Convergence of messaging

Suunnitteludokumentti

The Converge Group:
Mikko Hiipakka
Anssi Johansson
Joni Karppinen
Olli Pettay
Timo Ranta-Ojala
Tea Silander

Helsinki 10. joulukuuta 2002

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

0.0.1	21.10.2002	Dokumentti luotu	Olli Pettay
0.1	21.11.2002	Muutoksia rakenteeseen ym.	Joni Karppi
0.1.1	25.11.2002	Järjestelmän ydin -kappaletta täydennetty	Mikko Hiipal
0.1.2	25.11.2002	Ajastettujen sääntöjen käsittely	Olli Pettay
0.1.3	25.11.2002	Tietosuunnitelma	Olli Pettay
0.1.4	25.11.2002	Tiedonhaku	Anssi Johans
0.1.5	30.11.2002	Tiedonhakukappaletta täydennetty	Anssi Johans
0.2	2.12.2002	Asiakasmoduulia käsitteleviä kohtia päivitetty	Joni Karppi
0.2.1	3.12.2002	FTR korjauksia	Mikko Hiipal
1.0	5.12.2002	Jäädytetty	Mikko Hiipal

1 Johdanto

1.1 Dokumentin tarkoitus

Tämän suunnitteludokumentti esittää teknisen mallin ohjelmistotuotantoprojektin ”Convergence of Messaging” tuotteena syntyvästä järjestelmästä määrittelydokumentin vaatimusten pohjalta.

1.2 Määritelmät, termit ja lyhenteet

1.3 Dokumentin rakenne

Dokumentin luokkakuvausten kirjoittamisessa on noudatettu muutamaa sääntöä. Kaikki kuvatut luokat, kentät ja metodit ellei toisin ole mainittu ovat näkyvyydeltää public, mitä ei siis kirjoiteta kuvauksiin mukaan. Toinen noudatettu sääntö on, että luokkakuvauksissa aliluokkien kuvauksissa määritellään vain kokonaan uudet sekä yläluokan korvatut metodit.
Ensimmäinen luku esittelee dokumentin sisällön ja rakenteen sekä dokumentissa käytetyt termit ja lyhenteet.
Toisessa luvussa on mainittu ne lisäykset määrittelydokumenttiin, jotka ovat tulleet esiin määrittelydokumentin kirjoittamisen jälkeen.
Kolmas luku sisältää järjestelmän tietosuunnitelma.
Neljännessä luvussa kuvataan järjestelmän arkkitehtuuri yleisellä tasolla.
Viidennessä luvussa on kuvattuna järjestelmän ytimen luokkarakenne, luokkien tehtävät ja metodit, sekä ytimen rajapinnat.
Kuudennessa luvussa kuvataan tiedonhakumoduulin yhteydet sähköpostipalvelimille sekä rajapinnat järjestelmän muihin osiin.
Seitsemännessä luvussa kuvataan asiakasmoduulin tarjoamat rajapinnat asiakasohjelmalle ja järjestelmän ytimelle.
Kahdeksannessa luvussa kuvataan asiakasohjelman toiminta.

HTML	Hypertext Markup Language, WWW-sivujen kuvauksessa käytetty kieli http://www.w3.org/MarkUp/
HTTP	Hypertext Transfer Protocol http://www.w3.org/Protocols/
Javadoc	Java-kielen kommentointityökalu http://java.sun.com/j2se/javadoc/
XML	Extensible Markup Language http://www.w3.org/TR/REC-xml
DOM	Document Object Model, määritys dokumenttien luonnille, muuttamiselle ja läpikäynnille, käytetään usein XML-dokumenttien yhteydessä http://www.w3.org/DOM/
XPath	XML Path Language, kyselykieli XML:lle http://www.w3.org/TR/xpath
XUpdate	Kieli XML-dokumenttien muuttamiseksi, http://www.xmldb.org/xupdate/xupdate-wd.html
IMAP	Internet Message Access Protocol, protokolla sähköpostien noutoa varten http://www.imap.org/
SSL	Secure Sockets Layer, verkkoliikenteen salausmenetelmä http://wp.netscape.com/eng/ssl3/

Taulukko 1: Dokumentissa käytetyt termit ja lyhenteet

Yhdeksännessä luvussa määritellään ohjelmakoodin ulkonäköasioita. AttributeInterface -rajapinnan toteutusesimerkit on sisällytetty dokumenttiin liitteenä A.

2 Lisäyksiä määrittelydokumenttiin

2.1 Viestien lähetys

Järjestelmä tukee myös erilaisten viestien lähettämistä. Tietyntyyppisen viestin lähettämisen järjestelmän ulkopuolelle hoitaa yleensä se moduuli joka myös vastaanottaa kyseisentyyppisiä viestejä. Esimerkiksi jos järjestelmään joskus tehtäisiin liittymä USENET-uutisryhmiä varten, osaisi kyseinen lisäkomponentti sekä hakea viestit uutisryhmäpalvelimelta että lähettää palvelimelle uusia viestejä.

