUBLICTESTHUF
    [transactions] => Array
        (
            [0] => Array
                (
                    [salt] => WG9Uxej8IiU1seKt1ww8piAsRYnqynPj
                    [merchant] => PUBLICTESTHUF
                    [orderRef] => 101010515383923675972
                    [currency] => HUF
                    [customer] => v2 START Tester
                    [customerEmail] => sdk_test@otpmobil.com
                    [language] => HU
                    [twoStep] =>
                    [total] => 15
                    [shippingCost] => 0
                    [discount] => 0
                    [invoice] => Array
                        (
                            [company] =>
                            [country] => hu
                            [state] => Budapest
                            [city] => Budapest
                            [zip] => 1111
                            [address] => Address 1
                            [address2] =>
                            [phone] => 06203164978
                            [lname] => SimplePay V2 Tester
                        )

                    [delivery] => Array
                        (
                            [company] =>
```

```
                            [country] => hu
                            [state] => Budapest
                            [city] => Budapest
                            [zip] => 1111
                            [address] => Address 1
                            [address2] =>
                            [phone] => 06203164978
                            [lname] => SimplePay V2 Tester
                        )

                    [transactionId] => 99325516
                    [status] => FINISHED
                    [resultCode] => OK
                    [remainingTotal] => 0
                    [paymentDate] => 2018-10-01T13:12:47+02:00
                    [finishDate] => 2018-10-01T13:14:39+02:00
                    [method] => CARD
                )

        )

    [totalCount] => 1
)
```

A lekérdezés kiegészíthető a „**refunds**" paraméterrel, aminek hatására a tranzakció adatai mellett megkapható a rájuk indított visszatérítések adatai is.

```
$trx->addData('refunds', true);
```

Visszatérítésekkel kiegészített válasz esetén minden tranzakciónál a „refunds" tömb fogja tartalmazni a tranzakcióra indított visszatérítések adatait.

```
Array
(
    [responseBody] => {"salt":"QxQmqOfkqV9khWU6SHJxlKmYyuN74x1E","merchant":"PUBLICTESTHUF","t
ransactions":[{"salt":"kKlf2RZJdtn5wHiOGBOqfNKE8yFtGwNW","merchant":"PUBLICTESTHUF","orderRef"
:"101010515384686499284","transactionId":99326020,"status":"FINISHED","resultCode":"OK","refun
dStatus":"PARTIAL","refunds":[{"transactionId":99326030,"refundTotal":-5.0,"refundDate":"2018-
10-02T10:29:43+02:00","status":"FINISHED"}],"remainingTotal":10.0,"paymentDate":"2018-10-02T10
:24:09+02:00","finishDate":"2018-10-02T10:29:28+02:00","method":"CARD"}],"totalCount":1}
    [responseSignature] => sHitW57bS9UkeskilZh3mzoOLwuuIzxQBgFKDe77OSOFosfN08VLbFidUyNIMNaq
    [responseSignatureValid] => 1
    [salt] => QxQmqOfkqV9khWU6SHJxlKmYyuN74x1E
    [merchant] => PUBLICTESTHUF
    [transactions] => Array
        (
            [0] => Array
                (
                    [salt] => kKlf2RZJdtn5wHiOGBOqfNKE8yFtGwNW
                    [merchant] => PUBLICTESTHUF
                    [orderRef] => 101010515384686499284
                    [transactionId] => 99326020
                    [status] => FINISHED
                    [resultCode] => OK
                    [refundStatus] => PARTIAL
                    [refunds] => Array
                        (
                            [0] => Array
                                (
                                    [transactionId] => 99326030
                                    [refundTotal] => 5
                                    [refundDate] => 2018-10-02T10:29:43+02:00
                                    [status] => FINISHED
```

```
                                )

                            )

                        [remainingTotal] => 10
                        [paymentDate] => 2018-10-02T10:24:09+02:00
                        [finishDate] => 2018-10-02T10:29:28+02:00
                        [method] => CARD
                    )

            )

        [totalCount] => 1
)
```

5 Mintakódok

Az implementációt leíró fejezet a tranzakcióhoz szükséges küldött és fogadott adatokat tárgyalta. A JSON stringeket bármilyen a kereskedő által alkalmazható programnyelven ugyanabban a formában kell felépíteni, így a leírásuk során nem alkalmaztunk programnyelv függő példákat.

Emiatt a leírás nem érintett két kulcsfontosságú funkciót, a **hash generálását** és az **API hívások** kivitelezését, mivel ezek már programnyelvtől függően eltérők lehetnek.

Az alábbi kódrészletek konkrét programnyelven mutatnak be mintát a fenti funkciókhoz.

FIGYELEM: a példák minden esetben a funkció megvalósításához szükséges minimális mintakódot tartalmazzák. Az alábbiakban egy-egy lehetséges megoldást mutatunk be, azonban mindegyik esetben elképzelhető más mód is a szükséges funkció megvalósítására.

5.1 HASH kalkulálás, hitelesítés

Az üzenetek validálásához szükséges hash generálás logikája a **3.2** fejezetben található meg. A továbbiakban a többféle programnyelv segítségével történő megvalósítását mutatja be ez a fejezet.

Minden hash generálási példa esetében az alábbi változókat használjuk:

- **jsonString** tartalmazza az üzenet tartalmát
- **secretKey** tartalmazza a kereskedői fiók egyedi SECRET_KEY értékét

5.1.1 PHP megoldás

