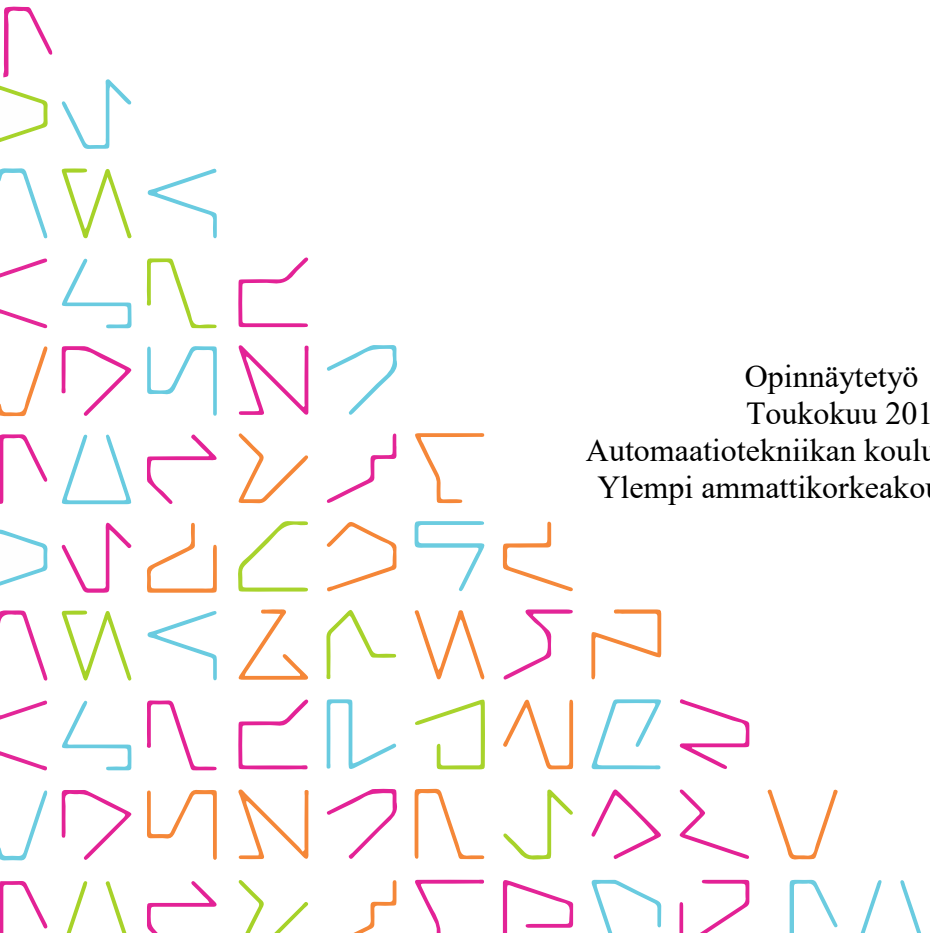
TAMPEREEN
AMMATTIKORKEAKOULU

OPC UA AUTOMAATION TIEDON-SIIRROSSA

Mikko Heikkilä



Opinnäytetyö
Toukokuu 2016
Automaatiotekniikan koulutusohjelma
Ylempi ammattikorkeakoulututkinto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma, ylempi ammattikorkeakoulututkinto

MIKKO HEIKKILÄ:
OPC UA automaation tiedonsiirrossa

Opinnäytetyö 69 sivua, joista liitteitä 5 sivua
Toukokuu 2016

Tämän opinnäytetyön tarkoituksena oli tuottaa suomenkielinen tekninen selvitys OPC UA -standardin rakenteesta, toiminnasta, ominaisuuksista, suorituskyvystä sekä käyttökohteista yleisellä tasolla. Lisäksi työssä tutustuttiin eräisiin markkinoilla oleviin laitteisiin, jotka käyttävät OPC UA -pohjaista tiedonsiirtoa kommunikointiin.

Työn tavoitteena oli tuottaa Cargotec Finland Oy:lle tietoa OPC UA -standardista, minkä pohjalta voitaisiin arvioida standardin soveltuvuutta yrityksen toimintaympäristöön.

OPC Unified Architecture on OPC Foundationin® laatima tiedonsiirtostandardi automaatioteollisuuteen, mikä mahdollistaa eri valmistajien ohjelmistojen, laitteiden sekä koneiden keskinäisen kommunikoinnin yhteisen rajapinnan avulla.

Työn lopputuloksena OPC UA -standardista saatiin tekninen yhteenveto, jota voidaan käyttää yrityksen tuotekehityksessä.

Asiasanat: opc ua, tiedonsiirto, tietoturva

ABSTRACT

Tampere University of Applied Sciences, Master's Degree
Department of Automation technology
MIKKO HEIKKILÄ:
OPC UA in Automation Communication

Master's thesis 69 pages, appendices 5 pages
May 2016

The main purpose of this thesis was to study the structure, functionality and use of the OPC UA standard. In addition a few devices which use OPC UA for communication were explored.

The objective of the work was to provide information about OPC UA standard for assessing it's feasibility in the operational environment of Cargotec Finland Ltd.

OPC Unified Architecture developed by OPC Foundation® industry consortium is a communication standard for industrial automation that enables machines and applications from different vendors to communicate via common interface.

Final result of the work was a technical summary of OPC UA standard that can be used in the company's product development.

Key words: opc ua, communication, security

SISÄLLYS

LYHENTEET

AES = Advanced Encryption Standard

API = Application Programming Interface

COM = Component Object Model

DCOM = Distributed Component Object Model

DCS = Distributed control system

ERP = Enterprise Resource Planning

HMI = Human Machine Interface

HTTPS = Hypertext Transfer Protocol Secure

IP = Internet Protocol

MES = Manufacturing Execution System

OLE = Object Linking and Embedding

OOP = Object-oriented programming

OPC = OLE for Process Control tai Openness, Productivity and Connectivity

OPC UA = OPC Unified Architecture

PKI = Public Key Infrastructure

PLC = Programmable Logic Controller

RPC = Remote Call Procedure

SCADA = Supervisory Control And Data Acquisition

SOA = Service Oriented Architecture

SOAP = Simple Object Access Protocol

SQL = Structured Query Language

TCP = Transmission Control Protocol

XML = Extended Markup Language

1 JOHDANTO

Automaation nykyinen ympäristö asettaa korkeat vaatimukset käytettävälle tiedonsiirtotekniikalle, ennen kaikkea tietoturvan, luotettavuuden ja skaalautuvuuden suhteen. Järjestelmiä yhdistettäessä eri valmistajien laitteilla avainsanoiksi muodostuvat helppous, nopeus ja taloudellisuus. OPC UA -standardi on käyttöjärjestelmästä riippumaton, skaalautuva ja palvelukeskeinen arkkitehtuuri, joka pyrkii tarjoamaan joustavan sekä turvallisen vaihtoehdon automaation järjestelmäintegraatioon.

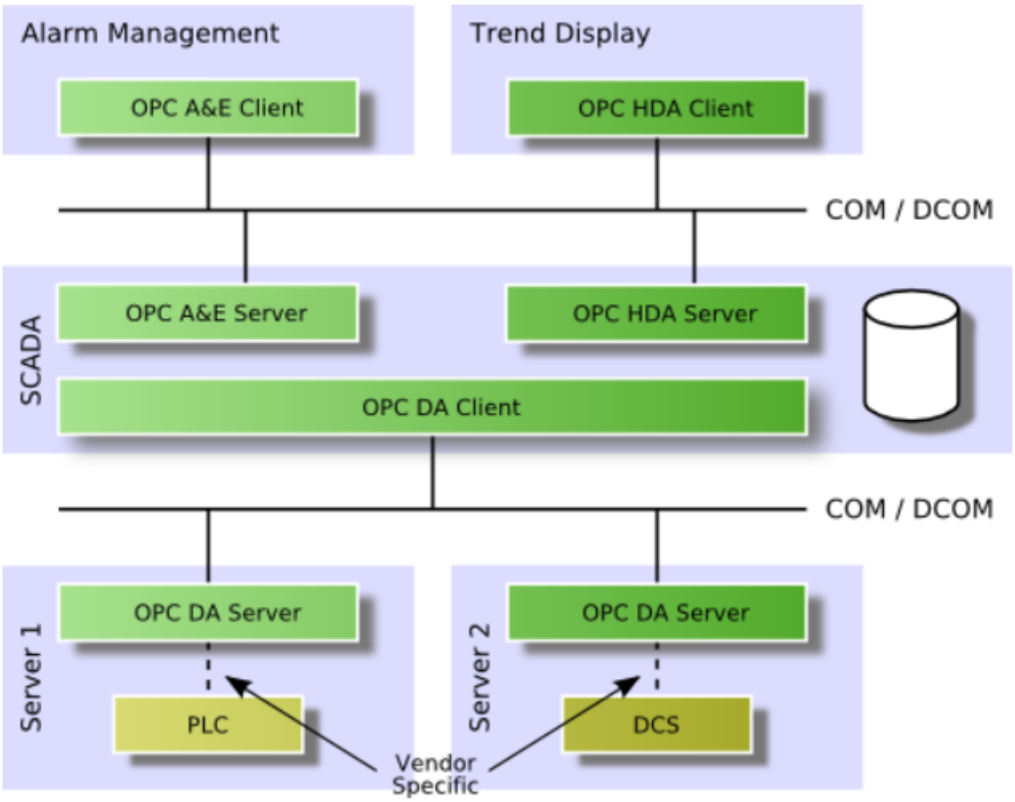
Työssä keskitytään standardin tärkeimpiin osa-alueisiin, joita ovat informaatiomalli, osoiteavaruus, palvelut sekä profiilit. Aluksi luvussa 2 käsitellään lyhyesti alkuperäistä OPC Classic -tiedonsiirtomenetelmää ja sen käyttötarkoitusta. Luvuissa 3−8 käsitellään OPC UA -standardin rakennetta, toimintaa, tiedonsiirtoa sekä tietoturvaa. Lopuksi luvussa 9 tutustutaan muutamiin markkinoilla oleviin eri valmistajien laitteisiin, joiden tiedonsiirto perustuu OPC UA -protokollaan.

2 OPC CLASSIC

OPC on avoin palvelin/asiakas *(server/client)* pohjainen ohjelmistorajapinta, mikä mahdollistaa Windows-ohjelmien kommunikoinnin teollisuudessa käytettävien laitteiden, kuten ohjelmoitavien logiikoiden, käyttöliittymien, valvomosovellusten ja kenttälaitteiden välillä (OPC Datahub).

OPC-määrittely perustui alun perin Microsoftin kehittämiin OLE-, COM- ja DCOM-teknologioihin ja se suunniteltiin tarjoamaan yhteinen tiedonsiirtomenetelmä Windows-pohjaisten sovellusten ja prosessilaitteiden välille. OPC Classic koostuu ryhmästä eri määrittelyjä, joita ovat OPC Data Access (OPC DA), OPC Alarms & Events (OPC AE) ja OPC Historical Data Access (OPC HDA). OPC Foundation® niminen organisaatio julkaisi nämä edellä mainitut määrittelyt vuosien 1996−2001 välillä ja nykyään järjestön tehtävänä on ylläpitää vanhoja, sekä laatia uusia ja avoimia määrittelyjä automaatiosovellusten integrointiin ja prosessidatan tiedonsiirtoon. OPC Foundation® järjestön jäsenenä ovat kaikki isoimmat ja merkittävimmät automaatioalan yritykset mukaan lukien Siemens, ABB, Ascolab, Beckhoff, sekä Emerson Process Management (Suomen Automaatioseura; OPC Foundation, What is OPC; National Instruments, Introduction to OPC; Mathworks OPC Data; Mahnke, Leitner, Damm, 2009, 3).

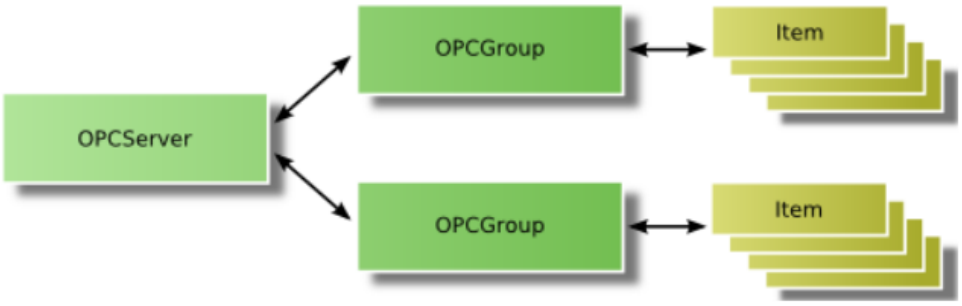
Teollisuuden eri sovellusten vaatimusten pohjalta siis kehitettiin kolme merkittävintä OPC-määrittelyä, joilla informaation siirto kenttälaite tasolta ylemmän tason järjestelmiin saatiin standardisoitua. Datan lukeminen ja kirjoittaminen on kuvattu OPC DA määrittelyssä, A&E kuvaa rajapinnan tapahtumapohjaiseen dataan sisältäen prosessihälytysten kuittauksen *(acknowledgement)* ja HDA toiminnot varastoidun datan käsittelyyn. OPCtiedonvaihto perustuu asiakas/palvelin-malliin, jossa OPC-palvelin kapseloi prosessidatan suoraan lähteeltä, esimerkiksi lämpötilaa mittaavalta kenttälaitteelta ja jakaa tiedon eteenpäin rajapintansa kautta. OPC-asiakas ottaa yhteyden palvelimeen ja pääsee kuluttamaan palvelimen tarjoamaa tietoa. Sovellukset, jotka tuottavat tai käyttävät dataa, voivat olla molemmat, sekä asiakas että palvelin. Kuvassa 1 on kuvattu tyypillinen OPC asiakas/palvelin-käyttötapaus (Suomen Automaatioseura; OPC Foundation; National Instruments, Introduction to OPC; Mathworks OPC Data; Mahnke ym. 2009, 3).



KUVA 1. Tyypillinen OPC asiakas/palvelin käyttötapaus (Lähde: Unified Automation)

2.1 OPC Data Access

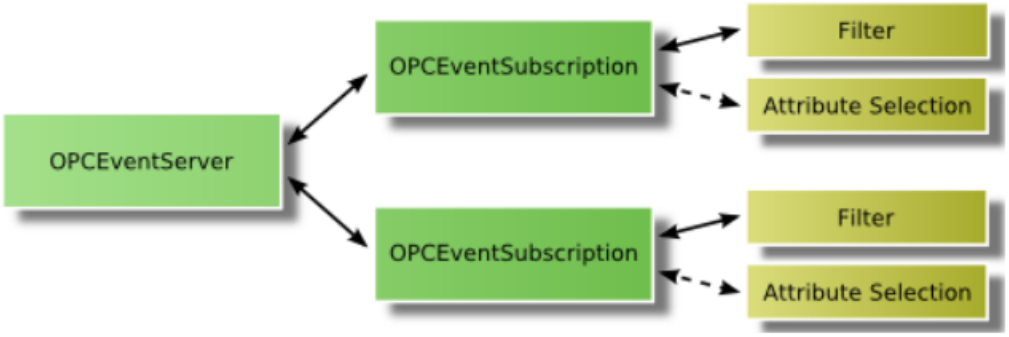
OPC Data Access on vanhin kaikista OPC-määrittelyistä ja sen rajapintaa käytetään prosessidatan lukemiseen, kirjoittamiseen sekä monitorointiin. Rajapintaa käytetään pääasiassa reaaliaikaisen tiedonsiirtämiseen esimerkiksi ohjelmoitavalta logiikalta (*PLC*) tai hajautetusta ohjausjärjestelmästä *(DCS)*, paikallisiin käyttöliittymiin *(HMI)*, sekä valvomosovelluksiin. OPC DA -asiakas valitsee ne data-muuttujat, joita halutaan lukea palvelimelta ja muodostaa yhteyden palvelimelle luomalla OPC-palvelin-olion *(engl. OPCserver object).* Palvelinolio, on ylimmän tason olio OPC-hierarkiassa, ja se tarjoaa metodit, joiden avulla osoiteavaruudesta paikannetaan etsittävä tietolähde (*item*). Näitä voivat olla esimerkiksi kenttälaitteet tai tietty muistialue ohjelmoitavassa logiikassa. Päästäkseen käsiksi varsinaiseen dataan, asiakas luo identtisillä asetuksilla OPC-ryhmäolioita *(OPC-Group object)* tietolähteille. Kuvassa 2 on havainnollistettu edellä kuvailtu prosessi (Mathworks OPC Data; Mahnke ym. 2009, 4; Frank Iwanitz, Jurgen Lange, 2002)



KUVA 2. OPC asiakkaan luomat eri oliot (Lähde: Unified Automation)

2.2 OPC Alarms and Events

OPC Alarms ja Events -rajapinta on tarkoitettu tapahtumiin perustuvien ilmoitusten ja hälytysten vastaanottamista varten. Tapahtumatiedot ovat yksittäisiä ilmoituksia esimerkiksi prosessissa määriteltyjen ehtojen täyttymisestä. Hälytykset ovat tiedoksiantoja, jotka ilmoittavat jonkin muutoksen. Tällainen muutos voi olla vaikkapa prosessiin määritellyn pinnankorkeuden ylä- tai alaraja-hälytys. Monesti tämän kaltaiset hälytykset vaativat kuittauksen, joka onkin mahdollista tehdä OPC A&E -rajapinnan kautta. On huomioitava, että OPC A&E ei itse tuota ilmoituksia tai hälytyksiä, vaan raportoi ne eteenpäin kaikille näistä tiedoista kiinnostuneille asiakkaille. Itse hälytykset ja tilamuutokset ovat määritelty jo aiemmin prosessissa, esimerkiksi ohjelmoitavassa logiikassa. Saadakseen hälytykset OPC-asiakas ottaa yhteyden palvelimeen, tilaa ilmoitukset sekä vastaanottaa ne (kuva 3). Jos asiakas ei halua vastaanottaa kaikkia ilmoituksia kerralla, voidaan käyttää suodatusta, jonka avulla vain kriteerit täyttävät ilmoitukset vastaanotetaan. (Mahnke ym. 2009, 5).



KUVA 3. OPC asiakkaan luomat eri oliot tapatumien vastaanottamista varten (Lähde: Unified Automation)

2.3 OPC Historical Data Access

Siinä missä OPC DA mahdollistaa pääsyn reaaliaikaiseen, jatkuvasti muuttuvaan dataan, OPC Historical Data Access puolestaan, nimensä mukaisesti, toimii rajapinta historiatietoihin, eli varastoituun dataan. Tällaisena varastona voi toimia esimerkiksi SQL-kyselykieltä (*Structured Query Language*) käyttävät palvelintietokannat, joita tarjoavat muun muassa Microsoft, Oracle ja IBM. OPC-asiakas ottaa yhteyden HDA-palvelimeen luomalla OPCHDA-palvelin-olion, joka tarjoaa kaikki tarvittavat rajapinnat ja metodit historiadatan lukemiseen ja päivittämiseen. Toinen olio, OPCHDA-selain on määritelty HDA-palvelimen osoiteavaruuden selaamista varten (Mahnke ym. 2009, 6).

OPC HDA:n pääasiallisena toiminnollisuutena on varastoidun datan lukeminen kolmella eri tavalla. Ensimmäinen mekanismi kerää dataa arkistoista, missä asiakas määrittelee yhden tai useamman muuttujan sekä aika-alueen (*time domain*), jolta data halutaan lukea, jonka jälkeen palvelin toimittaa kaiken määritellyn tiedon. Toinen mekanismi lukee yhden tai useamman muuttujan arvon tietyllä aikaleimalla (*time stamp*). Kolmas kerää ja laskee arvoja yhteen, kuten maksimi-, minimi- ja keskiarvoja (*Engl. aggregate values*) tietokannan datasta määritetyllä aika-alueella yhdelle tai useammalle muuttujalle. Muuttujien arvot sisältävät aina niihin liittyvät aika- ja laatuleimat. Yllämainittujen mekanismien lisäksi HDA:ssa määritellään metodit sille kuinka tietokannassa dataa voidaan muuttaa, lisätä, korvata sekä poistaa (Mahnke ym. 2009, 6).

3 OPC UNIFIED ARCHITECTURE – UUSI MÄÄRITTELY

OPC UA on käyttöjärjestelmästä riippumaton, palvelukeskeinen (*Engl. service oriented*) arkkitehtuuri, jonka ensimmäinen versio julkaistiin vuonna 2006. Se yhdistää kaikki yksittäiset OPC Classic -määrittelyt yhdeksi laajennetuksi määrittelyksi. OPC UA ei ole jatkoa vanhoille määrittelyille, vaan se on kokonaisvaltainen arkkitehtuuri, jonka tarkoituksena on korvata tulevaisuudessa aikaisemmat, irralliset OPC Classic-määrittelyt (OPC Foundation, What is OPC UA).

Miksi uusi arkkitehtuuri sitten kehitettiin ja mihin tarkoitukseen? Vastauksia on useita, mutta yleisesti voidaan sanoa, että automaation tietotekniikka ja tarpeet ovat muuttuneet paljon siitä, kun OPC DA julkaistiin 1990-luvulla. OPC Classic on riippuvainen Microsoft Windows-käyttöjärjestelmästä sekä osittain vanhentuneista COM/DCOM-tekniikoista, joiden tietoturva ei vastaa nykypäivän vaatimuksia. Verkottuminen ja eri informaatiojärjestelmien integrointi luovat lisävaatimuksia tiedonsiirtotekniikalle sekä luotettavuudelle että tietoturvalle ja juuri näihin tarpeisiin OPC UA pyrkii tarjoamaan ratkaisuja.

OPC Classic on laajalti hyväksytty valmistajasta riippumaton, avoimen tiedonsiirron standardi automaatioteollisuudessa, jossa nykyiset määrittelyt ovat saavuttaneet de facto aseman. Tällä hetkellä on olemassa yli 22 000 tuotetta 3 200 eri toimittajalta, joissa käytetään OPC-tiedonsiirtoa. Tämä menestys on myös yksi seuraus siihen, että OPC UA pitää sisällään OPC Classicin toiminnallisuudet, mutta myös parantaa sen puutteita. OPC UA soveltuu sekä lattiatason automaation tiedonsiirtoon, kuten kenttälaitteiden ja automaatiojärjestelmien väliseen kommunikointiin että ylemmän tason kommunikointiin kunnossapitojärjestelmien, valmistuksenohjausjärjestelmien MES (Manufacturing Execution Systems) ja toiminnanohjausjärjestelmien ERP (Enterprise Resource Planning) välille.Toisin kuin edeltäjänsä OPC UA ei ole riippuvainen Microsoft Windows -alustoista. Sovelluksia voidaan kehittää Java-, ANSI C-, C++- ja .NET-ympäristöihin laitetoimittajastariippumatta ja lisäksi uusi standardi parantaa siirrettävän datan tietoturvaa (National Istruments, Why OPC UA matters; Mahnke, Leitner ABB review 3/2009).

OPC UA:lle asetettiin kehitysvaiheessa useita eri vaatimuksia liittyen toiminnollisuuteen ja suorituskykyyn. Taulukossa 1 on listattu tärkeimpiä kommunikaatioon sekä tiedon mallintamiseen liittyviä vaatimuksia.

TAULUKKO 1. OPC UA-standardille asetettuja vaatimuksia. (Mahnke ym 2009, 9, muokattu)

Kommunikointi hajautettujen järjestelmien välillä	***Tiedon mallintaminen***

- **luotettavuus**
  o **vakaus ja viansietokyky**
  o **kahdennettavuus**
	- **yleinen tietomalli kaikelle OPC tiedolle**

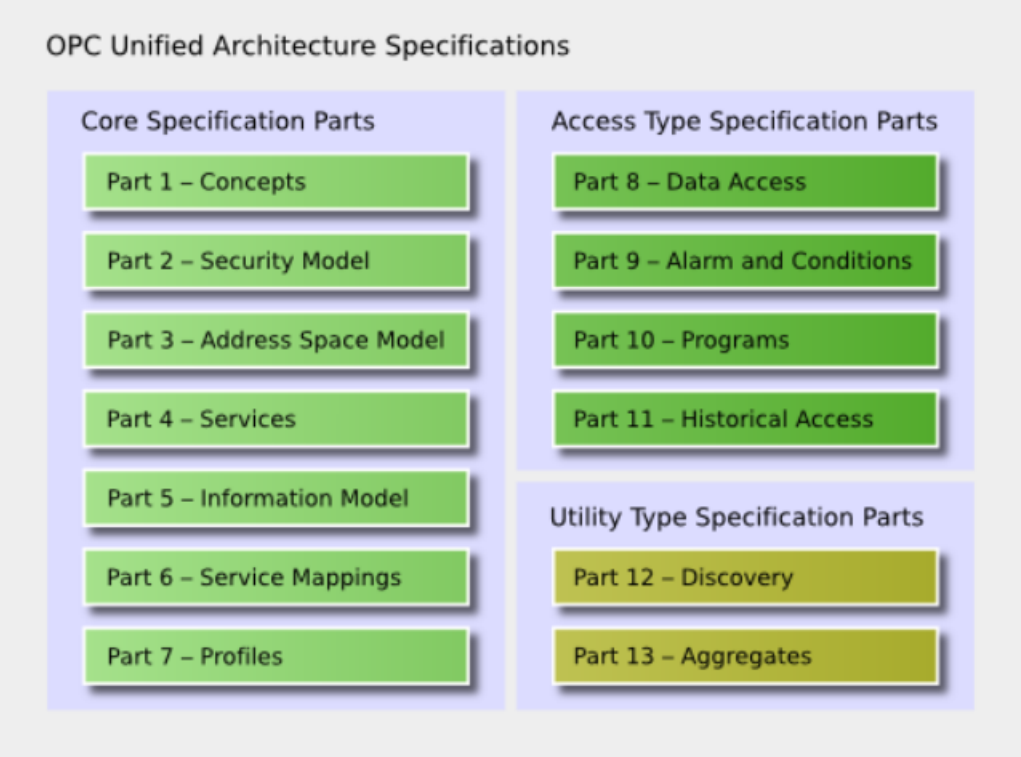
- **alustariippumattomuus**
  o **HW: PC, pilvipohjaiset serverit, PLC, mikro-kontrollerit (ARM, Intel, PPC)**
  o **Käyttöjärjestelmät: Windows, Apple OSX iOS, Android, Linux**
	- **olio-malli**
	  o **metatiedon tarjoaminen**
	  o **monitahoisen tiedon mallintaminen sekä laajennettavuus (extensible type system)**

- **skaalautuvuus**	- **skaalautuvuus yksinkertaisista monimutkaisiin tietomalleihin**
- **suorituskyky**	- **abstrakti kantatietomalli**
- **internet ja palomuurit**	- **perusta muille standardi-tietomalleille**
- **tietoturva ja pääsyn hallinta**	- **olio-ohjelmoinnin ominaisuuksilla kaikista monimutkaisimmatkin rakenteet voidaan mallintaa**
- **yhteen toimivuus**	- **OPC Classic (DA, A&E, HDA) -määrittelyt mallinnettiin OPC UA profiileihin**

Kommunikaation luotettavuus hajautettujen järjestelmien välillä on erittäin tärkeää, sillä häiriöt tuotannossa voivat aiheuttaa mittavia taloudellisia menetyksiä. Verkkoliikenne ei ole perusmäärittelyltään luotettavaa, minkä takia viansietokyky, luotettavuus ja kahdennettavuus ovat tärkeitä OPC UA -vaatimuksia. Alustariippumattomuus ja skaalautuvuus ovat myös tärkeitä ominaisuuksia, sillä ne mahdollistavat rajapintojen integroinnin riippumatta käytettävästä ohjelmistoalustasta.

