Helyzetfüggő Parlay alkalmazások fejlesztése

SCHULCZ RÓBERT

Budapesti Műszaki és Gazdaságtudományi Egyetem, Híradástechnikai Tanszék
rschulcz@mik.bme.hu

Kulcsszavak: Parlay Group, OSA, Parlay API

A cikkben a Parlay Group által kifejlesztett Parlay specifikáció ismertetésére kerül sor, egy egyszerű példán keresztül. A specifikációból a Framework szolgáltatásaival és a User Interaction szolgáltatással foglalkozunk részletesebben. A cikk első felében bemutatjuk a Parlay specifikáció keletkezését, ezek után pedig annak felépítésével, ezen belül mélyebben a Framework biztonsági megoldásaival ismerkedhetünk meg.

1. A távközlési hálózatok fejlődésének nehézségei

A gyors információ-technológiai fejlődés egyik akadálya abban rejlik, hogy a távközlési szolgáltatásokat jelenleg sok egyedi interfész és protokoll jellemzi. Ebből kifolyólag az alkalmazások hordozhatósága igen nehézkes lehet, ugyanannak a szolgáltatásnak a különböző hálózatokban történő üzemeltethetősége sok időt vesz igénybe, valamint a továbbfejlesztése és skálázhatósága is sok többletköltséggel jár. Problémát jelent a különböző hálózatok elszigeteltsége; a hálózatok összekapcsolásához minden egyes szolgáltatás esetén egyedi, az adott rendszerre szabott átjárók beüzemelésére van szükség. Ha meg akarjuk valósítani a hálózatok közötti átjárhatóságot, gyakran különböző protokollok közötti fordítókat kell alkalmaznunk. Az előbb említett problémának volt köszönhető például az, hogy az SMS szolgáltatás vezetékes és mobiltelefon hálózatok közötti átjárhatósága sokáig megoldatlan volt. Sok esetben nehézségeket okoz és hátráltatja a fejlődést a biztonságos hálózat elérés, a szolgáltatás hozzáférés kérdése, valamint a számlázási rendszerek kialakítása is.

Az *Open System Architecture* (OSA) szabvány [3] célja ezen problémakörök megoldása. A szabvány kialakításánál a szolgáltatás hordozhatóság megvalósítását, a hálózatok együttműködésének egy magasabb szintre emelését és a biztonsági, elszámolási kérdések megoldását tűzték ki célként. Ezen OSA elveknek egy konkrét alkalmazói megvalósítása a Parlay API.

A következőkben a Parlay API és felépítésének ismertetésére kerül sor. Szólunk a szolgáltatások eléréséről és igényléséről, majd pedig az utolsó fejezetben egy egyszerű példán keresztül betekintést nyerhetünk a User Location szolgáltatásba is.

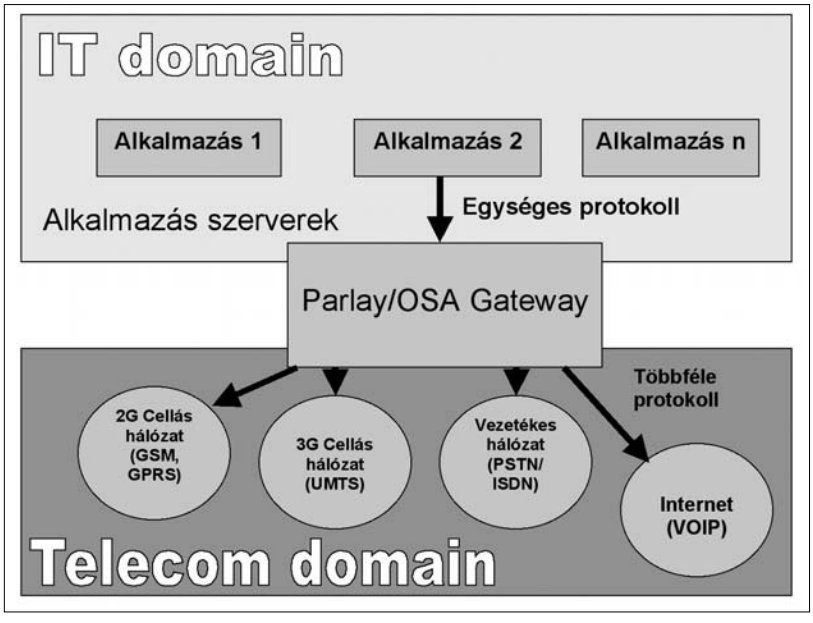
2. A Parlay API

A Parlay specifikáció egy nyílt szabvány, amelyet a Parlay Group dolgozott ki. A szabvány tulajdonképpen egy *Application Programming Interface*-t (API) határoz meg, amelynek segítségével egy alkalmazás úgynevezett Parlay Gateway-jel képes kommunikálni. A Parlay Gateway feladata az alkalmazás számára biztosítani az alatta lévő hálózati réteg által nyújtott szolgáltatásokat [1].

2.1. Parlay Group

A Parlay Group informatikai és távközlési cégekből létrejött konzorcium, mely 1998 áprilisában alakult. Jelenleg 69 tagjuk van. Ebből 13 teljes tag és 46 társtag, gyártók és szolgáltatók. Feladata olyan nyílt, technológiailag független API-k létrehozása, amelyeknek köszönhetően az alkalmazásfejlesztők egyszerűen képesek elérni a távközlési hálózatok által nyújtott szolgálta-

1. ábra A Parlay Gateway szerepe



tásokat, valamint új értéknövelt alkalmazásokat tudnak kialakítani. Egy ilyen nyílt interfész specifikáció a Parlay is, amelynek legelső verzióját 1998 decemberében publikálták, legújabb jelenleg az 5.0 verzió. A Parlay-alapú hordozható, hálózatfüggetlen alkalmazások összekötik az informatika és a telekommunikáció világát azáltal, hogy elfedik magát a hálózati struktúrát a fejlesztő elől.

2.2. A Parlay hatásai

Az internetes szolgáltatások fejlesztési modellje nyílt szabványokon és protokollokon alapul, melyeknek hatására csökkenhetnek a fejlesztések költségei, szélesebb fejlesztői bázis alakulhat ki, és nagy számban jöhetnek létre új szolgáltatások. A távközlési szolgáltatások innovációs modelljét ezzel szemben, a levédetett szabadalmak és a privát protokollok jellemzik. Az új szolgáltatások fejlesztése igen költséges, rengeteg erőforrást és időt igényel, valamint csak korlátozott számban állnak rendelkezésre olyan fejlesztői csoportok, amelyek mélyebben ismerik a felhasználásra kerülő terméket és tisztában vannak az adott hálózat minden lényeges tulajdonságával.