Toteutettavassa prototyypissä tuetaan sähköpostiviestien lähettämistä. Sitä ei kuitenkaan hoida sama IMAP-moduuli joka hakee viestejä palvelimelta, vaan tätä varten on poikkeuksellisesti erillinen, SMTP-protokollaa osaava, lähetysmoduuli.

3 Tietosuunnitelma

Järjestelmä käyttää tietojen säilyttämiseen Apache-organisaation Xindice XML-tietokantaa. Järjestelmän dynaamisesta luonteesta johtuen ei tietokannan dokumenttien rakenteelle voida asettaa tiukkoja muotovaatimuksia. Esimerkiksi viestien tallennuksessa käytettävän XML:n muoto riippuu muun muassa viestin tyypistä (onko se esimerkiksi sähköposti) ja siitä millaisia attribuuttimäärityksiä järjestelmään on tehty. Attribuuttiluokkia saatetaan tulevaisuudessa lisätä järjestelmään dynaamisesti, joten voi olla mahdollista että viestien muoto muuttuu järjestelmän toiminnan aikana.

Vaikka varsinaisille XML-dokumenteille ei tässä määritellä muotoa, tulee dokumenttien kantaantallettaminen määritellä. Xindicen avulla dokumenttien joukkoja ryhmitellään kokoelmiksi (collection). Tietokannan juuri on /db ja sen alaisuudessa seuraavat kokoelmat:

- /db/pref

Sisältää järjestelmän asetuksia, joihin tavalliset käyttäjät eivät pääse käsiksi.

- /db/users

  Sisältää kunkin käyttäjän nimisen alikokoelman käyttäjän tiedoille.

- /db/users/*user*

  Käyttäjän omat tiedot alikokoelmissa.

- /db/users/*user*/pref

  Käyttäjän asetukset dokumentissa preference.xml

- /db/users/*user*/profiles

  Käyttäjän profiilitiedot.

- /db/users/*user*/contextmodels

  Käyttäjän kontekstimallit.

- /db/users/*user*/messages

  Käyttäjän viestit erillisissä dokumenteissa.

- /db/users/*user*/views

  Käyttäjän näkymien hallintaan liittyvää informaatiota.

- /db/users/*user*/datasources

  Käyttäjän tiedonlähteisiin liittyvät asetukset.

- /db/groups/

  Sisältää ryhmien nimien mukaiset kokoelmat.

- /db/groups/*group*

  Ryhmän tiedot alikokoelmissa.

- /db/groups/*group*/pref

  Ryhmän asetukset

- /db/groups/*group*/profiles

  Ryhmän profiilit.

- /db/groups/*group*/contextmodels
  Ryhmän kontekstimallit.

- /db/groups/*group*/datasources
  Ryhmän tiedonlähteisiin liittämät asetukset.

  Lokitietoja kerätään erillisiin tiedostoihin.

4 Arkkitehtuuri

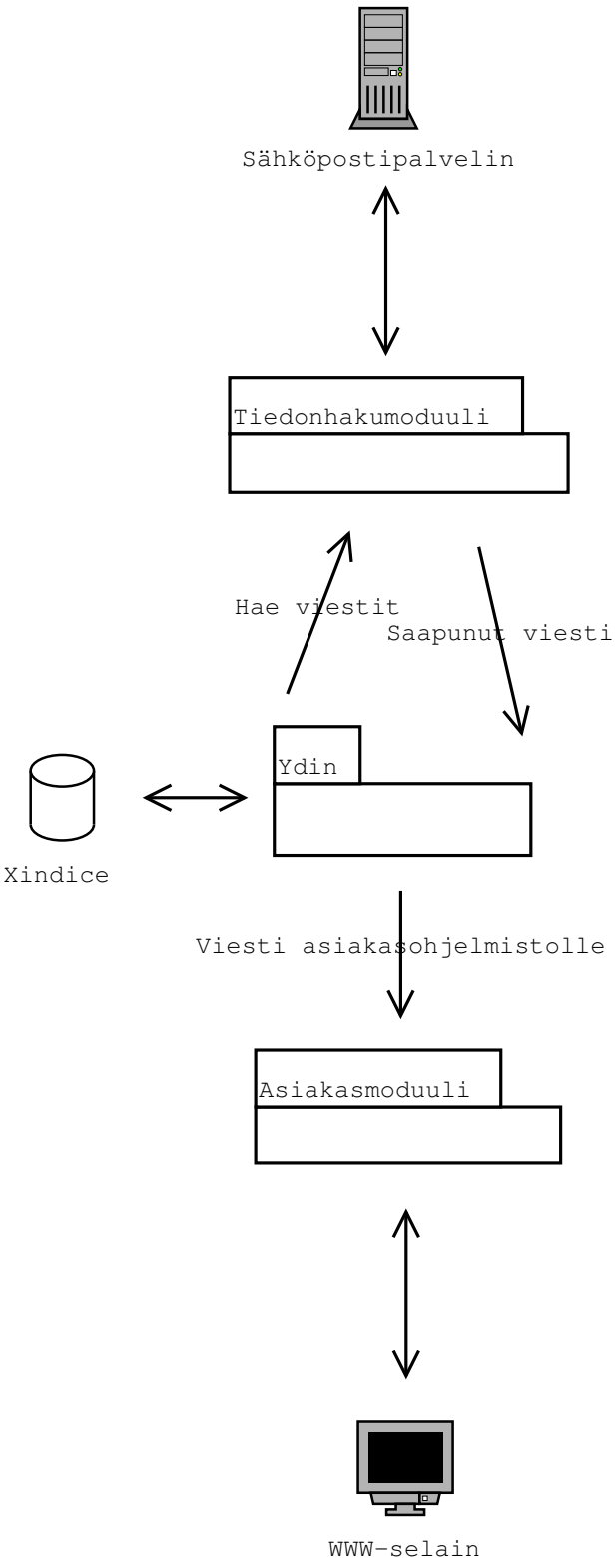
Järjestelmä voidaan jakaa toiminnallisesti kolmeen eri moduuliin: tiedonhakumoduuliin, järjestelmän ytimeen ja asiakasmoduuliin.

