Tomi Järvi

GSMA OneAPI-standardi kehittäjän ja operaattorin kannalta

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

17.5.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Tomi Järvi GSMA OneAPI-standardi kehittäjän ja operaattorin kannalta 31 sivua 17.5.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	
Ohjaaja	yliopettaja, Matti Puska

Tässä insinöörityössä oli tavoitteena perehtyä GSMA OneAPI-standardin versioon 3 ja siinä määriteltyihin REST-pohjaisiin API-rajapintoihin. Osana työtä käytiin läpi OneAPI standardi, REST-arkkitehtuurityyli ja kuinka REST-rajapinnat toimivat kehittäjän ja operaattorin kannalta.

GSMA OneAPI-standardin tarkoituksena on määritellä API-rajapintoja, jotka pohjautuvat REST-arkkitehtuurityyliin. Standardin myötä operaattoreille avautuu mahdollisuus julkaista verkkoresurssejaan kolmansille osapuolille standardoitujen API-rajapintojen avulla. Verkkoresurssit voivat olla mitä tahansa operaattorin omistamaa tietoa tai hallinnoimaa toiminnallisuutta. OneAPI versiossa 3 on määritelty seitsemän API-rajapintaa ja kolme beta-versiota.

Kehittäjän ja operaattorin kannalta OneAPI-standardi mahdollistaa yhtenäisen tavan verkkoresurssien julkaisuun erilaisten sovelluksien tarpeisiin. Insinöörityön tuloksena havaittiin, että OneAPI-standardiin perustuvia toteutuksia löytyy maailmalta vain muutamilta operaattoreilta, joten vielä ei voida sanoa, että tulevatko toteutukset yleistymään.

Avainsanat	GSMA OneAPI, REST, HTTP

Author	Tomi Järvi
Title	GSMA One API standard from developer and operator point of view
Number of Pages	31 pages
Date	17 May 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	
Instructor	Matti Puska, Principal Lecturer

This thesis focuses on the GSMA OneAPI standard, version three, where REST-based APIs are specified. In detail, this work concentrates on the OneAPI standard, the REST architectural style, and how the REST interfaces work from developer and operator point of view.

In general, the aim of the GSMA OneAPI standard is to define the APIs which are based on the REST architecture style. The standard gives an opportunity for operators to publish their network resources for third parties via standardized API interfaces. The network resources can include any information or functionality owned by an operator. In the OneAPI standard, version three, seven API interfaces and three beta versions are defined.

From the developer and operator point of view, the OneAPI standard allows a uniform way to publish network resources for various application needs. As a result of compiling this thesis, it can be concluded that only a small number of operators use OneAPI standard implementations worldwide, so it is not yet possible to determine if the implementations will be commonly used.

Keywords	GSMA OneAPI, REST, http

Sisällys

Lyhenteet

API	*Application Programming Interface.* Ohjelmointirajapinta, jonka mukaan eri sovellukset voivat keskustella keskenään.
CRUD	*Create, Read, Update, Delete.* Tiedon muutoksen kuvaamiseen käytettävä termistö.
DTMF	*Dual-tone Multi-frequency.* Puhelinlaitteissa käytetty äänitaajuusvalintatapa.
GSMA	*GSM Association.* Mobiili-operaattoreiden ja laitetoimittajien yhdistys.
HTTP	*Hypertext Transfer Protocol.* Tiedonsiirto protokolla, jota mm. selaimet käyttävät tiedonsiirtoon.
HTTPS	*Hypertext Transfer Protocol Secure.* HTTP-protokolla tiedon suojattuun siirtoon.
IMEI	*International Mobile Equipment Identity.* Matkapuhelimen laitetunnus.
IVR	*Interactive Voice Response.* Vuorovaikutteinen äänivaste, jolla voidaan totetuttaa palveluvalikoita.
JSON	*JavaScript Object Notation.* JavaScriptiin pohjautuva kevyt tiedonsiirtomuoto.
MMS	*Multimedia Message Service.* Mobiiliviestinnän palvelumuoto multimediaobjektien välittämiseen.
Nokia EAIF	*Nokia External Application Interface.* Nokian kehittämä rajapinta MMS-viestien välittämiseen.
OMA	*Open Mobile Alliance.* Mobiili-operaattoreiden ja laitetoimittajien standardointifoorumi.

REST	*Representational State Transfer.* HTTP-protokollaan perustuva arkkitehtuurityyli.
SDK	*Software Development Kit.* Kehitystyökalu sovelluksien luomiseen.
SMPP	*Short Message Peer-to-Peer.* SMS-viestien välitykseen tarkoitettu protokolla.
SMS	*Short Message Service.* Mobiiliviestinnän palvelumuoto tekstiviestien välittämiseen.
SOAP	*Simple Object Access Protocol.* Tiedonsiirtoprotokolla sovelluksien väliseen kommunikointiin.
SSL/TLS	*Secure Sockets Layer/Transport Layer Security.* Tietoliikenteen salausprotokolla, jolla voidaan suojata Internet-sovelluksien tietoliikenne.
UCP/EMI	*Universal Computer Protocol/External Machine Interface.* SMS-viestien välitykseen tarkoitettu protokolla.
URI	*Uniform Resource Identier.* Osoite, jolla viitataan tiettyyn resurssiin.
XML	*Extensible Markup Language.* Merkintäkieli, jonka avulla kuvataan tietoa tiedosta.

1 Johdanto

Insinöörityössä on tarkoitus tarkastella GSMA:n (GSM Association) OneAPI-standardin versiota kolme ja sen tarjoamia API-rajapintoja (Application Programming Interface) kolmansille osapuolille, kuten sovelluskehittäjille ja palveluntarjoajille. Vaikka OneAPI-standardi ei määrittele operaattorin verkkoresursseihin tehtäviä integraatiota ne tulee kuitenkin huomioida toteutusta suunniteltaessa. Osana työtä käydään läpi REST-arkkitehtuurityyli (Representational State Transfer), johon GSMA OneAPI-standardi pohjautuu.

Pääsääntöisesti operaattoreiden verkkoresurssit ovat olleet hyvin rajallisesti saatavilla ulkopuolisille, mutta OneAPI:n myötä verkkoresurssien avaaminen kehittäjille tehdään mahdolliseksi standardoitujen REST-rajapintojen kautta. Versiossa kolme rajapinnat on julkaistu ainoastaan REST-pohjaisina ja aikaisempien versioiden SOAP-rajapinnat (Simple Object Access Protocol) on jätetty pois. Kehittäjille tarjoutuu näin yksi yhtenäinen tapa hyödyntää operaattoreiden verkkoresursseja erilaisissa sovelluksissa.

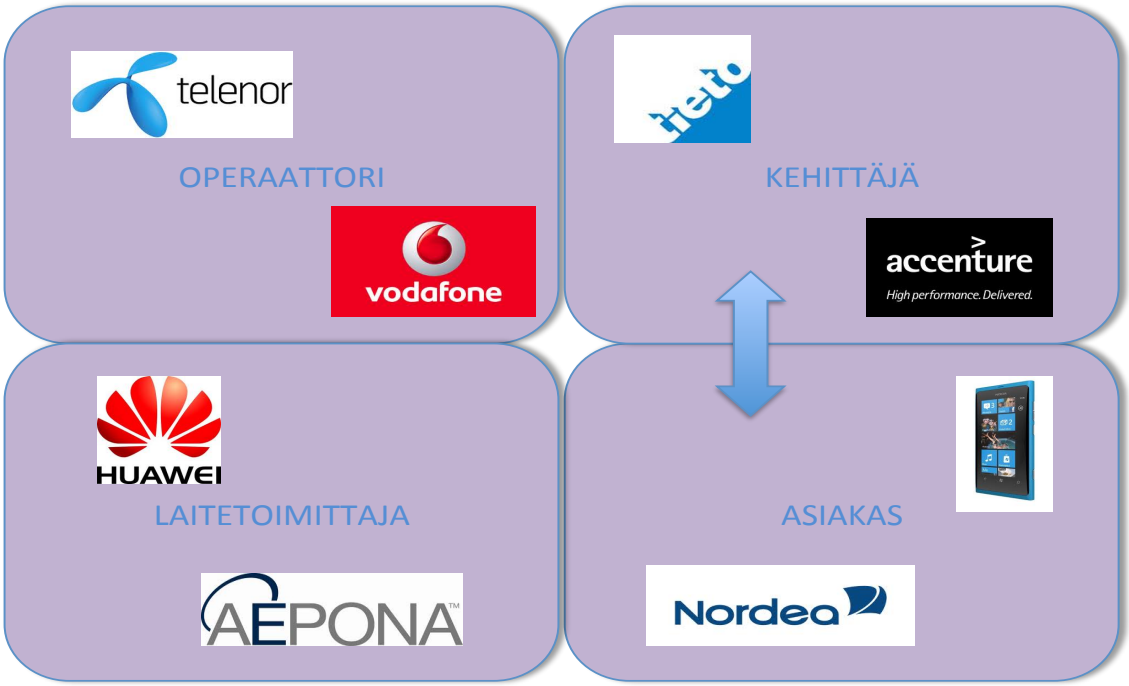
Työn keskeisenä tavoitteena on ymmärtää OneAPI-standardi, REST-arkkitehtuurityyli ja se kuinka REST-rajapinnat toimivat kehittäjän ja operaattorin kannalta. Työn kannalta merkittäviä asioita ovat itse standardi, rajapintojen hyödyntäminen sovelluksissa ja kuinka operaattorit toteuttavat integraatiot verkkoresursseihinsa.