Tiedon mallintamista varten OPC UA tarjoaa yhteisen oliokeskeisen tietomallin kaikelle tiedolle. Tätä abstraktia tietomallia käytetään pohjana kaikissa muissa eri sovellusten tietomalleissa, kuten esimerkiksi DA (Data Access), HA (Historical Access) ja A&E (Alarms & Events) –malleissa (Mahnke ym. 2009, 11).

OPC UA on IEC (International Electrotechnical Commission) standardi 62541, joka edelleen jaetaan kolmeentoista osaan (kuva 4)



KUVA 4. OPC UA määrittelyt (Lähde: Unified Automation)

Osa 1 kuvaa OPC UA:ta yleisellä tasolla ja osassa 2 käsitellään tietoturvavaatimuksia sekä OPC UA:n tietoturvamallia. Tärkeimmät ja oleellisimmat määrittelyt ovat osoiteavaruus [osa 3], palvelukeskeisyys [osa 4], informaatiomalli [osa 5] ja profiilit [osa 7].

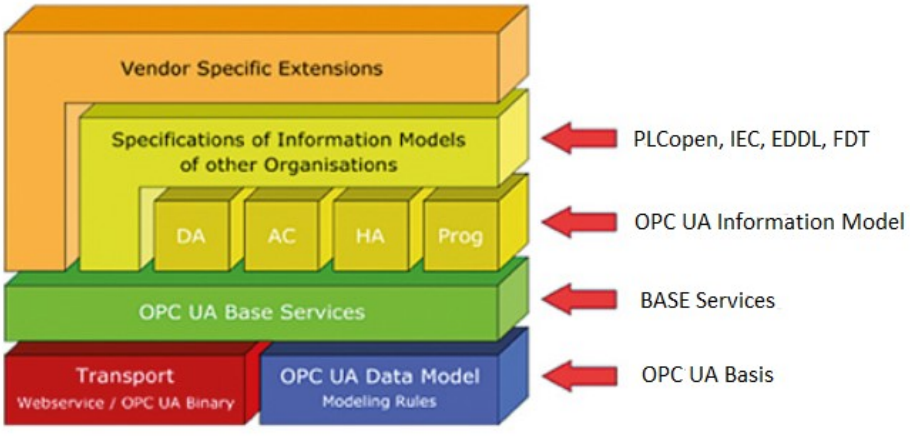
Osat 8-11 määrittelevät vanhan OPC Classic -määritysten toiminnan uudessa UA -pohjaisessa ympäristössä (OPC Foundation, Spesifikaatio osa 1, versio 1.03)

3.1 Informaatiomalli

OPC UA:n abstrakti informaatiomalli tarjoaa tehokkaan keinon tiedon semantiikan esittämiseen. Vanhassa OPC Classic:issa on saatavilla vain puhdas data, esimerkiksi painetta mittaavalta anturilta ja sisällön ymmärtämiseen ainoastaan tunnistenimi (*tag*) sekä yksikkö [Pa]. Edellä mainittujen tietojen lisäksi informaatiomalli voi paljastaa muun muassa mittaavan anturin tyypin ja mitä laitteita se tukee.

OPC UA:n kanta-informaatiomalli tarjoaa kuitenkin pelkän infrastruktuurin tiedon mallintamiseen, mutta koska malli on vapaasti laajennettavissa, voivat kolmannet osapuolet, kuten esimerkiksi laitevalmistajat ja muut standardointiryhmät, lisätä omia mallejaan osaksi UA:ta. Informaatiomalli tukee olio-ohjelmoinnin *(Object Oriented Programming)* periaatteita, kuten esimerkiksi tiedon abstraktia esitystä *(data abstracting)* sekä uusien luokkamääritysten tekemistä vanhojen määritysten pohjalta *(inheritance)*. Tietomallissa tieto voi olla näkyvillä useissa eri formaateissa, mukaan lukien binääritietona tai XML-(*Extensible Markup Language*) formaatissa (OPC Foundation, Spesifikaatio osa 1, versio 1.03; Mahnke ym. 2009,19)

Kuvassa 5 on OPC UA:n informaatiomalli ja täydentävät, muiden organisaatioiden sekä toimittajien määrittelemät informaatiomallit.



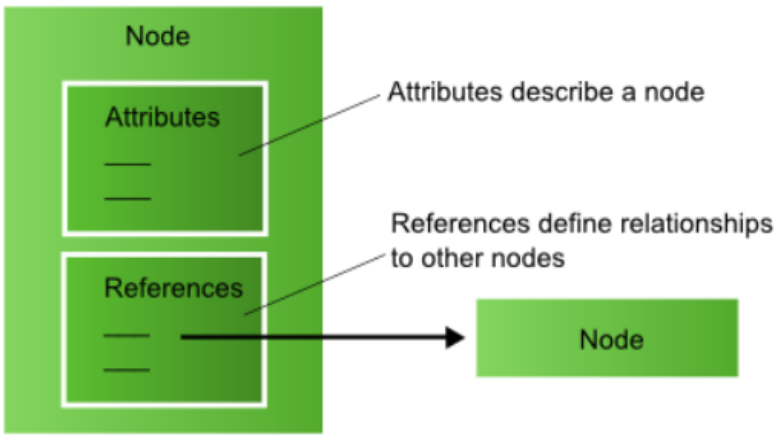
KUVA 5. OPC UA:n tietomalli ja täydentävät tietomallit (Lähde: Unified Automation

3.2 Osoiteavaruus

OPC UA:n osoiteavaruus muodostuu ryhmästä dataolioita *(objects)*, mikä mahdollistaa palvelimella olevien olioiden esittämisen standardin mukaisella tavalla asiakkaille. Osoiteavaruus rakentuu siten, että joukko dataolioita (tietopiste) muodostavat verkkomaisen rakenteen *(full meshed network)*, joka sallii tietojen yhdistämisen monella eri tavalla. Lisäksi olioiden välillä voi olla monia viittauksia ja riippuvuuksia toisiinsa. Osoiteavaruuden rakennuselementteinä toimivat solmut *(node)* sekä viittaukset *(reference)* solmujen välillä. Lisäksi jokainen solmu kuuluu johonkin luokkaan kuten olio-, muuttuja- tai metodi-luokkaan ja edustaa oliomallin eri elementtejä (Mahnke ym. 2009)

OPC UA määrittelee kahdeksan eri solmu-luokkaa ja jokainen solmu osoiteavaruudessa on jonkin luokan instanssi. Luokat puolestaan määrittelevät attribuutit ja viittaukset eri solmuille. Solmujen attribuutit eli määritteet riippuvat niiden luokasta, mutta tietty joukko perusmääritteitä on yhteisiä kaikille solmuille. Yhtenä tärkeimmistä määritteistä voisi mainita solmun tunnisteen *(nodeid),* joka yksilöi solmun ja jolla solmu tunnistetaan palvelimelta. Attribuutit ovat eräänlaisia dataelementtejä, jotka kuvailevat osoiteavaruuden solmuja. Asiakas pääsee käsiksi attribuutti-arvoihin käyttämällä Read-, Write-, Query-, ja Subscription/MonitoredItem-palveluita. Palvelut ovat määritelty OPC UA -spesifikaation osassa 4 ja niihin tutustutaan kappaleessa 4. Viittauksia käytetään solmujen yhdistämiseen ja viitauksiin pääsee kiinni käyttämällä browsing- ja querying-palveluita.

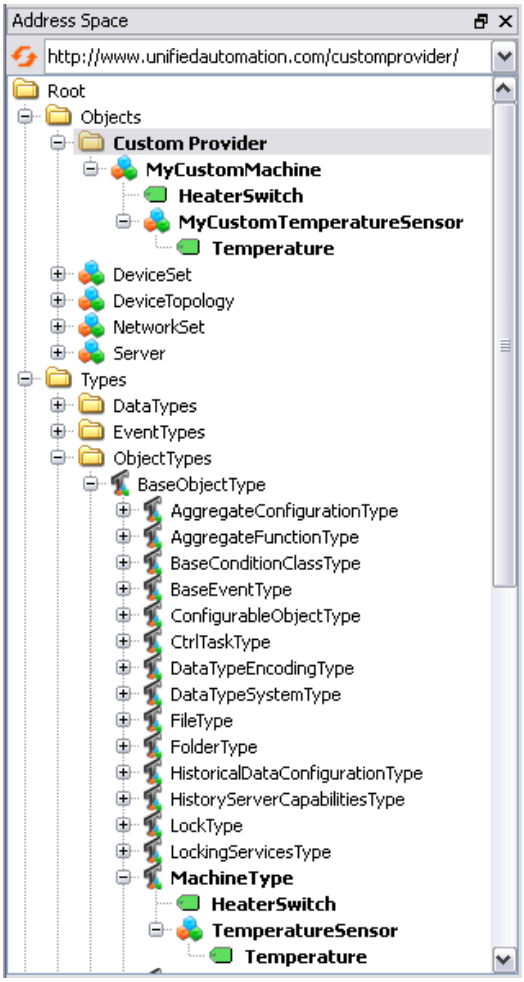
Kuvassa 6 on esitetty solmu sekä siihen kuuluvat elementit ja kuvassa 7 solmujen eri luokat. Liitteessä 1 on kerrottu tarkemmin solmujen luokista sekä niiden määrittelyistä (Mahnke ym. 2009; OPC Unified Architecture e-kirja).



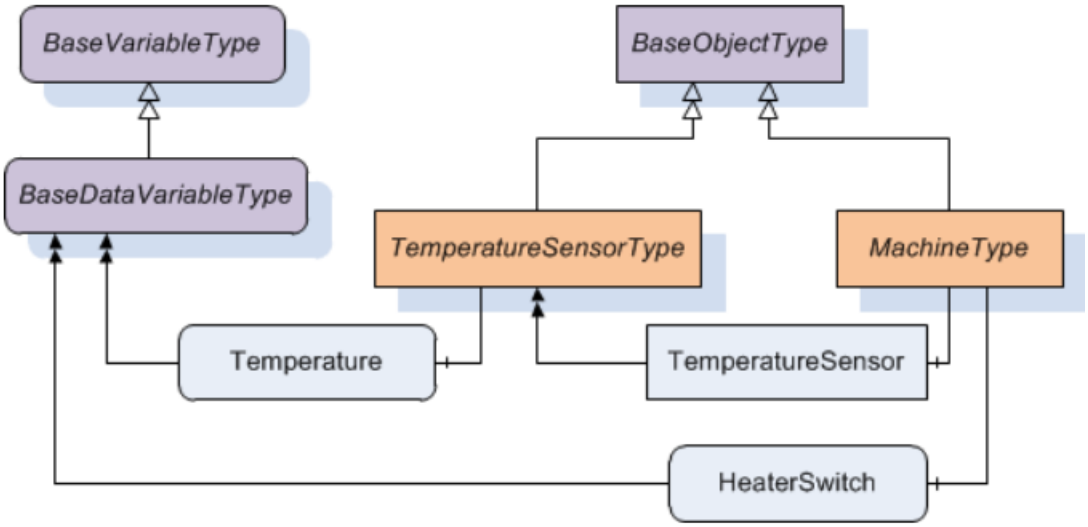
KUVA 6. OPC UA-osoiteavaruuden solmu-malli.(Lähde:Unified Automation)

KUVA 7. OPC UA-osoiteavaruuden luokat solmuille.(Lähde: Unified Automation)

Esimerkki osoiteavaruudesta on esitetty kuvassa 8, missä on kuvattu OPC UA-palvelimen oletusosoiteavaruus ja toinen, laajennettu osoiteavaruus (Kuva 9), jossa kuvataan johonkin laitteeseen kuuluva lämmityselementti ja lämpötila-anturi.



KUVA 8. Esimerkki OPC UA:n osoiteavaruudesta. (Lähde: Unified Automation)



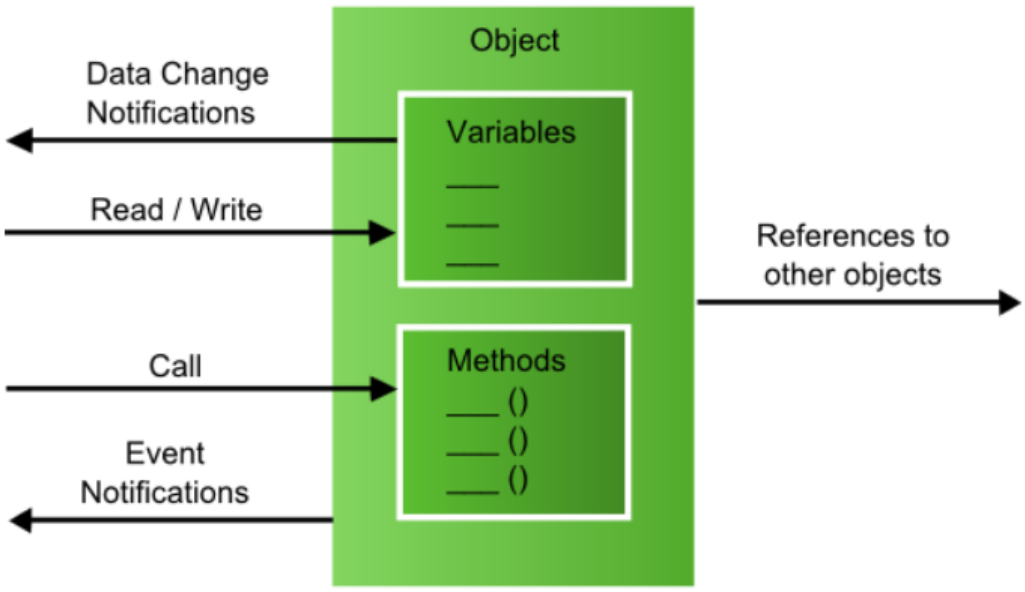
KUVA 9. OPC UA:n osoiteavaruuden laajennus.(Lähde: Unified Automation)

3.3 Oliomalli

Yksi OPC UA:n pääasiallisista tarkoituksista on tarjota palvelimille yhtenäinen menetelmä, jolla ne voivat esittää reaaliaikaisten prosessien tilan asiakkaille yksiselitteisesti riippumatta ohjelmistoalustasta. Toisin sanoen, oliomallia käytetään OPC UA -palvelimella olevan osoiteavaruuden rakenteen kuvaamiseen. Tätä tarkoitusta varten on suunniteltu oliomalli, joka määrittelee palvelimen dataoliot. Dataoliot sisältävät siihen liittyvät muuttujat *(variables),* menetelmät *(methods)* ja tapahtumat *(events)*. Olioilla voi olla myös viittauksia toisiin olioihin (OPC Unified Architecture e-kirja)

Dataoliot koostuvat muuttujista ja menetelmistä. Muuttujat esittävät jotakin arvoa ja sen tyyppi riippuu muuttujasta itsestään. Muuttujaa käytetään kuvaamaan esimerkiksi lämpötila-anturin mittaamaa lämpötilaa tai prosessisäätimen asetusarvoa. Asiakas voi lukea muuttujien esittämiä arvoja, kirjoittaa niihin ja saada ilmoituksia tilojen muutoksista (Mahnke ym. 2009; OPC Unified Architecture e-kirja).

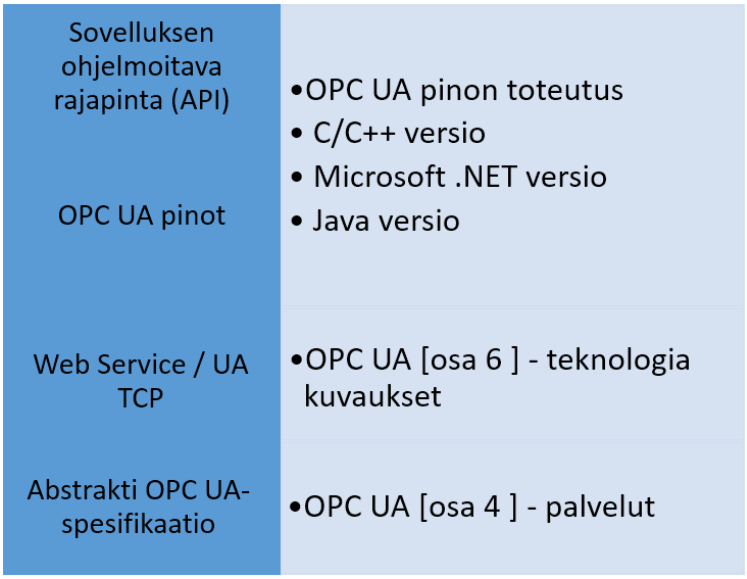
Menetelmä on sellainen osa palvelimen dataoliota, mitä asiakas voi kutsua ja johon palvelin vastaa lähettämällä tuloksen. Yleisesti metodeita on järkevää käyttää, kun joukko argumentteja on määritetty syötteeksi ja tulosteeksi, tai kun serverillä halutaan käynnistää tietty toiminto. Metodin perus ominaisuutena on suhteellisen hyvä suoritusnopeus. OPC UA:ssa käytetty oliomalli on esitetty kuvassa 10 (Mahnke ym. 2009; OPC Unified Architecture e-kirja).



KUVA 10. OPC UA:n oliomalli. (Lähde: Unified Automation).

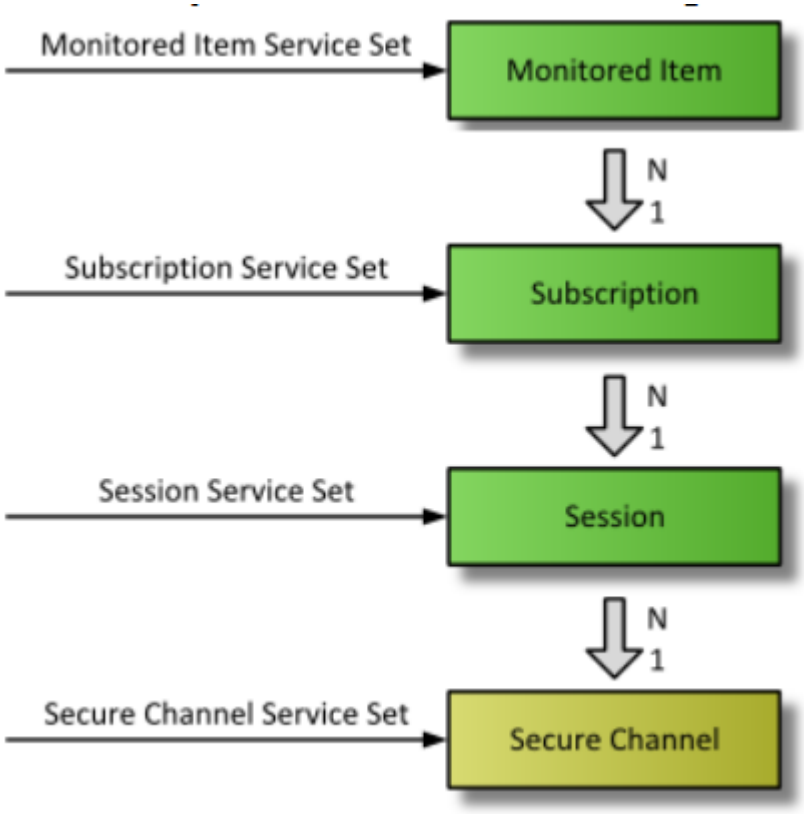
4 PALVELUT

OPC UA on palvelukeskeinen arkkitehtuuri (*Engl. Service Oriented Architecture),* joka noudattaa palvelupyyntö/vastaus-mallia. Palvelukeskeinen arkkitehtuuri tarkoittaa yleisellä tasolla ohjelmistotekniikan suunnittelutapaa, jossa tietojärjestelmien toiminnot tai prosessit ovat suunniteltu toimimaan itsenäisinä, avoimina sekä joustavina palveluina. Palvelujoukot määrittävät sovellustason kommunikaation palvelimen ja asiakkaan välillä. OPC UA -spesifikaation osassa 4 kuvatut palvelut ovat joukko abstrakteja RPC *(Remote Procedure Call)* -etäkutsuja, joilla kommunikointi palvelimien ja asiakkaiden välillä tapahtuu. Asiakas käyttää näitä palvelumetodeita päästäkseen kiinni palvelimen informaatiomallissa sijaitsevaan tietoon. OPC UA -palvelimen ei kuitenkaan tarvitse tukea jokaista määriteltyä palvelua, sillä tarvittavat palvelut määräytyvät profiilien mukaan. Profiilit määrittelevät sovellusten toiminnollisuudet ja ne jakautuvat eri kategorioihin, kuten esimerkiksi palvelin-, asiakas- tai tietoturvaprofiileihin. OPC UA:n riippumattomuus siirtoprotokollasta ja sovellusympäristöstä on oleellinen parannus verrattuna perinteisiin OPC rajapintoihin ja COM/DCOM-riippuvuuteen. Lisäksi palveluiden abstrakti määritys mahdollistaa niiden soveltamisen eri siirtomekanismeihin sovellustasolla ja eri palveluidenyhdistämisen verkkokerroksessa. Sovelluskohtaiset C/C++-, .NET tai Java-ohjelmointikielillä toteutettavat ohjelmoitavat rajapinnat ovat määritelty OPC UA -pinossa *(Engl.OPC UA- stack)*, joka perustuu OPC UA:n abstraktiin palvelumäärittelmään. Kuvassa 11 on esitetty OPC UA:n kommunikaatiokerrokset (Mahnke ym. 2009, 125; Wikipedia, Palvelukeskeinen arkkitehtuuri; OPC Foundation, Spesifikaatio osa 4: Palvelut, Versio 1.03).



KUVA 11. OPC UA kommunikaatiokerrokset.

Palveluita käytetään kommunikaation eri tasojen luomiseen ja muokkaamiseen. Suojattu tiedonsiirtokanava *(Secure Channel)* on matalan tason protokollariippuvainen kanava, joka suojaa palvelimen ja asiakkaan välisen kommunikaation ja viestien vaihdon. Kun tietoturvattu kanava luodaan sovellusten välille ensimmäisen kerran, sille määritellään kesto *(lifetime).* Tämä kanava tulee luoda uudelleen turvallisuussyistä ennen kuin sen elinaika umpeutuu. Seuraavaksi luodaan istunto *(Session)* kanavan päälle. Istunto varaa palvelimen resursseja, jotka voidaan vapauttaa myöhemmin istunnon päättyessä. Istunnon aikana voidaan luoda tilauksia *(Subscription).* Tilauksia käytetään palvelimen ja asiakkaan välillä datan ja tapahtumailmoitusten käsittelyssä. Valvottavat kohteet *(Monitored Items)* luodaan tilauksissa ja niitä käytetään verkon yksittäisten solmujen attribuuttien muuttuvan datan valvontaan sekä tapahtumailmoitusten monitorointiin. Yleinen kommunikaatiorakenne on havainnollistettu kuvassa 12 (Mahnke ym. 2009)

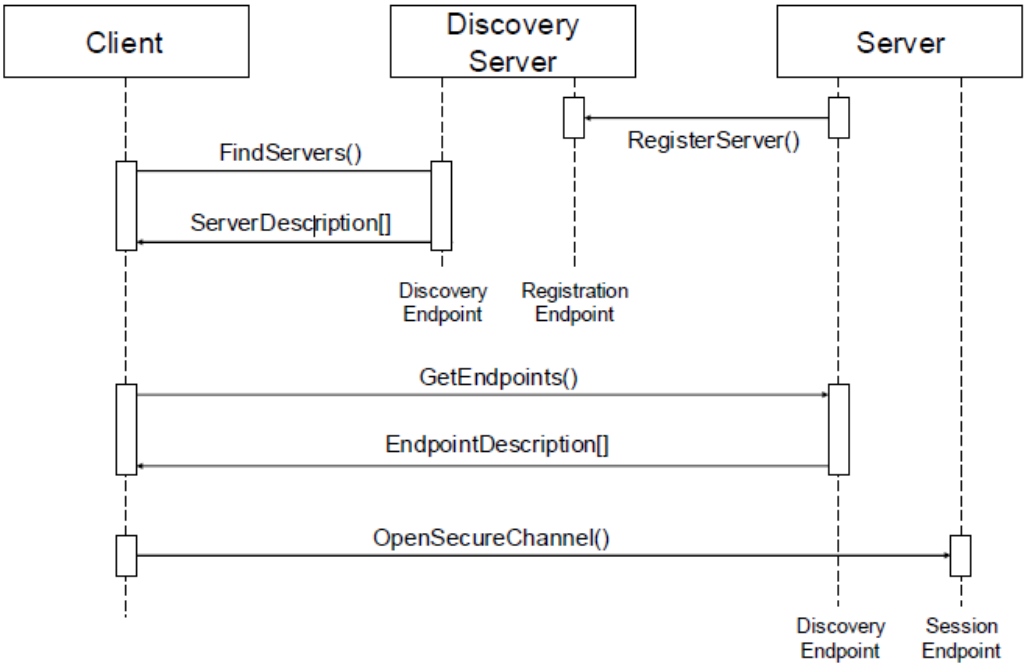


KUVA 12. OPC UA:n kommunikaatiorakenne eri tasoilla (Lähde: Unified Automation)

OPC UA:n palvelumääritykset ovat abstrakteja kuvauksia eivätkä näin ollen esitä määrittelyä toteutukselle. Palvelukuvausten ja niistä johdetun kommunikointipinon välinen yhteys on määritelty standardin osassa 6 [Mappings]. Jos kyseessä on Web Service implementaatio, niin OPC UA-palvelut vastaavat kyseisen implementaation toimintoja.