Ytimen rooli järjestelmässä on hallinnoida ryhmiä ja käyttäjiä sekä näihin liitettyjä tietoja kuten kontekstimallit ja profiilit. Toimintoja, joista ydin huolehtii ovat tietokantaoperaatiot, viestien käsittely kontekstimallivertailujen ja profiilisääntöjen perusteella sekä ajastettujen toimintojen hallinta. Sen tehtäviin kuuluu myös suuri osa asiakasmoduulin tarjoaman käyttöliittymän palveluista, muun muassa näkymien hallinta ja tiedonlähdeasetusten ylläpitäminen.

Tiedonhakumoduuli huolehtii yhteyksistä järjestelmän ulkopuolella oleville palvelimille. Näitä palvelimia ovat esimerkiksi sähköpostipalvelimet (sekä saapuvat että lähtevät viestit), uutisryhmäpalvelimet, WWW-palvelimet ym. Tiedonhakumoduuli myös muokkaa saapuneet viestit järjestelmän vaatimaan sisäiseen muotoon.

Asiakasmoduuli yhdistää järjestelmän ja järjestelmän käyttäjän. Kaikki tieto järjestelmältä asiakkaalle ja asiakkaalta järjestelmään välitetään asiakasmoduulin kautta.
Asiakasmoduuli huolehtii järjestelmän ytimestä tulevien viestien muuttamisesta järjestelmän sisäisestä muodosta kunkin asiakasohjelman ymmärtämään muotoon, esimerkiksi muuttaen viestin HTML-muotoon toimitettaessa viestiä WWW-sovellukselle. Asiakasmoduulissa muutetaan myös asiakkaalta tuleva tieto järjestelmän sisäiseen muotoon, esimerkiksi käyttäjän luodessa uusia kontekstimalleja

tai profiileja.