2 GSMA OneAPI-standardi

GSMA OneAPI-standardi julkaisee joukon OMA:n (Open Mobile Alliance) REST NetAPI-standardoituja ja yleisesti tuettuja Web-sovellusrajapintoja, jotka ovat kirjoitettu uudelleen pitäen silmällä helppokäyttöisyyttä ja ymmärrettävyyttä. Näiden avulla verkko-operaattorit voivat julkaista erilaisia verkkoresursseja ulkopuolisille kehittäjille. Julkaistujen APIen avulla voi muun muassa lähettää ja vastaanottaa teksti- ja multimediaviestejä, hyödyntää puhelimen paikkatietoa tai luoda erilaisia laskutusmalleja sovelluksien tarpeisiin. Tarkoituksena onkin tarjota yhtenäinen ja globaali tapa sovellusrajapintojen julkaisuun riippumatta verkko-operaattorista. Kehittäjän kannalta tämä tarkoittaa, että julkaistut rajapinnat ovat yhdenmukaiset operaattorista riippumatta ja samat sovellukset voidaan näin tarjota useiden operaattoreiden kautta laajemmalle kohderyhmälle [8].

Sidosryhmät

Yksi tärkeä osa API-rajapintojen julkaisussa on ymmärtää sidosryhmien roolit. Tässä tapauksessa sidosryhmiä ovat kuvassa 1 kuvatut neljä toimijaa: operaattori, kehittäjä, laitetoimittaja ja loppuasiakas.



Kuva 1. GSMA OneAPI sidosryhmät esimerkkeineen.

Operaattori

Operaattoreilla on tarjolla erilaisia API-ratkaisuja, joiden kautta on julkaistu verkkoresursseja yrityksille ja kehittäjille. Ratkaisut perustuvat API-kirjastojen tai protokollarajapintojen hyödyntämiseen kuten esimerkiksi Soneran Content Gateway-ratkaisussa. Soneran Content Gateway-ratkaisussa on mahdollistettu SMS:n (Short Message Service) ja MMS:n (Multimedia Message Service) lähetys ja vastaanotto, joka on toteutettu asiakkaalle asennettavasta Provider Server-ohjelmistosta ja Soneran hallitsemasta Operator Serveristä. Ratkaisun avulla yritys voi käyttää valmiita sovelluksia, kuten viestien lähetystä tai rakentaa API-kirjastoja ja protokollarajapintoja, käyttäen omia sovelluksia. Tuetut protokollat Soneran ratkaisussa ovat pääsääntöisesti ns. verkkoprotokollia, kuten UCP/EMI (Universal Computer Protocol/External Machine Interface) tai SMPP (Short Message Peer-to-Peer) SMS-viestin lähetykseen ja vastaanottoon, joita sovellukset pystyvät hyödyntämään [7, s. 4–8].

GSMA OneAPI:n myötä operaattoreilla on mahdollisuus julkaista standardoituja rajapintoja niin yrityksille kuin kehittäjille. Operaattorin kannalta tämä luo uusia liiketoimintamahdollisuuksia uusien sovelluksien muodossa ja laajemman kohdeyleisön tavoittamista niin kehittäjien kuin loppukäyttäjän näkökulmasta. Näitä sovellus esimerkkejä käydään läpi luvun neljä yhteydessä myöhemmin. Yhtenä etuna operaattorin kannalta on, se pystyy hyödyntämään olemassa olevia verkkoresursseja ja näin ollen investointeja verkkoresursseihin ei tarvitse tehdä [8].

Kehittäjä

Kehittäjän ja asiakkaan välinen riippuvuus on ilmeinen eivätkä nämä sidosryhmät aina olekaan eroteltavissa toisistaan. Kehittäjä voi toimia myös asiakkaan roolissa, jolloin kehittäjä, esimerkiksi Accenture voi toteuttaa sovelluksia itselleen, jolloin kehittäjä toimii myös asiakkaana.

Kehittäjän kannalta OneAPI:n käytön lisääntyminen on suorastaan ihanteellista, koska sovelluksien käyttö eri operaattoreiden verkoissa tulee mahdolliseksi ja näin voidaan välttää operaattorille räätälöidyt ratkaisut [8]. Tätä voidaankin verrata Applen App storen kaltaisiin toteutuksiin, jossa sovellukset ovat julkaistavissa operaattoririippumattomasti.

Toteutuksena GSMA OneAPI-pohjainen ympäristö on pyritty tekemään kehittäjän kannalta helpoksi toteuttaa. Helppokäyttöisyys näkyykin hyvin REST-pohjaisuutena ja HTTP:n (Hypertext Transfer Protocol) päälle tehdystä toteutuksesta, jolloin API:t ovat käytettävissä siihen soveltuvalla HTTP-clientilla. GSMA OneAPI:iin tutustumiseen ja testaukseen löytyy ns. Sandbox-toteutuksia, kuten Aeponan Developer Account https://oneapi.aepona.com/, jonka kautta kehittäjän on helppo tutustua julkaistuihin OneAPI-rajapintatoteutuksiin esimerkkeineen. OneAPI perustuukin avoimeen kommunikaatioon ja standardin mukaisesti julkaistuihin API-rajapintoihin [8].

Laitetoimittaja

Laitetoimittajille OneAPI-standardi antaa luonnollisen kehityspolun perinteisien palvelujärjestelmien rinnalle, kuten lyhytsanomakeskus (SMSC) tai multimediaviestikeskus (MMSC). Näitä hyödynnetään, kun verkkoresursseja julkaistaan kolmansille osapuolille. Periaatteessa OneAPI-toteutusta voidaan myös miettiä tilanteessa, jossa verkkoprotokolla, esimerkiksi SMPP lyhytsanomaviestien välityksessä, konvertoidaan noudattamaan OneAPI-standardia. Laitetoimittajien kannalta siirtymä OneAPI-standardin mukaiseen toteutukseen ei ole kovinkaan suuri panostus, mikäli laitetoimittajalla on jo kokemusta operaattoreiden palvelujärjestelmistä.

Asiakas

Käyttäjän kannalta OneAPI tuo varmasti uusia innovatiivisia palveluita ja sovelluksia tarjolle operaattoririippumattomasti, mikäli operaattorien OneAPI toteutukset yleistyvät.

3 REST-arkkitehtuurityyli

REST määrittelee arkkitehtuurillisia periaatteita ja rajoitteita, joiden avulla pystytään toteuttamaan webpalveluita hyödyntäen olemassa olevia standardeja, kuten HTTP, URI (Uniform Resource Identifier) ja XML (Extensible Markup Language). REST-arkkitehtuurityyli tähtää yhteentoimivuuden säilyttämiseen sellaisissa järjestelmissä, joissa eri osapuolet kehittyvät ja muuttuvat itsenäisesti toisistaan riippumatta. REST ei siis määrittele itse komponenttien toteutusta eikä edes protokollatason toimintaa [1].

REST:n (Representational State Transfer) avainkäsite on resurssi, joka voi olla mikä tahansa asia, joka voidaan nimetä, esimerkiksi dokumentti tai kuva. REST-arkkitehtuurin mukaisesti jokaisella resurssilla ainoa pakollinen asia on tunniste, joka RESTin tapauksessa on URI [2, s. 88–90]. URI on merkkijono, jolla resurssit pystytään erottamaan toisistaan yksilöllisesti. Informaatio, joka resurssin kutsumiseen ja tunnistamiseen tarvitaan, sisältyy URI:iin. Tavallisin URI-skeema on HTTP, jossa resurssilla on yksilöivä tunniste. Tarkempi URI:n rakenne on seuraava:

http://example.com/sovellus/resurssi/tunniste

Jos resurssin tunniste jätetään pois polun lopusta, viitataan koko resurssikokoelmaan ja palvelu yleensä tulostaa tässä tilanteessa kaikki mahdolliset resurssin jäsenet [1].

3.1 Metodikutsujen esitysmuoto

REST-periaatteiden mukaan asiakkaalla ja palvelimella on oltava yhteinen esitysmuoto, jonka molemmat osapuolet ymmärtävät. REST ei suoraan rajoita käytettäviä esitysmuotoja. REST webpalveluissa yksi resurssin URI-osoite voi tukea useita eri esitysmuotoja. Resurssin eri esitysmuodot ovat haettavissa metodikutsulla, jossa esitystapa erotetaan resurssin nimen perään liitettävällä tunnisteella, esimerkiksi .xml tai .json. Yleisimmin web-palveluissa käytössä olevia esitysmuotoja ovat xml ja json (JavaScript Object Notation) [1].

JSON

JSON on kevyt ja melko yksinkertainen tiedonvälityksen formaatti, joka perustuu JavaScriptin syntaksiin, kuten esimerkkikoodissa 1 esitetty. Lisäksi JSON on helppo ihmisen lukea, kirjoittaa ja koneen jäsentää ja generoida. Vaikka JSON perustuu JavaScriptiin, se on ohjelmointikielestä riippumaton, sen vuoksi sitä voidaan käyttää muun muassa C++-, Java- sekä Perl-ohjelmointikielissä. Nämä ominaisuudet tekevätkin siitä yleisesti käytetyn tiedonvälityksen formaatin [5].

```
{”lasku”:{
          ”saaja”: ”Matti Meikäläinen”,
”osoite”: ”Kielokuja 3”,
”summa”: ”250$”
}}
```

Esimerkkikoodi 1. JSON-formaatin esimerkki.

XML

XML on niin kutsuttu meta-/merkintäkieli, jonka avulla kuvataan tietoa tiedosta, eli sitä mitä tieto tarkoittaa. Tieto koostuu tekstimuotoisesta XML-dokumentista, jonka avulla tietoa pystytään siirtämään ja tallentamaan täsmällisessä muodossa. XML määrittääkin tiedon rakenteen, merkityksen, kuten esimerkkikoodissa 2 esitetty ja toimii laitteisto- ja ohjelmistoriippumattomasti [6].

```
<?xml version=”1.0” encoding=”UTF-8”?>
<lasku>
<saaja>Matti Meikäläinen</saaja>
<osoite>Kielokuja 3</osoite>
<summa>250$</summa>
</lasku>
```

Esimerkkikoodi 2. XML-formaatin esimerkki.