OPC UA:n palvelumääritykset koostuvat seuraavista palvelujoukoista:

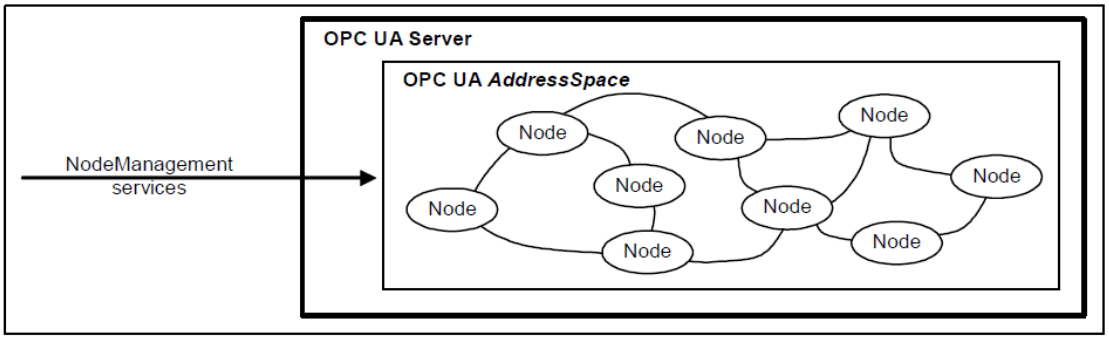
- *Discovery Service Set* -palvelujoukkoa käytetään palvelimien loppupisteiden *(Endpoints)* paikantamiseen sekä tietoturvamääritysten lukemiseen. Jokaisella palvelimella on *Discovery Endpoint*, johon asiakas pääsee luomatta istuntoa. Luodakseen suojatun tiedonsiirtokanavan, asiakas lukee tarvittavat tietoturvamääritykset palvelimelta kutsumalla *GetEndPoints-palvelua*. Tämän lisäksi palvelimet voivat rekisteröidä itsensä *LDS*-palvelimelle *(Local Discovery Server)* käyttämällä *RegisterServer*-palvelua, jos sellainen on käytössä. *LDS*-palvelin on paikallisten palvelimien hakuun tarkoitettu palvelin ja se sisältää tiedot palvelimista, jotka sijaitsevat samassa laitteessa missä se itsekin sijaitsee. Asiakkaat löytävät kaikki rekisteröidyt palvelimet kutsumalla *FindServers*-palvelua. Kuvassa 13 on esitetty *FindServers*-prosessi.



KUVA 13. Palvelinten etsintäprosessi (Lähde: OPC UA Spesifikaatio osa 4: Palvelut)

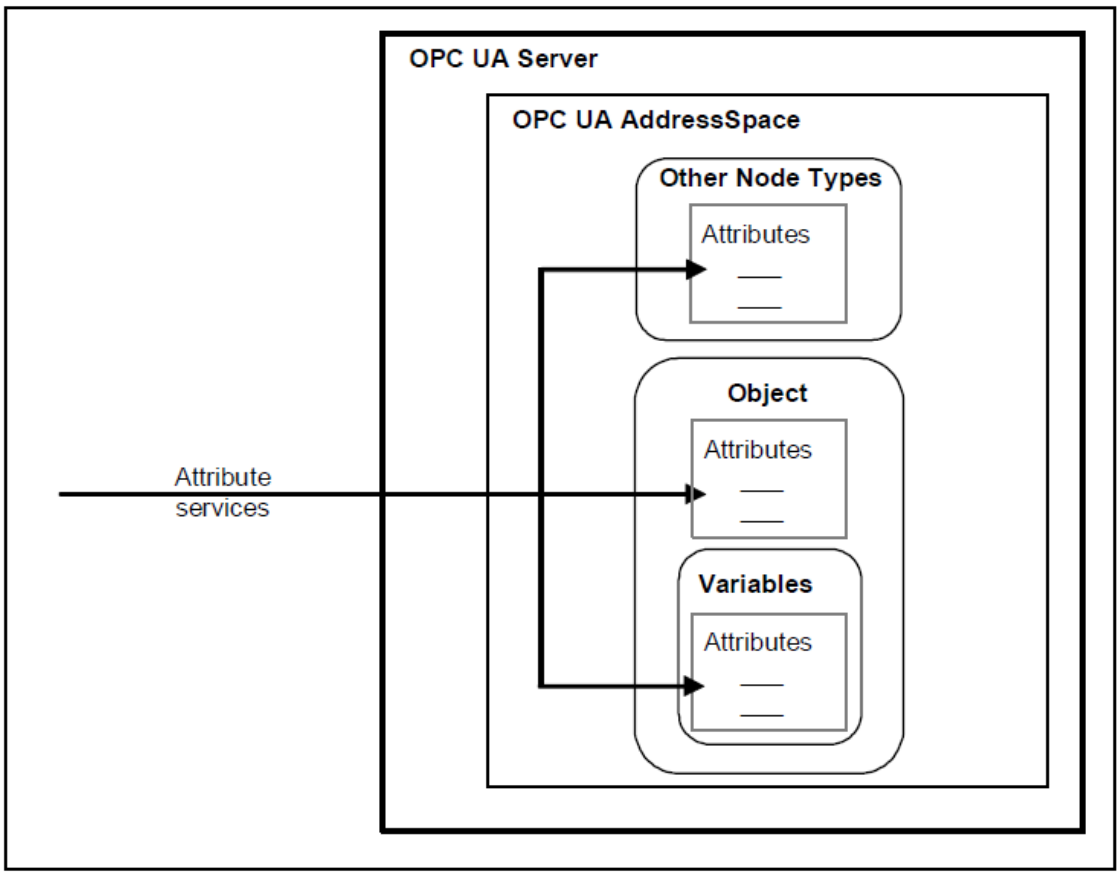
- *Secure Channel Service*- ja *Session Service* -palvelujoukot määrittelevät tietoturvaan liittyvät palvelut, joita sovellukset käyttävät. Palvelujoukkoja käytetään kommunikaatiokanavan avaamiseen palvelimen ja asiakkaan välille.

- *Node Management Service* -palvelujoukko, kuvassa 14, määrittelee palvelut, joilla osoiteavaruuden solmuja ja niiden välisiä viittauksia voidaan luoda tai poistaa.



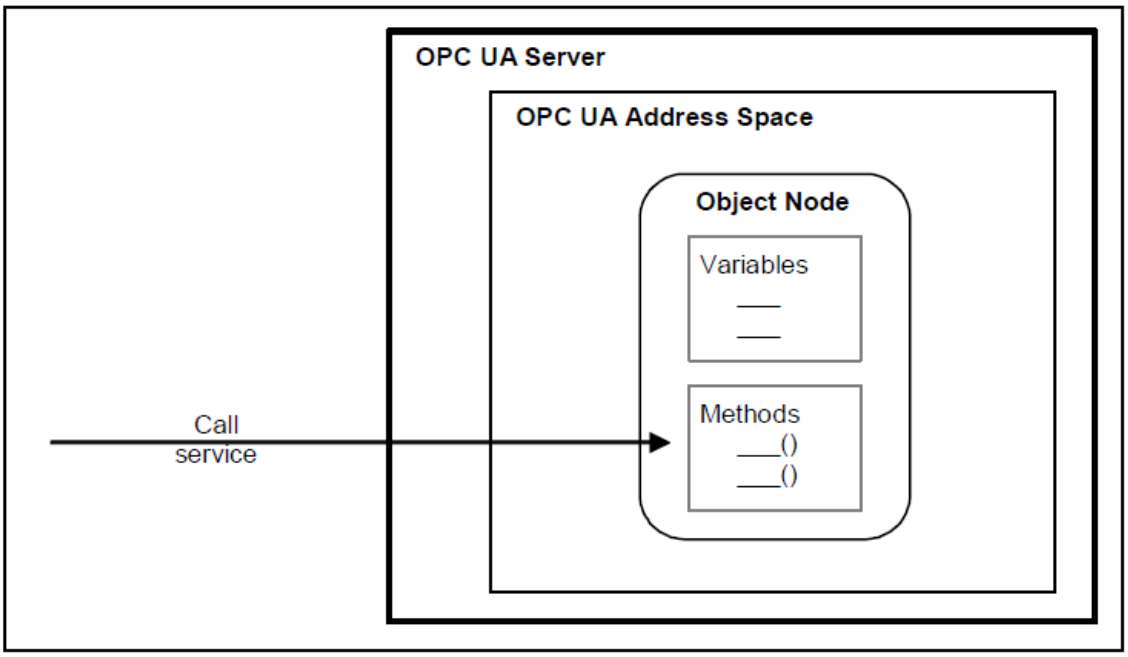
KUVA 14. Node management -palvelujoukko. (Lähde: OPC UA Spesifikaatio osa 4: Palvelut)

- *Attribute Service* -palvelujoukko, kuvassa 15, tarjoaa Read- ja Write- palvelut, joita käytetään reaaliaika- ja historia-datan lukemiseen ja kirjoittamiseen. Varsinainen data on solmujen attribuutteja.



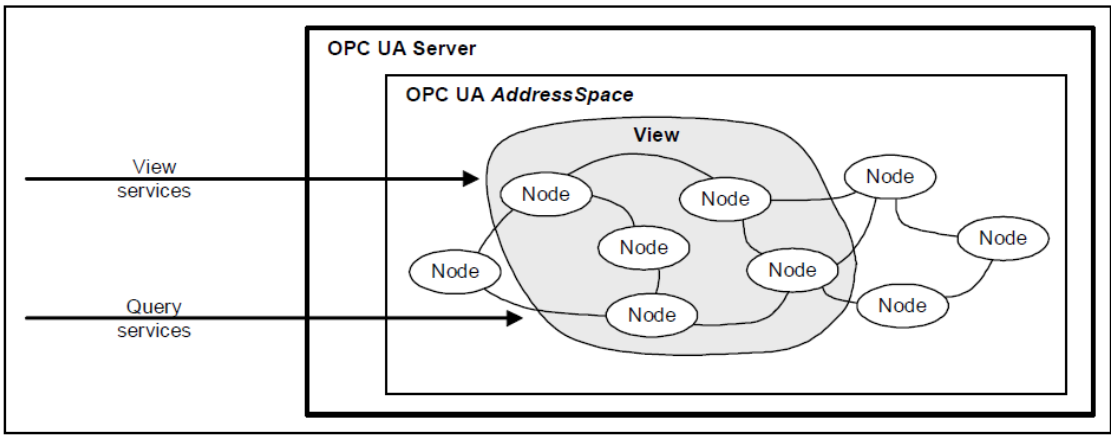
KUVA 15. Attribute-palvelujoukot. (Lähde: OPC UA Spesifikaatio osa 4: Palvelut)

- *Method Service* -palvelujoukko, kuvassa 16, määrittelee keinot metodien kutsumiseen. Metodi on olion komponentti ja *Call Service* -palvelua käytetään palvelimen metodien kutsumiseen.



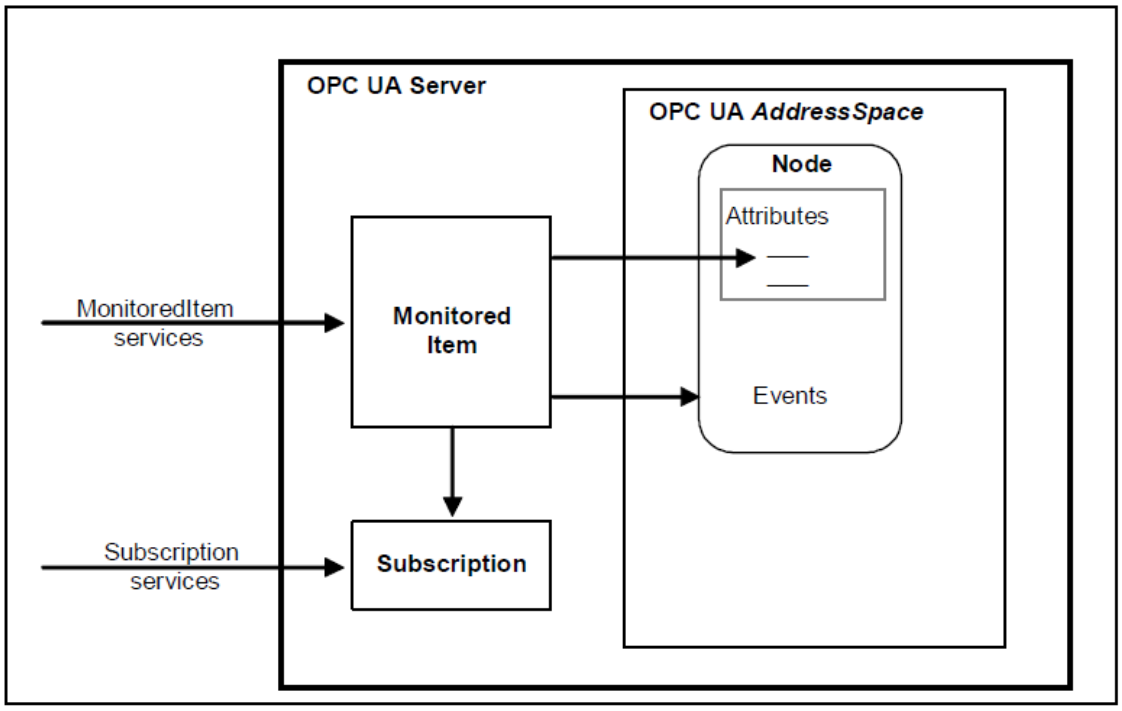
KUVA 16 Method service -palvelujoukko. (Lähde: OPC UA Spesifikaatio osa 4: palvelut)

- *Query* ja *View Service* -palvelujoukkoja käytetään palvelimen osoiteavaruudessa sijaitsevan tiedon löytämiseen. Kuvassa 17 on havainnollistettu *View- ja Query*-palvelujoukot.



KUVA 17. View- ja Query-palvelujoukot. (Lähde: OPC UA Spesifikaatio osa 4: Palvelut)

- *Monitored Item* -ja *Subscription Service* -palvelujoukot ovat asiakkaan määrittelemät palvelujoukot. Niitä käytetään datan ja tapahtumien tilaamiseen. Palvelujoukkojen toiminta on havainnollistettu kuvassa 18.



KUVA 18 Monitored Item -ja Subscription Service -palvelujoukot. (Lähde: OPC UA Spesifikaatio osa 4: Palvelut)

Palvelut käyttävät pyyntö/vastaus-mallia tiedonsiirtoon, mikä on tuttu www-sovelluspalvelussa *(Web Services).* Saadakseen halutun palvelun asiakas lähettää palvelimelle pyynnön, johon palvelin vastaa käsiteltyään tämän pyynnön. OPC UA:n kaikkia palveluita voidaan kutsua asynkronisesti tai synkronisesti, koska viestien vaihdon perusmekanismi on asynkroninen. Asynkronisessa kommunikaatiossa asiakas voi prosessoida muita toimintoja siihen asti, kunnes vastausviesti palvelimelta saapuu. Asynkroninen tiedonsiirto on yksi merkittävimmistä parannuksista vanhaan OPC Classic -tiedonsiirtoon, jossa liikennöinti on synkronista. Mahdollisia verkon toimintahäiriöitä varten OPC UA -kommunikaatiossa on käytössä aikakatkaisu-, *(timeout),* ja virheiden käsittelyä varten tilakoodit *(StatusCode)*. Jokaiselle palvelukutsulle voidaan yksilöllisesti määritellä aikakatkaisuajastus, jonka avulla verkkovirheet voidaan huomata. Palveluissa on käytössä kaksi eri virheilmoitustyyppiä, joista toinen on tilakoodi ja toinen vikadiagnostiikka *(Diagnostic Information)*. Tilakoodi on 32-bittinen etumerkitön kokonaisluku, missä eniten merkitsevät 16 bittiä kuvaa numeerisia arvoja, joita käytetään jonkin spesifisen virheen tai tilan

havaitsemiseen. Vähiten merkitsevät 16 bittiä ovat sarja bittejä *(Bit Flags)*, jotka sisältävät lisätietoja, mutta eivät kuitenkaan vaikuta tilakoodin sisältöön. Tilakoodin kaksi merkitsevämpää bittiä kuvaa yleisen tilan. Virheilmoituksen tiloja on kolme:

Good = onnistunut

Uncertain = varoitus

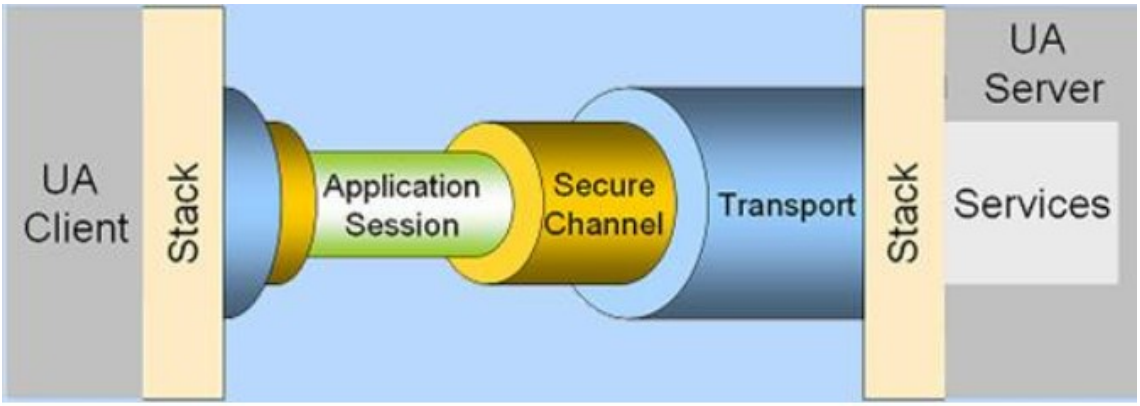
Bad = epäonnistunut

Vikadiagnostiikka voi puolestaan sisältää vikakoodien lisätietoja. Näitä lisätietoja voivat olla esimerkiksi laitetoimittajan määrittämät vikakoodit ja niiden tarkemmat lisätiedot. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 4: Palvelut, Versio 1.03)

OPC UA vaatii eri tason kommunikaatiokanavien luomisen varmistaakseen turvallisen, luotettavan ja joustavan tiedonsiirron asiakkaan ja palvelimen välillä.

Matalan tason verkon siirtokanava, mikä vastaa viestien vaihdosta ja loogisen suojatun kanavan, mikä varmistaa, että viestit käsitellään UA-pinoissa. Palvelimen ja asiakkaan sovellustason istunnoissa puolestaan varmennetaan käyttäjät. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 4: Palvelut, Versio 1.03).

OPC UA määrittelee yhteensä 37 palvelua, joista vain 16 on käytössä tiedonsiirtoon. Loppuja 21 palvelua käytetään eri kommunikaatiotasojen hallinointiin, kuten niiden luomiseen, ylläpitämiseen ja muokkaamiseen. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 4: Palvelut, Versio 1.03).



KUVA 19. OPC UA:n kommunikaatiokanavien eri tasot (Lähde: Automation.com)

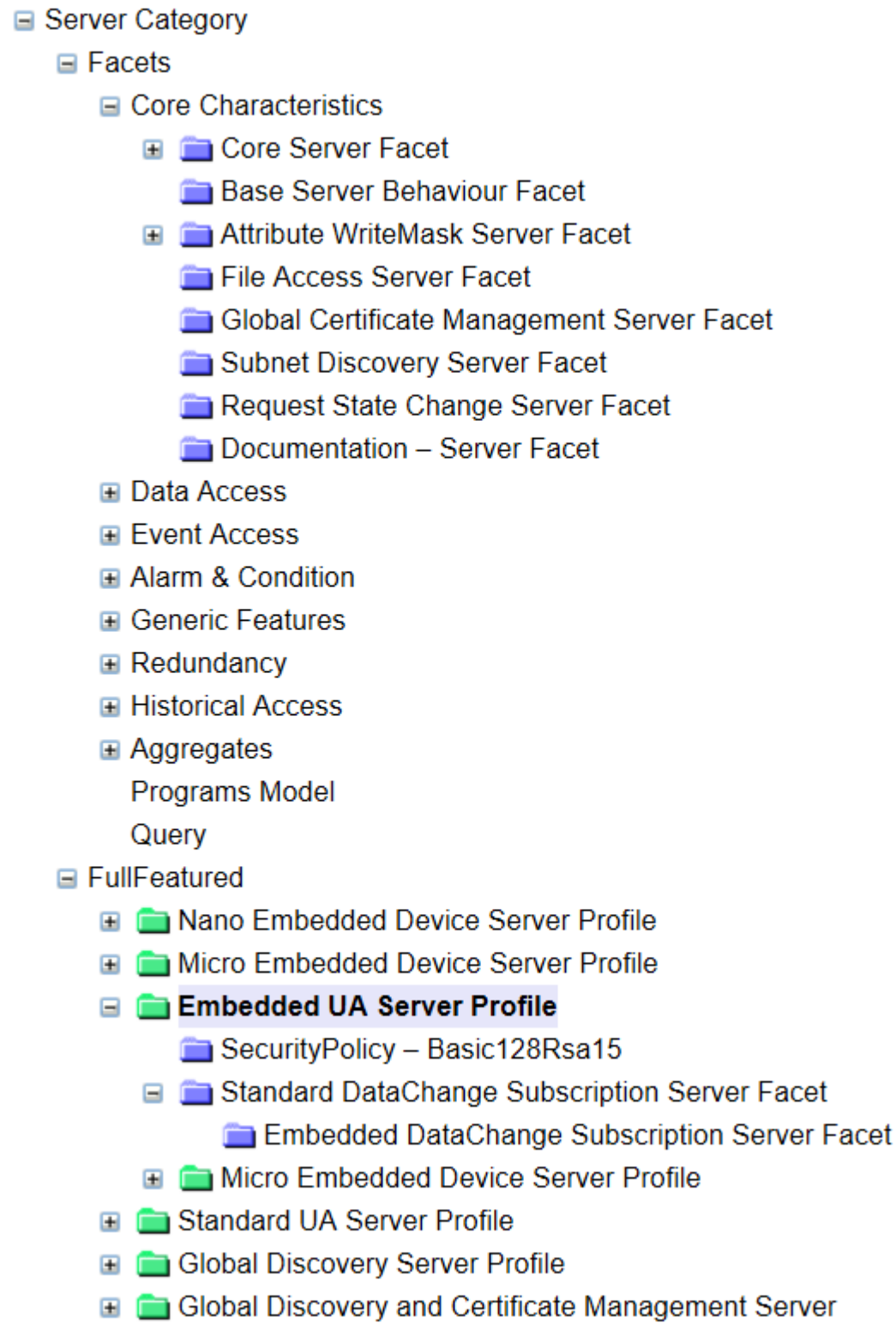
5 PROFIILIT

OPC UA yhdistää OPC DA-, OPC HDA- ja OPC A&E -toiminnollisuudet yhdeksi kokonaisuudeksi ja sen lisäksi esittelee joukon uusia lisäominaisuuksia, kuten metodit ja historia-tapahtumat. Kaikkien sovellusten ei tarvitse kuitenkaan tukea kaikkia mahdollisia toiminnallisuuksia. Esimerkiksi sulautetun laitteen palvelin ei välttämättä toimita ollenkaan historiatapahtumia tai ei tue tapahtumatilauksia. Lisäksi jotkin palvelimet pystyvät seuraamaan osoiteavaruudessa tapahtuvia muutoksia ja toiset eivät. Sama pätee asiakassovelluksiin, sillä jotkin pystyvät käsittelemään ainoastaan reaaliaikaista dataa ja toiset pystyvät vain tilaamaan tapahtumia. Profiilit määrittelevät sovellusten toiminnollisuudet. OPC UA -sovellus voi tukea useita eri profiileja ja jokainen profiili voi sisältää toisia profiileja. Profiilit jakautuvat neljään eri kategoriaan: asiakasprofiileihin, palvelinprofiileihin, tietoturvaprofiileihin ja siirtoprofiileihin. Sovelluksia voidaan testata käyttämällä erikseen määriteltyjä testejä, joilla varmistetaan mitä profiileja sovellus tukee. Testauksia suorittavat itsenäiset testaustahot, jotka myöntävät sovelluksille sertifikaatit. Nämä sertifikaatit sisältävät tiedot profiileista, joita sovellus tukee.

Profiilit on määritelty OPC UA -standardin osassa 7 ja profiilien testikuvaukset erillisissä testispesifikaatioissa [OPCTL-osa 8] ja [OPCTL-osa 9]. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 7: Profiilit, Versio 1.03).

5.1 Palvelin – ja asiakas sovellusten profiilit

Palvelinprofiileja on olemassa kahdenlaisia: täydennysprofiileja *(Facets)* ja täysien ominaisuuksien käsittäviä profiileja *(Full Feature)*. Näistä jälkimmäistä oletettavasti suurin osa sovelluksista tukee. Määrittelyssä vaaditaan, että palvelimen on tuettava vähintään yhtä täyden ominaisuuden käsittävää profiilia. Täydennysprofiilit määrittelevät eräitä yksittäisiä lisäominaisuuksia palvelimelle, esimerkiksi tuen tapahtumatilauksille. Asiakasprofiilit määrittelevät ainoastaan täydennysprofiilit, koska asiakassovellukset tukevat harvoin samoja profiiliryhmiä. Kuvassa 20 on lista palvelimen täydennysprofiileista ja täysien ominaisuuksien profiileista. Kuvassa näkyy laajennettuna yksittäinen lisäominaisuus *(Core Characteristic)* ja sulautettu palvelin-profiili *(Embedded UA Server Profile).* (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 7: Profiilit, Versio 1.03).