A Parlay API segítségével az internetes fejlesztési modell átmozgatható a távközlésbe és megőrzi annak biztonságos voltát. A Parlay API biztonságos hozzáférést valósít meg az elérhető hálózati szolgáltatásokhoz, ezen felül elfedi a szoftverfejlesztő elől az alkalmazói réteg alatt elhelyezkedő telekommunikációs hálózatot *(1. ábra).* Annak következtében, hogy az API specifikációja nyílt, szélesíti a szereplők köreit (például független szoftverszállítók, alkalmazásszerver-üzemeltetők megjelenése), hatására új fejlett alkalmazások, szolgáltatások kerülhetnek a távközlési piacra.

Fontos megjegyezni, hogy a Parlay Group, az ETSI (European Telecommunications Standards Institute) és a 3GPP (Third Generation Partnership Project) alkotja a Joint API Working Group-ot, amely felelős a Parlay API specifikációkért, miután a 3GPP-nél ez a specifikáció az OSA (Open Service Access) néven fut, ezért a szakirodalomban sokszor hivatkoznak az API-ra úgy is, mint OSA/Parlay vagy Parlay/OSA API.

A Parlay API elvonatkoztat a különböző hálózati protokolloktól. Az üzemeltető felelősége közé tartozik egy Parlay Gateway integrálása a hálózatba, melynek segítségével ő is és mások is használhatják az API-t. Ez az átjáró lefordítja a Parlay API hívásokat az alatta fekvő hálózat szintjére, így biztosítva az alkalmazásfejlesztők Parlay API alkalmazásainak hordozhatóvá válását.

2.3. A Parlay felépítése

Az Open Service Access jellegzetesen három elem köré csoportosul: az alkalmazás, a Framework, és a Service Capability Server (SCS) [4]. A kettő utóbb felsoroltat szokták együttesen Parlay Gateway-nek nevezni. A három egység összefüggéseit a *2. ábra* mutatja.

2.3.1. Framework

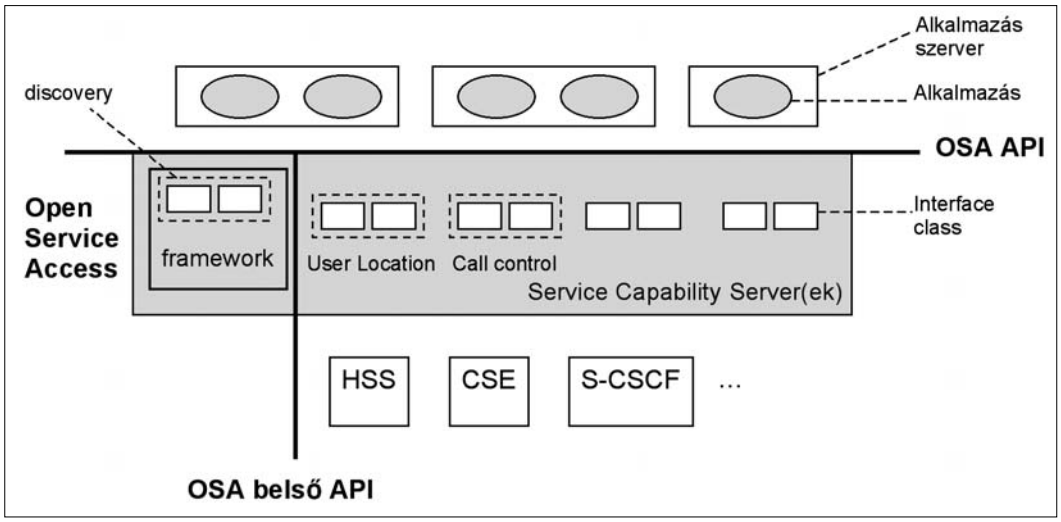
A Framework biztosítja az alkalmazások számára az alapvető mechanizmusokat. Ezen mechanizmusok segítségével az adott alkalmazás felhasználhatja az elérhető szolgáltatások által nyújtott lehetőségeket a hálózatban. A Framework felelős például az autentikációért és a szolgáltatások felderítéséért (discovery), valamint azok hozzáféréséért. Amennyiben egy alkalmazás szeretne csatlakozni egy Parlay Gateway-hez, akkor először csak a Framework-öt képes elérni. A többi szolgáltatást csak azután tudja használni, miután autentikálta magát a Framework segítségével.

2.3.2. Service Capability Server

Egy Parlay Gateway több Service Capability Serverekből (SCS) épül fel, amelyek hálózati funkciókat látnak el. A szerverek nyújtotta hálózati képességeket az alkalmazások az SCS-k által biztosított, Service Capability Feature-ön (SCF) keresztül érik el. Egy Parlay Gateway-ben kötelezően jelen van egy SCS, illetve egy speciális SCS, amely a Framework) [5].

2.3.3. Service Capability Features

A Service Capability Features-t (SCF), szolgáltatásnak is szokás nevezni, mert az alkalmazás az SCF-eken keresztül tudja elérni és felhasználni a hálózat képessé-



2. ábra
A Parlay felépítése

geit. Az SCF-ek elvonatkoztatnak a SCS alatt fekvő hálózattól és így elfedik az alkalmazás elől. Az SCF-ket interfészekként és azok függvényein keresztül specifikálják.

Ahhoz, hogy a Framework-től egy SCF-t el lehessen kérni, az SCF-nek regisztrálnia szükséges magát az OSA internal API-n keresztül. Ha egy arra jogosult autentikált szolgáltatás szeretne egy SCF-et elérni, akkor a Framework az OSA internal API-n keresztül tudja elérni azt és átadni az alkalmazásnak *(3. ábra)*.

A Parlay 5 specifikáció által definiált SCF-k funkcionálisan a következők lehetnek [6]:

– Framework
– Call Control
– Call Control Common Definitions
– Generic Call Control SCF
– Multi-Party Call Control SCF
– Multi-Media Call Control SCF
– Conference Call Control SCF
– User Interaction SCF
– Mobility SCF
– Terminal Capabilities SCF
– Data Session Control SCF
– Generic Messaging SCF
– Connectivity Manager SCF
– Account Management SCF
– Charging SCF
– Policy Management SCF
– Presence and Availability Management SCF
– Multi-Media Messaging SCF

3. Szolgáltatások elérése

Egy Parlay átjáróhoz való kapcsolódás három egyértelműen elkülönülő szakaszra bontható:

1) Kezdeti kapcsolat (Initial Access) felvétele a Framework-kel.
2) Framework felé történő autentikáció.
3) A Framework szolgáltatásainak és a hálózat által nyújtott szolgáltatásoknak (SCF) az elérése.