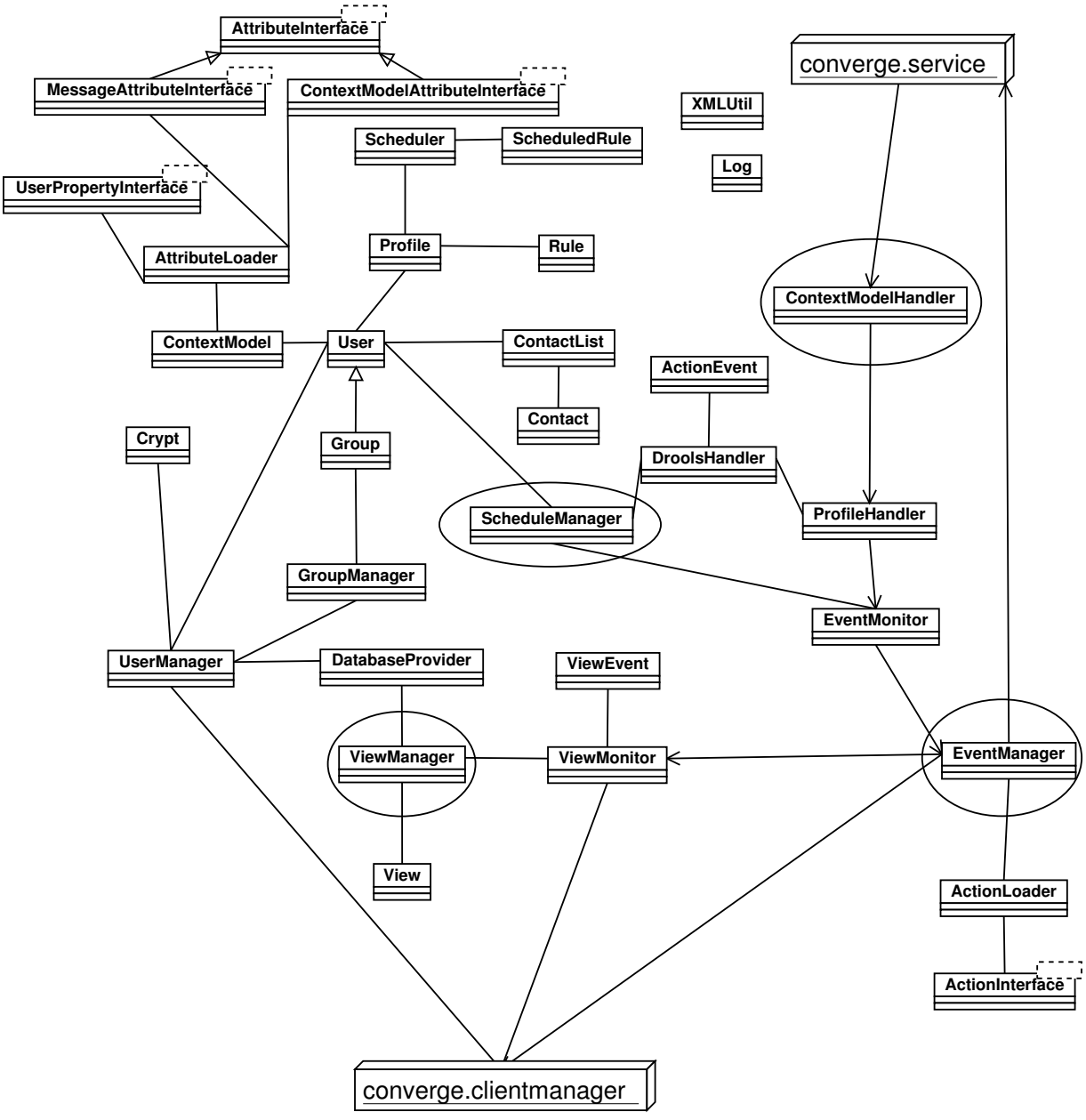


Kuva 1: Arkkitehtuuri

5 Järjestelmän ydin

5.1 Luokat

Pakkaus converge.kernel



Kuva 2: Ytimen luokkakaavio, jossa soikiot kuvaavat säikeitä

5.1.1 ContextModel

ContextModel-luokka tarjoaa attribuuttien hallintaa sekä vertailun viestiolion attribuutteihin. Kontekstimalli-ilmentymälle voidaan määritellä joukko attribuutteja, joihin liitetään arvo sekä vertailussa käytettävä laskusääntö. Näiden avulla luokka laskee itselleen arvon siitä kuinka hyvin viestiolion attribuuttien arvot toteuttavat laskusäännön määrittelemällä tavalla malliin liitettyjen attribuuttien arvovälejä. Saatu arvo liitetään viestiolioon ContextModelValueAttribute-tyyppisenä (kts. luku A.1, sivulla 68) attribuuttina.

Määrittely :

class ContextModel implements XMLContentInterface

Konstruktorit :

- **ContextModel**(String name)
  Alustaa uuden name-nimisen kontekstimalli ilmentymän valmiiksi, attribuuttien lisäämiselle
- **ContextModel**(org.w3c.dom.Element contextModelElement)
  Konstruktori, jolla kontekstimalli-ilmentymä muodostetaan antamalla ilmentymän tietosisältö XML-elementtinä.

Metodit :

- void **addAttribute**(String name, String expressionType, Object value, boolean forced, int weight)
  Lisää kontekstimalliin uuden *name*-nimisen attribuutin, jonka arvoväli määritellään *value*:ksi. Laskusääntö siitä miten tätä arvoväliä tullaan vertaamaan viestiolion samannimisen attribuutin arvoon annetaan *expressionType*:lla (laskenta suoritetaan ”Viestiolion.Attribuutin.Arvo” expressionType value). *Forced*-parametrilla asetetaan, onko määritelty attribuutti kontekstimallin arvon laskennalle ehdoton vaatimus eli kontekstimalli ei saa arvoa, jos kyseisen attribuutin vertailu arvoväliin ei ole

tosi. *weight*-parametri kertoo lisätyn attribuutin painoarvon kontekstimallissa.

- protected void **startProcessing**(Message msg)
  Laukaisee kontekstimalliolion suorituksen, jonka aikana malli laskee itselleen arvon. Tuo arvo liitetään MessageAttributeInterface:n toteuttavana attribuuttina *msg*-olioon, minkä nimeksi määräytyy ilmentymälle määritetty nimi.
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34
- String **getModelName**()
  Palauttaa kontekstimalli-ilmentymälle annetun nimen.
- ContextModelAttributeInterface[] **getAllAttributes**()
  Palauttaa kontekstimalli-ilmentymään liitetyt attribuutit taulukkona.