```
$signature = base64_encode(hash_hmac('sha384', $jsonString, $secretKey, true));
```

5.2 API hívások

Az API hívások általános üzenet formátuma a **3.1** fejezetben található meg. A továbbiakban a többféle programnyelv segítségével történő megvalósítását mutatja be ez a fejezet.

Minden API hívási példa esetében az alábbi változókat használjuk:

- **url** a SimplePay API végpontja
- **jsonString** tartalmazza az üzenet tartalmát
- **headers** tartalmazza az üzenet header-t

5.2.1 PHP megoldás

```
$curlData = curl_init();
curl_setopt($curlData, CURLOPT_URL, $url);
curl_setopt($curlData, CURLOPT_POST, true);
curl_setopt($curlData, CURLOPT_POSTFIELDS, $data);
curl_setopt($curlData, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curlData, CURLOPT_USERAGENT, 'curl');
curl_setopt($curlData, CURLOPT_TIMEOUT, 60);
curl_setopt($curlData, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($curlData, CURLOPT_HTTPHEADER, $headers);

//optional return header for result Signature check
curl_setopt($curlData, CURLOPT_HEADER, true);

$result = curl_exec($curlData);
curl_close($curlData);
```

6 Hibakódok

Hibakód	Leírás
0	Sikeres művelet
999	Általános hibakód.
1003	Elutasított művelet
1200	Érvénytelen aláírás
1400	Kártya tárolási hiba
1500	Tárolt kártya nem található
1529	SimplePay belső hiba
1600	Invalid külső azonosító
1650	A tranzakció már létezik (és nincs újraindíthatóként jelölve).
1800	Tárolt kártya esetén invalid összeg
1900	Belső hiba
2000	Invalid devizanem
2003	Megadott jelszó érvénytelen
2004	Általános hibakód
2006	Megadott kereskedő nem található
2008	Megadott e-mail nem megfelelő
2010	Megadott tranzakcióazonosító nem megfelelő
2013	Nincs elég fedezet a kártyán
2014	Fizetéshez jelszó szükséges
2016	A felhasználó megszakította a fizetés
2019	Időtúllépés az elfogadói kommunikációban
2020	Elfogadó bank oldali hiba
2021	Kártyakibocsátó interaktív 3DS ellenőrzést igényel
2030	Kártya nem törölhető, mert egyenlege pozitív / Megadott összeg helytelen
2040	Érvénytelen devizanem
2063	Kártya inaktív
2064	Hibás bankkártya adatok
2065	Megadott kártya bevonása szükséges / nem létező kártya
2066	Kártya nem terhelhető / limittúllépés miatt
2068	Kártya adat hiba / nem létező kártya
2070	Nem megfelelő kártya típus
2071	Hibás bankkártya adatok / nem létező kártya
2072	Kártya lejárat nem megfelelő / nem létező kártya
2073	A megadott CVC nem megfelelő / nem létező kártya
2074	Kártyabirtokos neve több, mint 32 karakter
2077	Érvénytelen CVC
2078	Általános hiba, a kártyakibocsátó bank nem adja meg a hiba okát
2079	A routingnak megfelelő elfogadó bank nem érhető el
2121	Érvénytelen recurring amount
2999	SimplePay belső hiba
3000	Általános 3DS hiba
3001	Érvénytelen 3DS válaszüzenet
3002	3DS folyamat hiba

3003	3DS folyamat hiba
3004	Redirect 3DS challenge folyamán (vásárló átirányítása szükséges a kártyakibocsátó ACS szerverére a kapott URL felhasználásával)
3005	3D secure interaktív azonosítás szükséges
3006	3DS hiba, banki válasz nem érkezik meg, banki válasz késve érkezik meg
3012	3DS folyamat hiba, nem 3DS képes kártya, 3DS valamelyik szereplőnél levő probléma
3013	3DS folyamat hiba
3101	Érvénytelen 3DS válaszüzenet
3102	Érvénytelen 3DS verzió
3103	3DS belső hiba
3201	3DS adathiány
3202	3DS adathiba
3203	3DS adat formátum hiba, vagy szükséges adat hiánya
3204	3DS belső hiba
3301	3DS belső hiba
3302	3DS belső hiba
3303	3DS belső hiba
3304	3DS belső hiba
3305	3DS adathiba
3306	3DS adathiba
3307	3DS adathiba
3402	3DS timeout
3403	3DS rendszerhiba
3404	3DS rendszerhiba
3405	3DS rendszerhiba
3500	3DS ACS általános hiba, kártyakibocsátó bank oldal
3501	3DS ACS kártyakibocsátó bank oldal, a felhasználó megszakította az interaktív ellenőrzés (challenge) folyamatát
3504	3DS ACS timeout, kártyakibocsátó bank oldal
3505	3DS ACS timeout, kártyakibocsátó bank oldalról nem-, vagy nem időben érkezik válasz
3506	3DS ACS tranzakció hiba, kártyakibocsátó bank oldali
3507	3DS ACS ismeretlen hiba, kártyakibocsátó bank oldal
3508	3DS ACS tranzakció hiba, kártyakibocsátó bank oldali komponensben
3911	3DS adathiba
3912	3DS adathiba
5000	Általános hibakód
5010	A kereskedői fiók nem található
5011	A tranzakció nem található
5012	A kereskedői fiók nem egyezik meg
5013	A tranzakció már létezik (és nincs újraindíthatóként jelölve).
5014	A tranzakció nem megfelelő típusú
5015	A tranzakció éppen fizetés alatt
5016	Tranzakció időtúllépés (elfogadói/acquirer oldal felől érkező kérés során).
5017	A tranzakció meg lett szakítva (elfogadói/acquirer oldal felől érkező kérés során).
5018	A tranzakció már kifizetésre került (így újabb művelet nem kezdeményezhető).
5020	A kérésben megadott érték vagy az eredeti tranzakcióösszeg ("originalTotal") ellenőrzése sikertelen

5021	A tranzakció már lezárásra került (így újabb Finish művelet nem kezdeményezhető).
5022	A tranzakció nem a kéréshez elvárt állapotban van.
5023	Ismeretlen / nem megfelelő fiók devizanem.
5026	Tranzakció letiltva (sikertelen fraud-vizsgálat következtében).
5029	A tranzakció jelenleg még nem refundolható (banki háttérműveletek)
5030	A művelet nem engedélyezett
5040	Tárolt kártya nem található
5041	Tárolt kártya lejárt
5042	Tárolt kártya inaktíválva
5043	Tárolt kártya előkészített, de még nem aktív
5044	Recurring nincs engedélyezve
5048	Recurring until szükséges
5049	Recurring until eltér
5071	Tárolt kártya érvénytelen hossz
5072	Tárolt kártya érvénytelen művelet
5081	Recurring token nem található
5082	Recurring token használatban
5083	Token times szükséges
5084	Token times túl nagy
5085	Token until szükséges
5086	Token until túl nagy
5087	Token maxAmount szükséges
5088	Token maxAmount túl nagy
5089	Recurring és oneclick regisztráció egyszerre nem indítható egy tranzakcióban