3.1. Kezdeti kapcsolat (Initial Access) felvétele

Mielőtt egy alkalmazás kihasználhatná az SCF-en keresztül a hálózat által nyújtott szolgáltatásokat, autentikálnia kell magát. Ennek eléréséhez szükség van egy hivatkozásra a kezdeti kapcsolatért (Initial Contact) felelős interfész számára (IpInitial). A hivatkozást megkaphatja egy URL-n, vagy névszolgáltatáson, vagy egy stringgé konvertált objektum referenciáján keresztül is. Az interfész csak az autentikáció megkezdéséhez szükséges függvényeket tartalmaz.

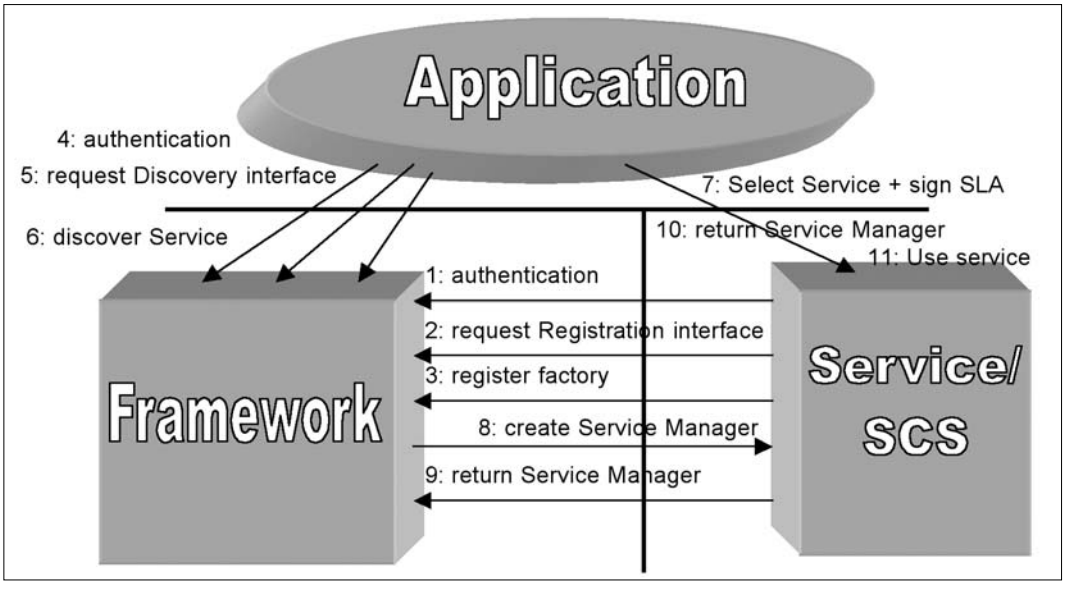
3.2. Autentikáció

A Parlay specifikáció alapértelmezésben a kihívás-válasz alapú PPP *Challenge Handshake Authentication Protocol* (CHAP) szerinti autentikációt használja. Ha az alkalmazás igényeinek nem felel meg a CHAP protokoll, akkor az autentikáció a korábban megkötött Service Level Agreement (SLA) szerinti protokollal megy végbe *(4. ábra)*.

3.2.1. Az autentikáció menete

1) Autentikáció inicializálása
A kliens meghívja az initiateAuthenticationWith Version() eljárást az Initial Contact (IpInitial) interfészen keresztül, amelyre válaszként megkapja a Framework autentikációs interfészének (API Level Authentication interface) referenciáját.

2) Autentikációs eljárás kiválasztása
A kliens futtatja a selectAuthenticationMechanism() eljárást az API Level Authentication interfészen keresztül és megnevezi, hogy milyen autentikációs algoritmust támogat a CHAP használtához.
A Framework eldönti, hogy melyik algoritmust használják a hitelesítéshez. Természetesen az autentikáció egy korábban megosztott közös titkon alapul. A Parlay CHAP alapuló autentikációt használ, amely előírja, hogy minimális elvárás az MD5 hash függvény támogatása.



3. ábra
SCF igénylés

Természetesen a hash függvény is csak abban az esetben kerülhet használatra, ha azt a Framework elfogadja.

3) A Framework autentikálása
A kliens szabadon választhat, hogy szeretné-e autentikálni a Framework-öt. Amennyiben igen, akkor a challenge() eljáráson keresztül küldhet kihívást a Framework-nek.

4) Sikeres autentikáció
A Kliens jelzi a Framework-nek, hogy az autentikáció sikeres.

5) A kliens autentikálása
A Framework meghívja a challenge() eljárást a kliens API Level Authentication interfészén keresztül. Ezen függvényhívás egymás után többször is megtörténhet. A Framework a challenge() eljáráson keresztül adja át a kihívást, amely eljárás visszatérési értéke a klienstől a kihívásra adott válasz. A Framework eldöntheti, hogy autentikálja-e magát addig, amíg a kliens nem tette azt meg. Ellentétben a Framework-kel, a kliensnek azonnal válaszolnia kell a kihívásra.

6) Sikeres autentikáció
A Kliens jelzi a Framework-nek, hogy az autentikáció sikeres.

7) Hozzáférés kérése
A sikeres autentikáció után a kliens futtathatja a requestAccess() eljárást a Framework API Level Authentication interfészén keresztül, amely egy hozzáférési (Access) interfész referenciával tér vissza. Ez az interfész minden kliens számára egyedi. A Framework autentikációjának sikerességétől függetlenül a kliens meghívhatja-e a requestAccess() eljárást.

8) Egyeztetés
A kliens és a Framework megállapodnak, hogy milyen aláíró algoritmust használnak a szolgáltatások eléréséhez szükséges megegyezéseknél.

9) A Framework interfész igénylése
A folyamatnak ezen része arra szolgál, hogy a kliens elérje a Framework által nyújtott funkciókat. Ilyen funkciók például a szolgáltatás-felderítés, vagy a szolgáltatás-regisztráció.

3.3. Az autentikációban résztvevő interfészek

A következőkben az autentikációban résztvevő interfészeket tekintjük át.

3.3.1. IpInitial interfész

Az IpInitial interfész biztosítja a kezdeti hozzáférést a Framework-höz, valamint az autentikációs folyamat elindításához szükséges függvényeket.