3.2 Resurssien tilamuutokset

Resurssin tilan käsite on yksi oleellinen osa REST:n periaatteita. Käytännössä metodikutsuilla muutetaan resurssin ja asiakkaan tilaa käyttäen HTTP-metodeita. REST web-palveluissa resurssien tai resurssiryhmien tilamuutokset eli CRUD-operaatiot tehdään käyttäen http-metodeita POST, GET, PUT, DELETE. Alla on listattu resurssien tilamuutoksissa käytetyt HTTP-metodit ja niiden yhteys CRUD-operaatioihin. Niillä pystytään

määrittelemään operaatio, joka tietylle resurssiryhmälle tai yksittäiselle resurssille halutaan tehdä [1; 10, s. 53–56].

HTTP-metodit:

- resurssin luonti eli Create käyttämällä POST-metodia
- resurssin luku eli Read käyttämällä GET-metodia
- resurssin tilan muutos eli Update käyttämällä PUT-metodia
- resurssin poisto eli Delete käyttämällä DELETE-metodia.

3.3 REST-arkkitehtuurin rajoitukset

Asiakas-palvelinarkkitehtuuri

Ensimmäinen rajoite on asiakas-palvelin arkkitehtuurimallin (eng. Client/Server) käyttö, joka muodostuu palvelinprosessista, yhdestä tai useammasta asiakasprosessista ja niille yhteisestä yhteyskäytännöstä eli protokollasta. Ideana on siis jakaa sovelluksen toiminnallisuus kahteen alueeseen, jolloin asiakas lähettää palvelimelle palvelusta pyynnön, johon palvelin vastaa tarjoamalla kyseistä palvelua tai hylkäämällä pyynnön [2, s. 45–46, 78].

Tilattomuus

Tilasiirtymät ovat REST-arkkitehtuuriperiaatteiden mukaisesti tilattomia. Kaikki informaatio joka resurssin hakuun tai resurssin muokkaamiseen tarvitaan sisältyy itse pyyntöön. Asiakkaan pyyntö voidaan ymmärtää ilman, että käsitellään asiakkaan tekemiä aikaisempia pyyntöjä. Näin ollen kaikki mahdolliset istunnon hallintaan liittyvät tiedot ovat tallennettuna asiakaspuolelle, eikä palvelin tallenna mitään asiakkaan istuntoa koskevaa tietoa. Edut joita tilattomuudella saavutetaan, liittyvät näkyvyyteen, luotettavuuteen ja skaalautuvuuteen. Näkyvyydellä tarkoitetaan järjestelmän monitorointia, jolloin järjestelmän ei tarvitse ottaa kantaa aikaisempiin pyyntöihin, mikäli halutaan selvittää yksittäisen pyynnön luonne. Luotettavuus ja skaalautuvuus paranevat, koska

pyyntöjä ei tallenneta palvelimelle, jolloin palautuminen virhetilanteista helpottuu ja palvelimen ei tarvitse pitää yllä aikaisempia pyyntöjä [2, s. 47–48, 78–79].

Yhtenäinen rajapinta

Keskeinen piirre, joka erottaa RESTin muista web-palvelukonsepteista, yhtenäinen rajapinta komponenttien välillä. Yhtenäisen rajapinnan takaamiseksi REST:ssä on määritelty neljä rajoitusta rajapintaan:

- resurssien identifiointi käyttäen tunnistetta
- resurssien tilamuutokset esitysmuodon kautta
- itsekuvautuvat viestit, jolloin pyynnössä on kaikki tarpeellinen tieto jatkokäsittelyä varten
- hypermedia sovelluksen tilan moottorina, jolloin palvelimelta saadut vastaukset sisältävät URIt kaikkeen, mitä voidaan seuraavaksi tehdä.

Nämä rajoitukset muodostavat suurimman eron muihin web-palvelukonsepteihin, joissa jokainen palvelu määrittelee oman rajapintansa [2, s. 81–82].

Välimuisti

Neljäntenä rajoituksena on mahdollisuus paikallisen välimuistin käyttöön, joka määritellään resurssikohtaisesti. Vaihtoehtoja on kolme: resurssia ei tallenneta välimuistiin, resurssi tallennetaan välimuistiin tai resurssi tallennetaan välimuistiin määritellyksi ajaksi. Mikäli resurssi on määritelty tallennettavaksi välimuistiin, voi asiakas tallentaa resurssin esityksen itselleen myöhempää vastaavaa esitystä varten [2, s. 79–81].

Ladattava koodi

Yhtenä valinnaisena rajoituksena mainittakoon palvelun laajentaminen lisäohjelmilla, joka sallii koodin (kuten appletit tai skriptit) käytön asiakkaan toiminnallisuuden laajentamiseksi ja parantamiseksi [2, s. 84–85].

Kerroksittainen järjestelmä

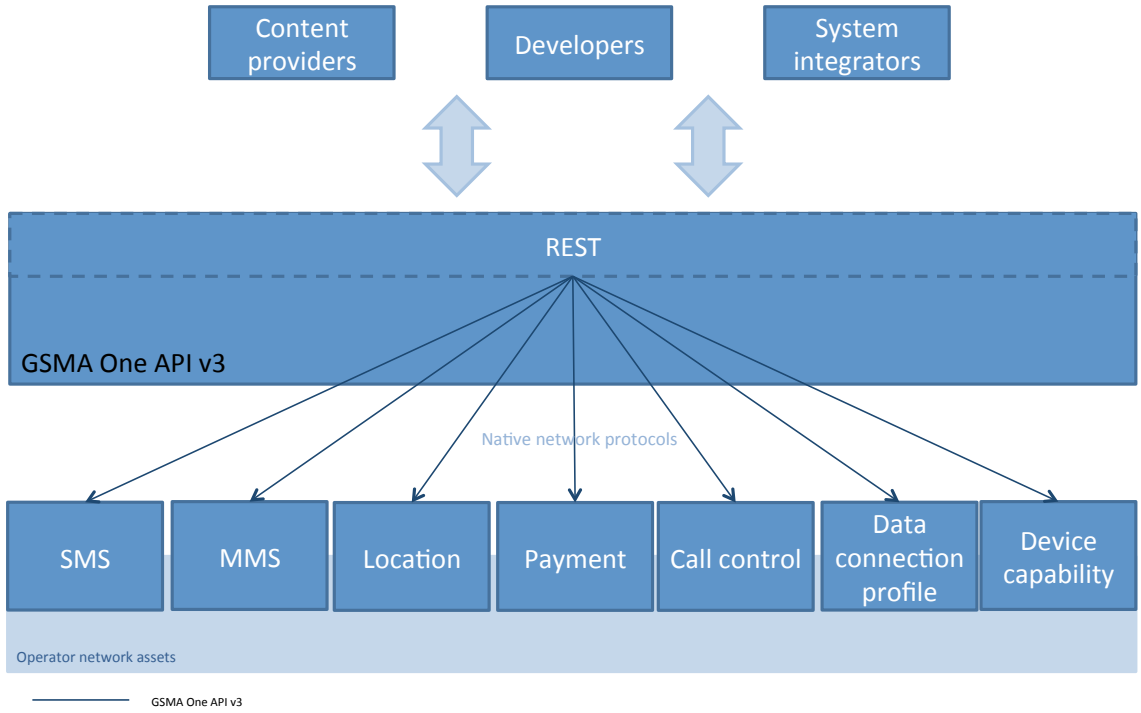
Viimeisenä rajoituksena on järjestelmän koostuminen kerroksista. Tällöin asiakaan ja palvelimen välille voidaan asettaa esimerkiksi palomuureja ja välityspalvelimia muuttamatta rajapintoja palvelimen ja asiakkaan välillä [2, s. 82–84].

3.4 Tietoturva

REST ei suoraan tarjoa tietoturvaan liittyviä rajoitteita tai ohjeistusta. REST käyttää kuitenkin HTTP-protokollaa, joten sen suojattu versio HTTPS (Hypertext Transfer Protocol Secure) tarjoaa luonnollisen tavan viestien suojaamiseen lähettäjän ja vastaanottajan välillä ja mahdollistaa myös käyttäjän todennuksen. HTTP keskittyy tietoturvatoteutuksessa käyttäjän todennukseen ja välitettävän tiedon salaukseen. Todennukseen HTTP tarjoaa kaksi tapaa, Basic- ja Digest-todennuksen. Basic-todennuksessa asiakkaan tunnus ja salasana lähetetään HTTP-otsikossa palvelimelle Base64-muodossa selväkielisenä salaamatta. HTTP Digest korjaa yhden Basic-todennuksen puutteen eli salasanan lähettämisen selväkielisenä verkon yli. Digest-todennus perustuu haaste-vasteperiaatteeseen, jossa asiakkaan tunnuksesta, salasanasta ja palvelimen antamista tunnisteista lasketaan tiiviste, jonka perusteella palvelin tunnistaa käyttäjän. Tiedon suojattuun siirtoon eli salaukseen, HTTP käyttää SSL/TSL-protokollia (Secure Sockets Layer/Transport Layer Security), joilla tiedot salataan ennen niiden lähettämistä [4, s. 238–241, 311; 11, s. 5–6].

4 GSMA OneAPI-standardin rajapinnat versiossa 3

GSMA OneAPI määrittelee listan yleisesti käytettyjä RESTful API-rajapintoja, jotka tarjoavat mahdollisuuden hyödyntää operaattoreiden verkkoresursseilta saatavaa tietoa/toiminnallisuutta erilaisten sovelluksien tarpeisiin. GSMA OneAPI versio 3 julkaisee profiilin, joka perustuu OMA REST NetAPI standardiin. One API versio 3 pitää sisällään seitsemän API-rajapintaa ja kolme BETA-versiota uusista API-rajapinnoista, kuten alla olevassa kuvassa 2 on esitetty [8].