5090	Recurring token szükséges
5091	Recurring token inaktív
5092	Recurring token lejárt
5093	Recurring account eltérés
5110	Nem megfelelő visszatérítendő összeg.
5111	Az orderRef és a transactionId közül az egyik küldése kötelező
5113	A hívó kliensprogram megnevezése,verziószáma ("sdkVersion") kötelező.
5201	A kereskedői fiók azonosítója ("merchant") hiányzik.
5213	A kereskedői tranzakcióazonosító ("orderRef") hiányzik.
5216	Érvénytelen szállítási összeg
5219	Email cím ("customerEmail") hiányzik, vagy nem email fotmátumu.
5220	A tranzakció nyelve ("language") nem megfelelő
5223	A tranzakció pénzneme ("currency") nem megfelelő, vagy hiányzik.
5302	Nem megfelelő aláírás (signature) a beérkező kérésben. (A kereskedői API-ra érkező hívás aláírás-ellenőrzése sikertelen.)
5303	Nem megfelelő aláírás (signature) a beérkező kérésben. (A kereskedői API-ra érkező hívás aláírás-ellenőrzése sikertelen.)
5304	Időtúllépés miatt sikertelen hívás.
5305	Sikertelen tranzakcióküldés a fizetési rendszer (elfogadói/acquirer oldal) felé.
5306	Sikertelen tranzakciólétrehozás
5307	A kérésben megadott devizanem ("currency") nem egyezik a fiókhoz beállítottal.
5308	A kérésben érkező kétlépcsős tranzakcióindítás nem engedélyezett a kereskedői fiókon

5309	Számlázási adatokban a címzett hiányzik ("name" természetes személy esetén, "company" jogi személy esetén).
5310	Számlázási adatokban a város kötelező.
5311	Számlázási adatokban az irányítószám kötelező.
5312	Számlázási adatokban a cím első sora kötelező.
5313	A megvásárlandó termékek listájában ("items") a termék neve ("title") kötelező.
5314	A megvásárlandó termékek listájában ("items") a termék egységára ("price") kötelező.
5315	A megvásárlandó termékek listájában ("items") a rendelt mennyiség ("amount") kötelező pozitív egész szám.
5316	Szállítási adatokban a címzett kötelező ("name" természetes személy esetén, "company" jogi személy esetén).
5317	Szállítási adatokban a város kötelező.
5318	Szállítási adatokban az irányítószám kötelező.
5319	Szállítási adatokban a cím első sora kötelező.
5320	A hívó kliensprogram megnevezése,verziószáma ("sdkVersion") kötelező.
5321	Formátumhiba / érvénytelen JSON string
5322	Érvénytelen ország
5323	Lezárás összege érvénytelen
5324	Termékek listája ("items"), vagy tranzakciófőösszeg ("total") szükséges
5325	Érvénytelen URL
5326	Hiányzó cardId
5327	Lekérdezendő kereskedői tranzakcióazonosítók ("orderRefs") maximális számának (50) túllépése.
5328	Lekérdezendő SimplePay tranzakcióazonosítók ("transactionIds") maximális számának (50) túllépése.
5329	Lekérdezendő tranzakcióindítás időszakában "from" az "until" időpontot meg kell előzze.
5330	Lekérdezendő tranzakcióindítás időszakában "from" és "until" együttesen adandó meg.
5333	Hiányzó tranzakció azonosító
5337	Hiba összetett adat szöveges formába írásakor.
5339	Lekérdezendő tranzakciókhoz tartozóan vagy az indítás időszaka ("from" és "until") vagy az azonosítólista ("orderRefs" vagy "transactionIds") megadandó.
5343	Nem megfelelő tranzakciósátusz
5344	Nem megfelelő tranzakciósátusz
5345	Áfa összege kisebb, mint 0
5349	A tranzakció nem engedélyezett az elszámoló fiókon (AMEX, TSP)
5350	Érvénytelen email
5351	Érvénytelen nap
5352	Simple business fiók hiba / nem létező fiók
5401	Érvénytelen salt, nem 32-64 hosszú
5413	Létrejött utalási tranzakció
5501	Böngésző accept kötelező
5502	Böngésző agent kötelező
5503	Böngésző ip kötelező
5504	Böngésző java kötelező
5505	Böngésző nyelv kötelező
5506	Böngésző szín kötelező

5507	Böngésző magasság kötelező
5508	Böngésző szélesség kötelező
5509	Böngésző tz kötelező
5511	Érvénytelen böngésző accept
5512	Érvénytelen böngésző agent
5513	Érvénytelen böngésző IP cím
5514	Érvénytelen böngésző java
5515	Érvénytelen böngésző nyelv
5516	Érvénytelen böngésző szín
5517	Érvénytelen böngésző magasság
5518	Érvénytelen böngésző szélesség
5519	Invalid browser tz
5530	Érvénytelen type
5813	Kártya elutasítva

Fizetési kérelem (RTP) hibakódok

6100	a fiókon nincs engedélyezve az RTP fizetés
6101	hiányzik vagy hibás a kereskedői csomag azonosító
6105	fizető neve nincs megadva
6107	fizető email címe nemhelytelen vagy hiányzó email cím
6108	fizető bankszámlaszáma helytelen
6123	a megadott kereskedői tranzakció azonosítók száma meghaladja a maximális értéket egy csomagban
6124	a from paraméter az until paraméter-nél későbbi dátum
6125	from és until paramétereket együtt kell megadni
6127	tranzakció nem visszavonható
6128	hibás tranzakció lejárati dátum, nem lehet múltbeli
6133	közlemény (additionalInfo) túl hosszú (>140 karakter)
6135	a megadott tranzakció azonosítók száma meghaladja a maximális értéket

7 Logók és tájékoztatók

A fizetési elfogadóhely állandóan látható részén (pl. a láblécen), vagy a fizetés kiválasztásakor a tranzakciónál szükséges megjeleníteni a SimplePay logót.

A SimplePay logó védjegyoltalom és szerzői jogi oltalom alatt áll, ezért a SimplePay logót a kereskedő csak a SimplePay ÁSZF-ben meghatározott módon és célra használhatja fel.

A logó nem lehet transzparens és csak a jól láthatóság mértékéig változtatható a mérete. A SimplePay logókat tartalmazó fájl az alábbi helyről tölthető le:
http://simplepartner.hu/download.php?target=logo

A logónak egyben linknek is kell, hogy legyen a fizetési tájékoztatóra. A logókon linkelendő Fizetési Tájékoztató az alábbi URL-en érhető el:
Magyar nyelven: http://simplepartner.hu/PaymentService/Fizetesi_tajekoztato.pdf
Angol nyelven: http://simplepartner.hu/PaymentService/Payment_information.pdf

A következő mintakóddal lehet megjeleníteni a logót és a linkelt fizetési tájékoztatót
Az src tartalma (pirossal jelölve) a logó elérési útja az Ön szerverén.