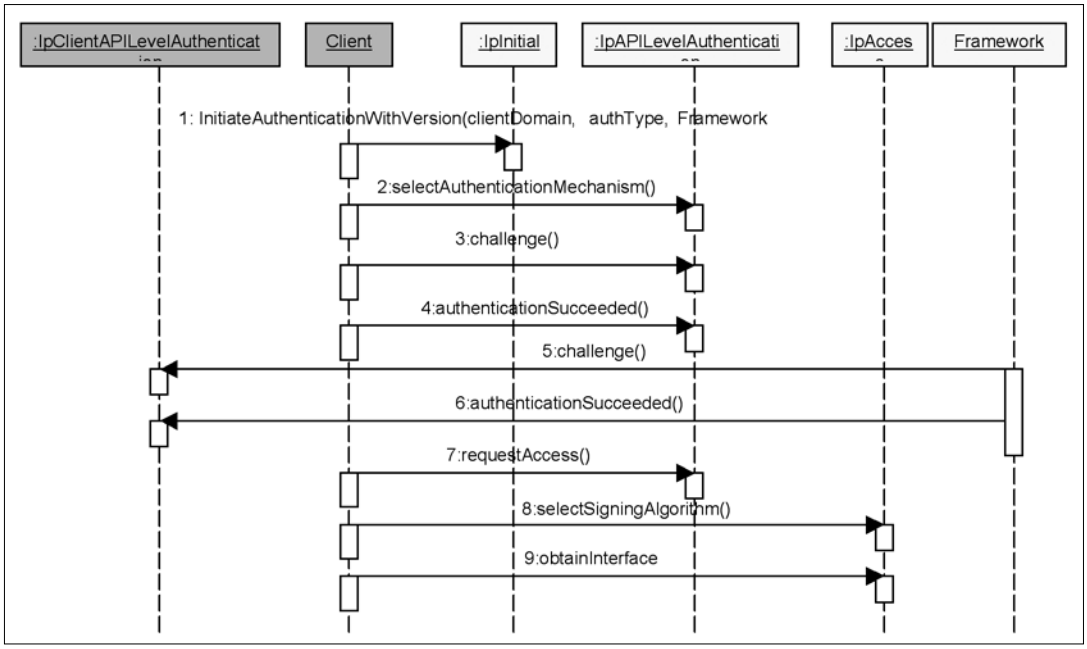
A clientDomain akárcsak a visszatérési érték, a fwDomain is egy struktúra alakú, amelynek segítségével adják át egymásnak az alkalmazás és a Framework az azonosítóját (domainID), valamint az autentikációért felelős interfészére mutató referenciát.

Az authType típusban mondja meg a kliens a Framework-nek, hogy milyen autentikációs módot választ. Az alap érték a P_OSA_AUTHENTICATION, amely esetben az átadott autentikációs interfészek az API Level Authentication interfész. Mind a kliens (IpAppAPILevelAuthentication) és mind a Framework (IpAPILevelAuthentication) részéről. az autentikáció ilyenkor a CHAP protokollt követi.

A P_AUTHENTICATION ezzel szemben azt jelenti, hogy kliens és a Framework már korábban megegyeztek egy alternatív autentikációs folyamatban, így ez esetben az átadott autentikációs interfészek ennek megfelelően alakulnak.

3.3.2. IpAPILevelAuthentication intefész

Az IpAPILevelAuthentication interfész tartalmazza azon függvényeket, amelyek a CHAP protokoll végrehajtásához szükségesek.



4. ábra
Az autentikáció folyamatábrája (Parlay 5-ös specifikáció)

A kliens a selectAuthenticationMechanism függvény segítségével fedi fel a Framework előtt, hogy milyen autentikációs eljárásokat támogat. A támogatott eljárásokat authMechanismList paraméterrel adja át, amelyek közül választja ki a Framework a számára legmegfelelőbbet. Az authMechanism visszatérési értéke a kiválasztott eljárás lesz. Amennyiben a Framework nem talál általa elfogadható eljárást, akkor a P_NO_ACCEPTABLE_AUTHENTICATION_MECHANISM hiba értéket generálja.

A specifikációban definiált alapértékek a következők:

- P_OSA_MD5:
  Az autentikáció az MD5 (RFC 1321) hash függvényt használja.
- P_OSA_HMAC_SHA1_96:
  Ebben az esetben HMAC-SHA1 (RFC 2404) hash függvény használják.
- P_OSA_HMAC_MD5_96:
  Ekkor viszont a HMAC-MD5 (RFC 2403) hash függvényt használják.

A challenge függvény módosítás nélkül a CHAP eljáráson alapul, amelyben a kihívásnak és a válasznak is a protokollban megadott formátumnak kell megfelelnie.

Az abortAuthentication függvény az autentikáció megszüntetésére szolgál. Ez a kliens oldalról csak akkor használható, ha kliens nem kíván a továbbiakban részt venni az autentikációs eljárásban és bont minden kapcsolatot a Framework-kel.

Az authenticationSucceeded() függvény, csak abban az esetben kerülhet meghívásra, ha az alkalmazás is autentikálja a Framework-öt, és mindemellett az autentikáció sikeres volt.

3.3.3. IpClientAPILevelAuthentication interfész

Az IpClientAPILevelAuthentication interfész funkcióban megegyezik a Framework oldali IpAPILevelAuthentication interfésszel. Függvényeik hasonlóak azzal a különbséggel, hogy a kliens oldali interfésznek nincs selectAuthenticationMechanism függvénye.

3.3.4. IpAuthentication interfész

Az IpAuthentication, valamint a belőle származtatott IpAPILevelAuthentication interfész biztosítja a requestAccess függvényt, melyet az autentikáció után kell meghívnia az alkalmazásnak.

Sikeres autentikáció után a kliens requestAccess függvénynek a segítségével tudja elkérni a Frameworktől a hozzáférési interfészét. Ha az autentikáció még nem zajlott le, a függvény a P_ACCESS_DENIED hibakóddal tér vissza. Az alkalmazás a clientAccessInterface paraméterrel adja át a kliens oldali hozzáférési interfészt. Az accessType paraméter segítségével pedig azt adja meg, hogy milyen módon szeretne a Frameworkhöz kapcsolódni.

Ha az accessType értéke P_OSA_ACCESS, akkor a kliens a hagyományos IpAccess interfészt kéri. Ha valamilyen speciálisan a kliens számára kialakított interfészen keresztül szeretné elérni a Framework-öt, akkor az accessType értéke egy előre meghatározott konstans (üzemeltető specifikus). Sikeres visszatérés esetén a kliens megkapja a Framework hozzáférési interfészét, a fwAccessInterface visszatérési értéken keresztül.

3.3.5. IpAccess interfész

Az IpAccess interfész a Framework hozzáférési interfésze, ezen keresztül tudja a kliens a Framework további szolgáltatásait elérni.