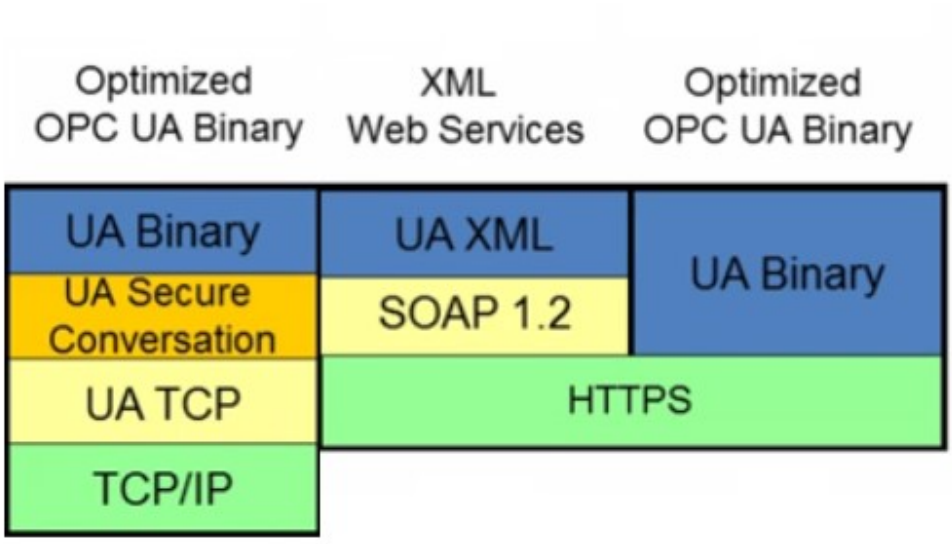
KUVA 20. Palvelinprofiilit. (Lähde: OPC Foundation)

5.2 Tietoturvaprofiilit

Tietoturvaprofiilit määrittelevät viestienvaihdossa käytettävät algoritmit ja avainten pituudet, mitä käytetään viestien allekirjoittamiseen ja salaukseen. Tietoturvaprofiileissa määritellään aina pelkästään yksittäisiä täydennysprofiileja. Tällä hetkellä profiileja on määritelty neljä: None, Basic128Rsa15, Basic256 ja Basic256Sha256. Määrityksiä tehdään tulevaisuudessa todennäköisesti lisää, koska nykyiset salausalgoritmit voidaan luultavasti murtaa laskentatehojen lisääntyessä. Basic128Rsa15-tietoturvaprofiilin mukaisessa salauksessa käytetään symmetristä AES-lohkosalaus algoritmia viestien salaukseen, 128 bitin mittaista avainta allekirjoitukseen, sekä RSA1.5 algoritmia asymmetriseen salaukseen (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 7: Profiilit, Versio 1.03).

5.3 Kuljetusprofiilit

Kuljetusprofiilit määrittelevät kommunikaatioprotokollat, joita OPC UA -sovellukset tukevat. Kuljetusprofiilit ovat kaikki täydennysprofiileja. UA-standardin versio 1.03 määrittelee profiileita kolme kappaletta, yhden jokaiselle järkevälle tietoturva-, siirto-, ja koodausprotokollayhdistelmälle. Kuvassa 21 on esitetty tiedonsiirtokerrosten eri protokollat: UA XML ja UA binäärikoodaus/dekoodaus protokollat, UA-Secure-tietoturvaprotokolla ja HTTPS sekä UA-TCP-siirtoprotokollat (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 7: Profiilit, Versio 1.03).



KUVA 21. Kommunikaatioprotokollat. (Lähde: Ascolab)

6 ARKKITEHTUURI

OPC UA -pino *(Engl. Stack)* on sovellusten yhteinen osa, joka pitää sisällään alemman tason toiminnallisuudet. Pino jaetaan neljään eri osaan: alustakerrokseen, koodaus/dekoodauskerrokseen, tietoturvakerrokseen ja siirtokerrokseen.

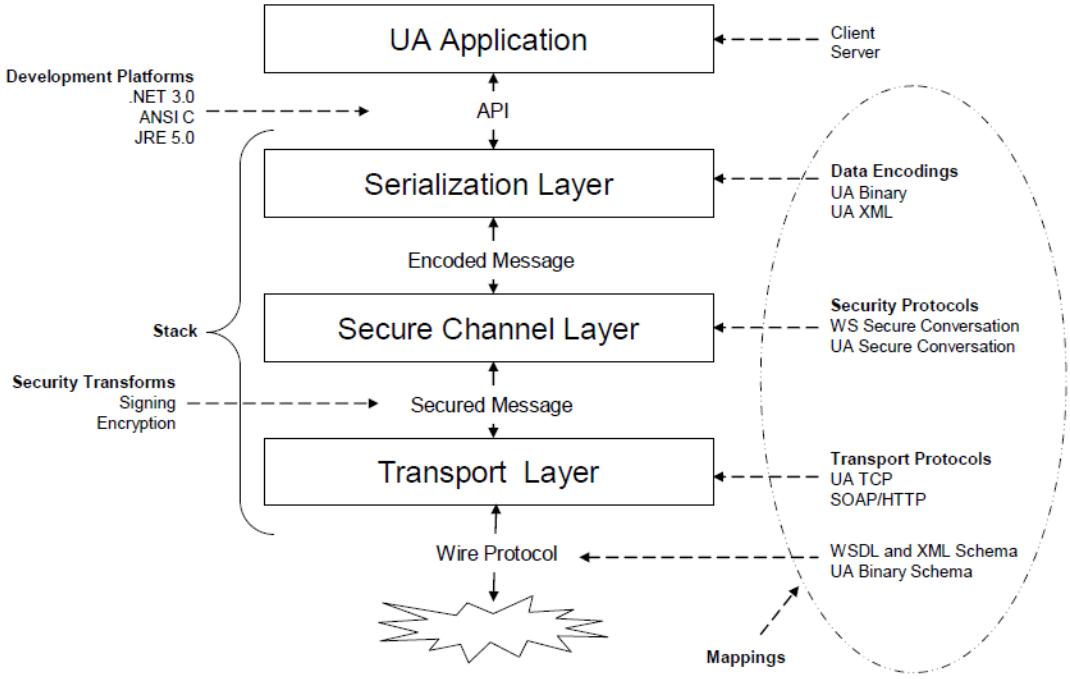
OPC UA -standardi määrittelee kaksi tapaa datan koodaukselle, OPC UA -binäärin ja OPC UA XML:n. Standardissa kuvataan tarkemmin kuinka viestit rakennetaan molemmissa tapauksissa. SDK- *(Software Development Toolkit)* ja sovelluskerrokset luovat yhteyden pinoon asiakas-ja palvelinkerrosten kautta, jotta viestienvaihto onnistuisi. Saatuaan viestin koodaus- ja dekoodaus-kerros suorittaa viestin purkamisen ja muokkaamisen sarjaksi tavuja *(Serialization).*

Tämän jälkeen viesti siirtyy tietoturvakerrokselle, joka suojaa koodatut viestit joko allekirjoituksella tai allekirjoituksella ja salauksella *(Sign – Sign and Encrypt)* riippuen konfiguraatiosta. Sovelluksia käytetään usein myös suljetuissa verkoissa, joissa viestien suojaamista ei välttämättä tarvita ollenkaan tai sitä ei haluta käyttää. Tällaisissa tapauksissa kommunikaatioasetuksista voidaan määritellä, että viestien suojausta ei tehdä. Viestien otsikot ja alatunnisteet sisältävät tiedon siitä kuinka itse viesti on suojattu. Lukemalla nämä tiedot vastaanottaja osaa käyttää oikeata menetelmää viestien purkamiseen ja allekirjoitusten varmentamiseen.

Tietoturvakerrokselta viesti kulkee siirtokerrokselle. Siirtokerros vastaanottaa ja välittää viestit sekä käsittelee verkossa mahdollisesti ilmenevät virheet. Ennen viestien lähettämistä siirtokerros lisää viesteihin otsikon, joista ilmenee muun muassa viestin pituus ja tyyppi. Vastaanottajan siirtokerros puolestaan käyttää näitä tietoja viestien eheyden varmentamiseen. Varmennuksessa vastaanottaja tunnistaa sekä viestityypin että viestin pituuden ennen kuin se lähettää viestin eteenpäin vastaanottajan tietoturvakerrokselle. Kuvassa 22 on esitetty OPC UA -pino ja sen neljä eri osaa sekä havainnollistettu viestien kulku pinon eri kerrosten välillä.

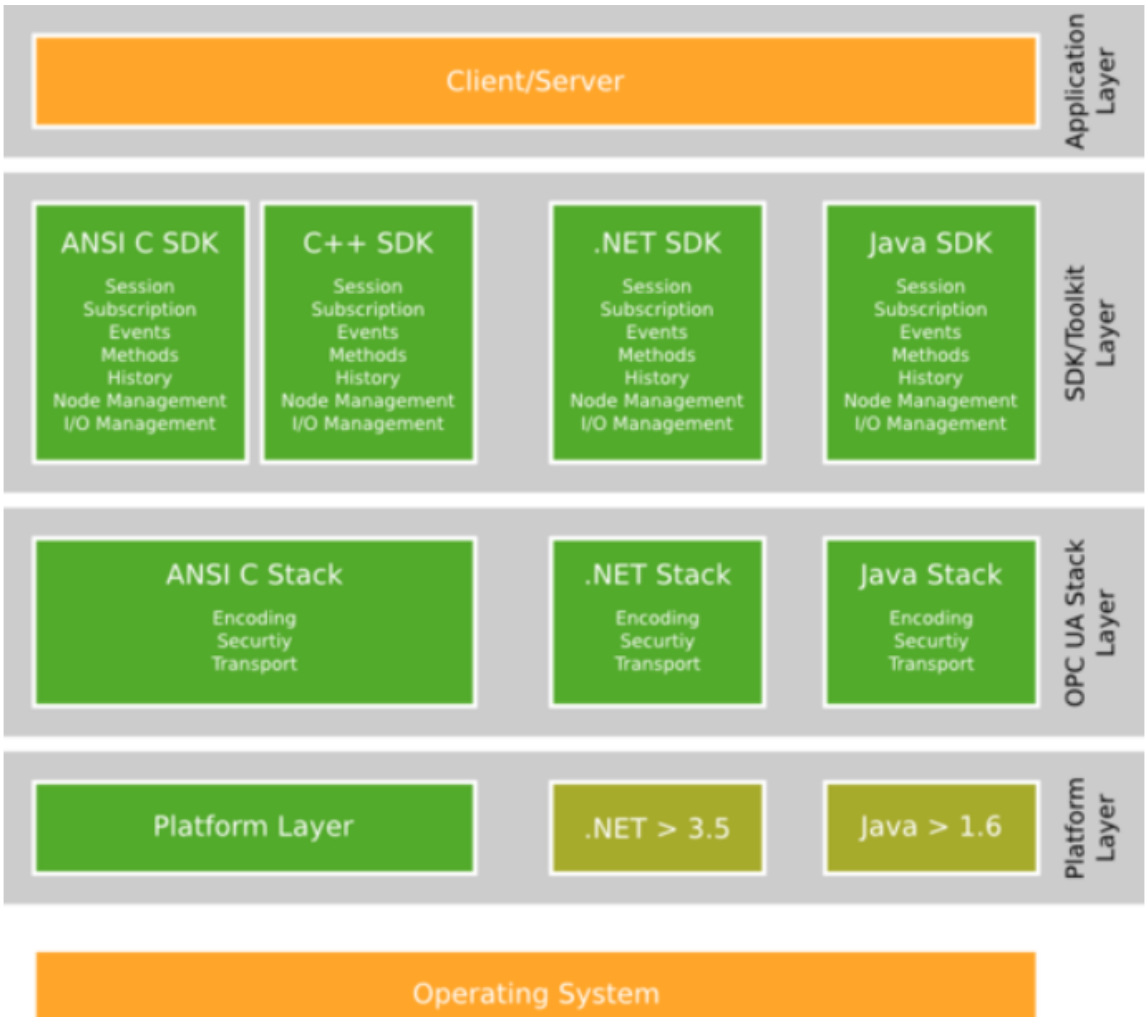
Pinon uudelleenkäyttöä voidaan parantaa lisäämällä siihen ohjelmistoalustakerros (*Platform Layer*). Perusajatuksena on se, että kaikki muut kerrokset voidaan kehittää ohjelmistoalustariippumattomalla tavalla. Täten ainoastaan alustakerros sisältäisi alustakohtaiset

kirjastot, kuten esimerkiksi pistokkeet *(Sockets)* ja säikeet *(Threads)*. Tämä tarkoittaa sitä, että vain alustakerrosta tarvitsee muokata, jos pinoa halutaan käyttää eri alustoilla ja muu koodi voidaan käyttää uudestaan, mikä helpottaa ja nopeuttaa ohjelmointia (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).



KUVA 22. OPC UA pino eri variaatioilla. (Lähde: OPC Foundation)

OPC UA -pinon päällä sijaitsee SDK-kerros, jonka tehtävänä on kattaa ylemmän tason toiminnot. Kerros jakautuu OPC UA -kohtaisiin toiminallisuuksiin, yhteisiin toiminnollisuuksiin ja asiakas/palvelin-rajapintoihin. OPC UA -kohtaiset toiminnot sisältävät OPC UA:n määrittämät konseptit ja palvelut, kuten istunnot, tapahtumat ja solmut. Nämä yhteiset toiminnot täytyy toteuttaa sekä asiakkaan että palvelimen päässä. Yhteiset toiminallisuudet pitävät sisällään yleisiä funktioita, jotka sovellusten täytyy toteuttaa, kuten esimerkiksi sertifikaattien validointi, kirjaukset ja konfiguraatiot. OPC UA:ssa määritellään vain sertifikaatin osa, joka sovellusten kuuluu tarkistaa, mutta se ei määrittele kuinka se kuuluisi tehdä. Sertifikaattien tarkistus voidaan tehdä myös pinossa, koska tarkistusmenettely on oletettavasti samanlainen kaikilla sovelluksilla. Sekalaisissa ympäristöissä sertifikaattien validointi on kuitenkin järkevämpää tehdä SDK-kerroksessa tai sovelluskerroksella varsinkin, jos niissä joudutaan käyttämään useita eri menettelytapoja sertifikaattien kokoamiseen. Kuvassa 22 on esitetty mihin SDK-kerros sijoittuu sekä eri ohjelmointikielillä tehdyt kirjastot (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).



KUVA 23. SDK-yleiskuvaus (Lähde:Unified Automation)

7 TIEDONSIIRTO

Säilyttääkseen avoimuuden tulevaisuuden tekniikoille ja taatakseen yhteensopivuuden eri tuotteiden välillä, OPC UA -työryhmä määritteli abstraktit palvelut ja konseptit, jotka voidaan yhdistää konkreettisiin, jo olemassa oleviin sekä tulevaisuuden teknologioihin. OPC UA:n tiedonsiirtokerros *(Communication Stack)* sekä muut kommunikointipinon kerrokset voidaan toteuttaa tällä hetkellä kahdella eri tavalla, joko OPC Foundation määrittelemällä UA TCP -protokollalla tai SOAP/HTTPS-protokollalla. Tiedonsiirtokerros ei ole kuitenkaan protokollariippuvainen (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

Tällä hetkellä standardin osassa kuusi määritellään kaksi tapaa datan koodaukselle; binääri ja XML, mutta standardi jättää mahdollisuuden muidenkin tekniikoiden käyttämiselle tulevaisuudessa.
Binäärikoodauksessa koodaus/dekoodaus on nopeaa ja viestien koko on pieni. Sekä binääri että XML-koodauksessa nojaudutaan useisiin primitiivisiin datatyyppeihin, kuten esimerkiksi kokonaislukuihin ja liukulukuihin. Näiden koodaussäännöt on määritelty selkeästi ja niistä voidaan koostaa rakenteita. Näitä primitiivisiä datatyyppejä kutsutaan sisäänrakennetuiksi datatyypeiksi *(Built-In Data Types).* Binääridatan koodaus/dekoodaus ei sisällä datan tyyppiä eikä nimialue-informaatiota, koska sovellusten oletetaan tietävän etukäteen tuetut palvelut ja rakenteet. Poikkeuksena tälle on kuitenkin laajennusolio *(Extensions Object)*, jota käytetään monimutkaisen datan tunnistamiseen ja mikä pitää sisällään tiedot sisältämästään datasta ja siitä, kuinka se on koodattu.

Binäärikoodaus muodostuu kääntämällä primitiiviset datatyypit binäärimuotoon ja kirjoittamalla niitä jaksottaisesti binäärijonoksi. Taulukossa 2 on esimerkki OPC UA:n binäärikoodauksesta, jossa merkkijono ”OPCUA” on koodattu käyttämällä UTF-8-merkkejä. Koodaus alkaa merkkijonon pituudella (5) ja jatkuu sarjalla UTF-8-merkkejä (OPCUA). Liitteessä 2 on listattu sisäänrakennetut datatyypit. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

TAULUKKO 2. OPC UA - merkkijono binäärikoodattuna. (Mahnke ym. 2009)

5				O	P	C	U	A
05	00	00	00	4F	50	43	55	41

Toinen kommunikaation tiedonvälitystapa on XML *(Extensible Markup Language)* WS *(Web Services)*. XML on merkintäkieli, jolla tiedon merkitys sekä rakenne voidaan kuvata tiedon sisään. XML-formaatti auttaa laajojen tietomassojen jäsentelyssä. Se on sekä koneen että ihmisen luettavissa. XML WS -pohjainen tiedonsiirto mahdollistaa kommunikaation minkä tahansa sellaisen järjestelmän kanssa, joka voi kommunikoida käyttäen Web-palvelua. XML:n kehittäjä on World Wide Web Consortium.

Suurin osa sisäänrakennetuista datatyypeistä on koodattu noudattamalla yleisiä XMLspesifikaatioita W3C04a ja W3C04b, mutta joitakin rajoituksia ja erikoistapauksiakin on. Kuvassa 24 on esitetty ”NodeId” XML-mallin mukaisella esitystavalla (Extensible Markup Language; Mahnke ym. 2009)

```
<xs:complexType name="NodeId">
  <xs:sequence>
    <xs:element name="Identifier" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

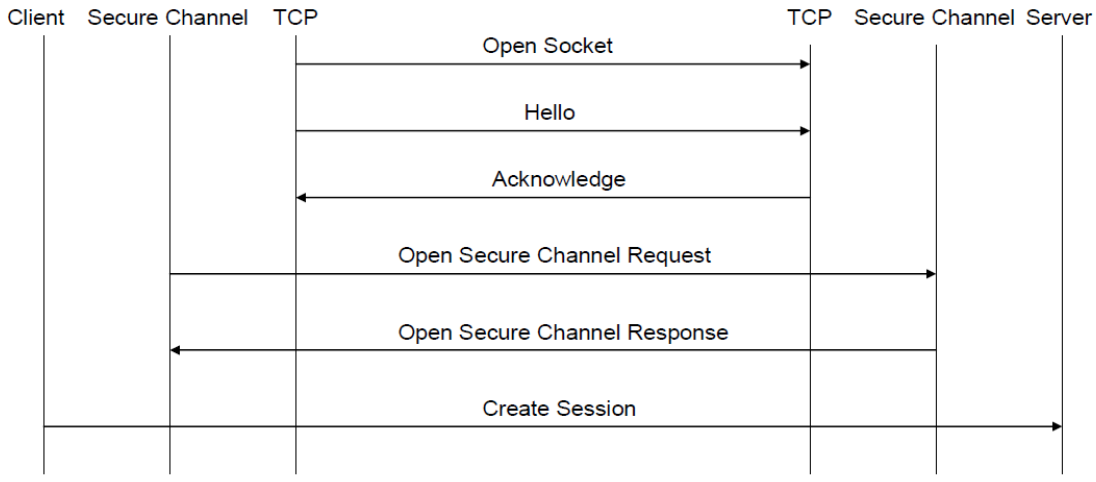
KUVA 24. OPC UA XML-malli.

7.1 Protokollat

OPC UA -standardin uusin versio 1.03 määrittelee kolme kommunikaatioprotokollaa tiedonsiirtoon: UA TCP:n, SOAP/HTTP:n ja HTTPS:n. UA TCP on tarkemmin ottaen binäärikoodattua dataa, joka lähetetään TCP/IP:n yli. UA TCP:n rekisteröity IANA *(Internet Assigned Numbers Authority)* -portti on 4840. Protokollia käytetään luomaan verkkotason yhteys palvelimen ja asiakkaan välille. Käyttöliittymätasolla UA-serverit identifioidaan käyttämällä URL-merkkijonoa samalla tavalla kuin Web-selaimessa. UA TCP protokollan etuliite on "opc.tcp://server" ja SOAP/HTTPS-protokollaa käytettäessä etuliite on ”https://server”. Lukuun ottamatta tätä eroa kahden protokollan välillä OPC UA toimii muuten täysin läpinäkyvästi suhteessa ohjelmointirajapintaan. UA TCP -viestin yleisrakenne koostuu viestin alkuosasta *(Header)* ja loppuosasta *(body)*. Viestin alkuosa sisältää tietoja viestin tyypistä ja pituudesta. Viestin loppuosa sisältää joko koodatut ja salatut palveluviestit, jotka kuljetetaan eteenpäin ylemmälle kerrokselle tai UA TCP -yhteyden luomiselle tarvittavia erityisviestejä, jotka kuvaillaan seuraavaksi lyhyesti. UA TCP viestityyppejä on kolme: Hello, Acknowledge ja Error message (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03)

Asiakas lähettää ”hello”-viestin palvelimelle luodakseen pistokeyhteyden *(socket connection)* palvelimen loppupisteeseen. Lisäksi asiakas neuvottelee palvelimen kanssa puskurikoot tiedon lähettämiseen ja vastaanottamiseen sekä viestin maksimikoon ja pituuden. Vastauksena tähän viestiin palvelin lähettää asiakkaalle acknowledgement-kuittausviestin, joka sisältää vahvistuksen asiakkaan pyytämiin tietoihin. Asiakkaalla ja palvelimella pitää olla yhteisymmärrys viestin maksimikoosta ja pituudesta, sillä ne vaikuttavat kommunikaation luotettavuuteen ja tietoturvaan. Kun kommunikaatioon liittyvät parametrit ovat molempien tiedossa, asiakas ja palvelin pystyvät vastustamaan esimerkiksi palvelunestohyökkäyksiä. Kolmatta viestityyppiä käytetään virheilmoituksien lähettämiseen. Virheilmoitus voidaan lähettää esimerkiksi, jos yhteydenluonti epäonnistuu viestin ylisuuren koon takia, jonka johdosta sitä ei voida prosessoida tai jos palvelin on ylikuormittunut (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

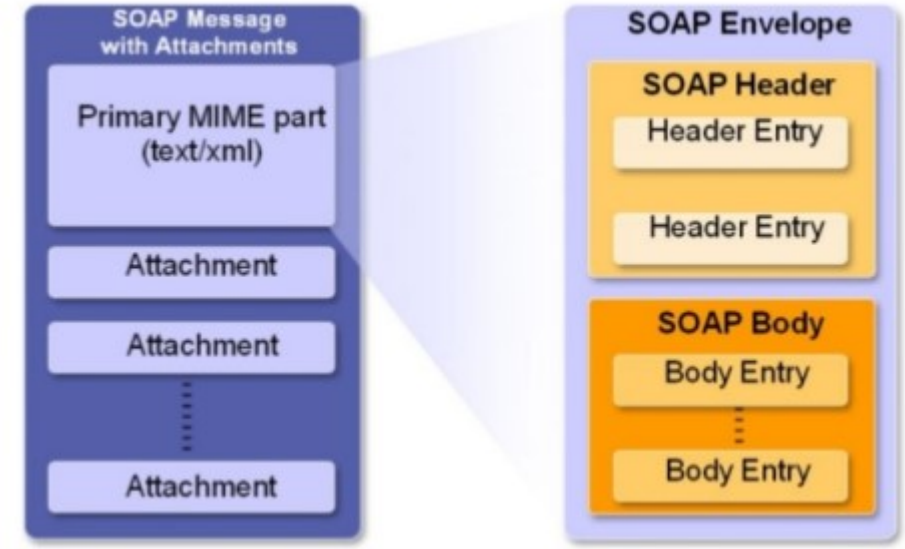
UA TCP on yksinkertainen TCP-pohjainen protokolla, joka muodostaa kaksisuuntaisen kommunikaatiokanavan palvelimen ja asiakkaan välille. UA TCP sallii vastausviestien palauttamisen missä tahansa järjestyksessä ja toiseksi vastausviestit voidaan palauttaa alkuperäisestä poikkeavaan pistokkeeseen, jos verkon kommunikaatiohäiriöt aiheuttavat tilapäisiä keskeytyksiä TCP-istuntoon. Jos yhteys katkeaa palvelimen ja asiakkaan välillä täytyy pistokeyhteys luoda uudelleen. Viestien lähetysjärjestys UA TCP -yhteydelle on esitetty kuvassa 25 (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).



KUVA 25. OPC UA TCP- yhteyden luominen. (Lähde: OPC Foundation).

SOAP/HTTPS-protokollaa käytetään hyvin yleisesti Web Service -ympäristöissä sen tietoturvan ja palomuuriystävällisyyden takia. SOAP/-HTTPS-protokollalla tarkoitetaan SOAP-kommunikaatiota HTTPS:n yli. SOAP on lyhenne sanoista *Simple Object Access Protocol* ja HTTPS sanoista *Hypertext Transfer Protocol Secure*. HTTPS-protokolla koostuu puolestaan HTTP-protokollasta, jossa käytetään TLS-salausmekanismia. TLS (*Transport Layer Security*) on salausprotokolla, jota käytetään verkon yli tapahtuvan tietoliikenteen suojaamiseen. TLS-yhteyden avaamisessa asiakas ja palvelin sopivat yhteyskohtaisista ja kertakäyttöisistä salausavaimista, joita käytetään istunnon aikana tiedon salaamiseen ja tulkintaan. UA-binäärin ja HTTPS:n yhdistelmää kutsutaan hybridi-protokollaksi, jossa yhdistyy molempien protokollien edut. Siinä binäärikoodattu hyötykuorma kuljetetaan HTTPS-kehyksessä. Hybridiprotokolla on tarkoitettu käytettäväksi pääasiassa Internetin yli tapahtuvaan kommunikaatioon, jossa on mahdollista käyttää ainoastaan TCP-porttia 443, joka on HTTPS-protokollan rekisteröity oletusportti. Näiden kahden protokollan välillä olennaisin ero on niiden operointi OSI-mallin eri tasoilla. HTTP on sovelluskerroksen protokolla, kun taas TCP on siirtokerroksen protokolla (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

SOAP on tietoliikenneprotokolla, jonka pääasiallisena tehtävänä on mahdollistaa proseduurien etäkutsu (RPC) ja järjestelmien välinen tiedonvaihto. SOAP nojautuu useisiin muihin standardeihin kuten esimerkiksi XML:ään, jota SOAP käyttää tiedon esittämiseen sekä HTTP että TCP-protokolliin tiedon kuljettamisessa. Kuvassa 26 on havainnollistettu SOAP-viestin rakenne.



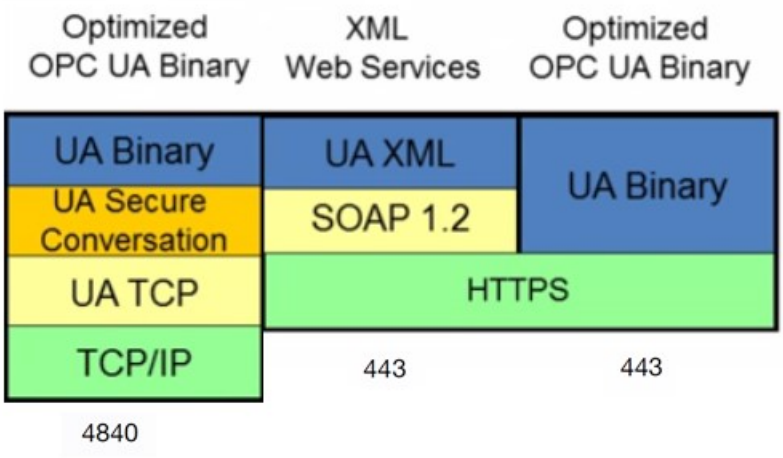
KUVA 26. SOAP-viestin rakenne. (Lähde: Java Web Services, Presentations for Java Web Services Course)

SOAP-viesti rakentuu otsikosta ja rungosta. Otsikko sisältää tietoja viestin osoitteesta sekä reitityksestä ja runko varsinaisen tietosisällön. XML-koodatut SOAP-viestit vaativat enemmän prosessointitehoa verrattuna binäärikoodattuun TCP-tiedonsiirtoon, jonka takia protokolla on suunnattu yritystason sovelluksien väliseen kommunikointiin Web Service ympäristöihin. Alla olevassa kuvassa 27 on kuvattu SOAP-viestin rakenne XML-muodossa (Java Web Services Architecture).