```
<a href="http://simplepartner.hu/PaymentService/Fizetesi_tajekoztato.pdf" target="_blank">
        <img width="197" height="27" src="images/simple_logo.png" title=" SimplePay - Online bankkártyás fizetés" alt=" SimplePay vásárlói tájékoztató">
</a>
```

8 Adattovábbítási nyilatkozat

Mivel a kereskedő harmadik félnek adja át a megrendelési/vásárlói adatokat, ezért **a vásárlónak az adattovábbítási nyilatkozatot** kifejezetten **el kell fogadnia**.
A nyilatkozat elhelyezésére több lehetőség is van
- a fizetésnél közvetlenül megjelenítve
- az oldal saját Általános Szerződési Feltételeiben
- az oldal saját Adattovábbítási nyilatkozatában

FONTOS: nyilatkozat elhelyezése a weboldalon önmagában nem elégséges, ha azzal a vásárló nem találkozik és nem fogadta el.

Az elfogadás történhet checkbox segítségével, vagy a tranzakció indításánál egyértelműen jelezve, hogy a fizetést elindítva egyben elfogadja a nyilatkozatot.
A lentebb megtalálható nyilatkozat szövegében a kiemelt részeken valós kereskedői adatokkal kell feltölteni a nyilatkozatot a következő módon.

Kereskedő cégneve: a szerződésben megadott cégnév.
Székhelye: a szerződésben megadott székhely.
Fizetési Elfogadóhely webcíme: a szerződésben megadott domain név, vagy applikáció esetén ezt kiegészítve az applikáció nevével.
Kereskedő által továbbított adatok megnevezése: mindazon vásárlói adatok, amik a tranzakció során át vannak adva, pl. név, e-mail cím, stb.

Magyar nyelvű nyilatkozat

Tudomásul veszem, hogy a(z) **[Kereskedő cégneve]** (**[székhelye]**) adatkezelő által a(z) **[Fizetési Elfogadóhely webcíme]** felhasználói adatbázisában tárolt alábbi személyes adataim átadásra kerülnek az OTP Mobil Kft., mint adatfeldolgozó részére. Az adatkezelő által továbbított adatok köre az alábbi: **[Kereskedő által továbbított adatok megnevezése]**

Az adatfeldolgozó által végzett adatfeldolgozási tevékenység jellege és célja a SimplePay Adatkezelési tájékoztatóban, az alábbi linken tekinthető meg:
https://simplepay.hu/adatkezelesi-tajekoztatok/

Angol nyelvű nyilatkozat

I acknowledge the following personal data stored in the user account of **[Company Name]** (**[Company address]**) in the user database of **[Paying Acceptance Web site]** will be handed over to OTP Mobil Ltd. and is trusted as data processor. The data transferred by the data controller are the following:
[data transmitted by the trader]

The nature and purpose of the data processing activity performed by the data processor in the SimplePay Privacy Policy can be found at the following link:
https://simplepay.hu/adatkezelesi-tajekoztatok/

9 Tesztelés

A SimplePay részéről minden élesítés előtt álló webáruház tesztelésen esik át.

9.1 Tesztelés megkezdése

A tesztelés a kereskedő vagy fejlesztője jelzésére történik meg. A fejlesztés befejezése után a SimplePay oldali ellenőrzésekhez jelezze az itsupport@otpmobil.com címen az elkészült fizetési rendszer elérhetőségét, illetve minden olyan tudnivalót, ami szükséges ahhoz, hogy el lehessen érni a fizetést.

9.2 A tesztek célja

A tesztek a fizetés megvalósítását ellenőrzik a SimplePay szolgáltatásra szerződött weboldalon a vásárló szemszögéből nézve.

9.3 A tesztelés helye

A fejlesztést el lehet végezni a kereskedői teszt rendszeren abban az esetben, ha rendelkezésre áll ilyen. A kereskedői teszt rendszernek nem szükséges a szerződésben szereplő domain néven legyen, így azt bárhol tesztelhető, ahol publikusan elérhető.

Abban az esetben, ha nem áll rendelkezésére különálló teszt rendszer, akkor a már működő weboldalon is elvégezhetők a szükséges ellenőrzések. Ebben az esetben a fizetési lehetőségek kiválasztásánál praktikus megjeleníteni a vásárlók számára azt, hogy a bankkártyás fizetés teszt alatt van, emiatt vásárláshoz ne használják.

9.4 A kereskedő rendszerének technikai háttere

A tesztek függetlenek attól, hogy a kereskedő milyen szerverkörnyezetben, operációs rendszeren, vagy milyen programnyelven végzi a fejlesztést és üzemelteti a rendszerét, amibe a SimplePay fizetést implementálja.

9.5 Harmadik fél megoldásainak használata

A teszt csak a szerződött kereskedő weboldalának/online fizetésének a SimplePay megfelelőségére vonatkozik. Ennek folyamán nincs vizsgálva az, hogy technikailag milyen egyéb fejlesztés, esetleg harmadik fél szoftverének, vagy online szolgáltatásának felhasználásával valósul meg a fizetési funkció, vagy annak különböző részei.

A fentiekből adódóan a tesztek nem a technikai (rész)megoldást szolgáltató rendszerre vonatkoznak, hanem minden esetben a SimplePay szolgáltatásra szerződő kereskedő partner weboldaláról indítható fizetésre.

Ilyen harmadik fél által fejlesztett, vagy üzemeltetett elemek lehetnek például:

- webáruházak
- beépíthető modulok
- gateway megoldások, API-k
- egyedileg fejlesztett szoftverek

Ha a kereskedő a SimplePay fizetési rendszer alkalmazásához harmadik fél megoldását használja fel, akkor minden esetben a kereskedő hatáskörébe tartozik az általa igénybe vett szoftver, vagy szolgáltatás egyszeri beállítása, vagy folyamatos üzemeltetése.