5.1.2 ContextModelHandler

ContextModelHandler-luokan toiminta alkaa kun tiedonhakumoduuli lähettää järjestelmään saapuneen viestin luokan käsiteltäväksi. Toteutuksen tulee tukea viestien jatkuvaa saapumista siten, että viestien vastaanotto on säikeistettyä toimintaa. Kun uusi säie on käynnistetty viestin käsittelyyn, suoritetaan kontekstimallimuuttujien arvovälien vertailu viestiolion tarjoamiin konteksti- ja viestiattribuutteihin. Tämä vertailu suoritetaan kaikille vastaanottajana (useamman vastaaottajan viestit käsitellään jokaiselle erikseen) käsiteltävään käyttäjään liitetyille kontekstimalleille (tämän tuloksena kontekstimallit liittävät itselleen saamat arvot viestiolioon nimetyiksi attribuuteiksi). Lopuksi suorittava säie antaa käsittelemänsä viestin edelleen käsiteltäväksi ProfileHandler-luokalle, jonka toiminta suoritetaan siis samassa säikeessä kuin ContextModelHandler:kin toiminta.

Määrittely :

Public class ContextModelHandler

Private class Process extends Thread

Metodit :

- static synchronized void **incomingMessage**(Message msg)
  Vastaanottaa uuden viestiolion ja luo uuden (Process) säikeen sen käsittelemiseksi.

5.1.3 ProfileHandler

ProfileHandlerissa aluksi käyttäjän oletusprofiilin säännöt, viesti ja tyhjä toimintatapahtuma annetaan DroolsHandlerille. Tämän jälkeen vertaillaan viestiolion eri kontekstimalliarvoja jokaisen profiilin hallinnoimien kontekstimallien kynnysarvoihin. Arvon ylityksestä seuraa viestin ja sääntöjen antaminen DroolsHandlerille, joka muokkaa toimintatapahtumaolioita sääntöjen mukaan. Saadut toimintatapahtumat annetaan viestin kanssa EventMonitorille.

Määrittely :

class ProfileHandler

Konstruktorit :

- **ProfileHandler**(Message msg)
  Luodaan uusi ProfileHandler-olio viestiä *msg* varten.

Metodit :

- void **startProcessing**()
  Tämän metodi kutsu aloittaa ProfileHandlerin toiminnan suorituksen.

5.1.4 RuleHandler

RuleHandler hoitaa käyttän profiiliin liitettyjen sääntöjen käsittelystä.

Määrittely :

class RuleHandler

Metodit :

- void **processMessage**(Message msg, Profile p, ActionEvent ae)
  Tällä metodilla viestin *msg* tiedoista kerätään tietoja halutuista toiminnoista profiilin *p* sääntöjen avulla ActionEvent-tyyppiseen *ae*-olioon.
- void **processScheduledRule**(User user, Scheduler sc, ActionEvent ae)
  Tällä metodilla ajastetun käyttäjän *user* ja säännön *sc* avulla kerätään tietoja halutuista toimenpiteistä *ae*-olioon.

5.1.5 ActionEvent

ActionEvent-olioita käytetään Droolsin sääntöjenkäsittelyn yhteydessä keräämään sääntöjen päättelyn seurauksena syntyneitä toimintoja EventManagerin ymmärtämään muotoon.

Määrittely :

class ActionEvent

Konstruktorit :

- **ActionEvent**()
  Oletuskonstruktori
- **ActionEvent**(ActionEvent event)
  Kopiokonstruktori

Metodit :

- void **addEvent**(String key, String value)
  Lisätään uusi tapahtuma, jonka nimi määritellään *key*-parametrin avulla ja arvo *value* -parametrillä. Tämän metodikutsun toistaminen samannimisellä tapahtumalla lisää tapahtumaan erillisiä arvoja.

- void **replaceEvent**(String key, String value)
  key-nimisen tapahtuman arvon muuttaminen siten, että kaikki vanhat arvot korvataan arvolla *value*.
- void **replaceEvent**(String key, String oldvalue, String newvalue)
  Etsitään *key*-nimisestä tapahtumasta arvo *oldvalue* ja korvataan se arvolla *newvalue*. Mikäli vanhaa arvoa ei löydy, suoritetaan addEvent-metodin kutsu.
- void **deleteEvent**(String key)
  Poistetaan kaikki *key*-nimisen tapahtuman arvot.
- void **deleteEvent**(String key, String oldvalue)
  Poistetaan *key*-nimisen tapahtuman arvo *oldvalue*.
- HashMap **getEvents**()
  Tämän metodin avulla EventManager pääsee käsiksi olion sisältämiin tapahtumiin.

5.1.6 EventMonitor

EventMonitorissa hallitaan ytimen sisään tulleiden viestien käsittelyssä ja ajastetuissa säännöissä syntyneiden tapahtumaolioiden tallentamista siihen saakka, kunnes omassa säikeessä toimiva EventManager hoitaa viestien jatkokäsittelyn.