```
<?xml version ="1.0"?>
< Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Header>
 </s:Header>
 <s:Body> (joko XML koodattu tai UA binäärikoodaatu palveluviesti)
 </s:Body>
</s:Envelope>
```

KUVA 27. SOAP-viesti XML-muodossa.

Protokollista binääriprotokolla on pakollinen, mitä kaikkien UA-ohjelmistojen täytyy tukea, kun taas Web Service (XML-SOAP) ja hybridi (UA-binääri HTTPS:n yli) ovat valinnaisia. Kuvassa 28 on havainnollistettu eri tiedonsiirtoprotokollat käytettävine TCPportteineen (OPC UA_brochure, 2013)



KUVA 27. OPC UA:n tiedonsiirtoprotokollat. (Lähde: Ascolab)

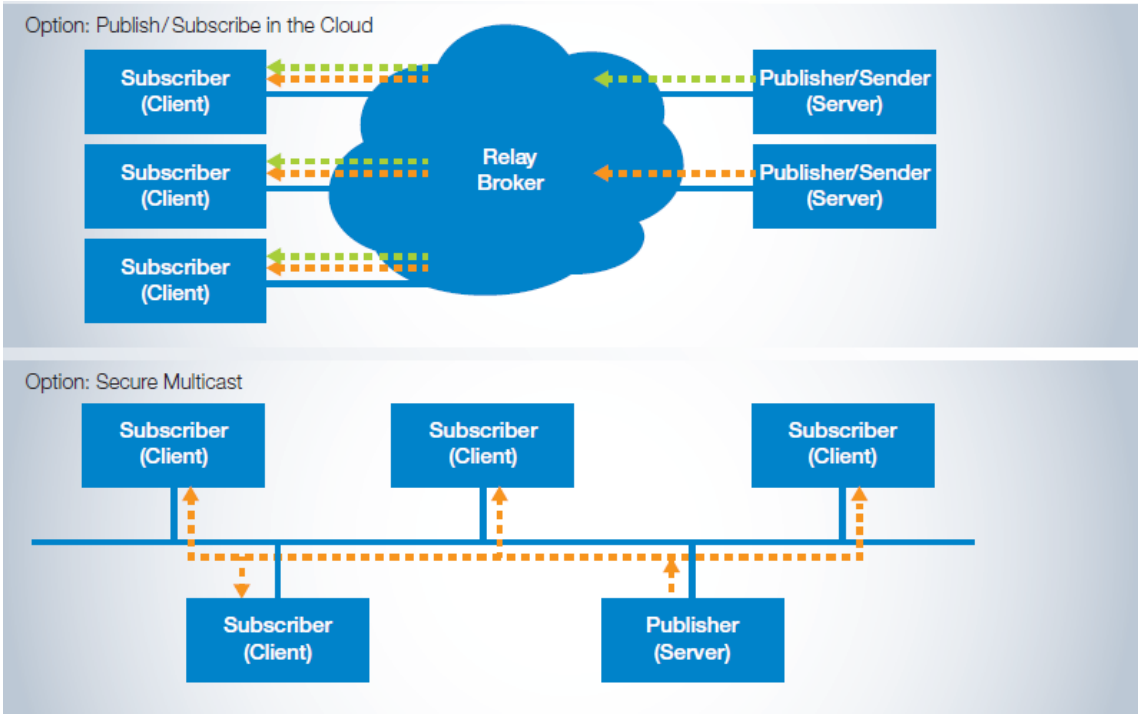
7.2 Skaalautuvuus

OPC UA:n sovellusympäristön laaja kirjo vaatii arkkitehtuurilta skaalautuvuutta mikrokontrolleritasolta sulautettuihin järjestelmiin sekä vielä laajemmalle MES- ja ERP-järjestelmiin. Saatavilla olevat OPC UA -kommunikaatiopinot kattavat melkein kaikki tämän päivän tunnetut alustat. Java-implementaatio vaatii Java JRE1.6 -ympäristön, jota voidaan käyttää Android-pohjaisissa laitteissa sekä Microsoft- tai Unix-pohjaisissa yritystason palvelimissa. .NET-implementaatio vaatii vähintään ohjelmistokomponenttikirjaston version 3.5, jonka voi asentaa Windows 7-,8-,8.1 - ja 10-versioihin (OPC Foundation, Interoperability for Industrie 4.0 and the Internet of Things)

OPC UA on palvelin/asiakas-pohjainen arkkitehtuuri, jossa yksi osa palvelimesta tarjoaa dataa ja toinen osa hankkii sitä. Kun toinen osa hankkii dataa ja suorittaa syklisiä toimenpiteitä, niin samaan aikaan toinen osa siirtää datan jollain aikavälillä eteenpäin. Tätä kutsutaan asynkroniseksi kommunikaatioksi. OPC UA -asiakkaan ja palvelimen välinen yhteys on aina kiinteä, niin kutsuttu point-to-point-yhteys, jonka päällä käytetään pyynti/vastausmekanismia. Asian eteen ollaan tekemässä kuitenkin parannuksia, sillä tätä

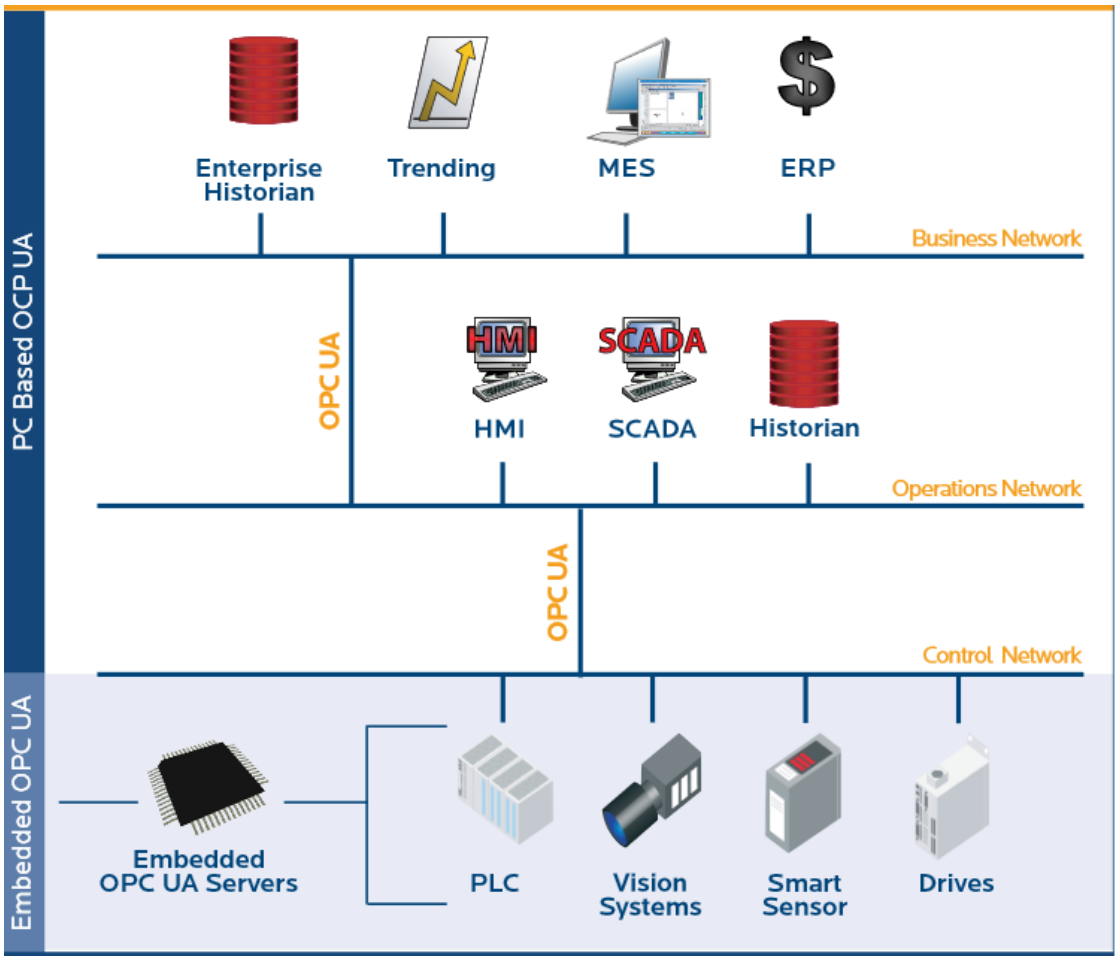
työtä kirjoittaessa OPC UA -työryhmä on laajentamassa standardin kommunikaatiota julkaisija/tilaaja *(publish/subscribe)* -pohjaiseen viestinvälitysarkkitehtuuriin perinteisen pyynti/vastaus mallin lisäksi. Julkaisija/tilaaja-mallissa julkaisija (palvelin) lähettää dataa yhteydettömästi verkkoon ja tilaajat (asiakas) käyttää dataa ilman resursseja kuluttavaa jatkuvaa yhteyttä (OPC Foundation, Interoperability for Industrie 4.0 and the Internet of Things)

Kuvassa 29 on havainnollistettu julkaisija/tilaaja viestinvälitys globaalissa ympäristössä palvelimelta välittäjälle *(broker)* ja edelleen asiakkaalle sekä paikallisen verkon viestinvälitys yhdeltä palvelimelta usealle asiakkaalle. Työryhmän mukaan mahdollisia tiedonsiirtoprotokollia julkaisija/tilaaja-mallille olisi ainakin UDP *(User Datagram Protocol)* ja AMQP *(Advanced Message Queuing Protocol)*. Lisäksi OPC UA -julkaisija/tilaaja viestinvälitysmekanismia ollaan yhdistämässä uuteen Ethernet-kommunikaatiolaajennukseen nimeltä TSN *(Time-Sensitive Networking)*, mikä mahdollistaisi reaaliaikaisen kommunikaation verkon laitteiden, kuten ohjelmoitavien logiikoiden ja antureiden välillä käyttämällä tavallista Ethernet-verkkoa. Uuden tiedonsiirtomekanismin lisääminen standardiin parantaisi OPC UA:n käytettävyyttä esimerkiksi M2M *(Machine To Machine)* ja IoT *(Internet of Things)* –sovellusympäristöissä (OPC UA is Enhanced for Publish-Subscribe 2015; Publish/Subscribe in a Mobile Environment, Huang, Gargia-Molina).



KUVA 29. Julkaisija/tilaaja-viestinvälitys. (Lähde: OPC Foundation)

Julkaisija/tilaaja-malli on perusmäärittelyltään asynkroninen ja anonyymi reititysmalli, jossa viestit lähetetään ryhmälähetyksenä *(multicast)* eli yhdeltä monelle. Anonyymius tarkoittaa sitä, että kommunikointiosapuolten ei tarvitse identifioida toisiaan. Tilaajan ei siis tarvitse tietää julkaisijaa saadakseen tapahtumia vaan se määrittelee tapahtumien ”tuntomerkit”, joiden perusteella tilaaja osaa vastaanottaa ne tapahtumat, joista se on kiinnostunut. Julkaisija/tilaaja-pohjainen viestinvälitys sopii hyvin dynaamiseen ympäristöön, jossa sekä julkaisijat sekä tilaajat voivat toistuvasti joko liittyä verkkoon tai kytkeytyä verkosta pois. Lisäksi julkaisija/tilaaja-viestinvälityspalvelu on käytännöllinen ympäristöihin, joissa etukäteen ei tiedetä kuka tarvitsee mitäkin tietoa. Malli soveltuu erityisen hyvin esimerkiksi mobiiliympäristöihin ja hajautettujen ohjelmistojen tiedonsiirtoon sekä globaalien verkkojen väliseen viestinvaihtoon. (OPC UA is Enhanced for PublishSubscribe,2015; Publish/Subscribe in a Mobile Environment, Huang, Gargia-Molina) Kuvassa 30 on yksi näkemys OPC UA:n skaalautuvuudesta aina lattiatason komponenteista yritystason järjestelmiin.



KUVA 30. OPC UA:n kommunikaatio eri tasoilla. (Lähde: MatrikonOPC)

7.3 Suorituskyky

OPC UA:n suunnittelussa tavoitteena oli sama tai parempi suorituskyky mitä OPC Classicin kanssa, mutta suorituskyky ei tarkoita pelkästään tiedonsiirtonopeutta. Se tarkoittaa myös pienempää kuormaa sekä resurssivaatimuksia käytettävältä järjestelmältä. Sulautetuissa järjestelmissä, joissa siirretään pieniä palasia dataa lyhyissä intervalleissa korostuvat minimaalinen resurssitarve, järjestelmän kuormitus sekä suorituskyky. Yritystason systeemeissä tärkeämpää voi olla esimerkiksi strukturoidun datan tehokas käsittely, jolloin absoluuttinen tiedonsiirtonopeus ei ole tärkein tekijä. OPC UA käyttää eri siirtotekniikoita kattaakseen mahdollisimman monet eri vaatimukset ja varmistaakseen standardin skaalautuvuuden. Automaatioympäristöjen UA-tuotteiden ja sulautettujen järjestelmien protokollaksi suositellaan optimoitua UA TCP -protokollaa binäärikoodauksella, kun taas yritystason järjestelmien tiedonsiirtomekanismi voi käyttää esimerkiksi Web Service -binääriä tai XML-koodausta.
Suorituskykyyn vaikuttaa moni muukin asia kuin käytettävä siirto- ja koodausmenetelmä. Näitä ovat esimerkiksi sovelluskerros, sovelluksen integrointi UA-pinon kanssa, kommunikaation määritelty tietoturvataso sekä käytettävä laitteisto. Suorituskyvystä on mahdotonta sanoa jotakin yleispätevää lukemaa yllämainituista syistä. OPC UA:n suorituskyvystä on tehty erilaisia tutkielmia, joista seuraavassa kaksi esimerkkiä:

Saksalaisen Ostwestfalen-Lippe Yliopiston Institute of Industrial IT -osaston tekemässä tutkielmassa testattiin OPC UA, CoAP ja MQTT -protokollien tiedonsiirtokäyttäytymistä emuloiduissa EDGE-, UMTS- ja LTE-verkoissa. Tutkielmassa vertailtiin näiden kolmen protokollan tiedonsiirtoaikoja laboratorioympäristössä sekä niiden soveltuvuutta M2Mtiedonsiirtoon. Testissä mitattiin tiedonsiirtoaikaa suhteessa viestin hyötykuorman kokoon (0-10kB) joko asiakkaalta palvelimelle ja takaisin, tai välittäjältä julkaisijalle ja takaisin, riippuen testattavasta protokollasta. Testissä selvisi, että OPC UA käytti vähiten aikaa tiedonsiirtoon kaikissa tapauksissa. Tuloksen selittää OPC UA:n tiedonsiirtomekanismi, jossa TCP-yhteys luodaan vain kerran asiakkaan ja palvelimen välille eikä erillisiä kuittausviestejä lähetetä kuin kerran. Tulos oli mielenkiintoinen, sillä aikaisemmin oli todettu, että vertailun protokollista OPC UA -viesti on yleiskustannukseltaan korkein aiheuttaen merkittävän lisäkuorman verkkoon varsinaisen hyötykuorman lisäksi. Mainittakoon, että tutkielmassa ei kerrottu mitä tietoturvaprofiilia OPC UA:n tapauksessa käytettiin. Taulukossa 3 on esitetty EDGE-verkkotestin mittaustuloksia (Durkop, Czybik, Jasperneite).

TAULUKKO 3. Suorituskykymittaukset OPC UA, MQTT ja CoAP-protokollilla (Durkop ym, muokattu)

Viestin hyöty-kuorma: 2500 tavua (EDGE-verkossa)	OPC UA (datalähde=palve-lin)	MQTT (datalähde=julkaisija)	CoAP (datalähde=palvelin)
Lähetettyjen tavujen kokonaismäärä (si-sältää IP-kerroksen ja MAC-osoitteet)	2959	3195	2839
Lähetettyjen viesti-kehyksien luku-määrä	4	11	6
Tiedonsiirtoon kulu-tettu kokonaisaika (ms)	840	1312	1775

Suomalainen ohjelmistoyritys Prosys Oy on tutkinut OPC UA:n suorituskykyä ja vertail-lut eri alustojen, tietoturvan sekä ohjelmistokirjastojen vaikutusta suorituskykyyn. Testissä tutkittiin muun muassa viestin salaukseen kulutettua absoluuttista (taulukko 4) aikaa sekä vertailtiin salauksen vaikutusta suoritusaikoihin (taulukko 5). Testissä käytet-tiin Prosys Oy:n OPC UA Java-ohjelmistokirjastoa.

TAULUKKO 4. Yhden salatun metodikutsun suoritusaika. (Prosys 2015, muokattu)

1 metodi-kutsun absoluutti-nen suoritusaika	Windows OS 64bit, Intel Core i7 2.7GHz (8 ydintä) 8GB RAM, SSD	Rasperry Pi Rasbian Linux 700 MHz (1 ydin ARM), 512MB RAM
salauksen vaikutus	~ 0.2 [ms]	~ 4 [ms]

TAULUKKO 5. Salauksen vaikutus luku-kutsujen suoritusaikaan. (Prosys 2015, muokattu)

salauksen vaikutus luku-kutsuihin sovellusten välillä Tietoturvaprofiili:128Rsa15	Windows OS 64bit, Intel Core i7 2.7GHz (8 ydintä) 8GB RAM, SSD	Rasperry Pi Rasbian Linux 700 MHz (1 ydin ARM), 512MB RAM
Tietoturva: ei käytössä (10 elementin tavujono)	0.5 [ms]	20 [ms]
Tietoturva: allekirjoitus (10 elementin tavujono)	0.6 [ms]	23 [ms]
Tietoturva: allekirjoitus ja salaus(10 elementin tavujono)	1.0 [ms]	32 [ms]

Testitulosten perusteella Prosys toteaa, että sulautetussa järjestelmässä pelkkä tiedon allekirjoitus ei vaikuta merkittävästi suorituskykyyn. Lisäksi on todettu, että sulautettu järjestelmä on noin 40 kertaa hitaampi verrattuna tehokkaampaan tietokoneeseen, jossa lukukutsujen määrä on 1000–2000/s riippuen käytettävästä tietoturvasta (OPC UA Performance Evaluation. Aro 2015).

TAULUKKO 6. Viestin koon vaikutus suoritusaikaan. (Prosys 2015, muokattu)

viestin koon vaikutus luku-kutsuihin sovellusten välillä Tietoturvaprofiili:128Rsa15	Windows OS 64bit, Intel Core i7 2.7GHz (8 ydintä) 8GB RAM, SSD	Rasperry Pi Rasbian Linux 700 MHz (1 ydin ARM), 512MB RAM
Tietoturva: ei käytössä (10k elementin tavujono)	0.5 [ms]	20 [ms]
Tietoturva: allekirjoitus (10k elementin tavujono)	~0.8[ms]	28 [ms]
Tietoturva: allekirjoitus ja salaus (10k elementin tavujono)	~1.4[ms]	78 [ms]

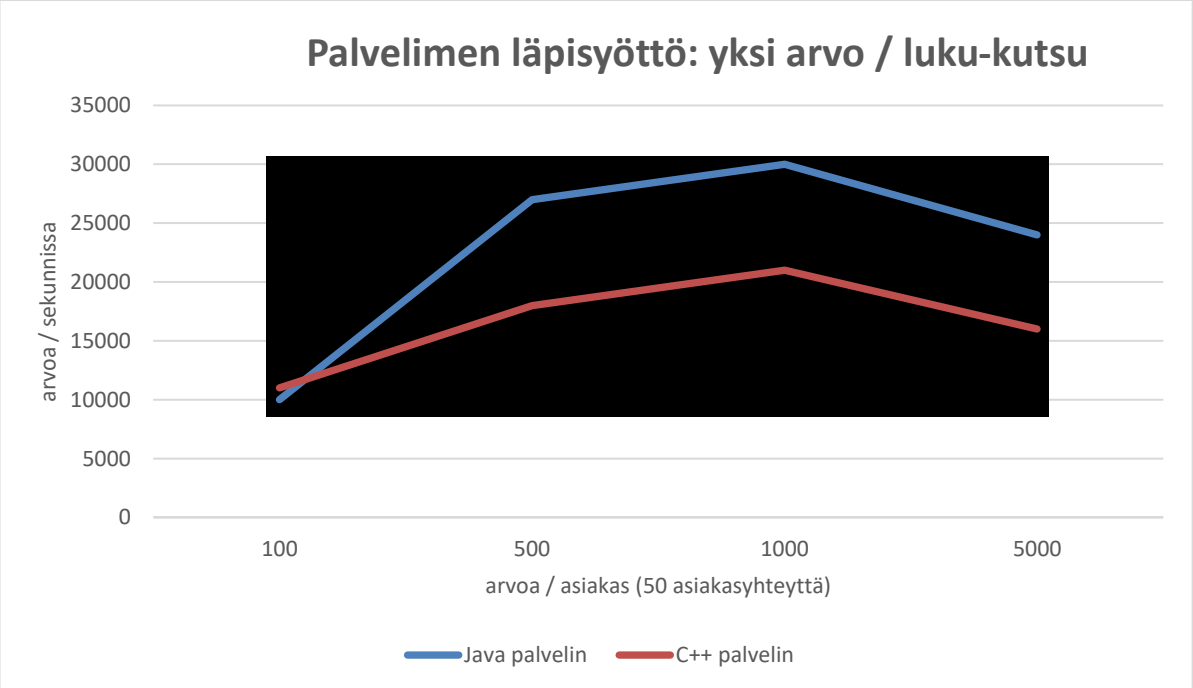
Tulosten perusteella huomataan, että mitä suurempi viesti niin sitä kauemmin salaukseen kuluu aikaa. Sulautetussa järjestelmässä viestin allekirjoitus ja salaus kestää kaksi kertaa pidempään verrattuna pienempään viestiin (taulukko 6).

Toisessa Prosys Oy:n suorittamassa testissä tutkittiin kahden eri ohjelmointikielen asynkronisten etälukukutsujen suorituskykyä. Testissä yhdeltä asiakassovellukselta luotiin viisikymmentä yhteyttä palvelimelle ja mitattiin sen läpisyöttökykyä asiakkaalle. UA:ssa on mahdollista lukea dataa joko niin, että yhdellä kutsulla luetaan yksi arvo kerrallaan (kuvio 1) tai niin, että yhdellä kutsulla luetaan useampi arvo kerrallaan (kuvio 2). Testi suoritettiin gigahertsin teoreettisen tiedonsiirtonopeuden mahdollistavassa tiedonsiirtoverkossa käyttäen taulukossa 7 lueteltuja alustoja ja ohjelmistokirjastoja. (OPC UA Performance Evaluation. Aro 2015).

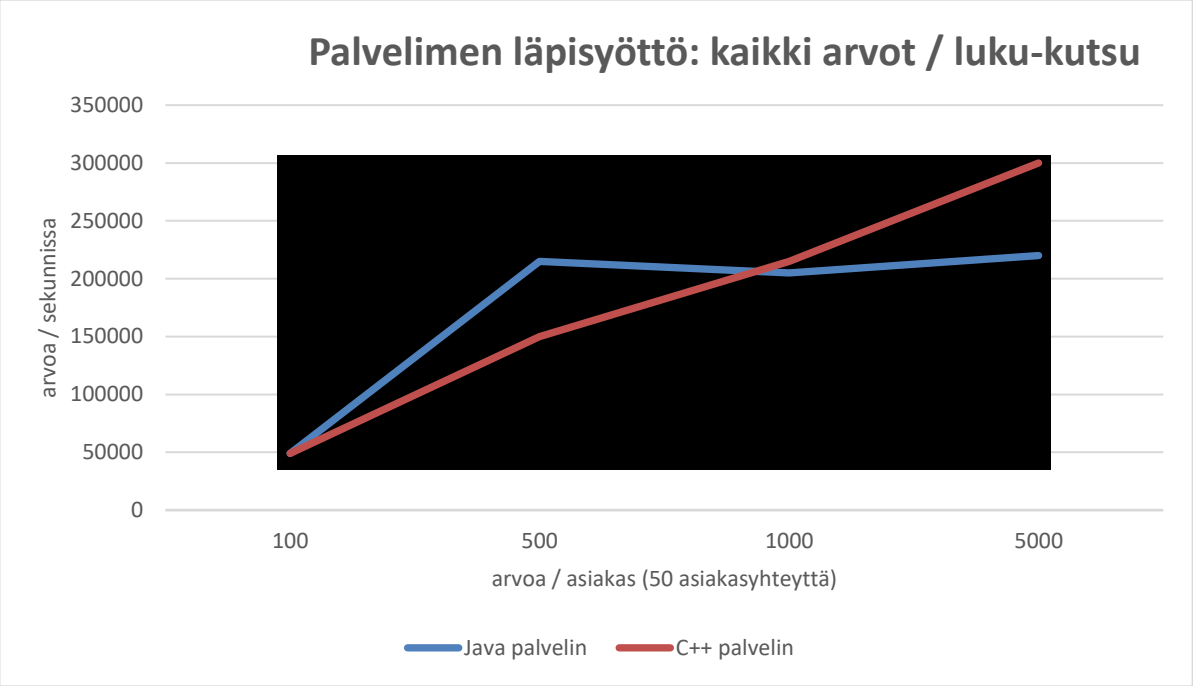
TAULUKKO 7. Testialustat ja ohjelmistokirjastot. (Prosys 2015, muokattu)

Asiakas-alusta	Palvelin-alusta	Palvelin-ohjelmistokirjastot
Windows OS 64bit, Intel Core i7 2.7GHz (8 ydintä) 8GB RAM, SSD	Intel Core 2 Quad Q9300 2.50GHz (4 ydintä), 4GB RAM, SATA	Prosys Java SDK Unified Automation C++ SDK

KUVIO 1. Yhden arvon lukeminen yhdellä luku-kutsulla. (Prosys 2015, muokattu)



KUVIO 2. Kaikkien arvojen prosessointi käyttäen yhtä luku-kutsua. (Prosys 2015, muokattu)

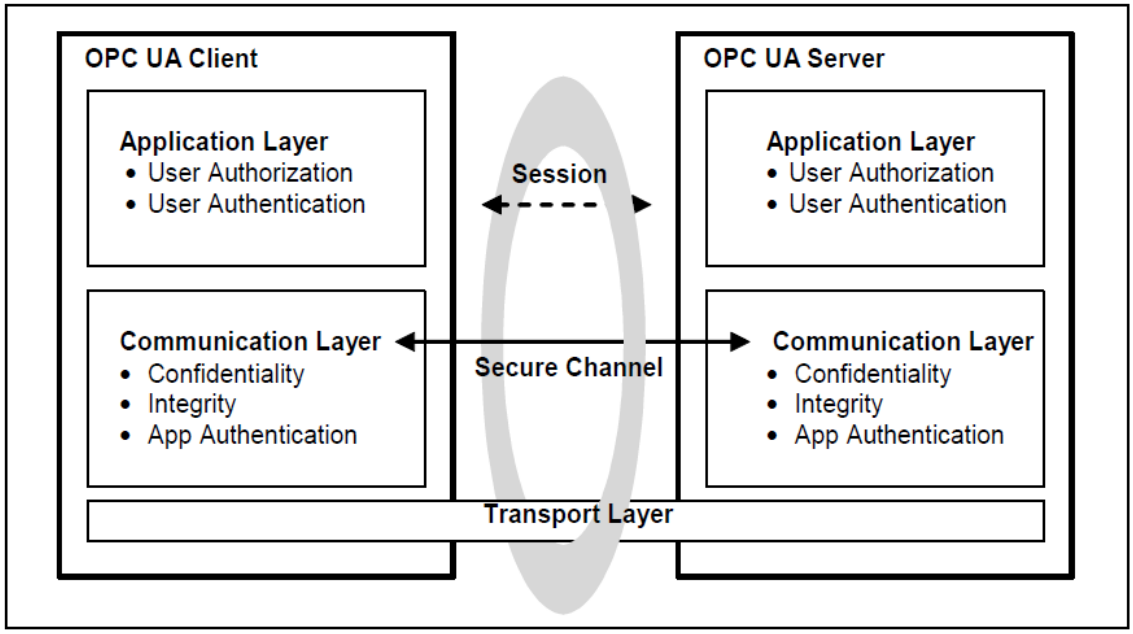


Testitulosten perusteella Prosys Oy toteaa, että OPC UA on nopea, mutta toisaalta tiedon salaaminen aiheuttaa huomattavia viiveitä ainakin sulautetussa järjestelmässä. Palvelin pystyy lähettämään jopa 300 000 arvoa sekunnissa, kun kaikki arvot luetaan yhdellä kutsulla. Lisäksi C++-ohjelmistokirjastolla toteutettu palvelin on nopeampi kuin Java-toteutus, mutta kaiken kaikkiaan ne tarjoavat yhtäläisen suorituskyvyn. Tulokset on julkaistu OPC Day Finland-tapahtumassa (OPC UA Performance Evaluation. Aro 2015).

8 TIETOTURVA

Koska UA on suunniteltu käytettäväksi erilaisissa ympäristöissä ja usealla eri tietojärjestelmätasolla, joissa tietoturvavaatimukset voivat vaihdella sovelluskohtaisesti, on sen tietoturva-arkkitehtuurin oltava yleiskäyttöinen ja joustava. Tässä kappaleessa tutustutaan tietoturvamalliin yleisellä tasolla.

OPC UA:n tietoturva-arkkitehtuuri on monikerroksinen konsepti, joka perustuu käyttäjän tunnistukseen, käyttöoikeuksiin, sertifikaatteihin sekä sovellusten todentamiseen ja turvattuun tiedonsiirtokanavaan. Kuvassa 31 on havainnollistettu tietoturva-arkkitehtuuri, joka jakautuu sovellus- ja kommunikaatiokerroksiin. Jokaisella kerroksella on vastuunsa tietoturvan toteutuksen suhteen. Sovelluskerros vastaa asiakasohjelmien käyttäjien todentamisesta sekä valtuuksista, kun taas kommunikointikerros huolehtii viestien suojaamisesta ja sovellusten tunnistamisesta (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).



KUVA 31. OPC UA-tietoturvamalli. (Lähde: OPC Foundation)

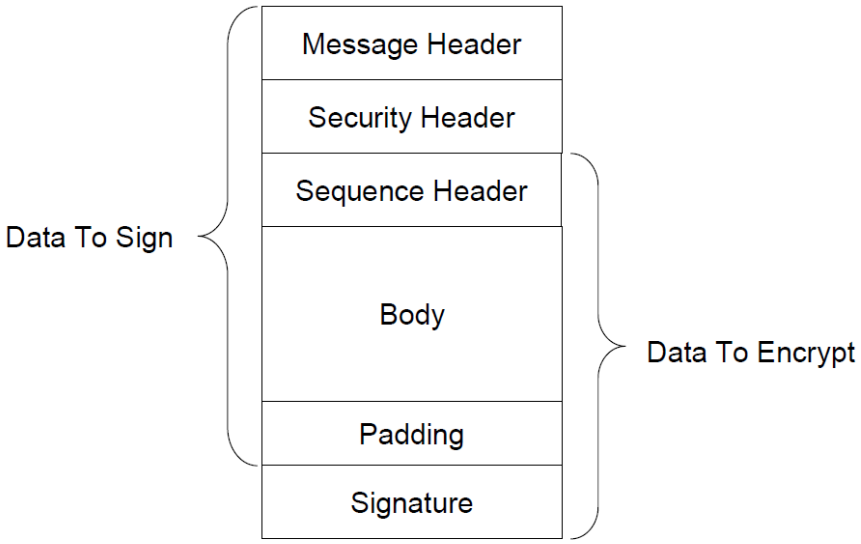
Yleiskäyttöisyys toteutetaan eri tietoturvaprofiileilla, jotka määrittelevät sovellusten välisen kommunikaation tietoturvatason. Tietoturvaprofiilit määrittelevät viestienvaihdossa algoritmit ja avaimet, joita käytetään viestien allekirjoittamiseen ja salaukseen.

UA-palvelimille voidaan valita sovellusalueen ja käyttötarkoituksen perusteella sopivin profiili. Internetin yli kommunikoivien sovellusten viestinvaihdossa voidaan käyttää esimerkiksi raskainta tietoturvaprofiilia, jossa viestit allekirjoitetaan ja salataan. Suljetun verkon sisällä kommunikoivien laitteiden viestinvaihto voidaan puolestaan toteuttaa vain allekirjoittamalla viestit, tai alkuperäisessä muodossa ilman salausta, jos laitteiden resurssit eivät riitä salattujen viestien prosessointiin (Mahnke ym, 2009).

Sovellukset voivat kommunikoida sekä binääri- että XML-muodossa. XML-tiedonsiirto perustuu web-sovelluspalvelujen käyttöön ja binääritiedonsiirto OPC Foundation:n määrittelemään OPC UA TCP -protokollaan. Kun tiedonsiirtoon käytetään binääristä TCPkerroksen päällä toimivaa menetelmää, toteutetaan tietoturva OPC UA Secure Conversation-protokollalla. UASC on binääri-versio Web Services Secure Conversation-spesifikaatiosta, mitä käytetään web-sovelluksien välisen tietoliikenteen turvaamiseen. UASC
on tarkoitettu tietoliikenteen suojaamiseen, kun viestinvälitykseen käytetään natiivia UAbinääri kuljetusprofiilia (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

OPC UA SC on suunniteltu toimimaan eri tiedonsiirtoprotokollien ja laitteiden kanssa, joilla on rajoitettu suorituskyky. UASC pätkii lähetettävät viestit lohkoihin *(Message-Chunks)*, jotka ovat puskurikooltaan pienempiä kuin käytettävän protokollan määrittelemä puskurikoko. UASC:n käyttövaatimuksena on kuitenkin vähintään 8196-kilotavun puskurikoko. Tietoturva on toteutettu erillisiin lohkoihin yksilöllisesti eikä kokonaiseen viestiin. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

UASC-viesti, joka on esitetty kuvassa 32, koostuu kolmesta otsikosta *(Header)*, runkoosasta *(Body),* alatunnisteesta *(padding)* ja allekirjoituksesta *(Signature).*



KUVA 32. OPC UA Secure Conversation-viestin rakenne. (Lähde: OPC Foundation).

Viestiotsikko identifioi viestityypin *(Message type)*, joka voi olla esimerkiksi pyyntö salatun tiedonsiirtokanavan avaamiseen, sulkemiseen tai sovellusten välisen istunnon luomiseen. *IsFinal*-kenttä kertoo onko lähetetty viestilohko viimeinen osa viestiä. *MessageSize* kuvaa viestilohkon koon ja *SecureChannelID* kanavan tunnisteen. Taulukossa 8 on esitelty viestiotsikon rakenne (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

TAULUKKO 8. UASC-viestin viestiotsikon rakenne. (Lähde: OPC Foundation)

Name	Data Type	Description
MessageType	Byte[3]	A three byte ASCII code that identifies the *Message* type. The following values are defined at this time: MSG A *Message* secured with the keys associated with a channel. OPN OpenSecureChannel *Message*. CLO CloseSecureChannel *Message*.
IsFinal	Byte	A one byte ASCII code that indicates whether the *MessageChunk* is the final chunk in a *Message*. The following values are defined at this time: C An intermediate chunk. F The final chunk. A The final chunk (used when an error occurred and the *Message* is aborted).
MessageSize	UInt32	The length of the *MessageChunk*, in bytes. This value includes size of the *Message* header.
SecureChannelId	UInt32	A unique identifier for the *SecureChannel* assigned by the *Server*. If a *Server* receives a SecureChannelId which it does not recognize it shall return an appropriate transport layer error. When a *Server* starts the first *SecureChannelId* used should be a value that is likely to be unique after each restart. This ensures that a *Server* restart does not cause previously connected *Clients* to accidently 'reuse' *SecureChannels* that did not belong to them.

Viestiotsikkoa seuraa joko symmetrinen tai asymmetrinen tietoturvaotsikko *(Security Header)*. Asymmetrinen otsikko sisältää tiedot viestiin sovelletuista tietoturvamenettelyistä, kuten salaus-algoritmit sekä lähettäjän sertifikaatin, mitä käyttämällä vastaanottaja

voi purkaa ja varmentaa viestin. Asymmetrisiä salausalgoritmeja käytetään ainoastaan salaamaan *OpenSecureChannel*-palvelupyyntöihin liittyvät viestit, koska se on ainut palvelu, jossa viestit turvataan julkisilla ja yksityisillä avaimilla. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

Asymmetrisen viestiotsikon rakenne on kuvattu taulukossa yhdeksän ja se sisältää seuraavat kentät.

SecurityPolicyUriLength-kenttä kuvaa tietoturvapolitiikan URI:in tavuina. *SenderCertificate*-kenttä kuvaa X509-sertifikaatin, joka on myönnetty viestin lähettävälle sovellukselle. Viesti on koodattu DER -*(Distinguished Encoding Rules)* menetelmällä ja se indikoi mitä yksityistä avainta viestilohkon allekirjoitukseen on käytetty. *ReceiverCertificateThumbprint*-kenttä kuvaa vastaanottajan SHA-1 -algoritmin mukaisen sormenjäljen, joka on muodostettu X509-sertifikaatista sekä viestin salaamisessa käytetyn julkisen avaimen (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03)