9.6 Kötelező teszt pontok bankkártyás fizetések esetére

9.6.1 Sikeres tranzakció

- Tranzakció megfelelően végig fut

- A **back** oldalon a „**Sikeres bankkártyás fizetés**" fejezetben leírt tájékoztatások megjelennek
- IPN üzenet fogadása és megfelelő visszajelzés a **0** fejezet alapján

9.6.2 Sikertelen tranzakció

- Tranzakció megfelelően végig fut
- A **back** oldalon a „**Sikertelen bankkártyás fizetés**" fejezetben leírt tájékoztatások megjelennek

9.6.3 Időtúllépés

- Tranzakció megfelelően végig fut
- A **timeout** oldalon a „**Időtúllépés**" fejezetben leírt tájékoztatások megjelennek

9.6.4 Megszakított tranzakció

- Tranzakció megfelelően végig fut
- A **cancel** oldalon a „**Megszakított fizetés**" fejezetben leírt tájékoztatások megjelennek

9.6.5 SimplePay Logo megjelenítése

- A SimplePay logo a **„Logók és tájékoztatók"** c. fejezetben leírtaknak megfelelően megjelenik

9.6.6 Adattovábbítási nyilatkozat

- A szükséges nyilatkozat az **„Adattovábbítási nyilatkozat"** c. fejezetben leírtaknak megfelelően a tranzakció indítás előtt megjelenik

9.7 Nem tesztelt elemek

A SimplePay tesztek minden esetben böngészőn keresztül történnek és kizárólag a SimplePay követelményekkel, fizetési tranzakciókkal kapcsolatosak. Ebből adódóan **az alábbiakat a tesztek nem tartalmazzák**:

- a weboldalt kiszolgáló szerverre (ssh, vagy bármilyen egyéb csatornán) belépés
- a weboldalt kiszolgáló szerver admin felületére webes belépés
- a weboldalt kiszolgáló szerver/adatbázis konfigurálása
- a weboldal adatbázisába belépés
- a weboldal admin felületére belépés
- a weboldal forráskódjának ellenőrzése, szerkesztése
- a weboldal (SimplePay követelményeken túlmenő) technikai működésének tesztelése
- a weboldal (SimplePay követelményeken túlmenő) üzleti logikájának tesztelése
- a weboldal (SimplePay követelményeken túlmenő) kinézetével kapcsolatos tesztelése
- a weboldalba beépített harmadik fél által fejlesztett szoftverek külön tesztelése, konfigurálása
- a weboldalba beépített harmadik fél által üzemeltetett szolgáltatások (gateway, API) külön tesztelése, konfigurálása

10 Támogatás

További információért, technikai támogatásért kérjük lépjen kapcsolatba velünk az itsupport@otpmobil.com címen.

Kérjük, a hatékony ügyintézés végett minden esetben írja meg nekünk azt az adatot, ami alapján be tudjuk azonosítani a problémát, vagy a kérdését.

Tranzakció
Tranzakcióval kapcsolatos kérdés esetén a fizetés **SimplePay** azonosítóját adja meg nekünk. Az sandbox esetében jelenleg **1xxxxxxx**, éles esetén jelenleg **8xxxxxxx** formátumú.

Interface
A tranzakció a V1, vagy a V2 API használatával lett indítva.

Kereskedői fiók
Kereskedői technikai beállításokkal kapcsolatban a SimplePay rendszeren belüli **kereskedői fiók** azonosítót. Az azonosító a fiók **MERCHANT értéke**.

Fizetési rendszer
Melyik rendszerrel kapcsolatos a kérdése. A **sandbox rendszer** csak tesztek esetén, az **éles rendszer** a valós fizetési tranzakciók esetén.

Élesítés
Élesítési tesztek esetén kérjük, hogy jelezze nekünk az itsupport@otpmobil.com címen, hogy

- melyik szerződött domain névhez készült a tesztelhető fizetés
- melyik fiókot használják (MERCHANT)
- hol érjük el a tesztelhető rendszert

Mellékletek

I. Fizetőoldal implementáció mobil kliensbe / social belépés Simple fiókba

A SimplePay fizetőoldal desktop mellett mobil kliensbe is beágyazható. Ilyen esetben is a desktop esetén alkalmazott folyamatok történnek, azaz a fizetőoldalra való

átirányítás majd onnan a redirect vissza a kereskedői rendszerbe, azonban ez a mobil applikációkra jellemző technikai megoldást tesz szükségessé.

Továbbá a SimplePay fizetőoldalon a vásárlónak ebben az esetben is lehetősége van a Simple mobilapplikáció rendszerében regisztrált bankkártyája használatával elvégezni a fizetést. Ilyenkor nem kell a fizetőoldalon kártyaadatot megadni. A regisztrált kártya használatához a vásárlónak a fizetőoldalon be kell lépjen a Simple rendszerébe, ahol el tudja indítani a fizetést. A belépéshez használhatja a social logint is, azaz a Google, vagy a Facebook fiókját.

A social login használata nem igényel semmilyen kereskedő oldali fejlesztést, ugyanakkor a fizetőoldalnak a mobil applikációba történő beágyazásának módja jelentősen befolyásolhatja a folyamat biztonságosságát és hatékonyságát.

A kereskedői **mobil applikációban a CustomTab, illetve SFSafariViewController alkalmazását javasoljuk**, a lentebbi példák szerint.

Ugyanakkor **nem javasoljuk** a SimplePay fizetőoldal WebView / UIWebView-ben történő megjelenítését sérülékenységi okokból kifolyólag, illetve problémát jelenthet a Simple fiókba történő Social (Google / Facebook) belépéskor is.

Android

Chrome Custom Tab használata (Preferált megoldás)
Mintakód:
http://simplepartner.hu/download.php?target=androidexample

A WebView helyett Chrome Custom Tab megnyitása a következőképpen történik