Määrittely :

class EventMonitor

Metodit :

- synchronized **incomingEvent**(Object o, ActionEvent evt)
  Uuden tapahtumaolion lisääminen parametrien *o* ja *evt* avulla.
- synchronized Pair **removeEvent**()
  Vanhimman käsittelemättömän olio-tapahtumaolio -parin poisto EventMonitorista.

- static EventMonitor **getInstance**()
  Palauttaa arvonaan järjestelmän ainoan EventMonitor-olion.

5.1.7 EventManager

EventManager-luokka suorittaa tapahtumaolioiden määrittelemiä toimintoja. ActionInterface-tyyppisten olioiden avulla se suorittaa muun muassa viestien kantaanlisäyksen, näkymien hallinnan ja viestien lähettämiseen liittyvät toiminnot.

Määrittely :

class EventManager extends Thread

Metodit :

- static EventManager **getInstance**()
  Palauttaa arvonaan järjestelmän ainoan EventManager-olion.
- void **run**()
  EventManager-säikeen käynnistyessä alkaa run-metodin suoritus, jossa EventMonitorilta haetaan viesti-tapahtumaolio tai ajastettu sääntö-tapahtumaolio -pareja ja toimitaan tapahtumaolion määritysten mukaisesti.

5.1.8 ActionLoader

ActionLoader-luokalla ladataan (tarvittaessa dynaamisesti) ActionInterface-rajapinnan toteuttavia luokkia järjestelmän käyttöön.

Määrittely :

class ActionLoader

Metodit :

- static ActionInterface **newActionInstance**(String name)
  Palveluna luodaan ActionInterface ilmentymä, joka tarjoaa toteutuksen *name*-nimiselle toiminnolle. Jos luonti epäonnistuu palautetaan null.
- static synchronized void **loadClasses**()
  Ladataan Action-toteutusluokat, ja tarkastetaan niiden tarjoamat toteutukset.

5.1.9 View

View-tyyppisillä olioilla kuvataan tietokannan näkymien sisältöä.

Määrittely :

class View implements XMLContentInterface

Metodit :

- String **getName**()
  Palautetaan näkymän nimi.
- String[] **getMessageIDs**()
  Palautetaan kaikkien näkymään liitettyjen viestien tunnukset.

5.1.10 ViewMonitor

Järjestelmässä ViewMonitor-luokka huolehtii viestien ja näkymien muutosten synkronoinnista.

Määrittely :

class ViewMonitor

Metodit :

- static void **setMessageToView**(Message msg, String view)
  Luodaan pyyntö viestin *msg* lisäämisestä näkymään *view*.

- static void **setMessageToView**(User user, String id, String view)
  Luodaan pyyntö viestin *id* liittämisestä käyttäjän *user* näkymään *view*.
- static void **deleteMessageFromView**(User user, String id, String view)
  Viestin *id* poistaminen käyttäjän *user* näkymästä *view*.
- static void **deleteMessage**(User user, String id)
  Käyttäjän *user* viestin *id* poistaminen kaikista näkymistä ja kannasta.
- static View **getView**(User user, String view)
  Palautetaan käyttäjän *user* näkymä *view*.
- static Document **getMessage**(User user, String id)
  Palautetaan käyttäjän *user* viesti *id*.
- static String[] **getViewNames**(User user)
  Palautetaan käyttäjän *user* kaikkien näkymien nimet.

5.1.11 ViewEvent

ViewEvent on apuluokka ViewMonitorin ja ViewManagerin välillä. ViewEvent-olioiden avulla ViewManager suorittaa viestiin ja näkymiin liitettyviä pyydettyjä toimintoja.

Määrittely :

class ViewEvent

Konstruktorit :

- **ViewEvent**(int type, Message msg, String view)
  Luodaan uusi ViewEvent tyypin *type*, viestin *msg* ja näkymän *view* avulla.
- **ViewEvent**(int type, MessageID id, String view)
  Luodaan uusi ViewEvent tyypin *type*, viestin tunnisteen *id* ja näkymän *view* avulla.

- **ViewEvent**(int type, Message msg)
  Luodaan uusi ViewEvent tyypin *type* ja viestin *msg* avulla.
- **ViewEvent**(int type, MessageID id)
  Luodaan uusi ViewEvent tyypin *type* ja viestin tunnisteen *id* avulla.
- **ViewEvent**(User user, int type, String view)
  Luodaan uusi ViewEvent käyttäjän *user*, tyypin *type* ja näkymän *view* avulla.

Kentät :

- final int **DELETE_MESSAGE_TOTAL**
  Tyyppi: tuhoa viesti järjestelmästä.
- final int **DELETE_MESSAGE_VIEW**
  Tyyppi: tuhoa viesti mäkymästä.
- final int **ADD_MESSAGE**
  Tyyppi: Liitä viesti näkymään.
- final int **ADD_BY_MESSAGE_ID**
  Tyyppi: Liitä viesti näkymään viestin tunnisteen avulla.
- final int **GET_VIEW**
  Tyyppi: Hae näkymä.
- final int **GET_MESSAGE**
  Tyyppi: Hae viesti.