Kuva 2. GSMA OneAPI v3:ssa julkaistut REST-rajapinnat [8].

4.1 SMS-rajapinta

SMS-rajapinta mahdollistaa tekstiviestien lähettämisen ja vastaanottamisen käyttäen Web-sovelluksia. Hyvänä sovellusesimerkkinä voidaan mainita erilaiset äänestyspalvelut massatapahtumien yhteydessä [8].

RESTful API hyödyntää HTTP:n metodeita POST, GET, PUT ja DELETE, joiden avulla resursseja kutsutaan palvelimelta määritellyn URI:n avulla. SMS API:ssa URI:t on määritelty seuraavasti viestin lähetyksessä ja vastaanotossa [8].

SMS lähetys (MT)
SMS:n lähetys yhdelle tai useammalle päätelaitteelle
http://example.com/smsmessaging/v1/outbound/{senderAddress}/requests
SMS:n toimituskuittaus
http://example.com/smsmessaging/v1/outbound/{senderAddress}/requests/{requestId}/deliveryInfos
Toimituskuittaus tilauksen aloitus ja lopetus lähettämillesi viesteille
http://example.com/smsmessaging/v1/outbound/{senderAddress}/subscriptions http://example.com/smsmessaging/v1/outbound/{senderAddress}/subscriptions/{subscriptionID}
SMS vastaanotto (MO)
SMS:n vastaanotto sovellukselle
http://example.com/smsmessaging/v1/inbound/registrations/{registrationId}/messages
Toimitusilmoituksen tilauksen aloitus ja lopetus sovellukselle lähetetyistä viesteistä
http://example.com/smsmessaging/v1/inbound/subscriptions

SMS API:ssa POST-metodille tuettuja esitysmuotoja ovat application/xml ja application/json. Yleisesti tuettu esitysmuoto vastauksille SMS API:ssa on application/json [8].

Seuraavassa on esimerkki kutsu ja vastaus MT SMS lähetykselle.

Request
 	
POST http://example.com/smsmessaging/v1/outbound/
tel%3A%2B12345678/requests HTTP/1.1
Host: example.com:80
Content-Type: application/json
Accept: application/json
 	
{"outboundSMSMessageRequest":{
"address":["tel:+35800000991", "tel:+35800000992"],
"senderAddress":"tel:12345678",
"outboundSMSTextMessage":{"message":"Hello"},
"clientCorrelator":"123456",
"receiptRequest":
{"notifyURL":"http://example.com/notifications",
"callbackData":"some-data-useful-to-the-requester"},
"senderName":“COMPANY"}
}
	POST-metodilla luodaan SMS aikaisemmin esitetyn URI määrityksen mukaisesti, jossa POST-metodi esitetty application/json muodossa ja vastaus pyydetty application/json muodossa.
	
	Pakollisia parametrejä SMS:n lähetyksessä ovat:
	address(vastaanottajan MSISDN) tel:+358401234567
	senderAddress(lähettäjän MSISDN)
	tel:+358501234567
	message(Viesti) Hello World
	
	Valinnaisia parametrejä ovat:
	clientCorrelator, jolla identifioidaan kyseinen viestin lähetys mahdollisen uudelleenlähetyksen varalta.
	notifyURL, URI johon toimituskuittaus lähetetään
	callbackData, käytettävissä mikäli toimituskuittaus pyydetty.
	senderName, lähettäjän haluama lähettäjä tieto, joka näkyy vastaanottajan terminaalissa.

Response
 	
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/smsmessaging/v1/outbound/
tel%3A%2B12345678/requests/abc123
Content-Length: 12345
Date: Thu, 04 Jun 2009 02:51:59 GMT
 	
{"outboundSMSMessageRequest":{
"address":["tel:+35800000991", "tel:+35800000992"],
"deliveryInfoList" : {
"deliveryInfo" : [
{ "address" : "tel:+35800000991",
"deliveryStatus" : "MessageWaiting" },
{ "address" : "tel:+13500000992",
"deliveryStatus" : "MessageWaiting" },
],
"resourceURL":
"http://example.com/smsmessaging/v1/outbound/
tel%3A%2B12345678/requests/abc123/deliveryInfos"
},
"senderAddress":"tel:12345678",
"outboundSMSTextMessage":{"message":"Hello"},
"clientCorrelator":"123456",
"receiptRequest":
{"notifyURL":"http://.example.com/notifications",
"callbackData":"some-data-useful-to-the-requester"},
"senderName":“COMPANY",
"resourceURL":
"http://example.com/smsmessaging/v1/outbound/
tel%3A%2B12345678/requests/abc123"
}
	HTTP 201 osoittaa, että viesti luotu ja Location tieto esittää URI:n jossa luotu viesti sijaitsee.
	
	Vastaus sisältää myös alkuperäisen kutsun parametrit.

4.2 MMS-rajapinta

MMS-rajapinta mahdollistaa multimediaviestien lähettämisen ja vastaanottamisen käyttäen www-sovelluksia. Loppukäyttäjän kannalta tämä mahdollistaa muun muassa

opastuspalvelut, joissa palvelun tarjoaja lähettää tilatun kartan tarvittavine lisätietoineen loppukäyttäjälle [8].

RESTful API hyödyntää HTTP:n metodeita POST, GET, PUT ja DELETE, joiden avulla resursseja kutsutaan palvelimelta määritellyn URI:n avulla. MMS API:ssa URI:t on määritelty seuraavasti viestin lähetyksessä ja vastaanotossa [8].

MMS lähetys (MT)
MMS:n lähetys yhdelle tai useammalle päätelaitteelle
http://example.com/messaging/v1/outbound/{senderAddress}/requests
MMS:n toimitustiedon kysely
http://example.com/messaging/v1/outbound/{senderAddress}/requests/{requestId}/deliveryInfos
Toimituskuittaus tilauksen aloitus ja lopetus lähettämillesi viesteille
http://example.com/messaging/v1/outbound/{senderAddress}/subscriptions
MMS vastaanotto (MO)
MMS:n vastaanotto sovellukselle
http://example.com/messaging/v1/inbound/registrations/{registrationId}/messages
Toimitusilmoituksen tilauksen aloitus ja lopetus sovellukselle lähetetyistä viesteistä
http://example.com/messaging/v1/inbound/subscriptions

MMS API:ssa POST-metodille tuettuja esitysmuotoja ovat application/xml ja application/json. Yleisesti tuettu esitysmuoto vastauksille MMS API:ssa on application/json. MMS-viesteissä käytetään MIME-multipart-koodausta, joka määrittelee viestiin liitettyjen tiedostojen formaatin. MIME-multipart-koodaus on välttämätön MMS-viestien lähetyksessä ja vastaanotossa, koska viesti pitää sisällään useita liitetiedostoja [8].

Seuraavassa on esimerkki MMS-viestin lähetyksestä vastauksineen, josta käy ilmi kuinka viesti MIME on koodattu ja kuinka eri osat ovat eroteltu toisistaan [8].

Request
 	
POST http://example.com/messaging/v1/outbound/
tel%3A%2B12345678/requests HTTP/1.1
Content-Length: 12345
Content-Type: multipart/form-data;
boundary="==========123456==";
 	
MIME-Version: 1.0
Host: www.example.com
Date: Thu, 04 Jun 2009 02:51:59 GMT
 	
--==========123456==
Content-Disposition: form-data; name="root-fields"
Content-Type: application/json;
 	
{"outboundMessageRequest": {
"address": [
"tel:+13500000991",
"tel:+13500000992"
],
"clientCorrelator": "123456",
"outboundMMSMessage": {
"subject": "My message"
},
"receiptRequest": {
"callbackData": "12345",
"notifyURL": "http://example-application.com/notifications"
},
"senderAddress": "tel:+12345678",
"senderName": "Gameworld"
}}
 	
--==========123456==
Content-Disposition: form-data; name=”attachments”;
filename=”picture.jpg”
Content-Type: image/gif
 	
GIF89a...binary image data...
--==========123456==--
	POST-metodilla lähetetään MMS viesti aikaisemmin esitetyn URI määrityksen mukaisesti, jossa POST-metodi esitetty application/json muodossa ja vastaus pyydetty application/json muodossa.
	
	Pakollisia parametrejä MMS:n lähetyksessä ovat:
	address(vastaanottajan MSISDN) tel:+358401234567
	senderAddress(lähettäjän MSISDN)
	tel:+358501234567
	
	Valinnaisia parametrejä ovat:
	Body/Text, viestin teksti osuus
	Subject, viestin aihe
	clientCorrelator, jolla identifioidaan kyseinen viestin lähetys mahdollisen uudelleenlähetyksen varalta.
	notifyURL, URI johon toimituskuittaus lähetetään
	callbackData, käytettävissä mikäli toimituskuittaus pyydetty.
	senderName, lähettäjän haluama lähettäjä tieto, joka näkyy vastaanottajan terminaalissa.

Response
 	
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/messaging/v1/outbound/
tel%3A%2B12345678/requests/abc123
Content-Length: 12345
Date: Thu, 04 Jun 2009 02:51:59 GMT
 	
{"outboundMessageRequest": {
"address": [
"tel:+13500000991",
"tel:+13500000992"
],
"clientCorrelator": "123456",
"outboundMMSMessage": {
"subject": "My message"
},
"receiptRequest": {
"callbackData": "12345",
"notifyURL": "http://example-application.com/notifications"
},
"resourceURL":
"http://example.com/messaging/v1/outbound/tel%3A%2B123
45678/requests/abc123",
"senderAddress": "tel:+12345678",
"senderName": "Gameworld"
}}
	HTTP 201 osoittaa, että viesti luotu ja Location tieto esittää URI:n jossa luotu viesti sijaitsee.