```
CustomTabsIntent.Builder builder = new CustomTabsIntent.Builder();
CustomTabsIntent customTabsIntent = builder.build();
customTabsIntent.launchUrl(MainActivity.this, Uri.parse(url));
```

A CustomTab-ot megnyitó Activity-n az AndroidManifest.xml fájlban kell beállítani a launchMode-ot singleTop, singleTask vagy singleInstance-ra, az adott app felépítésétől függően. Általában a singleTop megfelelő, bonyolultabb alkalmazásoknál jöhet szóba a többi.

A CustomTab-ot megnyitó Activity-re Intent-filter definiálása teljesen egyedi scheme-el az AndroidManifest.xml fájlban

```
<activity android:name=".MainActivity"
    android:launchMode="singleTop">
    <intent-filter android:priority="100">
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <action android:name="android.intent.action.VIEW" />
```

```
        <data
            android:host="uniqueHost"
            android:scheme="uniqueScheme" />
    </intent-filter>
</activity>
```

A tranzakció indításkor az előző pontban megadott egyedi **scheme://host** felépítésnek megfelelő url-t kell átadni redirect url-nek.

További dokumentáció:
https://developer.chrome.com/docs/android/custom-tabs/overview/

iOS

A preferált megoldás a UIWebView UI komponens helyett SFSafariViewController használata. Ezzel a megoldással gyakorlatilag továbbra is az alkalmazáson belül működik a bejelentkezés.
Mintakód:
http://simplepartner.hu/download.php?target=iosexample

```
let safariViewController = SFSafariViewController(url: url)
safariViewController?.delegate = self
present(safariViewController!, animated: true, completion: nil)
```

Másik lehetséges megoldás a Safari webböngésző megnyitása. Ezzel a megoldással kilép az alkalmazásból és megnyit a telefonon egy böngészőt, majd a sikeres bejelentkezést követően a felhasználót visszanavigálja az alkalmazásba.

```
UIApplication.shared.open(url, options: [:], completionHandler: nil)
```

II. Simple applikáció és kereskedői applikáció közötti redirect (deeplink)

Fizetés előtti deeplink kereskedői app-ból Simple app-ba

Abban az esetben, ha a fizetés kereskedői mobil applikációból van indítva és onnan csak Simple appos fizetést szeretne indítani, akkor a start hívásban közvetlenül a Simple app-ra mutató deeplink generálható.
Ebben az esetben a start végponton kapott válasz tartalmában a **paymentUrl** mezőben a deeplinket kapja vissza a kereskedői rendszer.