TAULUKKO 9. Asymmetrisen algoritmin tietoturvaotsikon rakenne. (Lähde: OPC Foundation)

Name	Data Type	Description
SecurityPolicyUriLength	Int32	The length of the *SecurityPolicyUri* in bytes. This value shall not exceed 255 bytes. If a URI is not specified this value may be 0 or -1.
SecurityPolicyUri	Byte[*]	The URI of the *Security Policy* used to secure the *Message*. This field is encoded as a UTF8 string without a null terminator.
SenderCertificateLength	Int32	The length of the *SenderCertificate* in bytes. This value shall not exceed *MaxCertificateSize* bytes. If a certificate is not specified this value may be 0 or -1.
SenderCertificate	Byte[*]	The X509v3 *Certificate* assigned to the sending *Application Instance*. This is a DER encoded blob. The structure of an X509 *Certificate* is defined in X509. The DER format for a *Certificate* is defined in X690 This indicates what *Private Key* was used to sign the *MessageChunk*. The *Stack* shall close the channel and report an error to the *Application* if the *SenderCertificate* is too large for the buffer size supported by the transport layer. This field shall be null if the *Message* is not signed. If the *Certificate* is signed by a CA the DER encoded CA *Certificate* may be appended after the Certificate in the byte array. If the CA *Certificate* is also signed by another CA this process is repeated until the entire Certificate chain is in the buffer or if *MaxSenderCertificateSize* limit is reached (the process stops after the last whole *Certificate* that can be added without exceeding the *MaxSenderCertificateSize* limit). Receivers can extract the *Certificates* from the byte array by using the *Certificate* size contained in DER header (see X509). Receivers that do not handle *Certificate* chains shall ignore the extra bytes.
ReceiverCertificateThumbprintLength	Int32	The length of the *ReceiverCertificateThumbprint* in bytes. If a thumbprint is specified, the length of this field is 20 bytes. If a thumbprint is not specified this value may be 0 or -1.
ReceiverCertificateThumbprint	Byte[*]	The thumbprint of the X509v3 *Certificate* assigned to the receiving *Application Instance*. The thumbprint is the SHA1 digest of the DER encoded form of the *Certificate*. This indicates what public key was used to encrypt the *MessageChunk*. This field shall be null if the *Message* is not encrypted.

Symmetrisen salauksen tietoturvaotsikko sisältää ainoastaan tunnisteen käytetylle tietoturvavaltuudella, joka identifioi viestien salauksessa ja allekirjoituksessa käytetyt avaimet. Symmetristä salausta käytetään kaiken muun liikenteen salaamiseen lukuun ottamatta *OpenSecureChannel*-palveluita.

TokenId-kenttä, taulukossa 10, kuvaa yksilöllisen tunnisteen tietoturvatulle kanavalle ja sen valtuudelle. (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

TAULUKKO 10. Symmetrisen algoritmin tietoturvaotsikon rakenne. (Lähde: OPC Foundation)

Name	Data Type	Description
TokenId	UInt32	A unique identifier for the *SecureChannel SecurityToken* used to secure the *Message*. This identifier is returned by the *Server* in an *OpenSecureChannel* response *Message*. If a *Server* receives a TokenId which it does not recognize it shall return an appropriate transport layer error.

Sekvenssiotsikko sisältää yksilöllisen numerotunnuksen, jolla lähetetyn viestin lohkot tunnistetaan. Otsikkoa käytetään silloin, jos viestin sisältämä hyötykuorma ei mahdu yksittäiseen lohkoon ja se joudutaan pilkkomaan useampaan osaan. Sekvenssiotsikko pitää huolen myös siitä, että jokaisen salattavan viestin ensimmäinen salattava viestilohko alkaa erilaisella tiedolla (Mahnke ym. 2009; OPC Foundation, Spesifikaatio osa 6: Mappings, Versio 1.03).

TAULUKKO 11. UASC-viestin sekvenssiotsikon rakenne. (Lähde: OPC Foundation)

Name	Data Type	Description
SequenceNumber	UInt32	A monotonically increasing sequence number assigned by the sender to each *MessageChunk* sent over the *SecureChannel*.
RequestId	UInt32	An identifier assigned by the *Client* to OPC UA request *Message*. All *MessageChunks* for the request and the associated response use the same identifier.

Viestin alatunniste sisältää allekirjoituksen, jolla varmistetaan, ettei allekirjoitetun viestin sisältö ole muuttunut lähetyksen jälkeen. Lisäksi allekirjoituksella varmennetaan, että lähetetty viesti on peräisin siltä lähettäjältä, joka on mainittu tietoturvaotsikon sisältämässä sertifikaatissa. Allekirjoitus varmennetaan lähettäjän sertifikaatin julkisella avaimella. Alatunnisteen rakenne on kuvattu taulukossa 12.

TAULUKKO 12. UASC-viestin alatunnisteen rakenne. (Lähde: OPC Foundation)

Name	Data Type	Description
PaddingSize	Byte	The number of padding bytes (not including the byte for the PaddingSize).
Padding	Byte[*]	Padding added to the end of the *Message* to ensure length of the data to encrypt is an integer multiple of the encryption block size. The value of each byte of the padding is equal to PaddingSize.
ExtraPaddingSize	Byte	The most significant byte of a two byte integer used to specify the padding size when the key used to encrypt the message chunk is larger than 2048 bits. This field is omitted if the key length is less than or equal to 2048 bits.
Signature	Byte[*]	The signature for the *MessageChunk*. The signature includes the all headers, all *Message* data, the PaddingSize and the Padding.

Kuljetusprofiilit määrittelevät kommunikointiprotokollat UA-sovelluksille. Natiivia pinokuvausta käytettäessä tietoturva toteutetaan OPC Foundationin® määrittelemällä UASC-menetelmällä, joka siis perustuu yksittäisten viestilohkojen salaamiseen, kuten yllä kuvattiin. Standardin määrittelemät kaksi muuta kuljetusprofiilia toteuttavat tietoturvan suojaamalla tiedonsiirtokanavan käyttämällä HTTPS/TLS-tekniikoita. Yhteenvetona voidaan todeta, että tietoturva sovellusten välisessä kommunikoinnissa voidaan toteuttaa joko viestitasolla tai yhteystasolla riippuen sovellusympäristöstä.

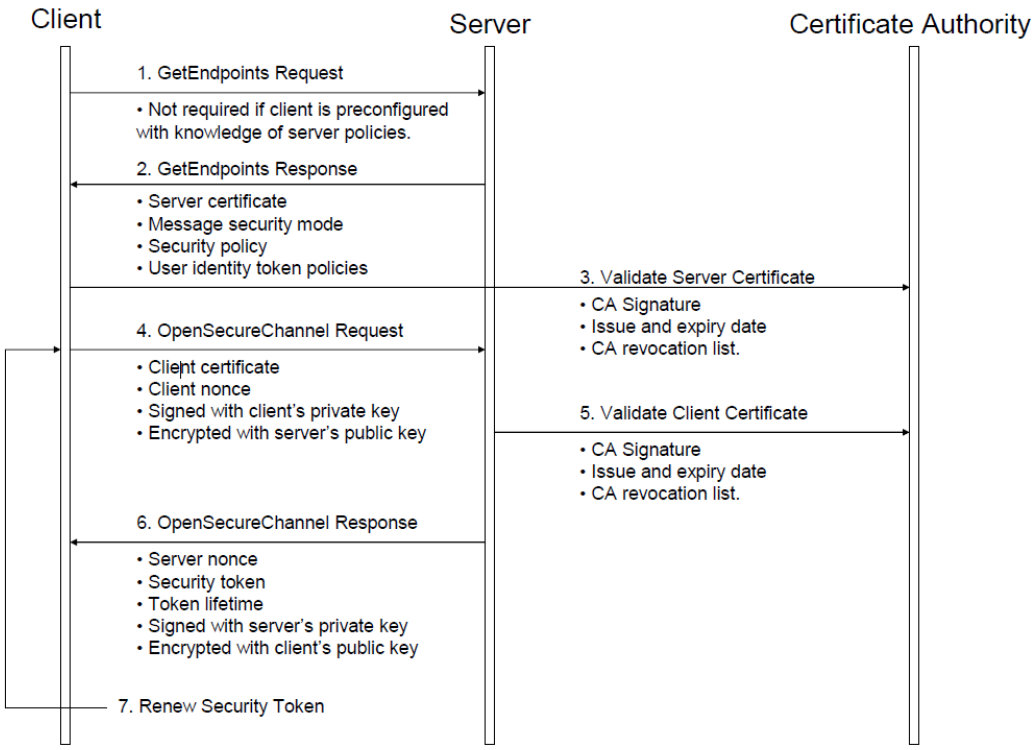
8.1 Tietoturvakerros ja tietoturvattu kommunikointikanava

Tietoturvakerroksen mekanismit määritellään pinokuvauksien sekä profiilien avulla. Tietoturvakerros toteuttaa tietoturvatun kanavan *(Secure Channel),* joka on pitkäkestoinen, looginen yhteys UA-asiakassovelluksen ja palvelimen välillä. Kanava varmistaa viestien tietosuojan sekä eheyden. Kanava on luotava sovellusten välille, jotta varsinainen viestien vaihto voisi alkaa. Tällä hetkellä standardi määrittelee neljä eri tietoturvaprofiilia: None, Basic128Rsa15, Basic256 ja Basic256Sha256. Profiilien lisäksi tietoturvatason määrittämiseen käytetään tietoturvamoodeja. Standardi määrittelee kolme eri laista moodia, jotka ovat *None, Sign* sekä *Sign ja Encrypt.* None-moodissa viestejä ei salata laisinkaan, Sign-moodissa viestit allekirjoitetaan ja *Sign&Encrypt*-moodisssa viestit myös salataan allekirjoituksen lisäksi (Mahnke ym. 2009; OPC Unified Architecture Spesifikaatio Osa 7: Profiilit. Versio 1.03).

Asiakkaan ja palvelimen välisen turvatun tiedonsiirtokanavan avaaminen voidaan jakaa kahteen palvelujoukkoon, jotka ovat *Discovery* sekä *Secure Channel* -palvelut.

Asiakas kutsuu *Discovery*-palvelujoukon *Get EndPoints- palvelua* ja lukee vaadittavat tietoturva-asetukset, jotka tarvitaan sekä turvatun tiedonsiirtokanavan että istunnon luomiseen. Palvelinsovelluksen yhteyspisteellä on neljä eri tietoturva asetusta: palvelimen sovellussertifikaatti *(Server Application Instance Certificate)*, viestin tietoturvamoodi *(Message Security Mode)*, tietoturvapolitiikka *(Security Policy)* sekä käyttäjän tunnistetiedot *(Supported User Identity Tokens)* (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

Palvelimen sovellussertifikaattia käytetään suojaamaan turvatun tiedonsiirtokanavan avaamisen yhteydessä lähetettävät viestit. Viestin tietoturvamoodi ja käytäntö kertoo asiakassovellukselle kuinka sen tulisi suojata/salata viestit, jotka kuljetetaan tiedonsiirtokanavaa pitkin. Turvatun tiedonsiirtokanavan luomisen edellytyksenä on molempien, sekä asiakas että palvelinsovellusten, pääsy sertifikaatteihin, joita käytetään viestien salaukseen ja allekirjoitukseen. Kuvan 33 esimerkissä suojatun kanavan luomisessa oletamuksena on, että asiakas ja palvelinsovelluksilla on yhteys varmenneviranomaiseen *(Certificate Authority),* joka ylläpitää sertifikaatteja. Varmenneviranomaisella tarkoitetaan tahoa, esimerkiksi ylläpitäjää tai organisaatiota, joka tarkistaa sertifikaatteihin sisälletyn tiedon oikeellisuuden sekä lisää siihen varmentajan digitaalisen allekirjoituksen. Allekirjoituksen perusteella toinen osapuoli voi varmentaa sertifikaatin luottamuksellisuuden. (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).



KUVA 33. Turvatun tiedonsiirtokanavan luominen. (Lähde: OPC Foundation)

Kun asiakas on saanut tarvittavat tietoturva-asetukset yhteyden luomiseen, lähettää se seuraavaksi palvelimelle julkisen avaimensa sovellussertifikaatissa, sekä salaisen satunnaismerkkijonon yhdessä *OpenSecureChannel*-palvelupyynnön kanssa. Tämä palvelupyyntö on turvattu asymmetrisellä salauksella, jossa on käytetty palvelimen julkista avainta sekä asymmetrisellä allekirjoituksella, mikä on puolestaan allekirjoitettu asiakkaan yksityisellä avaimella. Sovellussertifikaatti itsessään lähetetään kuitenkin salaamattomana, jotta vastaanottaja voi käyttää sitä varmentamaan asymmetrisen allekirjoituksen (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

Tämän jälkeen palvelin purkaa asiakkaan lähettämän viestin käyttämällä omaa yksityistä avainta sekä varmentaa asymmetrisen allekirjoituksen asiakkaan julkisella avaimella, jonka jälkeen se luo tietoturvatun kanavan asiakkaan ja itsensä välille. Kun palvelin avaa kanavan, se määrittelee kanavalle tunnisteen, jota vain kanavan avaamisesta sopineet sovellukset voivat käyttää. *OpenSecureChannel*-palvelupyynnön parametrit vaikuttavat kanavan tietoturva-asetuksiin. Seuraavaksi käydään lyhyesti läpi pyynnön ja vastausviestien rakenteet, jotka on listattu taulukoissa 13 ja 14 (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

TAULUKKO 13. Asiakkaan lähettämän *OpenSecureChannel*-pyynnön rakenne. (Lähde: OPC Foundation)

Name	Type	Description
Request		
requestHeader	RequestHeader	Common request parameters. The *authenticationToken* is always omitted. The type *RequestHeader* is defined in 7.28.
clientCertificate	ApplicationInstance Certificate	A *Certificate* that identifies the *Client*. The *OpenSecureChannel* request shall be signed with this *Certificate*. The *ApplicationInstanceCertificate* type is defined in 7.2. If the *securityPolicyUri* is None, the *Server* shall ignore the *ApplicationInstanceCertificate*.
requestType	Enum SecurityToken RequestType	The type of *SecurityToken* request: An enumeration that shall be one of the following: ISSUE_0 creates a new *SecurityToken* for a new *SecureChannel*. RENEW_1 creates a new *SecurityToken* for an existing *SecureChannel*.
secureChannelId	ByteString	The identifier for the *SecureChannel* that the new token should belong to. This parameter shall be null when creating a new *SecureChannel*.
securityMode	Enum MessageSecurityMode	The type of security to apply to the messages. The type *MessageSecurityMode* type is defined in 7.15. A *SecureChannel* may have to be created even if the *securityMode* is NONE. The exact behaviour depends on the mapping used and is described in the Part 6.
securityPolicyUri	String	The URI for *SecurityPolicy* to use when securing messages sent over the *SecureChannel*. The set of known URIs and the *SecurityPolicies* associated with them are defined in Part 7.
clientNonce	ByteString	A random number that shall not be used in any other request. A new *clientNonce* shall be generated for each time a *SecureChannel* is renewed. This parameter shall have a length equal to key size used for the symmetric encryption algorithm that is identified by the *securityPolicyUri*.
requestedLifetime	Duration	The requested lifetime, in milliseconds, for the new *SecurityToken*. It specifies when the *Client* expects to renew the *SecureChannel* by calling the *OpenSecureChannel Service* again. If a *SecureChannel* is not renewed, then all *Messages* sent using the current *SecurityTokens* shall be rejected by the receiver.

RequestHeader-kentässä kuvataan yleisiä parametreja, kuten lähetetyn viestin aikaleima *(Timestamp)* sekä tunnisteen istunnon tunnistamista varten *(SessionAuthenticationToken). ClientCertificate* on asiakassovelluksen identifioiva sertifikaatti, jota palvelin käyttää varmennukseen. Asiakkaan lähettämä pyyntö allekirjoitetaan tällä sertifikaatilla. *SecurityToken RequestType* -kentässä määritellään onko kyseessä uuden kanavan luonti vai olemassa olevan kanavan turvallisuusvaltuuksien uudistaminen. *SecureChannelID*kenttä kuvaa parametrin, jolla kanava yksilöidään. *SecurityMode*-kenttä määrittää kommunikaation salausmoodin. Eri moodit on selitetty kappaleessa 5.1. *SecurityPolicyUri*kenttä kuvaa käytettävän tietoturvapolitiikan, jota käytetään viestien turvaamisessa. *ClientNonce* on pyyntöön liittyvä satunnaismerkkijono, jota käytetään ainoastaan *OpenSecureChannel*-palvelupyynnön yhteydessä. Merkkijonon avulla lasketaan symmetrisessä salauksessa käytettävät salausavaimet (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).
Palvelimen lähettämän *OpenSecureChannel*-vastausviestin rakenne alkaa otsikolla *ResponseHeader*, joka sisältää muun muassa aikaleiman sekä tilakoodin, joka on selitetty sivulla 25. *SecurityToken*-kenttä kuvaa palvelimen määrittämän tietoturvavaltuuden. *ChannelID*-on vastaava kanavan tunniste, mikä löytyy myös pyynnön parametreista. *TokenID*-kuvaa yksittäisen tietoturvavaltuuden tunnisteen, joka välitetään aina kun viestit on suojattu symmetrisellä algoritmilla. Jos tunnistetta ei tunnisteta ollenkaan, niin palvelin lähettää virheilmoituksen. *CreatedAt* on tietoturvavaltuuden luonnin aikaleima ja *RevisedLifetime* kuvaa tietoturvavaltuuden elinajan. *ServerNonce* on palvelimen lähettämä satunnaismerkkijono, jonka avulla sekä palvelin että asiakas voi johtaa yhteiset symmetriseen salaukseen- ja allekirjoituksiin liittyvät avaimet. Tämä prosessi on selitetty tarkemmin standardin osassa 6, sivulla 39 (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

TAULUKKO 14. Palvelimen lähettämän *OpenSecureChannel*-vastausviestin rakenne. (Lähde: OPC Foundation)

Response		
responseHeader	ResponseHeader	Common response parameters (see 7.29 for *ResponseHeader* type definition).
securityToken	ChannelSecurityToken	Describes the new *SecurityToken* issued by the *Server*. This structure is defined in-line with the following indented items.
channelId	ByteString	A unique identifier for the *SecureChannel*. This is the identifier that shall be supplied whenever the *SecureChannel* is renewed.
tokenId	ByteString	A unique identifier for a single *SecurityToken* within the channel. This is the identifier that shall be passed with each *Message* secured with the *SecurityToken*.
createdAt	UtcTime	The time when the *SecurityToken* was created.
revisedLifetime	Duration	The lifetime of the *SecurityToken* in milliseconds. The UTC expiration time for the token may be calculated by adding the lifetime to the *createdAt* time.
serverNonce	ByteString	A random number that shall not be used in any other request. A new *serverNonce* shall be generated for each time a *SecureChannel* is renewed. This parameter shall have a length equal to key size used for the symmetric encryption algorithm that is identified by the *securityPolicyUri*.

Kun tietoturvattu kanava on luotu, pitää sen päälle luoda vielä istunto, jotta varsinainen tiedon, asetusten sekä eri komentojen välitys voidaan aloittaa. Seuraavaksi käsitellään sovelluskerroksen tietoturvaa sekä siihen liittyviä mekanismeja (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

8.2 Sovelluskerroksen tietoturva

Istunto *(Session)* on sovelluskerroksen tietoturvakäytäntö, joka vastaa sovellusten käyttäjien tunnistamisesta sekä käyttöoikeuksien tarkastamisesta. Käyttäjän tunnistus *(UserIdentityToken)* hoidetaan tunnistetiedoilla, joita taulukon 15 mukaan on neljää eri tyyppiä: käyttäjätunnus ja salasana, X.509-sertifikaatti, web-sovellusvaltuus sekä anonymous, jossa käyttäjätietoja ei ole saatavilla. Istuntopalveluita on määritelty kolme: CreateSession, ActivateSession sekä CloseSession. Käyttäjän tunnistetiedot välitetään palvelimelle istunnon aktivointipyynnön yhteydessä (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

TAULUKKO 15. Tunnistetiedot (Lähde: OPC Foundation)

Symbolic Id	**Description**
AnonymousIdentityToken	No user information is available.
UserNameIdentityToken	A user identified by user name and password.
X509IdentityToken	A user identified by an X509v3 *Certificate*.
IssuedIdentityToken	A user identified by a WS-*SecurityToken*.

Asiakas aloittaa istunnon luomisen lähettämällä *CreateSession*-palvelupyynnön palvelimelle. Viesti on turvattu käytettävän tietoturvamoodin mukaan, mutta tässä vaiheessa viestien salaukseen käytetään symmetrisen salauksen avaimia asymmetristen avaimien sijaan (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

Asiakkaan lähettämä pyyntö sisältää useita parametreja, joista tärkeimmät ovat sovellussertifikaatti *(ClientSertificate)* sekä satunnaismerkkijono *(ClientNonce). ClientNonce*-parametria käyttämällä palvelin pystyy todistamaan omistuksensa sertifikaattiin, jonka se lähettää vastausviestissään. Lisäksi palvelin varmentaa asiakkaan lähettämän sertifikaatin varmistuakseen siitä, että sertifikaatti on sama, mitä käytettiin turvatun tiedonsiirtokanavan luonnissa (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