4.3 Paikannusrajapinta

Paikannus (location) API mahdollistaa päätelaitteen paikkatiedon kysymisen operaattorin verkosta näin sallittaessa. Paikannukselle löytyy useita sovellusalueita. Tällaisia ovat muun muassa opastuspalvelut, joissa paikkatietoa pystytään hyödyntämään haettaessa esimerkiksi lähintä kukkakauppaa [8].

Paikannusrajapinta hyödyntää HTTP-metodia GET, jonka avulla resurssia kutsutaan palvelimelta määritellyn URI:n avulla. Paikannuspyynnössä määritellään paikannuksen kohde (address) ja paikannuksen tarkkuus (requestedAccuracy). Paikannusrajapinnassa URI on määritelty seuraavasti paikkatiedon hakemiseen yhdelle tai useammalle päätelaitteelle [8].

Päätelaitteen paikannuspyyntö
Paikannus kysely yhdelle tai useammalle päätelaitteelle
http://example.com/location/v1/queries/location?address={address}&requestedAccuracy={metres}

Paikannusrajapinnassa tuettu esitysmuoto on application/json. Seuraavassa on esimerkki paikkatiedon kyselystä yhdelle päätelaitteelle ja siihen saadusta vastauksesta paikkatietoineen [8].

Request GET http://example.com/location/v1/queries/location?address=tel:+16309700001&requestedAccuracy=1000 HTTP/1.1 Host: example.com:80 Accept: application/json	Esimerkissä GET-metodilla kysytään yhden päätelaiteen paikkatieto aikaisemmin esitetyn URI määrityksen mukaisesti, jossa GET-metodi esitetty application/json muodossa ja vastaus pyydetty application/json muodossa. Pakollisia parametrejä paikkatiedon kyselyssä on: address(paikannettava MSISDN) tel:+16309700001 requestedAccuracy(Paikkatiedon tarkkuus metreinä) =1000 Valinnaisia parametrejä paikannus API:ssa ei ole!

Response
 	
HTTP/1.1 200 OK Content-Type: application/json
Content-Length: 1234
Date: Thu, 04 Jun 2009 02:51:59 GMT
 	
{"terminalLocationList": {"terminalLocation": {
"address": "tel:16309700001",
"currentLocation": {
"accuracy": "1000",
"altitude": "1001.0",
"latitude": "-80.86302",
"longitude": "41.277306",
"timestamp": "2009-06-03T00:27:23.000Z"
},
"locationRetrievalStatus": "Retrieved"
}}}
	HTTP 200 osoittaa, että paikannuskysely vastaanotettu onnistuneesti.
	
	currentLocation objektin sisällä päätelaitteen paikkatieto
	- accuracy, paikannuksen tarkkuun metreina
	- altitude, korkeus metreinä
	- latitude, leveysaste ISO 6709 mukaisesti
	- longitude, pituusaste ISO 6709 mukaisesti
	
	locationRetrievalStatus arvot,
	- Retrieved, paikkatieto saatu onnistuneesti
	- Not Retrieved, paikkatietoa ei saatu vastaanotettua
	- Error, virhe paikkatiedon vastaanotossa

4.4 Laskutusrajapinta

Laskutus (payment) API mahdollistaa laskutustapahtuman luonnin liittymälle esimerkiksi halutusta sovelluksen tai sisällön ostamisesta. API:n kautta pystyy tekemään muun muassa suoran veloituksen liittymälle tai varauksen tietylle summalle. Varaus tietylle summalle on erityisen tärkeä, kun puhutaan prepaid-liittymistä, joissa ennen ostotapahtumaa täytyy tarkistaa saldo ja tehdä varaus ennen ostotapahtuman suorittamista ja mahdollisesti vapauttaa tehty varaus, mikäli ostotapahtumaa ei suoriteta loppuun [8].

Esimerkkinä laskutusrajapinnan käytöstä voidaan mainita erityismaksulliset SMS-viestit. Tässä tapauksessa tarkoituksenmukaista olisi käyttää SMS API:a viestin vastaanottoon, lähetykseen ja laskutusrajapintaa laskutustapahtuman luontiin halutunlaisena.

Laskutusrajapinta hyödyntää HTTP-metodeita POST ja GET, joiden avulla resursseja kutsutaan palvelimelta määriteltyjen URI:jen avulla. Laskutusrajapinnassa URI:t on määritelty seuraavasti riippuen tapahtumasta [8].

Laskutus
Laskutustapahtuman luonti/palautus/listaus
http://example.com/payment/v1/{endUserId}/transactions/amount
Laskutustapahtuman statuksen kysely
http://example.com/payment/v1/{endUserId}/transactions/amount/{transactionId}
Laskutusvarauksen luonti/palautus
http://example.com/payment/v1/{endUserId}/transactions/amountReservation

Laskutusrajapinnassa tuettuja esitysmuotoja POST-operaatiolle ovat application/xml ja application/json. Yleisesti tuettu esitysmuoto vastauksille API:ssa on application/json, mutta vaihtoehtona on myös application/xml [8].

Seuraavassa on esimerkki laskutustapahtuman luonnista, jossa liittymältä veloitetaan tietty summa onnistuneen tapahtuman yhteydessä.

Request

```
POST http://example.com/payment/v1/ tel:+16309700001/
transactions/amount  HTTP/1.1
Accept: application/json
Host: example.com:80
Content-Type: application/json
Content-Length: 12345
Date: Thu, 04 Jun 2009 02:51:59 GMT

{"amountTransaction": {
  "clientCorrelator": "54321",
  "endUserId": "tel:+16309700001",
  "paymentAmount": {
    "chargingInformation": {
      "amount": "10",
      "currency": ”EUR",
      ”description": ”Angry Birds Rio"
    },
    "chargingMetaData" : {
      "onBehalfOf" : "Example Games",
      "purchaseCategoryCode" : "Game",
      "channel" : ”SMS",
      "taxAmount" : "0"
    }
   },
  "referenceCode": "REF-12345",
  "transactionOperationStatus": "Charged",
}}
```

Esimerkissä POST-metodilla tehdään laskutus operaatio aikaisemmin esitetyn URI määrityksen mukaisesti, jossa POST-metodi esitetty application/json muodossa ja vastaus pyydetty application/json muodossa.

Pakollisia parametrejä ovat:
endUserId, loppukäyttäjän indentifioiva MSISDN tai acr
description, laskutus tapahtuman kuvaus
referenceCode, identifioi kyselyn, mikäli kyseistä tapahtumaa täytyy selvittää jälkeenpäin
transactionOperationStatus, määrittelee laskutus tapahtuman, joka tässä tapauksessa veloitus

Valinnaisia parametejä ovat:
amount(, määritetään laskutettava summa. Pakollinen mikäli code parametriä ei ole annettu
currency, laskutettava valuutta. Pakollinen mikäli code parametriä ei ole annettu
onBehalfOf, määritteleen laskutustapahtuman saajan kun kyseessä esim. agregaattori/partneri
purchaseCategoryCode, määrittelee laskutettavan sisällön tyypin
channel, määrittelee käytetyn kanavan, jota palvelutapahtuma käyttänyt
taxAmount, veloitettavan veron määrä

Response

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 12345
Date: Thu, 04 Jun 2009 02:51:59 GMT
Location: http://example.com/payment/v1/ tel%3A
%2B16309700001/transactions/amount/abc123