Miután a kliensnek hozzáférést szerzett az IpAccess interfészhez, elsőként a selectSigningAlgoritm függvényt kell meghívnia, amely a Parlay 4 specifikációban jelent meg, mint új függvény. Segédletével a kliens megállapodhat a Framework-kel az aláíró algoritmusról. Hasonlóan az autentikációnál felhasznált selectAuthenticationMechanism függvényhez, a kliens egy listát küld az általa támogatott algoritmusokról, amely alapján a Framework kiválasztja a számára megfelelő algoritmust.

A kiválasztott algoritmus lesz a függvény visszatérési értéke. A lehetséges értékek a következők lehetnek:

- P_MD5_RSA_512
- P_MD5_RSA_1024
- P_RSASSA_PKCS1_v1_5_SHA1_1024
- P_SHA1_DSA

A kliens e függvény újrahívásával megváltoztathatja az aláíró algoritmust a működés során, de ha ezt egy aláíró eljárás közben tenné, akkor azt a procedúrát, még a korábbi algoritmus szerint kell végrehajtani.

A Parlay4 specifikációban jelenik meg az új terminateAccess függvény is, amely a kapcsolatbontásra szolgál. A terminationText a bontás okát jelzi, míg a digitalSig, a terminationText digitális aláírása, amellyel a kliens igazolja magát a Framework felé. Ezzel bizonyítja be, hogy a függvényt tényleg ő hívta meg. Amennyiben a digitális aláírás nem egyezik meg a kliensével, akkor a Framework egy P_INVALID_SIGNITURE hibát dob.

Az obtainInterface függvényt akkor kell a kliensnek használnia, ha szeretne elérni egy Framework által nyújtott szolgáltatást. A függvény bemeneti paramétere a szolgáltatás neve (például a Service Discover szolgáltatás, melynek konstansa a P_DISCOVERY). A visszatérési érték a kért szolgáltatás interfészére mutató referencia.

Az obtainInterfaceWithCallback hasonló az obtainInterface függvényhez, azonban akkor van rá szükség, ha olyan szolgáltatást szeretne elérni a kliens, amelynek biztosítani muszáj egy úgynevezett visszahívó (callback) interfészt. Ennek megoldására plusz bemeneti paraméterként megjelenik a clientInterfaceRef interfész referencia, amelyen keresztül a Framework eléri a klienst. Például az obtainInterfaceWithCallback függvényt kell használni, ha a Framework Service Agreement Management szolgáltatását (a szolgáltatás konstansa a P_SERVICE_AGREEMENT_MANAGEMENT) szeretnénk használni.

4. A Framework szolgáltatatásainak igénylése

Miután a kliens hitelesítette magát, a Framework különböző interfészein keresztül hozzáférhet a Framework nyújtotta szolgáltatásokhoz.

Az interfészeket a kliens az obtainInterface, valamint, ha szükséges, akkor az obtainInterfaceWithCallback függvények segítségével képes elérni. A két legfontosabb szolgáltatás a Service Discovery, valamint a Service Agreement Management szolgáltatás.

4.1. Service Discovery

A Service Discovery szolgáltatáson kereszetül tudja az alkalmazás feltérképezni a Framework-nél regisztrált különböző szolgáltatásokat (SCF). Interfésze a Parlay specifikációban az IpServiceDiscovery illesztő egység. A szolgáltatások lekérdezéséhez az interfésznek a listServiceTypes függvényét kell meghívni.

Egy másik fontos tulajdonsága a Discovery szolgáltatásnak, hogy alkalmazásával lehet az egyes szolgáltatok azonosítóját (serciveID) is lekérdezni, amire a szolgáltatások igénylésénél van szükség. A Service Agreement Management szolgáltatás selectService függvényének ezt az azonosítót kell átadni.

4.2. Service Agreement Management

4.2.1 On-line szerződéskötés menete

A Service Agreement Management szolgáltásnak köszönhetően biztosítja a Framework az on-line szerződéskötést. Miután ez megtörtént, a kliens megkapja az úgynevezett SCF-et, és használni tudja az általa nyújtott szolgáltatást.

A szerződéskötés első lépéseként a kliens kiválasztja a szolgáltatást (selectService()), majd megtörténik az on-line szerződéskötés (signServiceAgreement()), amely folyamat végén a kliens megkapja a Framework által aláírt szerződést, valamint egy a kiválasztott SCF kezelői interfészére mutató referenciát *(5. ábra)*.

Az on-line szerződés kötésnél két interfész van jelen. Egy a kliens oldalon (IpAppServiceAgreementManagement), egy pedig Framework oldalán (IpServiceAgreementManagement). A kliens oldali interfész kettő, míg a Framework oldali interfész négy függvényt definiál.

4.2.2. IpServiceAgreementManagement intefész

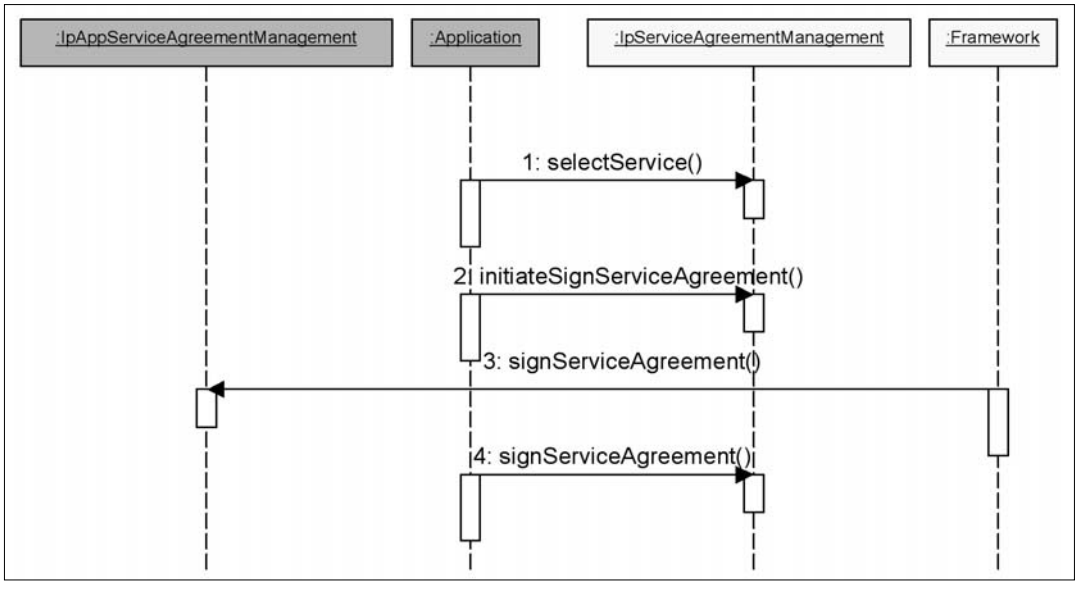
Miután a kliens kiválasztotta, mely SCF-t kívánja elérni, meghívja a Framework Service Agreement Management interfészén keresztül a selectService függvényt. A függvény bemeneti paramétere a kiválasztott szolgáltatás azonosítója (serviceID), melyet még a Service Discovery interfésztől kapott és amelyre válaszul a Framework visszaad egy serviceToken-t.