RequestHeader-kenttä sisältää yleiset parametrit, kuten aikaleiman, *ClientApplication*kuvaa sovelluksen nimen sekä yksikäsitteisen URI-tunnisteen. *ServerUri*- on palvelimen yhteyspiste, joka määritellään silloin, jos palvelimeen luodaan yhteys gateway-palvelimen kautta. *EndPointUri*-kenttä kuvaa palvelimen yhteyspisteen, *SessionName* kertoo istunnon nimen. (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).
Asiakkaan lähettämän palvelupyynnön parametrit ovat kuvattu taulukossa 16

TAULUKKO 16. Asiakkaan lähettämän *CreateSession*-pyynnön rakenne (Lähde: OPC Foundation)

Name	Type	Description
Request		
requestHeader	RequestHeader	Common request parameters. The *authenticationToken* is always omitted. The type *RequestHeader* is defined in 7.28.
clientDescription	Application Description	Information that describes the *Client* application. The type *ApplicationDescription* is defined in 7.1.
serverUri	String	This value is only specified if the *EndpointDescription* has a *gatewayServerUri*. This value is the *applicationUri* from the *EndpointDescription* which is the *applicationUri* for the underlying *Server*. The type *EndpointDescription* is defined in 7.10.
endpointUrl	String	The network address that the *Client* used to access the *Session Endpoint*. The *HostName* portion of the URL should be one of the *HostNames* for the application that are specified in the *Server's ApplicationInstanceCertificate* (see 7.2). The *Server* shall raise an *AuditUrlMismatchEventType* event if the URL does not match the Server's *HostNames*. *AuditUrlMismatchEventType* event type is defined in Part 5. The Server uses this information for diagnostics and to determine the set of *EndpointDescriptions* to return in the response.
sessionName	String	Human readable string that identifies the *Session*. The *Server* makes this name and the *sessionId* visible in its *AddressSpace* for diagnostic purposes. The *Client* should provide a name that is unique for the instance of the *Client*. If this parameter is not specified the *Server* shall assign a value.
clientNonce	ByteString	A random number that should never be used in any other request. This number shall have a minimum length of 32 bytes. Profiles may increase the required length. The *Server* shall use this value to prove possession of its *Application Instance Certificate* in the response.
clientCertificate	ApplicationInstance Certificate	The *Application Instance Certificate* issued to the *Client*. The *ApplicationInstanceCertificate* type is defined in 7.2. If the *securityPolicyUri* is None, the *Server* shall ignore the *ApplicationInstanceCertificate*.
Requested SessionTimeout	Duration	Requested maximum number of milliseconds that a *Session* should remain open without activity. If the *Client* fails to issue a *Service* request within this interval, then the *Server* shall automatically terminate the *Client Session*.
maxResponse MessageSize	UInt32	The maximum size, in bytes, for the body of any response message. The *Server* should return a *Bad_ResponseTooLarge* service fault if a response message exceeds this limit. The value zero indicates that this parameter is not used. More information on the use of this parameter is provided in 5.3.

Palvelimen vastaanotettua asiakkaan lähettämän *CreateSession*-pyynnön, lähettää palvelin vastineensa pyydettyihin tietoihin, joista tärkeimmät selitetään seuraavassa kappaleessa.

SessionID on yksilöllinen tunniste, jota käytetään istuntoon liittyvien diagnostiikkatietojen selaamiseen. *AuthenticationToken* on myös yksilöllinen tunniste, jolla tunnistetaan istunto. *RevisedSessionTimeOut* on istunnolle määritelty maksimiaika, joka kertoo kuinka kauan istunto pysyy auki ilman aktiviteettia. *ServerNonce* on palvelimen muodostama satunnaismerkkijono, jolla asiakas pystyy todistamaan omistuksensa sertifikaattiin. *ServerCertificate*-kenttä sisältää palvelimen sertifikaatin (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

Palvelimen vastausviestin rakenne kokonaisuudessaan on kuvattu taulukossa 17.

TAULUKKO 17. Palvelimen lähettämän *CreateSession*-vastausviestin rakenne (Lähde: OPC Foundation)

Response		
responseHeader	ResponseHeader	Common response parameters (see 7.29 for *ResponseHeader* type).
sessionId	NodeId	A unique *NodeId* assigned by the *Server* to the *Session*. This identifier is used to access the diagnostics information for the *Session* in the *Server AddressSpace*. It is also used in the audit logs and any events that report information related to the *Session*. The *Session* diagnostic information is described in Part 5. Audit logs and their related events are described in 6.2
authentication Token	Session AuthenticationToken	A unique identifier assigned by the *Server* to the *Session*. This identifier shall be passed in the *RequestHeader* of each request and is used with the *SecureChannelId* to determine whether a *Client* has access to the *Session*. This identifier shall not be reused in a way that the *Client* or the *Server* has a chance of confusing them with a previous or existing *Session*. The *SessionAuthenticationToken* type is described in 7.31.
revisedSession Timeout	Duration	Actual maximum number of milliseconds that a *Session* shall remain open without activity. The *Server* should attempt to honour the *Client* request for this parameter, but may negotiate this value up or down to meet its own constraints.
serverNonce	ByteString	A random number that should never be used in any other request. This number shall have a minimum length of 32 bytes. The *Client* shall use this value to prove possession of its *Application Instance Certificate* in the *ActivateSession* request. This value may also be used to prove possession of the *userIdentityToken* it specified in the *ActivateSession* request.
serverCertificate	ApplicationInstance Certificate	The *Application Instance Certificate* issued to the *Server*. A *Server* shall prove possession by using the private key to sign the *Nonce* provided by the *Client* in the request. The *Client* shall verify that this *Certificate* is the same as the one it used to create the *SecureChannel*. The *ApplicationInstanceCertificate* type is defined in 7.2. If the *securityPolicyUri* is NONE and none of the *UserTokenPolicies* requires encryption, the *Client* shall ignore the *ApplicationInstanceCertificate*.
serverEndpoints []	Endpoint Description	List of *Endpoints* that the server supports. The *Server* shall return a set of *EndpointDescriptions* available for the *serverUri* specified in the request. The *EndpointDescription* type is defined in 7.10. The *Client* shall verify this list with the list from a *Discovery Endpoint* if it used a *Discovery*
Name	**Type**	**Description**
		Endpoint to fetch the *EndpointDescriptions*. It is recommended that *Servers* only include the *endpointUrl*, *securityMode*, *securityPolicyUri*, *userIdentityTokens*, *transportProfileUri* and *securityLevel* with all other parameters set to null. Only the recommended parameters shall be verified by the client.
serverSoftware Certificates []	SignedSoftware Certificate	This parameter is deprecated and the array shall be empty. The *SoftwareCertificates* are provided in the *Server AddressSpace* as defined in Part 5.
serverSignature	SignatureData	This is a signature generated with the private key associated with the *serverCertificate*. This parameter is calculated by appending the *clientNonce* to the *clientCertificate* and signing the resulting sequence of bytes. The *SignatureAlgorithm* shall be the *AsymmetricSignatureAlgorithm* specified in the *SecurityPolicy* for the *Endpoint*. The *SignatureData* type is defined in 7.32.
maxRequest MessageSize	UInt32	The maximum size, in bytes, for the body of any request message. The *Client Communication Stack* should return a *Bad_RequestTooLarge* error to the application if a request message exceeds this limit. The value zero indicates that this parameter is not used. 5.3 provides more information on the use of this parameter.

Istunto pitää vielä aktivoida ennen kuin sitä voidaan käyttää. Aktivointi tapahtuu *ActivateSession*-palvelulla. Asiakas lähettää palvelimelle istunnon aktivointipyynnön, joka sisältää käyttäjän valtuutukset sekä ohjelmistosertifikaatin. Aktivoinnin tarkoituksena on toimittaa palvelimelle käyttäjän tunnistamiseen tarvittavat tiedot sekä profiilit, joita asiakas tukee. Taulukossa 18 on listattu istunnon aktivointi-parametrit
(Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

TAULUKKO 18. *ActivateSession*-palvelun parametrit (Lähde: OPC Foundation)

Name	Type	Description
Request		
requestHeader	RequestHeader	Common request parameters. The type *RequestHeader* is defined in 7.28.
clientSignature	SignatureData	This is a signature generated with the private key associated with the *clientCertificate*. The *SignatureAlgorithm* shall be the *AsymmetricSignatureAlgorithm* specified in the *SecurityPolicy* for the *Endpoint*. The *SignatureData* type is defined in 7.32.
clientSoftwareCertificates []	SignedSoftware Certificate	Reserved for future use. The *SignedSoftwareCertificate* type is defined in 7.33.
localeIds []	LocaleId	List of locale ids in priority order for localized strings. The first *LocaleId* in the list has the highest priority. If the *Server* returns a localized string to the *Client*, the *Server* shall return the translation with the highest priority that it can. If it does not have a translation for any of the locales identified in this list, then it shall return the string value that it has and include the locale id with the string. See Part 3 for more detail on locale ids. If the *Client* fails to specify at least one locale id, the *Server* shall use any that it has. This parameter only needs to be specified during the first call to *ActivateSession* during a single application *Session*. If it is not specified the *Server* shall keep using the current *localeIds* for the *Session*.
userIdentityToken	Extensible Parameter UserIdentityToken	The credentials of the user associated with the *Client* application. The *Server* uses these credentials to determine whether the *Client* should be allowed to activate a *Session* and what resources the *Client* has access to during this *Session*. The *UserIdentityToken* is an extensible parameter type defined in 7.36. The EndpointDescription specifies what *UserIdentityTokens* the Server shall accept. Null or empty user token shall always be interpreted as anonymous.
userTokenSignature	SignatureData	If the *Client* specified a user identity token that supports digital signatures, then it shall create a signature and pass it as this parameter. Otherwise the parameter is omitted. The *SignatureAlgorithm* depends on the identity token type. The *SignatureData* type is defined in 7.32.
Response		
responseHeader	ResponseHeader	Common response parameters (see 7.29 for *ResponseHeader* definition).
serverNonce	ByteString	A random number that should never be used in any other request. This number shall have a minimum length of 32 bytes. The *Client* shall use this value to prove possession of its *Application Instance Certificate* in the next call to *ActivateSession* request.
results []	StatusCode	List of validation results for the *SoftwareCertificates* (see 7.34 for *StatusCode* definition).
diagnosticInfos []	DiagnosticInfo	List of diagnostic information associated with *SoftwareCertificate* validation errors (see 7.8 for *DiagnosticInfo* definition). This list is empty if diagnostics information was not requested in the request header or if no diagnostic information was encountered in processing of the request.

ClientSignature-parametri on allekirjoitus, joka luodaan asiakasohjelman yksityisellä avaimella. *Localeid* on lista, joka kertoo lokalisoinnin tärkeysjärjestyksen. Listan ensimmäisellä id:llä on korkein tärkeysjärjestys, toisella sitä seuraava tärkeysjärjestys jne. *UserIdentityToken*-parametri sisältää käyttäjätunnisteen. Palvelin päättelee tunnisteen perusteella onko asiakkaalla oikeudet istunnon aktivointiin. *UserTokenSignature*-parametria käytetään ainoastaan sellaisen käyttäjäntunniste tiedon kanssa, joka tukee digitaalisia allekirjoituksia.

Palvelin vastaa aktivointipyyntöön lähettämällä omat viestiparametrinsa. ServerNonce-satunnaismerkkijonolla asiakas todistaa omistuksensa omaan sovellussertifikaattiinsa, silloin kun istunnon aktivointi palvelua käytetään seuraavan kerran. *Results*-parametri on tilakooditieto. Tilakoodi kertoo onnistuiko palvelin vahvistamaan asiakkaan ohjelmisto-

sertifikaatin *DiagnosticInfos*- parametri sisältää diagnostiikkatietoa liittyen ohjelmistosertifikaatin validointiin. (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osat 2,4 ja 7).

8.3 Sertifikaatit

OPC UA -sovellukset käyttävät kolmea eri X.509-sertifikaattia: sovellussertifikaatteja *(Appliction Instance Sertificate)* sovellusten varmentamiseen, ohjelmistosertifikaatteja *(Software Certificates)* tuotteen varmentamiseen sekä käyttäjäsertifikaatteja *(User Certificates)* asiakkaan käyttäjän varmentamiseen. Lisäksi sertifikaatteja käytetään kommunikoivien osapuolten tunnistamiseen sekä julkisten avainten säilyttämiseen asymmetrisen salauksen operaatioita varten. Kahden ensimmäisen sertifikaatin käyttäminen UA-sovelluksissa on pakollista, kun taas käyttäjäsertifikaatin käyttö on vapaavalinnaista (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osa 2).

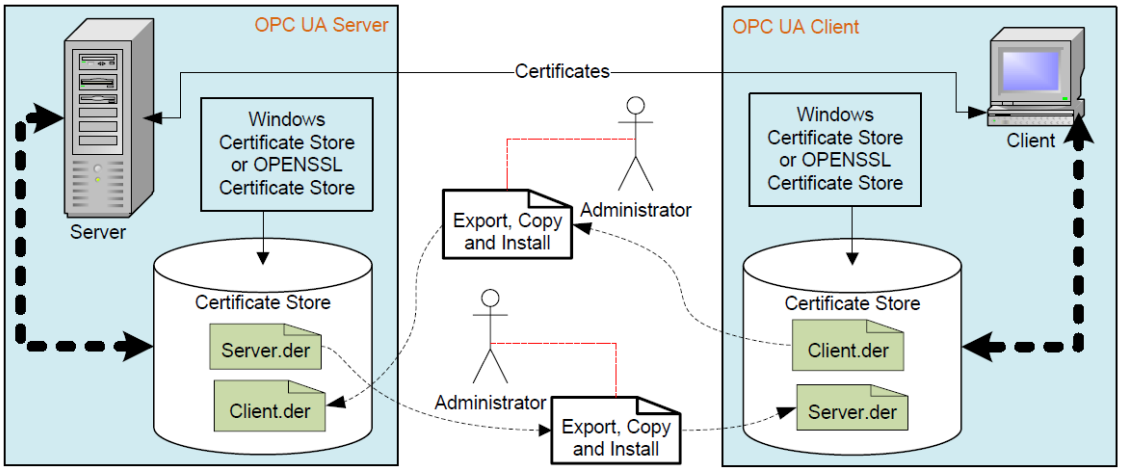
Sovellussertifikaatti on digitaalinen sertifikaatti, jota käytetään yksittäisen sovelluksen tunnistamiseen. Sitä voidaan käyttää myös viestien allekirjoitukseen ja salaukseen asymmetrisessä tietoturvamenettelyssä, kuten tietoturvatun tiedonsiirtokanavan avaamisessa. Ohjelmistosertifikaattia käytetään kyseessä olevan tuotteen tunnistamiseen. Sertifikaatti sisältää tiedot profiileista, joita sovellus tukee. Sovellukset välittävät tiedot tuetuista ominaisuuksista istunnon luomisen yhteydessä, jonka perusteella sovellukset päättelevät voivatko ne kommunikoida keskenään asianmukaisesti. Palvelin voi kieltäytyä aktivoimasta istuntoa, jos asiakkaan sertifikaatti ei vastaa palvelimen vaatimuksia. Toimittaja toimittaa aina ohjelmistosertifikaatin tuotteen mukana (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osa 2).

Käyttäjäsertifikaattia voidaan käyttää tunnistamaan sovelluksen käyttäjä-istunnon aktivoinnin yhteydessä. Muita mahdollisuuksia käyttäjän tunnistamiseen on esimerkiksi käyttäjätunnus/salasana-yhdistelmä tai Web-sovellusvaltuus. Sovelluksen käyttö anonyymisti on myös mahdollista ja siinä tapauksessa käyttäjää ei tunnisteta.

Sertifikaattien hallinnointijärjestelmä tulisi valita järjestelmän laajuuden ja vaatimusten mukaan. Laajoissa järjestelmissä ei ole järkevää ylläpitää sertifikaatteja manuaalisesti, koska niiden levittäminen jokaiselle sovellukselle erikseen olisi hyvin työlästä ja kuluttavaa. Järjestelmän ylläpitäjä joutuisi kopioimaan palvelinten julkisen avaimen ja siirtä-

mään sen kaikkiin asiakassovelluksiin, joiden kanssa kommunikointi pitäisi tapahtua ja toisinpäin. Tämän lisäksi sertifikaateilla on vanhenemispäivä, jonka takia ne joudutaan uusimaan jossain vaiheessa, mikä johtaa uuteen kopioimiskierrokseen. Kuvassa 34 on havainnollistettu manuaalinen sertifikaattien käsittely.

Ainoastaan pienissä ja tarkoin määritellyissä järjestelmissä manuaalinen sertifikaattien asentaminen voisi olla perusteltua (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osa 2).



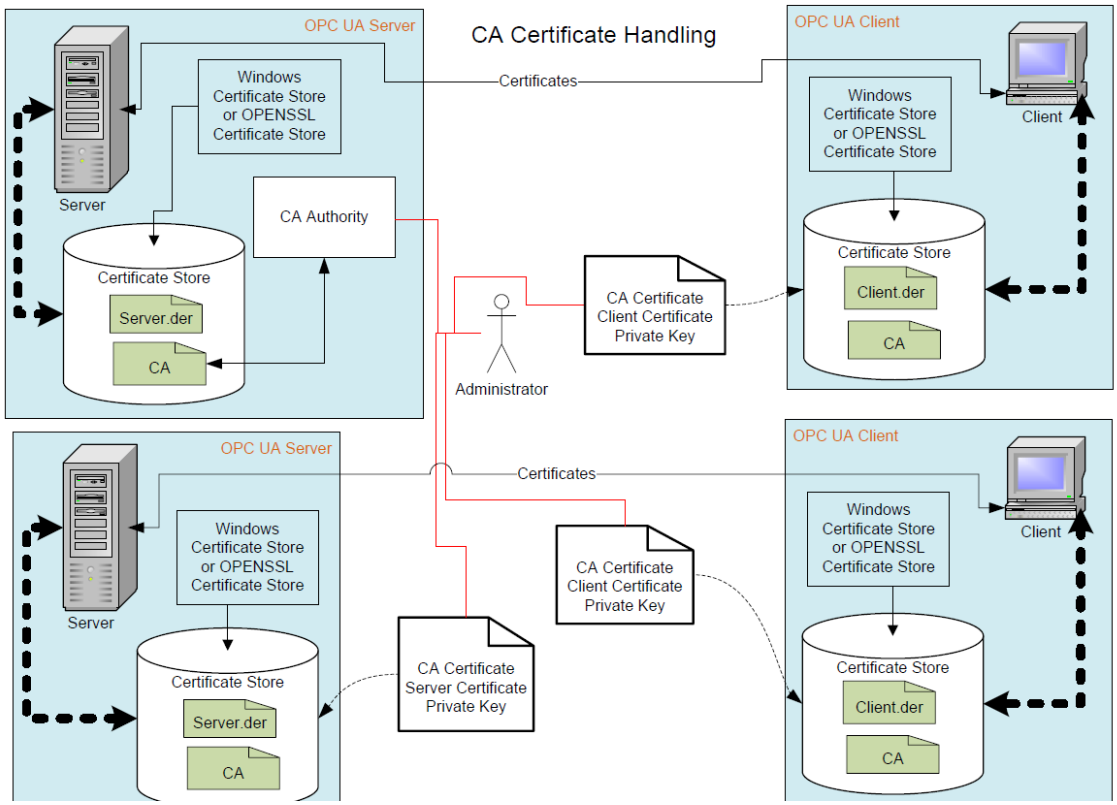
KUVA 34. Sertifikaattien manuaalinen hallinnointi. (Lähde: OPC Foundation).

Vaihtoehto manuaaliselle sertifikaattien asennukselle olisi PKI:n, eli julkisen avaimen infrastruktuurin käyttö, millä sertifikaattien hallinnointi ei olisi niin työlästä. PKI-järjestelmä tarjoaa tekniset ja organisaationaaliset perusteet digitaalisten sertifikaattien hallintaan, kuten allekirjoitusten ja varmenteiden luontiin sekä niiden jakamiseen. PKI-järjestelmässä sertifikaatit varmennetaan kolmannen osapuolen avulla, kuten varmenneviranomaisella CA. Laajoissa järjestelmissä PKI-infrastruktuurin käyttö on perusteltua, koska se helpottaa ja yksinkertaistaa asennusprosessia ja konfigurointia (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osa 2).

PKI-infrastruktuurissa ylläpitäjän *(Administrator)* tarvitsee luoda sovellussertifikaatti kaikille järjestelmän asiakkaille ja palvelimille, minkä varmenneviranomainen on allekirjoittanut, ja asentaa ainoastaan varmenneviranomaisen julkinen avain kaikille koneille.

Sitten kun sertifikaatti vanhenee, niin ylläpitäjän ei tarvitse tehdä muuta kuin korvata vanhentunut sertifikaatti, sillä julkisia avaimia ei tarvitse kopioida enää minnekään. Ku-

vassa 35 on havainnollistettu sertifikaattien hallinnointi käyttämällä varmenneviranomaista (Mahnke ym. 2009; OPC Unified Architecture OPC Unified Architecture: osa 2).



KUVA 35. Sertifikaattien hallinta kolmannen osapuolen toimesta. (Lähde: OPC Foundation)

9 MARKKINOILLA OLEVIA TUOTTEITA

OPC UA on alkanut saamaan jalansijaa teollisuudessa viime vuosien aikana yhä enemmän, minkä johdosta UA-tuotteiden lukumäärä on kasvussa. Useat eri OPC Foundation®-järjestöön kuuluvat yritykset, kuten Unified Automation, Prosys, MatrikonOPC sekä Softing tarjoavat sekä maksullisia että maksuttomia (evaluointi) SDK-kirjastopaketteja eri ohjelmointikielillä OPC UA -tuotekehitykseen. Lisäksi UA on saatavilla ilmaiseksi avoimen lähdekoodin periaatteella LGPL-lisenssin alla. C-pohjainen toteutus on ladattavissa osoitteesta: https://github.com/open62541/open62541.

Myös monet automaation laitevalmistajat tarjoavat erilaisia laitteita, joiden avulla esimerkiksi ohjelmoitava logiikka voidaan integroida OPC UA -yhteensopivaksi, minkä avulla dataa voidaan siirtää vaikka Internetin yli toiseen järjestelmään