{"amountTransaction": {
  "clientCorrelator": "54321",
  "endUserId": "tel:+16309700001",
  "paymentAmount": {
    "chargingInformation": {
      "amount": "10",
      "currency": ”EUR",
      "description": ”Angry Birds Rio"
    },
    "totalAmountCharged": "12.99",
    "chargingMetaData" : {
      "onBehalfOf" : "Example Games",
      "purchaseCategoryCode" : "Game",
      "channel" : ”SMS",
      "taxAmount" : "0"
     }
  },
  "referenceCode": "REF-12345",
  "serverReferenceCode": "ABC-123",
  "resourceURL": " http://example.com/payment/v1/    tel%3A
%2B16309700001/transactions/amount/abc123",
"transactionOperationStatus": "Charged"
}}
```

HTTP 201 osoittaa, että laskutus tapahtuma suoritettu onnistuneesti.

Yllä esitetyn operaation sisältämien parametrien lisäksi vastaus sisältää seuraavat parametrit:

Location, URI, joka määrittelee resurssin sijainnin
transactionOperationStatus, laskutus tapahtuman tila, joka tässä tapauksessa vahvistettu veloitetuksi

Valinnaisia parametrejä vastauksessa:

serverReferenceCode, serverin palauttama identifikaatio kyseiselle laskutus tapahtumalle, jonka avulla tapahtumaa pystytään selvittämään jälkeenpäin
totalAmountCharged, veloitettava summa sisältäen esim. verot

4.5 Puhelunohjausrajapinta

Puhelunohjaus (Call Control) API mahdollistaa sovelluksille puheluiden aloituksen ja niiden hallinnan käyttämällä API:n mahdollistamia operaatioita. Puhelu voi olla kahden tai useamman osallistujan välisiä tai ns. automaattisia IVR-ohjattuja (Interactive Voice Response), jolloin IVR:n avulla pystytään toteuttamaan palveluvalikoita, joissa asiakasta ohjataan tallennetuilla DTMF-komennoilla (Dual-tone Multi-frequency) haluttuun toiminnallisuuteen. Hyvänä esimerkkinä sovelluksesta voisi olla asiakaspalvelu, jossa IVR:n avulla vodaan tarkentaa asiakaan tarpeet ja ohjata näin puhelu oikealle henkilölle käsiteltäväksi [8].

Puhelunohjaus API hyödyntää HTTP metodeita POST, GET ja DELETE, joiden avulla resursseja kutsutaan palvelimelta määriteltyjen URI:jen avulla. Puhelunohjaus API:ssa URI:t on määritelty seuraavasti riippuen tapahtumasta [8].

Puhelun luonti/hallinta
Puhelun luonti yhden tai useamman osallistujan kesken
http://example.com/{api version}/thirdpartycall/callSessions
Osallistujien lisäys/poisto ja tietojen kysely
http://example.com/{api version}//thirdpartycall/callSessions/{callSessionID}/participants
Puhelun lopetus ja statuksen kysely
http://example.com/{api version}//thirdpartycall/callSessions/{callSessionID}
Ilmoiuksen tilaus puhelun tapahtumista
http://example.com/{api version}/callnotification/subscriptions/callEvent
Ilmoituksen lopetus puhelun tapahtumista ja perusteiden kysely
http://example.com/{api version}/callnotification/subscriptions/callEvent/{subscriptionID}
Ääniviesti toisto/lopetus/status puhelun osallistujille ja IVR-toiminteet
Ääniviestin toiston aloitus
http://example.com/{api version}/ audiocall/messages/audio
Ääniviestin toiston lopetus ja status kenelle se on toistettu
http://example.com/{api version}/audiocall/messages/audio/{abc123}
Ääniviestin toisto ja DTMF-vastaanotto
http://example.com/{api version}/audiocall/interactions/collection
Ääniviestin toiston ja DTMF-vastaanoton lopetus
http://example.com/{api version}/audiocall/interactions/collection/{abc123}
Notifikaation lähetyksen aloitus/lopetus riippuen loppukäyttäjän antamista DTMF-komennoista
http://example.com/{api version}/callnotification/subscriptions/collection

Puhelunohjausrajapinnassa tuettu esitysmuoto on application/json. Seuraavassa on esimerkki API:n käytöstä, jossa perustetaan puhelu kahden loppukäyttäjän välille onnistuneesti [8].

Request	

POST /exampleAPI/thirdpartycall/callSessions HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 1234
Accept: application/json
 	
{"callSessionInformation": {
"clientCorrelator": "104567",
"participant": [
{ "participantAddress": "tel:+4912345678901",
"participantName": "Max"},
{"participantAddress": "tel:+4412345678901",
"participantName": "Peter"
}
]
}}
	Esimerkissä POST-metodilla tehdään puhelun muodostus operaatio aikaisemmin esitetyn URI määrityksen mukaisesti, jossa POST-metodi esitetty application/json muodossa ja vastaus pyydetty application/json muodossa.
	
	Puhelua muodostettaessa pakollisia parametejä ovat ainoastaan participant-ryhmässä määritellyt vastaanottajat, numeron ja nimen muodossa
	
	Valinnaisia parametejä ovat:
	clientCorrelator, määrittää kyselylle yksiselitteisen indentifikaation

Response	

HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/
thirdpartycall/callSessions/cs001
Content-Length: 1234
Date: Mon, 28 Jun 2010 17:51:59 GMT
 	
{"callSessionInformation": {
"clientCorrelator": "104567",
"participant": [
{"participantAddress": "tel:+4912345678901",
"participantName": "Max",
"participantStatus": "CallParticipantConnected",
"resourceURL": "http://example.com/exampleAPI/1/
thirdpartycall/callSessions/cs001/participants/pt001",
"startTime": "2010-06-28T17:50:51"
},
{"participantAddress": "tel:+4412345678901",
"participantName": "Peter",
"participantStatus": “CallParticipantInitial”,
"resourceURL": "http://example.com/exampleAPI/1/
thirdpartycall/callSessions/cs001/participants/pt002"
 	
}
],
"resourceURL": "http://example.com/exampleAPI/1/
thirdpartycall/callSessions/cs001",
"terminated": "false"
}}
	HTTP 201 osoittaa, että puhelu muodostettu onnistuneesti.
	
	Location, URI, joka määrittelee resurssin sijainnin, joka sisältää puhelu Id:n cs001
	
	Yllä esitetyn operaation sisältämien participant – ryhmän parametrien lisäksi vastaus sisältää
	
	participantStatus, palauttaa osallistujan tilan, joka voi olla jokin seuraavista:
	
	CallParticipantInitial
	CallParticipantConnected
	CallParticipantTerminated
	
	resourceURL, jonka avulla pystytään kysymään osallistujan tila myöhemmin
	startTime, ilmoitaa koska osallistuja yhdistetty puheluun
	
	resourceURL, jonka avulla pystytään kysymään kyseisen puhelun tila Id:n perusteella

4.6 Datayhteysprofiilirajapinta

Datayhteysprofiili (Data Connection Profile) API mahdollistaa kyselyn, jonka vastauksena saadaan selville mikä on päätelaitteen käyttämä datayhteys. Päätelaitteen käytössä voi olla esimerkiksi 3G-yhteys ja päätelaite kytkeytynyt muuhun kuin kotiverkkoon, jolloin kyse on verkkovierailusta. Datayhteysprofiilirajapinnan hyödyntäminen sovelluksissa voisi tulla kysymykseen esimerkiksi videonlatauspalvelussa, jolloin ladattavan videon laatua pystytään muuttamaan riippuen käytettävästä datayhteydestä [8].

Datayhteysprofiilirajapinta hyödyntää HTTP-metodia GET, jonka avulla resursseja kutsutaan palvelimelta määriteltyjen URI:jen avulla. Yhteysprofiilirajapinnassa URI:t on määritelty seuraavasti riippuen tapahtumasta [8].

Yhteys tyyppi
Päätelaitteen data-yhteys tyypin kysyminen
http://example.com/terminalstatus/v1/queries/connectionType?address={terminalId}
Päätelaitteen verkkovierailu kysely
http://example.com/terminalstatus/v1/queries/roamingStatus?address={terminalId}

Datayhteysprofiili API:ssa tuettu esitysmuoto on application/json. Seuraavassa on esimerkki API:n käytöstä, jossa kysytään terminaalin käyttämä datayhteystyyppi ja saadaan siihen vastaus, josta käy ilmi terminaalin käyttämä yhteystyyppi [8].

Request GET http://example.com/terminalstatus/v1/queries/ connectionType? address=tel%3A%2B15555550100 HTTP/1.1 Accept: application/json Host: example.com	Esimerkissä GET-metodilla tehdään kysely terminaalin käyttämästä yhteystyypistä, jossa GET-metodi esitetty application/json muodossa ja vastaus pyydetty application/json muodossa. Yhteystyypin kyselyssä pakollisia parametejä ovat ainoastaan address, jossa määritellään kyseltävien päätelaitteiden MSISDN tai ACR.

Response
 	
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Thu, 04 Jun 2009 02:51:59 GMT
 	
{"terminalConnectionTypeList": {
"connectionType": {
"address": "tel:+15555550100",
"currentConnectionType": ”EDGE",
"retrievalStatus": "Retrieved"
},
"resourceURL":
"http://example.com/terminalstatus/v1/
queries/connectionType"
}}
	HTTP 200 osoittaa, että kysely suoritettu onnistuneesti.
	
	Vastauksena saadaan terminalConnectionTypeList objekti, joka sisältää päätelaitteen käyttämän yhteystyypin, joka tässä tapauksessa on EDGE.
	
	retrievalStatus parametrin arvo voi olla joko Retrieved tai Error
	
	resourceURL, joka määrittelee resurssin sijainnin

4.7 Päätelaitteen kyvykkyysrajapinta

Päätelaitteen kyvykkyys (Device Capability) API mahdollistaa päätelaitteen kyvykkyyksien kysymisen, minkä vastauksena saadaan tietoa päätelaitteesta. Vastaus sisältää muun muassa päätelaitteen IMEI:n (International Mobile Equipment Identity) ja linkin laitteen User Agent Profiiliin, josta käy ilmi tarkemmin esimerkiksi näytön koko ja resoluutio. Vastauksena saadun tiedon perusteella päätelaitteen käyttämään sisältöä voidaan optimoida päälaitteelle sopivaksi esimerkiksi näytön koon perusteella [8].

Päätelaitteen kyvykkyysrajapinta hyödyntää HTTP-metodia GET, jonka avulla resursseja kutsutaan palvelimelta määriteltyjen URI:jen avulla. Päätelaitteen kyvykkyysrajapinnassa URI on määritelty seuraavasti [8].

Terminaali kyvykkyys
Terminaalin kyvykkyyden kysyminen
http://example.com/devicecapabilities/v1/{address}/capabilities HTTP/1.1