```
"paymentUrl": "simple://psp/pspHU.SqXNQXMQWzLfuOw7#link"
```

A funkció a kereskedői admin felületen a „Technikai beállítások" oldalon a „Fizető oldali beállítások" panelen a „Simple App navigációs deeplink" checkbox segítségével kapcsolható be/ki.

Fizetés utáni redirect deeplink Simple app-ból a kereskedői app-ba

Abban az esetben, ha a kereskedő mobil applikációjába CustomTab / SFSafariViewController segítségével van beágyazva a SimplePay fizetőoldal, akkor a vásárlónak több lehetősége is van a fizetés elindítására:

- megadhatja a kártya adatokat
- beléphet a Simple rendszerébe e-mail/jelszó használatával, vagy social loginnel
- használhatja a Simple applikációt

Ha a felhasználó a Simple applikációt választja a fizetőoldalon, tehát a kereskedői applikációból átnavigál a Simple applikációba, majd ott végzi el a fizetést a tárolt bankkártyájával, akkor fontos lehet, hogy a fizetés után vissza is tudjon navigálni a Simple app-ból a kereskedői applikációba.

Ehhez a kereskedői rendszernek a tranzakció indításakor a start hívásban meg kell adja a két applikáció közötti vissza irányításhoz szükséges deeplink-et.

A deeplink URL felépítése a kereskedőre van bízva, de követnie kell a szabvány szerinti **schema://host** felépítést, például: merchant://payment/<transactionID>

```
myAppS01234://payment/101010515833121594393
```

A deeplink értéke a **start** hívásban a mobilApp értékeként kell legyen átadva, az alábbi minta szerint

```
"mobilApp": {
          "simpleAppBackUrl": "myAppS01234://payment/123456789"
},
```

Ugyanez az SDK **start** hívásában az alábbi módon adható meg

```
$trx->addGroupData('mobilApp', 'simpleAppBackUrl', 'myAppS01234://payment/123456789');
```

A kereskedői mobil applikációnak fel kell legyen készítve arra, hogy a megadott útvonalon fogadni tudja a Simple applikációból visszaérkező vásárlót. Amennyiben nem megfelelően, vagy hiányosan kerül átadásra a deeplink URL, azt a SimplePay nem tudja javítani, ebben az esetben a navigáció nem biztosított.

A navigáció nem tesz különbséget sikeres és sikertelen tranzakció között, egy pontra tud navigálni, így a kereskedői alkalmazásnak kell kezelnie a megfelelő záróképernyő megjelenítését.
A záróképenyő tartalmához használható a **query** hívás, vagy a backenden fogadott **ipn** adattartalma.

A kereskedő arra a képernyőre is vissza navigálhat, ahonnan a SimplePay fizetőoldalát megnyitotta. A Simple appból történő átirányítás után az ott látható SimplePay fizetőoldal automatikusan elvégzi a további átirányítást, ami a korábban leírt **back** redirect, azaz normál böngészőben történő tranzakcióként lezárható. A redirect utáni folyamat nagyban függhet a kereskedői applikáció egyedi működésétől.

Android

Az AndroidManifest.xml fájlban definiálni kell, hogy az adott deeplinket az alkalmazás kezelni tudja

```
...
<activity
          android:name=".DeepLinkActivity"
          android:label="@string/deeplink_name">
          <intent-filter>
                    <category android:name="android.intent.category.DEFAULT" />
                    <category android:name="android.intent.category.BROWSABLE" />
                    <action android:name="android.intent.action.VIEW" />
                    <data
                              android:host="payment/123456789"
                              android:scheme=" myAppS01234" />
          </intent-filter>
</activity>
...
```

Amint példában használt DeepLinkActivity osztály meghívódik az átirányítás során, abban már elérhetők az adatok:

```
public class DeepLinkActivity extends Activity {
          @Override    protected void onCreate(Bundle savedInstanceState) {
                    ...
                    Uri data = getIntent().getData();
                    ...
          }
}
```

iOS

Az Info.plist fájlban definiálni kell, hogy az adott deeplinket az alkalmazás kezelni tudja.

```
<key>CFBundleURLTypes</key>
    <array>
        <dict>
      ...
            <key>CFBundleURLSchemes</key>
            <array>
                <string>myAppS01234</string>
            </array>
      ...
        </dict>
    </array>
</key>
```

Az URL sémára érkező kérések leszűrhetők az AppDelegate osztályban.