5.1.12 ViewManager

ViewManager huolehtii ViewMonitorille annettujen näkymien ja viestien muutospyyntöjen käsittelemisestä. Järjestelmässä oleva yksi ainoa ViewManager kuuntelee ViewMonitorille tulevia pyyntöjä ja suorittaa niitä saapumisjärjestyksessä. Mahdolliset muutokset tietokantaan hoidetaan DatabaseProviderin kautta.

Määrittely :

class ViewManager

Metodit :

- static ViewManager **getInstance**()
  Palautetaan järjestelmän ainoa ViewManager-olio.

5.1.13 DatabaseProvider

DatabaseProvider tarjoaa tietokantapalveluita muille järjestelmän olioille. Tietokantana se käyttää Apache organisaation Xindice XML-tietokantaa, ja tästä syystä suoritettavat kyselyt ovat lähinnä DOM-puiden läpikäyntiä. Tulevaisuudessa myös XPath-kyselyt ja XUpdate-päivitykset voivat olla mahdollisia.

Määrittely :

class DatabaseProvider

Metodit :

- static DatabaseProvider **getInstance**() Palautetaan järjestelmän ainoa DatabaseProvide olio.
- protected synchronized Collection **getUserCollection**(String name)
  Palautetaan käyttäjän *name* kokoelma.
- protected synchronized Collection **newUserCollection**(String name)
  Luodaan uusi kokoelma käyttäjää *name* varten.
- protected synchronized void **removeUserCollection**(String name)
  Poistetaan käyttäjän *name* kokoelma kannasta.
- protected synchronized void **setDocument**(Collection col, String sub, String id, Document dom)
  Asetaan dokumentti *dom* tunnisteella *id* kokoelmaan *col*, tai mikäli *sub* ei ole null, kokoelman *col* alikokoelmaan *sub*.

- protected synchronized Document **getDocument**(Collection col, String sub, String id)
  Palautetaan dokumentti *id* kokoelmasta *col*, tai mikäli *sub* ei ole null, kokoelman *col* alikokoelmasta *sub*.
- protected synchronized void **deleteDocument**(Collection col, String sub, String id)
  Poistetaan dokumentti *id* kokoelmasta *col*, tai mikäli *sub* ei ole null, kokoelman *col* alikokoelmasta *sub*.

5.1.14 Crypt

Crypt-luokka tarjoaa palveluita datan salaamiseksi salasanan avulla.

Määrittely :

class Crypt

Metodit :

- static String **encode**(String pw, byte[] data)
  Salataan *data*-parametrin sisältämä tieto salasanalla *pw* ja palautetaan salattu tietosisältö base64-muotoisena String-oliona.
- static String **decode**(String pw, String crypted)
  Salasanan *pw* avulla puretaan *crypted*-parametrin base64-muodossa antama salattu tieto ja palautetaan salaamaton tieto.

5.1.15 User

User-luokan ilmentymillä ylläpidetään järjestelmässä eri käyttäjien tietoja. Ilmentymät sisältävät profiileja, kontekstimalleja, viittauksia tiedonlähteisiin, ominaisuusolioita ja kontaktilistan.

Määrittely :

class User

Konstruktorit :

- protected **User**(String name, String password)
  Uuden User-olion luominen on sallittua vain ytimessä (pakkaus converge.kernel). Parametreinä nimi, *name* ja salasana, *password*

Metodit :

- synchronized void **readLock**(Object locker)
  Olio voidaan lukulukita, jolloin kukaan ei voi muuttaa olion tietoja. Lukulukkoja User-oliota kohti voi olla useita. Parametrinä *locker* saadaan se olio, joka lukituksen haluaa tehdä.
- synchronized void **writeLock**(Object locker)
  User-olio voidaan kirjoituslukita, jolloin lukuoperaatiot on estetty. Kirjoituslukon tulisi olla päällä mahdollisimman vähän aikaa, sillä lukituksen ollessa päällä vain yksi olio tai luokka pystyy käsittelemään User-ilmentymää. Parametrinä *locker* saadaan se olio, joka lukituksen haluaa tehdä.
- synchronized void **releaseLock**(Object locker)
  Lukon vapautus, toimii sekä luku- että kirjoituslukoille. Parametrinä *locker* saadaan se olio, jonka saama lukko vapautetaan.
- synchronized boolean **hasLock**(Object locker)
  Voidaan testata onko oliolla *locker* User-olion lukko.
- ArrayList **getContextModels**()
  Palautetaan käyttäjän kaikki konteksimallit.
- void **addContextModel**(ContextModel model)
  Lisätään kontekstimalli *model*.
- ArrayList **getProfiles**()
  Palautetaan kaikki käyttäjän profiilit.