Päätelaitteen kyvykkyysrajapinnassa tuettu esitysmuoto on application/json. Seuraavassa on esimerkki API:n käytöstä, jossa kysytään terminaalin kyvykkyys ja saadaan siihen vastaus onnistuneesti [8].

Request GET http://example.com/devicecapabilities/v1/ tel%3A%2B15555550100/capabilities HTTP/1.1 Accept: application/json Host: example.com	Esimerkissä GET-metodilla tehdään kysely terminaalin kyvykkyydestä, jossa GET-metodi esitetty application/json muodossa ja vastaus pyydetty application/json muodossa. Terminaalin kyvykkyys kyselyssä pakollisia parametejä ovat ainoastaan address, jossa määritellään kyseltävien päätelaitteiden MSISDN tai ACR.

Response
 	
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Thu, 04 Jun 2010 02:52:00 GMT
 	
{"deviceCapabilities": {
"deviceId": "123456789012345",
"link": {
"href": "http://example.com/exampleconfigurations/
exampledeviceprofiles/A1234xyz123.xml",
"rel": "UserAgentProfileReference"
},
"name": ”Lumia 920",
"resourceURL": "http://example.com/devicecapabilities/v1/tel
%3A%2B15555550100/capabilities"
}}
	HTTP 200 osoittaa, että kysely suoritettu onnistuneesti.
	
	Vastauksena saadaan deviceCapabilities objekti, joka sisältää seuraavat parametrit:
	
	deviceId, terminaalin yksilöivä IMEi-koodi
	link, joka käsittää linkin User Agent Profile tiedostoon
	name, terminaalin nimi
	
	resourceURL, joka määrittelee resurssin sijainnin

4.8 BETA API:t

GSMA OneAPI v3:ssa on julkaistu kolme Beta API:a, joita ovat ACR (Anonymous Customer Reference), Customer Profile ja Zonal Presence. Beta API:t ovat ns. draft-versioita, joihin odotetaan tulevan vielä muutoksia riippuen sidosryhmien palautteesta, joka ohjaa vahvasti niiden kehittymistä [8].

ACR (Anonymous Customer Reference)

Useissa maissa yksityisyyden suojaan liittyvä regulaatio ja lainsäädäntö estävät käyttäjän MSISDN:n välittämisen operaattorin verkosta sovelluksille. Tätä varten on mahdollista luoda ACR API:n avulla, joka on viittaus käyttäjän MSISDN:ään, jolloin itse MSISDN pysyy operaattorin hallussa ja sovellukselle välitetään vain ACR. ACR:ää voidaan käyttää kaikissa API-kutsuissa, joissa se korvaa MSISDN:n [8].

Asiakasprofiili

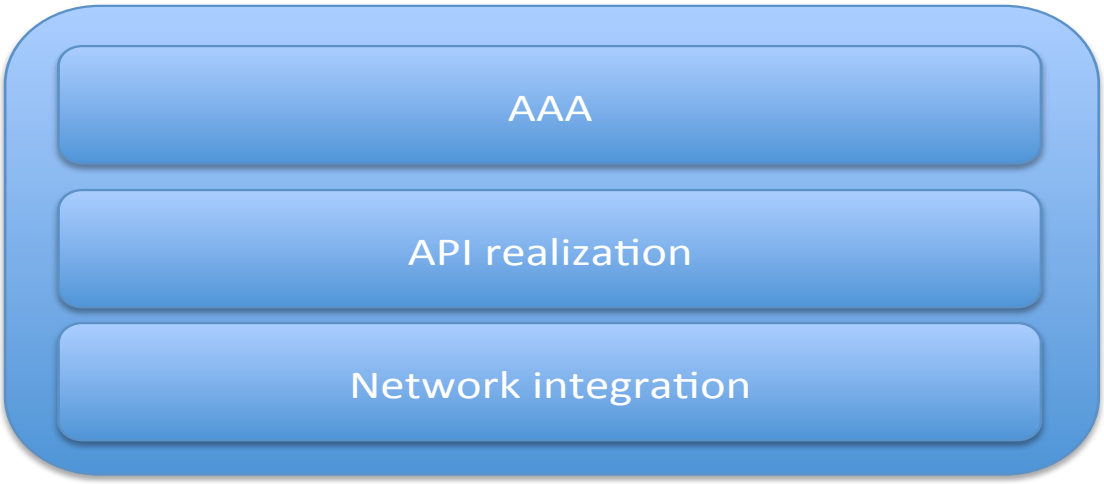
Asiakasprofiili (customer profile) API mahdollistaa asiakkaalle määritellyn profiilin kysymisen sovelluksen tarpeisiin operaattorin asiakasjärjestelmistä. Asiakasprofiilii voi pitää sisällään muun muassa nimi-, ikä- ja osoitetietoja huomioiden myös, mihin tietoihin asiakas on antanut julkaisuluvan. [8]

Alueellinen tilatieto

Alueellinen tilatieto (zonal presence) API:n avulla sovellus pystyy kysymään alueellisesti käyttäjämääriä ja asettamaan ilmoituksen muutoksista alueella, esimerkiksi lähtijät alueelta ja tulijat alueelle. Alue voi olla 3G-verkon solu tai WLAN (langaton verkko), johon loppukäyttäjä on kytkeytynyt. Tätä tietoa hyödyntäen voidaan esimerkiksi laskea kävijämääriä kaupassa tai sen kuinka paljon asiakkaita kävelee ohi, mikäli alue pystytään rajaamaan kaupan sisälle tai sen välittömään läheisyyteen [8].

5 OneAPI-standardin toteutukset

Standardiin perustuvaa toteutusta suunniteltaessa on hyvä ottaa huomioon muutamia perusasioita. One API-standardi määrittelee joukon REST-pohjaisia API-rajapintoja, jotka näkyvät suoraan kolmansille osapuolille. Operaattorin kannalta standardi antaa suuntaviivat toteutukselle, mutta paljon jää vielä mietittäväksi toteutusta suunniteltaessa. Jos asiaa mietitään kuvan 3 pohjalta, operaattorille jää ratkaistavaksi, kuinka se toteuttaa todennuksen ja valvoo pääsyä vain sallittuihin resursseihin (AAA). Näiden lisäksi toteutuksessa on otettava huomioon erilaiset rajoitukset ja säännöstöt itse toteutuksessa (API realization). Riippuen API:sta operaattorin on myös valittava tapa, jolla verkkointegraatio (Network integration) tarvittaviin verkkoresursseihin tehdään.



Kuva 3. API julkaisun suuntaviivat toteutukselle.

AAA (Authentication, Authorization, Accounting)

Ennen kuin asiakas pääsee pyyntöjä tekemään, pitää asiakkaalle tehdä todennus, jolla varmistetaan kuka pyyntöjä tekee. Kuten kohdassa 3.4 mainittiin, yksi tapa todennuksen toteuttamiseen on HTTP:n tarjoamat Basic- tai Digest-todennus. Auktorisoinnilla varmistetaan, että asiakkaalla on oikeus tehdä niitä pyyntöjä, joita asiakas yrittää tehdä. Todennus on siis tehtävä ennen auktorisointia.

API-toteutus (API realization)

Toteutuksessa huomioitavia asioita ovat muun muassa erilaiset käyttöön liittyvät rajoitukset ja säännöstöt, jotka voivat liittyä esimerkiksi lähettäjän numeron tarkastamiseen SMS–viestin lähetyksessä ja tapahtumatiketin luomiseen. Toteutuksessa on mietittävä myös, kuinka verkkoprotokolla konvertoidaan julkaistavaksi REST API–rajapinnaksi kaikkien parametrien osalta, esimerkiksi SMSC:n liikennöinnissä käytettävä SMPP.

Yksi vaihtoehto on jakaa toiminnallisuudet teknisiin ja liiketoiminnallisiin, jotka vaihtelevat usein API –kohtaisesti. Taulukossa 1 esitetty ajatuksia siitä minkälaisia toiminnallisuuksia tulisi toteutusvaiheessa miettiä.

Taulukko 1. OneAPI toteutuksen toiminnallisuus esimerkit.

Toiminnallisuus	Kuvaus
API käyttäjien hallinta	Käyttäjien todennus Käyttäjän rekisteröinti ja uuden API profiilin luonti API käyttäjän liittymän/tilin hallinta Käyttäjän API profiilin hallinta
API hallinta toiminteet	API luonti/päivitys/poisto API versio ja ryhmien/pakettien hallinta
API suojaus toimineteet	Liikenteen suojaus API käyttörajoitteiden hallinta
API operationaaliset toiminteet	Laskutus- ja toimintatikettien luonti Käytön monitorointi Logien kirjoittaminen API kutsujen reititys
Verkkointegraatiot	Integraatio pisteiden valitseminen Protokolla konversio julkaistaville API-rajapinnoille esim. SMPP <-> REST

Verkkointegraatio (Network integration)

Verkkointegraatiolla tarkoitetaan API-ratkaisun integroitumista operaattorin verkkoresursseihin. Integraatiopistettä valittaessa on käytävä läpi käytettävä protokolla ja se, ovatko tarvittavat tiedot saatavilla julkaistavaksi kolmansille osapuolille. Esimerkkinä voisi olla SMS API, jolloin luonteva integraatiopiste on SMSC ja protokollana SMPP.

5.1 Kehittäjäohjelmat maailmalta

Maailmalta löytyy muutamia operaattoreiden julkaisemia ratkaisuja perustuen OneAPI-standardiin. Ne tarjoavat kolmansille osapuolille mahdollisuuden hyödyntää julkaistuja API-rajapintoja heidän sovelluksissaan. Kuvassa 4 listattuna operaattoreiden kehittäjäohjelmia, joista löytyy OneAPI-standardiin perustuvia API-rajapintoja:

Kuva 4. Operaattoreiden kehittäjäohjelmat.

5.2 Developer garden-toteutus

Developer garden-kehittäjäohjelma on Deutsche Telecomin perustama. Ohjelman tarkoituksena on sallia kehittäjille pääsy operaattorin verkkoresursseihin REST-pohjaisten API-rajapintojen kautta. Developer carden tarjoaa julkaistuja REST API-rajapintoja viestien lähetykseen sovellukselta terminaaliin (SMS, MMS), puhelun perustamiseen yhden tai useamman osallistujan välille sekä laskutustapahtumien luontiin [9].

Seuraavaksi käydään läpi prosessikuva 5, jossa esitetään, kuinka kehittäjäohjelmaan liitytään ja kuinka API:n käyttöönotto ja sovelluksen luominen tapahtuu. Rekisteröinnin ja tilin luomisen jälkeen kehittäjällä on mahdollisuus aktivoida käytettäviä API-toiminnallisuuksia, kuten tässä tapauksessa aktivoitu Send MMS API. Aktivoinnin jälkeen luodaan itse sovellus, joka käyttää aktivoitua API-toiminnallisuutta. Tässä yhteydessä järjestelmä tallentaa sovelluksen ja palauttaa tarvittavat parametrit myöhempää sovelluksen käyttöä varten [9].

Kuva 5. Developer gardenin käyttöönotto.

Send MMS-sovelluksen luonnin jälkeen sovellus on testattavissa ns. sandbox- tai mock-ympäristössä ilman laskutustapahtumien luontia. Sovelluksen testaukseen voidaan käyttää REST-clientia, joita on tarjolla esimerkiksi liitännäisenä Firefox-selaimeen tai SoapUI-ohjelmistoa, joilla tarvittavat REST-kutsut on mahdollista luoda. Itse ohjelmointiin Developer garden tarjoaa REST SDK:t Java-, Python-, PHP- ja C-ohjelmointikielille [9].

Sovelluksen käyttöön liittyy kolme vaihetta: käyttäjän todennus, lähettäjän validointi ja itse viestin lähetys. Näistä kaksi ensimmäistä ovat operaattorin valitsemia toteutustapoja, joihin GSMA OneAPI ei ota kantaa. Käyttäjän todennus on toteutettu käyttäen oAuth 2.0-todennusta, josta saadaan palvelukutsuissa tarvittava käyttöoikeuskoodi. Seuraavassa on kuvaus oAuth 2.0 todennuksen yksittäisistä vaiheista [9].

1. Uudelleenohjaa käyttäjän käyttöoikeusvaltuutuksen saamiseksi Deutsche Telecomin ylläpitämän palvelimelle. Tässä yhteydessä palvelin palauttaa valtuutuskoodin [9].

2. Käyttäjän hyväksyttyä käyttöoikeuden palvelin vastaanottaa kutsun, jossa on käyttäjän saama valtuutuskoodi [9].

3. Palvelimen vahvistettua valtuutuskoodin käyttäjä pyytää käyttöoikeuskoodin ja mikäli virheitä ei ilmene vastaus sisältää palvelukutsuissa käytettävän käyttöoikeuskoodin lisätietoineen, kuten voimassaoloajan [9].

Ennen viestin lähetystä tehdään vielä lähettäjän käyttämän numeron validointi, joka on kertaluonteinen operaatio. Tällä varmistetaan, että numeron omistaa kyseinen henkilö/yritys. Tämän jälkeen voidaan aloittaa viestien lähetys. Seuraavassa on esimerkki onnistuneesta SMS-lähetyksen palvelupyynnöstä ja siihen saadusta vastauksesta [9].

Request