A serviceToken egy szöveges token, amely formátuma szabadon választott, korlátozott ideig érvényes és ha az idő lejárta után szeretnék felhasználni, akkor a Framework egy P_INVALID_SERVICE_TOKEN hibakóddal tér vissza a kliens felé. Ezáltal véd a token újrafelhasználása ellen.

A kliens miután megkapta a ServiceToken-t, meghívja az initiateSignServiceAgreement függvényt, paraméterként átadva neki a tokent. Ezzel jelzi a Frameworknek, hogy készen áll a szerződés aláírására. A Framework ennek hatására meghívja a kliens Service Agreement Management interfészén (amelyet még az obtainInterfaceWithCallback() hívás során adott meg az alkalmazás) a signServiceAgreement() függvényt.

A Framework oldali signServiceAgreement függvény segítségével irattatja alá a kliens a Framework-kel a szerződést, és szerzi meg az SCF interfészét. A függvényt addig nem lehet meghívni, amíg az alkalmazás alá nem írja a szerződést a kliensoldali signServiceAgreement segítségével, amelyet a Framework hív meg. Ha mégis megpróbálná a kliens meghívni a függvényt az aláírás előtt, akkor a P_INVALID_STATE kivételt dob a Framework.

A signServiceAgreement függvény bemeneti paraméterei, a serviceToken, az agreementText és a signingAlgorithm. A serviceToken megegyezik a selectService függvény által visszaadott típussal, és a szerződés azonosítására szolgál. Az agreementText a Framework által generált szerződés, amelyet a kliens még a saját interfészén meghívott signServiceAgreement függvénnyel kapott meg. A signingAlgorithm paraméter pedig-len az aláíró algoritmust adja meg.



5. ábra
A szolgáltatás kiválasztása

Az aláíró algoritmusról a Parlay 4 specifikációtól kezdve a kliens és a Framework már korábban megállapodik az IpAccess interfésznek selectSigningAlgorithm függvénye segítségével. A függvény visszatérési értéke a SignatureAndServiceMgr két értéket tartalmaz. Az egyik az aláírt szerződés, a másik pedig az SCF interfésze.

Ha a kliens szeretné megszüntetni a szerződést, akkor meghívja a terminationServiceAgreement függvényt. A függvény bemeneti paraméterei a szerződést azonosító serviceToken, a bontás célját megmagyarázó terminationText és a kliens aláírás, az előző két értéken. Az aláírásra azért van szükség, hogy biztosítva legyen a kliens hitelessége.

4.2.3. IpAppServiceAgreementManagement

Miután a kliens jelezte a Framework felé szerződéskötési szándékát, a Framework meghívja a kliens interfészén a signServiceAgreement függvényt, amely teljesen megegyezik a Framework IpServiceAgreementManagement interfészén található signServiceAgreement függvénnyel. Annyi a különbség, hogy a visszatérési értéke az elektronikus aláírás.

A terminateServiceAgreement függvény teljesen ugyanaz, mint a Framework oldali függvény. Természetesen ezt a függvényt a Framework használja a szerződés megszűntetésére.

5. Helymeghatározás a Parlay API segítségével

Ebben a szakaszban bemutatjuk a Parlay API User Location szolgáltatás használatát a H-OSA User Interaction szolgáltatás segítségével.

5.1. Az alkalmazás rövid leírása

Az alkalmazás egy helyzetmeghatározó szolgáltatást (User Location) valósít meg, a felhasználó SMS-t küld egy szolgáltatói telefonszámra, amelyre válaszul kap egy térképet a pozíciójával MMS üzenetben. Az üzenetek kezeléséért a H-OSA User Interaction szolgáltatás a felelős.

5.2. Az alkalmazás megtervezése

Az alkalmazás életciklusa során három jól elkülöníthető részre bontható. Ez az inicializálás, a szolgáltatás, és leállításkor az erőforrások felszabadítása. Az inicializálás során elsőként kapcsolódnia kell az Gateway-hez a Framework-ön keresztül. Ezután a szükséges SCF-ket kell igényelnie a Framework-től. A szolgáltatások megszerzése után regisztrálnia kell az Gateway-ben, hogy milyen számon történő SMS üzenet esetén kér értesítést. Ezek után el kell látnia szolgáltatói feladatát, végezetül pedig, ha leállítják az alkalmazást, fel kell szabadítania a használt erőforrásokat.

5.2.1. Inicializáció

Framework-höz való kapcsolódás

A Framework-höz való kapcsolódást, és SCF igénylést egy a H-OSA specifikációban definiált FWproxy objektum hajtja végre, amely az első Framework-höz intézett kérés előtt végrehajtja az autentikációt, majd a kérést.

SCF igénylése

Az SCF igénylést a FWproxy objektum obtainSCF() függvénye hajtja végre, ahol paraméterként meg kell adni a kért SCF egyedi azonosítóját. Mivel alkalmazásban üzenetek kezelésére és pozíció lekérdezésre van szükség, két különálló SCF kerül felhasználásra. Az egyik a H-OSA User Interaction, mely az üzenetekért, míg a másik a User Location a pozicióért felelős.