```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
          ...
          func application(_ app: UIApplication, open url: URL, options: [UIApplication.OpenU
RLOptionsKey : Any] = [:]) -> Bool {
          guard url.absoluteString.hasPrefix("myAppS01234") else { return false }
          ...
          return true
          }
}
```

III. EMV 3D Secure

Az Európai Unió, illetve a tagországi törvényhozó és felügyeleti szervek, a vonatkozó Európai Uniós direktíva (Az Európai Parlament és a Tanács 2015. november 25-i (EU) 2015/2366 irányelve; **PSD2**), illetve ennek tagországi jogszabályi adaptálásaalapján ún. kötelező erős ügyfél hitelesítés alkalmazását **(Strong Customer Authentication - SCA)** írják elő, **2019. szeptember 14**-i hatályba lépéssel, minden Európai Gazdasági Térségen belül kibocsátott készpénz-helyettesítő fizetőeszköz elfogadás során.

Az MNB a piaci szereplőkkel történt egyeztetések hatására 12 hónapos meghosszabbított átállási időszakot biztosít a hazai szereplők részére a fenti dátumhoz képest, így az új határidő 2020 szeptember 30.

Hazai jogszabályi háttér

Az Európai Parlament és a Tanács 2015. november 25-i (EU) 2015/2366 irányelve az alábbi jogszabályokba került adaptálásra:

- 2013. évi CCXXXVII. törvény a hitelintézetekről és a pénzügyi vállalkozásokról
- 2009. évi LXXXV. törvény a pénzforgalmi szolgáltatás nyújtásáról
- 2013. évi CCXXXV. törvény az egyes fizetési szolgáltatókról

A PSD2-es módosításokat tartalmazza többek között a 2017. évi 184. Magyar Közlönyben szereplő 2017. évi CXLV. salátatörvény is, továbbá az Európai Bizottság 2018/389 felhatalmazáson alapuló rendelete.
A bankkártya elfogadás tekintetében a szabályozás kiterjed az internetes (úgynevezett VPOS) elfogadásra is. A PSD2 megfelelés értelmében és az internetes kártyaelfogadás biztonságosabbá tétele érdekében, 2020. szeptember 14-ig be kell vezetni és teljeskörűen meg kell valósítani a kártyabirtokos fokozott biztonságú ellenőrzését, az úgynevezett EMV 3D Secure 2.0 szolgáltatást.

3D Secure a gyakorlatban

A rendelethez kapcsolódó technológia bevezetését teljes mértékben elvégzi az OTP Mobil Kft. Abban az esetben, **ha jelen dokumentáció alapján** ún. háromszereplős módon **van igénybe véve a bankkártyás fizetés**, tehát a kereskedői rendszer nem használ kártyatárolásra épülő extra szolgáltatási elemeket, **akkor a kereskedőnek ezzel kapcsolatban nincs további teendője**.

A 3D Secure megvalósulásához ugyanakkor elengedhetetlen a kereskedői rendszerből történő, az EMV 3D Secure 2.0 **szabvány által előírt adatok szolgáltatása**. A kereskedői rendszernek ehhez jelen dokumentációban a „**start**" hívás leírásában megjelölt adatokkal szükséges minden elindított tranzakciót paraméterezni.

Az EMV 3D Secure szabvány követelményéről az alábbi linken tájékozódhat:
https://www.emvco.com/emv-technoligies/3d-secure/

Mi az EMV 3D Secure 2.0 szolgáltatás és miért van szükség rá?

Az utóbbi években széles körben elterjedtek az elektronikus úton történő, kényelmes, akár mobil eszközről kezdeményezett internetes vásárlások. Ezzel együtt megnőtt a

csalások, a számítógépes vagy internetes visszaélések, az adatlopások száma és volumene is.

A bankkártyás fizetések biztonságos és megbízható lebonyolítása érdekében nemcsak a szabályozó szervek, hanem a kártyatársaságok és a bankok is folyamatosan dolgoznak újabb, hatékonyabb megoldásokon. Az internetes vásárlásoknál jelenleg is működő 3D Secure 1.0 a böngészőből indított internetes vásárlások során teszi lehetővé a fizető fél azonosítását. Ez a lehetőség okos eszközről indított vásárlás esetén nem elérhető, mobiltelefonról pedig csak abban az esetben érhető el, ha böngészőn keresztül (nem alkalmazásból) történik a fizetés kezdeményezése.

Az EMV 3D Secure 2.0 egy még biztonságosabb ügyfél hitelesítést tesz lehetővé, és alkalmazható nemcsak a böngésző által vezérelt felületekről indított vásárláskor, hanem az alkalmazásokon belüli (in app) vásárlások, mobiltelefonon és más okos eszközökön bonyolított fizetések során. Az EMV 3D Secure 2.0 olyan ügyfélhitelesítésre alkalmas azonosítási módszerekre támaszkodik, mint a biometria (pl. ujjlenyomatok vagy arcfelismerés), vagy egyszeri jelszavak. A tranzakció során a jelenleginél több adat kerül átadásra a kibocsátó bankok felé, ezzel téve lehetővé az ügyfél megalapozottabb minősítését, az esetleges csalások és visszaélések gyors és hatékony kiszűrését.

Milyen technikai felkészülést igényel a 3D Secure 2.0 bevezetése?

A jogszabály értelmében a 3D Secure 2.0 során az Európai Gazdasági Térségen belül kibocsátott bankkártyákkal az Európai Gazdasági Térségen belül működő elfogadóhelyen végrehajtott tranzakcióknál a kártyabirtokost –néhány kivételtől eltekintve– erős ügyfél hitelesítéssel kell azonosítani. A megfelelő hitelesítés érdekében mind a kibocsátó banknak, mind az elfogadó banknak (vagy az elfogadónak) rendelkeznie kell 3D Secure megoldással (a 3D Secure 1.0-ban az elfogadói megoldás MPI vagy Merchant Plug-in-ként ismert), amely segítségével a vásárlást kezdeményező fél azonosítható.

A jelen dokumentáció alapján fejlesztett és üzemeltetett internetes kártyaelfogadási folyamat a standard háromszereplős modell, amely azt jelenti, hogy az internetes vásárlások során a kártyabirtokos a webáruházi felületről átirányításra kerül az OTP Mobil Kft. által biztosított SimplePay internetes felületre. Ezen a fizetőoldalon adja meg a szükséges kártyaadatokat, a tranzakció ezen a felületen megy végbe, majd ezt követően a vásárló visszairányításra kerül a webshopba.

Ezen fizetési mód alkalmazásakor a bankkártya adatok biztonságos kezeléséért és a továbbításáért (a fizetőfelület és a bank között) a bank felel. Kereskedői oldalon ezzel nincs technikai teendő, ha a 3D Secure folyamathoz szükséges vásárlási adatokat küldi a kereskedő.