9.1 Ohjelmistokehitys

Unified Automation GmbH tarjoaa viittä eri kirjastoa ohjelmistokehitykseen. Yrityksen tuotevalikoimiin kuuluu ainakin C99, C++, ANSI C, .NET sekä Java-pohjaiset SDK-kirjastot. Ohjelmistojen lisensointiin on neljä eri vaihtoehtoa, jotka ovat listattu alla:

- Evaluation-lisenssi testikäyttöön, jonka kesto on normaalisti 3kk ensi asennuksesta
-Runtime-lisenssi, oikeus asentaa ohjelmia ostettujen lisenssien mukaan.
- Binary Developer-lisenssi, sovelluskehitykseen
-Source Code-lisenssi, edellä mainitun lisäksi haltijalla on velvollisuus säilyttää lähdekoodin tietosuoja.

Suomalainen ohjelmistoyhtiö Prosys on erikoistunut OPC UA:n Java-kehitykseen ja tarjoaa useita itse kehittämiään ohjelmistoja, kuten esimerkiksi Prosys Java Client, Android Client ja Simulation Server-ohjelmia.

OPC Foundation® tarjoaa rekisteröityneille yritysjäsenilleen ilmaiseksi SDK:n, joka tukee .NET ja C++ kehitysympäristöä. Spesifikaatio on nykyisin kenen tahansa ladattavissa Foundationin verkkosivuilta.

9.2 Laitteita

EmbeddedLabs:n valmistama mikropiiritason palvelin mahdollistaa UA-integraation muihin tuotteisiin kohtalaisen vaivattomasti. Moduli, kuvassa 36, voidaan integroida käyttämällä UART, SPI tai I2C-protokollia. Modulin ominaisuuksia on:
- 72MHz ARM Cortex-M3
-Palvelinpino (Embedded Labs)
-Täysin konfiguroitava osoiteavaruus
-ANSI C- kirjasto sarjaliikenteelle
-10/100M ethernet
-Sisäinen reaaliaikakello, SNTP aikasynkronoinnilla

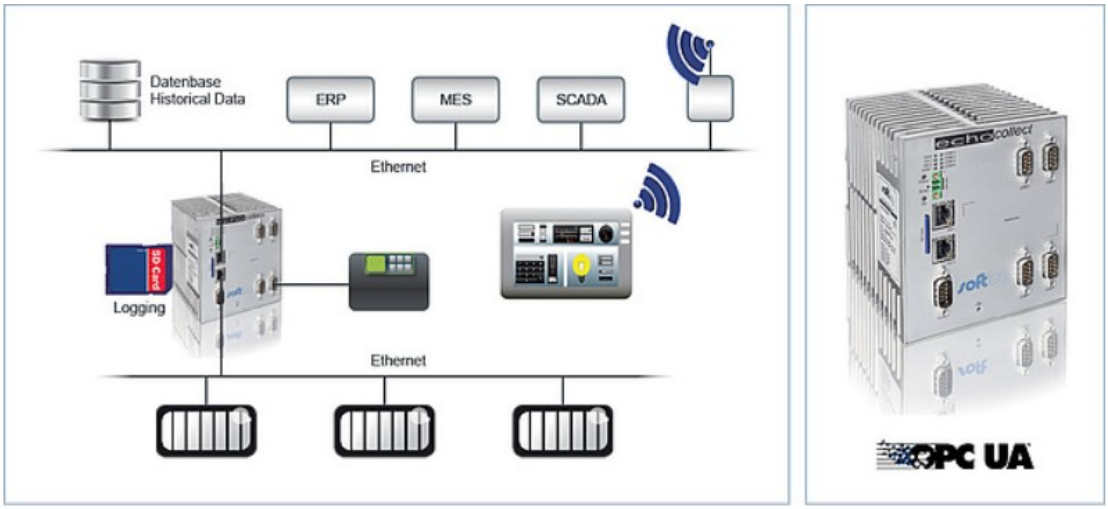
MatrikonOPC:lta on saatavissa OPC UA Embedded Server SDK-kirjastopaketti sulautettuille laitteille, mikä skaalautuu 128kB RAM mikrokontrollerilta (esim. Faslink) aina ARM9 ja Intel Atom-prosessoreille. /31/.

KUVA 36. FasaLink OPC UA-palvelinmoduli. (Lähde: Embedde Labs)

IBHLink, sulautettu OPC UA-palvelin/asiakas, kuvassa 37, on tarkoitettu yhdyskäytäväksi laitteiden sekä järjestelmien välille. Laite on suunniteltu yhdistämään Siemens SIMATIC S5, S7-200, S7-300, S7-400, S7-1200 ja S7-1500-sarjan logiikat OPC UA-kommunikaatioon. Modulissa on neljä ethernet-porttia, joista kolme on tarkoitettu konetason laitteille ja yksi portti UA-kommunikaatiolle. Tasot ovat suojattu ja erotettu palomuurilla. Konfiguroinnin voi suorittaa ilmaisella IBH OPC-editorilla, STEP7 tai TIA-ohjelmilla. IBHLink tukee samanaikaisesti sekä asiakkaan että palvelimen toimintoja, mikä mahdollistaa eri palvelimien, esimerkiksi eri valmistajien logiikoiden ja muiden laitteiden tiedonvaihdon. (IBH-Link).

KUVA 37. IBH-LINK (Lähde IBHSoftec).

Softing:in valmistama Echocollect on myös UA-palvelin, joka toimii yhdyskäytävänä laitteiden ja järjestelmien välillä, mutta tarjoaa sen lisäksi web-pohjaisen prosessidatan visualisoinnin sekä datan kirjaamisen *(logging)*. Lisäksi palvelin tukee useiden valmistajien, kuten Siemens S7, Rockwell, Wago, Beckhoff, Phoenix, Schneider sekä Mitsubishin valmistamia laitteita. Echocollectin käyttö automaation tiedonsiirrossa on havainnollistettu kuvassa 38. Molemmat sekä IBH-LINK, että Echocollect soveltuu hyvin jo olemassa olevien laitteiden liittämiseen UA-tiedonsiirtoon (Softing).



KUVA 38. Echocollect-yhdyskäytävä. (Lähde:Softing)

Wapice Oy:n WRM247+, kuvassa 39, on suunniteltu eri koneiden sekä laitteiden ja verkon väliseksi rajapinnaksi. Laitetta voidaan käyttää mittaukseen, etähallintaan sekä ohjaukseen. Laite on räätälöitävissä käyttötarkoituksen ja asiakkaan toiveiden mukaan, joten käyttämättömät liitännät voidaan jättää pois. WRM247+ sopii erityisesti liikkuvien koneiden tiedonkeruuseen kattavien liitäntöjen, kuten CAN-, RS232/485, Ethernet- ja USB-liittimien johdosta. Lisäksi siinä on sisäänrakennettu kiihtyvyysanturi ja monta tiedonsiirtovaihtoehtoa, kuten WLAN, GSM, GPRS, 3G, Bluetooth sekä GNSS-paikannus. Laite tukee Linux RT-käyttöjärjestelmää. Laitetta voidaan käyttää itsenäinesti tai osana IoT-ticket järjestelmää, jolloin järjestelmiin voidaan integroitua suoraan käyttämällä standardikommunikointiprotokollia kuten OPC-,OPC UA- sekä Modbus-protokollia (Wapice WRM-järjestelmä)

KUVA 39. WRM 247+ etähallintajärjestelmä.

10 POHDINTA

OPC UA:n skaalautuva, palvelukeskeinen sekä käyttöjärjestelmästä riippumaton arkkitehtuuri soveltuu koneiden, laitteiden ja automaatiojärjestelmien väliseen kommunikointiin, mobiiliympäristöihin sekä globaalien verkkojen väliseen tiedonsiirtoon, missä luotettavuus ja tietoturva tärkeitä elementtejä. Standardi on selvästi suunniteltu turvallisuuden ja tiedonmallinnuksen näkökulmista, mutta sen moniosaisuus saattaa tosin aiheuttaa vaikeuksia toiminnan ymmärtämisessä, sillä pelkästään tietoturva on aiheena hyvinkin laaja.

Vaikka standardin ensijulkaisusta on jo 10 vuotta, niin se kehittyy ja elää edelleen. Mielenkiintoisimpina lähitulevaisuuden asioina standardin kannalta näen kommunikaation laajennuksen julkaisija/tilaaja-mallilla yhdessä kehitteillä olevan TSN-Ethernet-laajennuksen kanssa, millä kommunikaatioon saataisiin reaaliaikaisuus-elementti.

Tämän työn luonnollisena jatkona näkisin käytännön osuuden, missä arkkitehtuurin suorituskykyä ja käytettävyyttä tutkittaisiin oikeassa satamalaitteiden toimintaympäristössä.

LÄHTEET

OPC Datahub. Cogent Real-Time Systems Inc. What is OPC? Luettu 30.11.2015.
http://www.opcdatahub.com/WhatIsOPC.html

Wolfgang Mahnke, Stefan-Helmut Leitner, Matthias Damm, Springer 2009. OPC Unified Architecture.

What is OPC? Luettu 1.12.2016.
https://opcfoundation.org/about/what-is-opc/

OPC:n uudet tuulet, Suomen Automaatioseura Ry. Luettu 15.10.2015.
http://www.automaatioseura.fi/index/tiedostot/OPC_Lisatieto.pdf

National Instruments, Introduction to OPC. Luettu 13.11.2015.
http://www.ni.com/white-paper/7451/en/

MathWorks, Introduction to OPC data. Luettu 5.8.2015.
http://se.mathworks.com/help/opc/ug/understanding-opc-data-value-quality-and-timestamp.html?s_tid=gn_loc_drop

Frank Iwanitz, Jurgen Lange, Huthig 2002, 2nd Edition. OPC Fundamentals, Implementation and Application.

OPC Foundation, what is OPC UA. Luettu 28.11.2015.
https://opcfoundation.org/about/opc-technologies/opc-ua/

National Instruments, white paper, why OPC UA matters. Luettu 25.12.2015.
http://www.ni.com/white-paper/13843/en/

OPC Unified Architecture,The future standard for communication and information modeling in automation Wolfgang Mahnke, Stefan-Helmut Leitner. Luettu 15.12.2015.
https://library.e.abb.com/public/75d70c47268d78bfc125762d00481f78/56-61%203M903_ENG72dpi.pdf

OPC Foundation. OPC Unified Architecture Spesification Part 1: Overview and Concepts. Versio 1.03, Helmikuu 2016.

OPC Unified Architecture e-book. Luettu 1.2.2016.
http://www.commsvr.com/UAModelDesigner/Index.aspx

Palvelukeskeinen arkkitehtuuri. Luettu 5.2.2016.
https://fi.wikipedia.org/wiki/Palvelukeskeinen_arkkitehtuuri

OPC Foundation. OPC Unified Architecture Spesification Part 4: Services. Versio 1.03, Helmikuu 2016.

OPC Foundation. OPC Unified Architecture Spesification Part 7: Profiles. Versio 1.03, Helmikuu 2016.

OPC Foundation. OPC Unified Architecture Spesification Part 6: Mappings. Versio 1.03, Helmikuu 2016.

Unified Automation, luettu 28.2.2016.
https://www.unified-automation.com/products/sdk-overview/licenses.html

Prosys: Multiplatform OPC UA Solutions, luettu 28.2.2016.
https://www.prosysopc.com

Extensible Markup Language, luettu 3.3.2016.
https://www.w3.org/TR/2008/REC-xml-20081126/#sec-intro

Java Web Services Architecture (The Morgan Kaufmann Series in Data Management Systems) Toukokuu 12, 2003. Luettu 2.3.2016.
http://flylib.com/books/en/2.844.1.38/1/

OPC UA_brochure. v.2013. Luettu 8.3.2016.
https://www.atvise.com/de/component/phocadownload/category/6-brochures

Publish/Subscribe in a Mobile Environment. Luettu 10.3.2016.
http://ilpubs.stanford.edu:8090/490/1/2001-15.pdf

OPC UA is Enhanced for Publish-Subscribe, artikkeli. Luettu 10.3.2016.
https://www.unified-automation.com/news/news-details/article/1217-opc-ua-is-enhanced-for-publish-subscribe-pubsub.html

OPC Foundation, Interoperability for Industrie 4.0 and the Internet of Things, Artikkeli. Luettu 12.3.2016
https://opcfoundation.org/wp-content/uploads/2015/09/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN-v4.pdf

Performance Evaluation of M2M Protocols Over Cellular Networks in a Lab Environment, Lars Durkop, Bjorn Czybik Jurgen Jasperneite. Luettu 13.3.2016.
https://www.hs-owl.de/init/uploads/tx_initdb/Duerkop__Czybik__Jasperneite_-_Performance_Evaluation_of_M2M_Protocols_Over_Cellular_Networks_01.pdf

OPC UA Performance Evaluation. Jouni Aro, Prosys. OPC Day Finland 2015. Julkaistu 29.10.2015, katsottu 16.3.2016.
https://www.youtube.com/watch?v=op8GqpH22Uw&index=12&list=PLryT9WIT6VQEXFL3b1Szp6VC7jXndXWAz

OPC Foundation. OPC Unified Architecture Specification Part 2: Security model. Versio 1.03, Helmikuu 2016.

IBH-Link. OPC UA-yhdyskäytävä. Luettu 10.4.2016.
http://www.ibhsoftec.com/IBH-Link-UA-Eng

Echocollect UA-palvelin. Luettu 10.4.2016.
http://industrial.softing.com/en/products/software/opc-servers/multiprotocol-embedded/echocollect-opcrm.html

Wapice WRM-järjestelmä. Luettu 10.4.2016.
https://www.wapice.com/fi/tuotteet/iot-ticket-internet-of-things-ratkaisut

Fasalink. Luettu 10.4.2016.
http://www.promelec.ru/pdf/Embedded%20Labs%20OPC-UA%20Solutions.pdf

LIITTEET

Liite 1. OPC UA:n osoiteavaruuden solmuluokat ja määritykset.

1(4)

Unified Automaton, OPC UA Fundamentals

http://documentation.unifiedautomation.com/uasdkcpp/1.2.0/L2UaNodeClasses.html
Luettu 3.2.2016.

Base Node-luokka. Muuttujat, jotka ovat yhteisiä kaikille solmuille.			
Base Node Class			
Attribute	**Use**	**DataType**	**Description**
NodeId	Mandatory	NodeId	Uniquely identifies a Node in an OPC UA server and is used to address the Node in the OPC UA Services.
NodeClass	Mandatory	NodeClass	An enumeration identifying the NodeClass of a Node such as Object, Variable or Method.
BrowseName	Mandatory	QualifiedName	Identifies the Node when browsing the OPC UA server. It is not localized.
DisplayName	Mandatory	LocalizedText	Contains the Name of the Node that should be used to display the name in a user interface. Therefore, it is localized.
Description	Optional	LocalizedText	This optional Attribute contains a localized textual description of the Node.
WriteMask	Optional	UInt32	Is optional and specifies which Attributes of the Node are writable, i.e., can be modified by an OPC UA client.
UserWriteMask	Optional	UInt32	Is optional and specifies which Attributes of the Node can be modified by the user currently connected to the server.
ObjectNode-luokkaa käytetään kuvaamaan järjestelmää, järjestelmäkomponentteja, reaali- ja ohjelmisto-olioita.			
Object			
Attribute	**Use**	**DataType**	**Description**
EventNotifier	Mandatory	Byte	This Attribute represents a bit mask that identifies whether the Object can be used to subscribe to Events and whether the history of Events is accessible and changeable.

(jatkuu)

VariableNode-luokkaa käytetään olion sisällön kuvaamiseen.
Muuttujat toimittavat varsinaisen datan.

Variable			
Attribute	**Use**	**DataType**	**Description**
Value	Mandatory	Base Data Type	The actual value of the Variable. The data type of the value is specified by the DataType, ValueRank, and ArrayDimensions Attributes.
DataType	Mandatory	NodeId	DataTypes are represented as Nodes in the Address Space. This Attribute contains a NodeId of such a Node and thus defines the DataType of the Value Attribute.
ValueRank	Mandatory	Int32	Identifies if the value is an array and when it is an array it allows specifying the dimensions of the array.
ArrayDimensions	Optional	UInt32[]	This optional Attribute allows specifying the size of an array and can only be used if the value is an array. For each dimension of the array a corresponding entry defines the length of the dimension.
AccessLevel	Mandatory	Byte	A bit mask indicating whether the current value of the Value Attribute is readable and writable as well as whether the history of the value is readable and changeable.
UserAccessLevel	Mandatory	Byte	Contains the same information as the AccessLevel but takes user access rights into account.
MinimumSamplingInterval	Optional	Double	This optional Attribute provides the information how fast the OPC UA server can detect changes of the Value Attribute. For Values not directly managed by the server, e.g., the temperature of a temperature sensor, the server may need to scan the device for changes (polling) and thus is not able to detect changes faster than this minimum interval.
Historizing	Mandatory	Boolean	Indicates whether the server currently collects history for the Value. The AccessLevel Attribute does not provide that information, it only specifies whether some history is available.

(jatkuu)

MethodNode-luokkaa käytetään kuvaamaan palvelimen osoiteavaruuden metodeita.

Method

Attribute	Use	DataType	Description
Executable	Mandatory	Boolean	A flag indicating if the Method can be invoked at the moment.
UserExecutable	Mandatory	Boolean	Same as the Executable Attribute taking user access rights into account.

ReferenceType-luokkaa käytetään ilmaisemaan palvelimen käyttämää viittaustyyppiä.

ReferenceType

Attribute	Use	DataType	Description
IsAbstract	Mandatory	Boolean	Specifies if the ReferenceType can be used for References or is only used for organizational purposes in the ReferenceType hierarchy
Symmetric	Mandatory	Boolean	Indicates whether the Reference is symmetric, i.e., whether the meaning is the same in forward and inverse direction
InverseName	Optional	LocalizedText	This optional Attribute specifies the semantic of the Reference in inverse direction. It can only be applied for nonsymmetric References and must be provided if such a ReferenceType is not abstract

ObjectType-luokkaa käytetään ilmaisemaan palvelimen osoiteavaruudessa olevien olioiden solmutyyppiä.

ObjectType

Attribute	Use	DataType	Description
IsAbstract	Mandatory	Boolean	This Attribute indicates whether the ObjectType is concrete or abstract and therefore cannot directly be used as type definition

(jatkuu)

VariableType-solmuluokkaa käytetään esittämään muuttujien solmuluokka palvelimen osoiteavaruudessa.

VariableType

Attribute	Use	DataType	Description
Value	Optional	Base Data Type	This optional Attribute defines a default value for instances of this VariableType. The data type of the value is specified by the DataType, ValueRank, and ArrayDimensions Attributes
DataType	Mandatory	NodeId	Defines the DataType of the Value Attribute for instances of this type as well as for the Value Attribute of the VariableType if provided
ValueRank	Mandatory	Int32	Identifies if the value is an array and when it is an array it allows specifying the dimensions of the array
ArrayDimensions	Optional	UInt32[]	This optional Attribute allows specifying the size of an array and can only be used if the value is an array. For each dimension of the array a corresponding entry defines the length of the dimension
IsAbstract	Mandatory	Boolean	This Attribute indicates if the VariableType is abstract and therefore cannot directly be used as type definition

DataType- kaikki tietoluokat osoiteavaruudessa esitetään DataType-solmuluokan solmuina.

DataType

Attribute	Use	DataType	Description
IsAbstract	Mandatory	Boolean	Indicates whether the DataType is abstract. An abstract DataType can be used in the DataType Attribute. However, concrete values must be of a concrete DataType

View-luokkaa käytetään rajoittamaan osoiteavaruudessa näkyvien solmujen ja viittausten määrää isoissa osoiteavaruuksissa.

View

Attribute	Use	DataType	Description
ContainsNoLoops	Mandatory	Boolean	This Attributes indicates whether the Nodes contained in the View do span a nonlooping hierarchy when following hierarchical References
EventNotifier	Mandatory	Byte	This Attribute represents a bit mask that identifies whether the View can be used to subscribe to Events and whether the history of Events is accessible and changeable

Liite 2. OPC UA:n sisäänrakennetut datatyypit

OPC Unified Architecture e-book. Luettu 2.3.2016.
http://www.commsvr.com/UAModelDesigner/Index.aspx?topic=html/7f816c6d-3aa2-4f38-9992-16d892e6fe4a.htm

ID	Name	Description
1	Boolean	A two-state logical value (true or false).
2	SByte	An integer value between -128 and 127.
3	Byte	An integer value between 0 and 256.
4	Int16	An integer value between -32768 and 32767.
5	UInt16	An integer value between 0 and 65535.
6	Int32	An integer value between – 2147483648 and 2147483647.
7	UInt32	An integer value between 0 and 4294967295.
8	Int64	An integer value between – 9223372036854775808 and 9223372036854775807
9	UInt64	An integer value between 0 and 18446744073709551615.
10	Float	An IEEE single precision (32 bit) floating point value.
11	Double	An IEEE double precision (64 bit) floating point value.
12	String	A sequence of Unicode characters.
13	DateTime	An instance in time.
14	Guid	A 16 byte value that can be used as a globally unique identifier.
15	ByteString	A sequence of octets.
16	XmlElement	An XML element.
17	NodeId	An identifier for a node in the address space of an OPC UA server.
18	ExpandedNodeId	A NodeId that allows the namespace URI to be specified instead of an index.
19	StatusCode	A numeric identifier for a error or condition that is associated with a value or an operation.
20	QualifiedName	A name qualified by a namespace.
21	LocalizedText	Human readable text with an optional locale identifier.
22	ExtensionObject	A structure that contains an application specific data type that may not be recognized by the receiver.
23	DataValue	A data value with an associated status code and timestamps.
24	Variant	A union of all of the types specified above.
25	DiagnosticInfo	A structure that contains detailed error and diagnostic information associated with a StatusCode.