SMS figyelés létrehozása

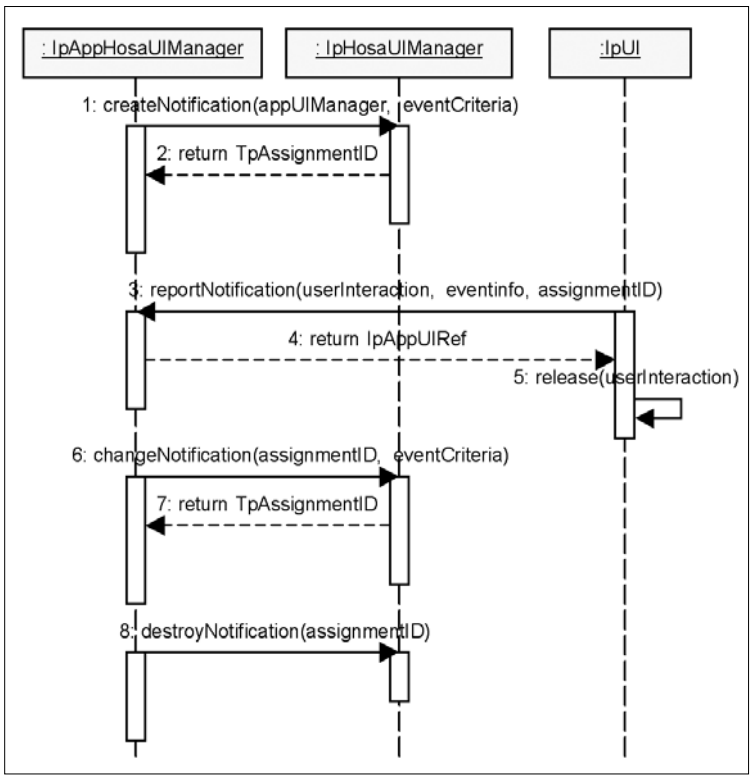
Ahhoz, hogy egy alkalmazás megkapjon bizonyos üzeneteket, először jeleznie kell a Gateway felé, hogy mely üzenetek érkezéséről értesítse az. Hogy az alkalmazás mely üzenetekről kér értesítést, a H-OSA User Interaction Manager interfészének createNotification() függvényvel adhatja meg. A Gateway az értesítéseket és az üzeneteket a kliens interfészének a reportNotification() függvényének segítségével adhatja át.

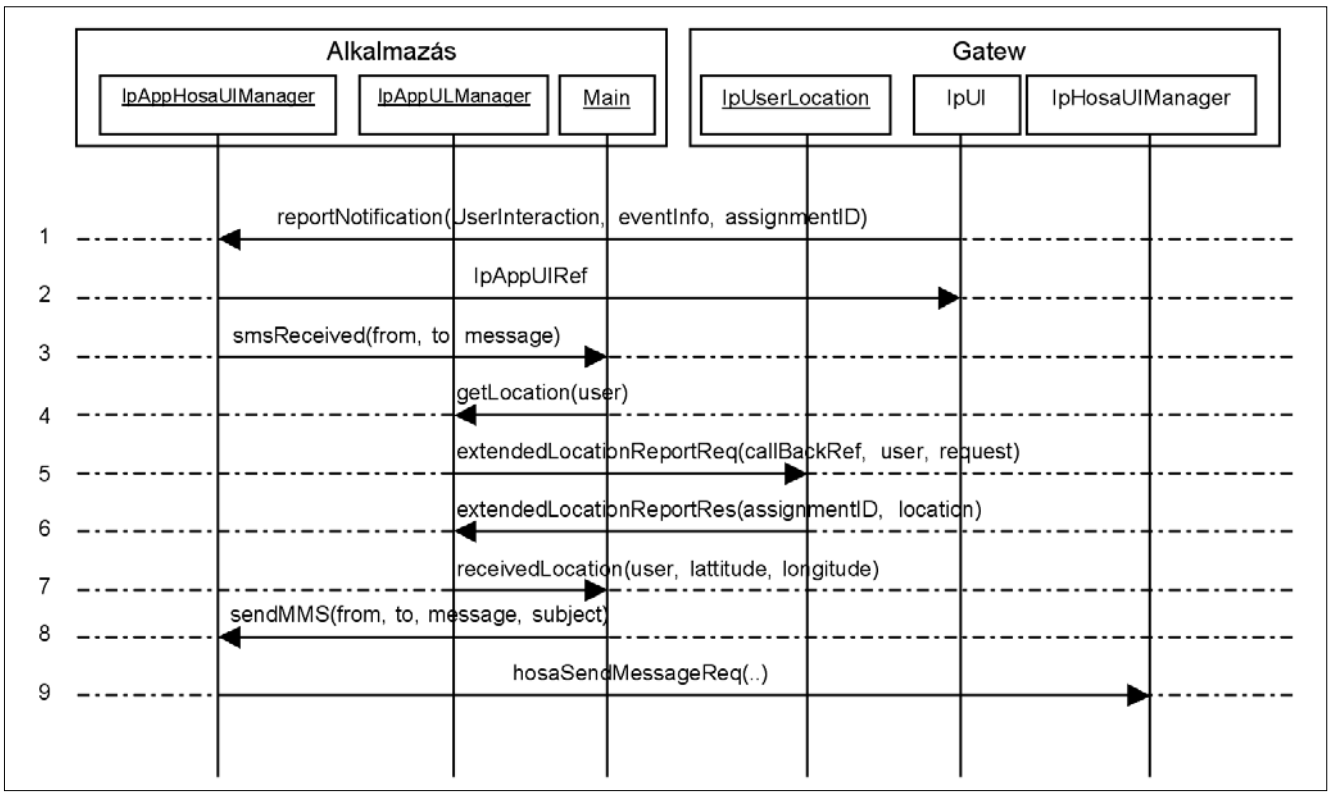
Az üzenetek kezelését a *6. ábra* mutatja.

5.2.2. Szolgáltatás

A szolgáltatás folyamata egyszerűen leírható. Ha üzenetet küld egy felhasználó a szolgáltatásunk SMS számára, akkor azt a Gateway továbbítja az alkalmazásnak, amely ezután lekérdezi a felhasználó pozícióját a User Location szolgáltatáson keresztül.

6. ábra Üzenetek kezelése





7. ábra A szolgáltatás menete

Ha a felhasználó helyzetének pontos értékét megkapja az alkalmazás, akkor generál egy térképet, megjelölve rajta a felhasználó pontos helyzetét. Végül ezt a képet MMS formájában elküldi a felhasználó készülékére.