```
POST
https://gateway.developer.telekom.com/sms/rest/premium/smsmessaging/v1/outbound/tel%3A%2B12345678/request
s HTTP/1.1
Host: example.com:80
Authorization: OAuth realm="developergarden.com",oauth_token="KDfueW4OphJ..."
Content-Type: application/json
Accept: application/json

{"outboundSMSMessageRequest":{
  "address":["tel:+13500000991", "tel:+13500000992"],
  "senderAddress":"tel:12345678",
  "outboundSMSTextMessage": {"message":"Hello "},
     "outboundEncoding":"7bitGSM",
     "clientCorrelator":"123456",
  "receiptRequest":
{"notifyURL":"http://application.example.com/notifications/DeliveryInfoNotification",
"callbackData":"some-data-useful-to-the-requester"},
"senderName":"COMPANY"}
}
```

Response

```
HTTP/1.1 201 Created
Content-Type: application/json
Location:https://gateway.developer.telekom.com/sms/rest/premium/smsmessaging/v1/outbound/tel%3A%2B1234567
8/requests/abc123
Content-Length: 12345
Date: Thu, 04 Jun 2009 02:51:59 GMT

{"outboundSMSMessageRequest":{
  "address":["tel:+13500000991", "tel:+13500000992"],
  "deliveryInfoList" : {
  "deliveryInfo" : [
  { "address" : "tel:+13500000991",
  "deliveryStatus" : "MessageWaiting" },
  { "address" : "tel:+13500000992",
  "deliveryStatus" : "MessageWaiting" },
],
"resourceURL": "http://example.com/smsmessaging/v1/outbound/
tel%3A%2B12345678/requests/abc123/deliveryInfos"
},
 "senderAddress":"tel:12345678",
  "outboundSMSTextMessage":{"message":"Hello  World"},
  "outboundEncoding":"7bitGSM",
  "clientCorrelator":"123456",
  "receiptRequest": {"notifyURL":"http://application.example.com/notifications/DeliveryInfoNotification",
  "callbackData":"some-data-useful-to-the-requester"},
  "senderName":"COMPANY",
  "resourceURL": "http://example.com/smsmessaging/v1/outbound/
  tel%3A%2B12345678/requests/abc123"
}
```

Mikäli sendSMS:n toteutusta verrataan OneAPI v3-standardiin, muutamia poikkeuksia on havaittavissa. Ensimmäiseksi Developer garden toteutuksessa viestin sisältö on laajennettu tukemaan binääriviestejä. Toisena laajennuksena on parametri, jolla pystytään määrittämään viestin koodaus. Tämä huomio koskee myös muita API-toiminteita kuin sendSMS:ää. Developer garden API-toteutusta tarkasteltaessa yleisesti voidaankin sanoa, että toteutusta ohjaa standardi, mutta toteutus ei standardia orjallisesti noudata. Erityisesti huomiota herättää se, että standardin API-toiminnallisuudet on toteutettu ainoastaan osittain. Esimerkkinä viestintään liittyvät SMS- ja MMS-rajapinnat, joissa toteutus tehty ainoastaan viestien lähetykselle ja viestien vastaanotto jätetty toteuttamatta [9].

Kehittäjäohjelman käyttöönoton vaiheita tarkasteltaessa voidaan mainita, että itse rekisteröinti, API-rajapintojen käyttöönotto ja sovelluksien luominen oli suoraviivainen ilman aikaavieviä käyttäjän hyväksyntäprosesseja. Informaatio liittyen rekisteröintiin, käyttöönottoon ja sovelluksien luomiseen oli myös hyvin saatavilla. Oletuksena API-rajapintojen käyttöön on, että kehittäjällä on olemassa oleva asiakassuhde Deutsche Telecomin kanssa, jolloin laskutustapahtumat pystytään veloittamaan asiakkaalta.

6 Yhteenveto

GSMA OneAPI-standardin tarkoituksena on julkaista REST-pohjaiset API-rajapinnat sovelluskehittäjille. OneAPI-standardin versio 3 onkin jo määritellyt seitsemän REST API-rajapintaa ja kolme betaversiota uusista rajapinnoista. OneAPI-standardin etuna onkin, että tärkeimmät sidosryhmät, operaattorit ja laitetoimittajat ovat olleet tiiviisti mukana standardia määriteltäessä GSMA:ssa. Tällöin varmistetaan ainakin, että toteutus on määritelty sidosryhmien tarpeista.

REST-arkkitehtuurityyli tarjoaa hyvin web-palveluihin soveltuvat rajoitteet ja opasteet pitäen mielessä helppokäyttöisyyden ja skaalautuvuuden. REST onkin onnistunut luomaan periaatteet, jotka tarjoavat hyvän pohjan toteuttaa API-rajapintoja sovelluksien tarpeisiin.

Operaattoreiden verkkoresurssit ovat olleet kuitenkin rajallisesti saatavilla ulkopuolisille sovelluskehittäjille. Ajatus siitä, että operaattoreilla olisi käytössä yhtenäinen tapa API-rajapintojen julkaisuun sovelluskehittäjien tarpeisiin operaattoririippumattomasti olisikin ihanteellinen ja tavoiteltava päämäärä. Käytännössä tavoitteeseen pääsemiseksi on operaattoreiden hyväksyttävä tämä tapa laajemmin ja noudatettava tehtyjä määrityksiä. Yhtenä esteenä REST-pohjaisten API-rajapintojen yleistymiselle on operaattoreiden nykyiset API-ratkaisut ja niiden korvaaminen, joissa API-rajapintoja on julkaistu hyödyntäen ns. verkkoprotokollia esimerkiksi SMPP-protokollaan SMS-viestien lähetyksessä ja vastaanotossa. Standardoitujen REST API-rajapintojen saatavuus riippuukin, kuinka halukkaita operaattorit ovat tukemaan toteutusta ja nähdäänkö tässä liiketoiminnallisesti houkuttelevia mahdollisuuksia.

Lähteet

1 RESTful Web services: The basics. 2008. Verkkodokumentti. IBM. http://www.ibm.com/developerworks/webservices/library/ws-restful/. Päivitetty 6.11.2008. Luettu 30.3.2013.

2 Roy Thomas Fielding. 2000. Architectural Styles and the Design of Network-based Software Architectures. University of California.

3 Representational State Transfer. 2009. Verkkodokumentti. Wikipedia. http://en.wikipedia.org/wiki/Representational_state_transfer. Päivitetty 7.2.2013. Luettu 18.3.2013.

4 Leonard Richardson & Sam Ruby. 2007. RERTful Web Services 2007 First Edition. O'Reilly.

5 Introducing JSON. http://www.json.org/. Verkkodokumentti. Json.org. Luettu 18.3.2012.

6 W3C Extensible Markup Language (XML) 1.0 (Fifth Edition). 2008. Verkkodokumenti. W3C. http://www.w3.org/TR/xml/. Päivitetty 1.3.2012. Luettu 18.3.2012.

7 Sonera Content Gateway. Verkkodokumentti. Sonera. http://www.sonera.fi/yrityksille/tuotteet+ja+palvelut/liiketoiminta+ja+tukipalvelut/palveluntarjoajille/content+gateway. Luettu 18.3.2012.

8 GSMA OneAPI. Verkkodokumentti. GSMA. http://www.gsma.com/oneapi/ Luettu 30.3.2013

9 Developer garden. Verkkodokumentti. Deutsche Telecom. http://www.developergarden.com/. Luettu 30.3.2013

10 HTTP/1.1 RFC 2616. 1999. Verkkodokumentti. The Internet Society. http://tools.ietf.org/html/rfc2616. Luettu 15.4.2013

11 HTTP authentication RCF 2617. 1999. Verkkodokumentti. The Internet Society. http://tools.ietf.org/html/rfc2617. Luettu 15.4.2013