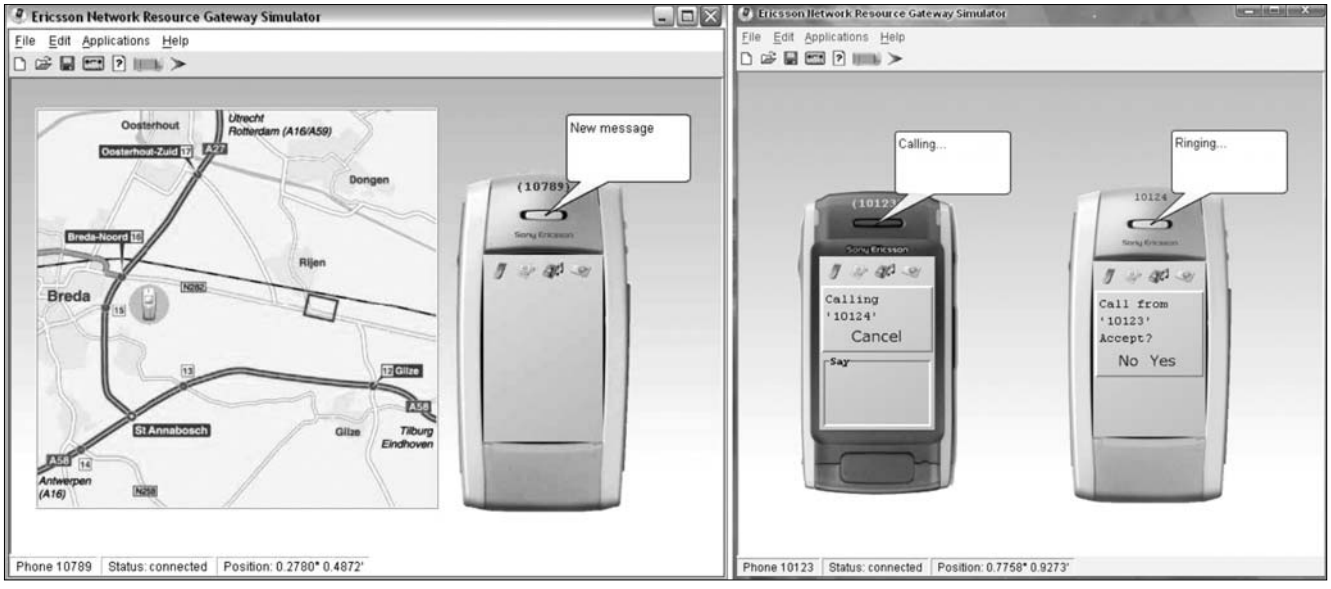
Az alkalmazás szolgáltatásának folyamatát a *7. ábra* szemlélteti.

SMS fogadása

Ahhoz, hogy egy alkalmazás SMS üzenetet kaphasson, létre kell hozni egy objektumot, mely tartalmazza az ehhez szükséges interfész (IpAppHosaUIManager) implementációját. Ugyanis miután létrehoztuk az értesítést a createNotification() segítségével, ezen objektum reportNotification() függvényét fogja meghívni a Gateway, ha üzenetet küldött egy felhasználó a szolgáltatás címére. Ha ez megtörtént, a függvény értesíti a főosztályt, hogy SMS üzenet érkezett, valamint továbbítja a feladó számát és az üzenet tartalmát. Ezt követően a főosztály lekérdezi a felhasználó pozícióját.

Pozíció lekérdezése

A pozíció megszerzéshez az alkalmazás oldalán implementálni kell a pAppULManager interfészt, amely interfészen keresztül tudja az NRG [9] visszaadni nekünk a felhasználó pozícióját. A pozíció megszerzéséhez két függvényre van szükség. Az egyik extendedLocation-



ReportReq(), amit az NRG User Location interfészén keresztül tudunk meghívni. Miután az NRG lekérdezte a felhasználó pozícióját, visszaadja azt az alkalmazásnak az IpAppULManager interfész extendedLocationReportRes() függvénynek segítségével.

MMS küldése

Miután az alkalmazás megszerezte a pozíciót, létrehoz egy képet, amely egy térkép, bejelölve rajta a felhasználó helyzetét, majd ezt a képet elküldi a felhasználónak MMS üzenet formájában. Az MMS üzenet elküldéséhez a kliens a H-OSA User Interaction szolgáltatást használja, amelynek segédletével az SMS figyelést is létrehozta.

Az üzenet elküldéséhez a hosaSendMessageReq() függvényt kell majd használnia az alkalmazásnak, valamint implementálni kell a hosaSendMessageRes(), illetve a hosaSendMessageErr() függvényeket, amelyek nyugtázásra, illetve hibakezelésre szolgálnak.

5.2.3. Erőforrások felszabadítása

Az alkalmazás leállításakor mindenképpen gondoskodni kell a lefoglalt erőforrások felszabadításáról is. Ezen erőforrások az általunk létrehozott objektumok, valamint az elkért szolgáltatások.

Az elkért szolgáltatásokat az on-line szerződések megszüntetésével (releaseSCF()) lehet felszabadítani. Miután az összes általunk lefoglalt erőforrást megszüntettük, bontanunk kell a kapcsolatot a Framework-kel (endAccess()).

6. Összefoglalás

A Parlay API specifikációról elmondható, hogy nagyon jó alternatívát biztosít az internetes világ és a telekommunikációs világ összekötésére. Megoldja a telekommunikációs hálózatok fejlesztési problémáit, ráadásul nyílt specifikáció, így bárki által elérhető, ezzel megoldva a könnyebb átjárhatóságot is.

Teszteszközeinek segítségével, sokkal könnyebben megjósolható egy kifejleszteni kívánt alkalmazás sikere, így csökkenti a veszteséges beruházási kockázatot. Az is látható, hogy a Parlay API megfelelő specifikációjának köszönhetően, viszonylag könnyen és gyorsan lehet értékes alkalmazásokat fejleszteni.

Irodalom

[1] Parlay Group, http://www.parlay.org

[2] 3rd Generation Partnership Project (3GPP), http://www.3gpp.org

[3] OSA API Joint Working Group Document Management Policy, http://portal.etsi.org/docbox/tispan/open/osa/ETSI_Parlay_3GPP_Correspondence.pps

[4] 3rd Generation Partnership Project; Technical Specification GroupServices and System Aspects; Virtual Home Environment (VHE) / Open Service Access (OSA) (Release 6), 3GPP TS 23.127 V6.1.0 (2004-06)

[5] Chelo Abarca et. all: Parlay/OSA: an open API for service development http://portal.etsi.org/docbox/tispan/open/osa/AllAboutParlayOSA.pps

[6] ETSI Open Service Access (OSA); Application Programming Interface(API); Part 1: Overview (Parlay 5), ETSI ES 203 915-1 V1.1.1 (2005-04)

[7] ETSI Open Service Access (OSA); Application Programming Interface(API); Part 3: Framework(Parlay3), ETSI ES 201 915-3 V1.5.1 (2005-02)

[8] ETSI Open Service Access (OSA); Application Programming Interface(API); Part 3: Framework(Parlay5), ETSI ES 203 915-3 V1.1.1 (2005-04)

[9] Ericsson Network Resource Gateway Software Development Kit User Guide: http://www.ericsson.com/mobilityworld/sub/open/technologies/parlay/tools/parlay_sdk

[10] Ericsson H-OSA Interface Specification Generic User Interaction (2005-11-15), 11/155 19-1/FAM 901 253 Uen A