- void **addProfile**(Profile profile)
  Lisätään profiili *profile*.
- void **addDatasource**(Datasource src)
  Lisätään uusi tiedonlähde *src*.
- void **deleteDatasource**(Datasource src)
  Poistetaan tiedonlähde *src*.
- Datasource **getDatasource**(String name)
  Palautetaan tiedonlähde nimen *name* perusteella.
- String[] **getDatasourceNames**()
  Palautetaan kaikkien User-olioon liitettyjen Datasource-olioiden nimet.
- ContactList **getContactList**()
  Palautetaan käyttäjän kontaktilista.
- UserPropertyInterface **getProperty**(String name)
  Pyydetään käyttäjäoliolta ominaisuusoliota nimen *name* perusteella.
- void **setProperty**(String name, Object value)
  Asetetaan ominaisuusolion *name* arvoksi *value*.
- void **save**()
  Kun käyttäjän tiedot halutaan tallentaa tietokantaan, kutsutaan tätä metodia.

5.1.16 UserManager

UserManager hallinnoi käyttäjäolioita ja huolehtii niiden kantaantallennuksesta.

Määrittely :

class UserManager

Metodit :

- static UserManager **getInstance**()
  Palautetaan järjestelmän ainoa UserManager-olio.

- User **getUser**(String name, String password)
  Palautetaan User-olio nimen ja salasanan perusteella.
- User **newUser**(String name, String password)
  Luodaan uusi käyttäjä nimellä *name* ja salasanalla *password*, mikäli sellaista ei jo järjestelmässä ole.
- boolean **hasUser**(String name)
  Onko järjestelmässä käyttäjää nimeltä *name*.
- synchronized void **saveUser**(User user)
  Tallennetaan käyttäjän *user* tiedot kantaan, mikäli käyttäjän tiedot ovat muuttuneet edellisen tallennuksen jälkeen.
- protected String[] **getUserNames**()
  Palautetaan järjestelmän käyttäjien nimet.
- static void **main**(String[] args)
  Järjestelmän käynnistäminen.

5.1.17 Group

Järjestelmän ylläpitäjä voi muodostaa käyttäjäryhmiä, joita kuvaa Group-luokka. Käyttäjäryhmien toiminta on useilta osin sama kuin varsinaisilla käyttäjillä, mutta ryhmillä ei ole näkymiä eikä kontaktilistoja. Ryhmälle tullut viesti käsitellään samoin kuin tavallisetkin viestit, mutta ProfileHandler ei annakaan viestiä EventMonitorille, vaan siitä luodaan kopio jokaista ryhmän jäsentä varten, mitkä käsitellään uudestaan jokaisen ryhmän jäsenen kontekstimallien ja sääntöjen avulla.

Määrittely :

class Group extends User implements XMLContentInterface

Konstruktorit :

- protected **Group**(String name, String password, String[] userNames)

Uuden ryhmän *name* luominen salasanalla *password* ja käyttäjillä *userNames*.

Metodit :

- ContactList **getContactList**()
  Group-oliot eivät sisällä kontaktilistoja, joten tämä metodi palauttaa aina null-arvon.
- String[] **getUserNames**()
  Palautetaan merkkijonotaulukkona tieto kaikista ryhmään kuuluvista käyttäjistä.
- void **addUser**(String name)
  Uuden käyttäjän *name* lisääminen.
- boolean **hasUser**(String name)
  Kuuluuko ryhmään käyttäjä *name*.
- void **removeUser**(String name)
  Käyttäjän *name* poistaminen ryhmästä.
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

5.1.18 GroupManager

GroupManager hallinnoi ryhmäolioita ja huolehtii niiden kantaantallennuksesta.

Määrittely :

class GroupManager

Metodit :

- static GroupManager **getInstance**()
  Palauttaa järjestelmän ainoan GroupManager olion.
- protected synchronized Group **newGroup**(Group grp)
  Uuden ryhmän lisääminen järjestelmään.
- synchronized Group **getGroup**(String name, String password)
  Palautetaan ryhmä *name*, jos salasana *password* on oikein. Muuten palautetaan null-arvo.
- synchronized boolean **hasGroup**(String name)
  Onko järjestelmässä ryhmää nimeltä *name*.
- synchronized void **saveGroup**(Group grp)
  Tallennetaan ryhmä *grp*, jos sen tiedot ovat muuttuneet edellisen tallennuksen jälkeen.
- protected synchronized void **saveAllGroups**()
  Kutsutaan saveGroup-metodia kaikille ryhmille.
- protected synchronized void **loadGroups**()
  Ladataan ryhmät järjestelmään.

5.1.19 Profile

Profile-oliolla kuvataan järjestelmässä käyttäjään liitettyä profiilia. Se sisältää tiedon siihen liitettyjen kontekstimallien kynnysarvoista, ajastetuista säännöistä ja varsinaisista viestin hallintaan liittyvistä säännöistä.

Määrittely :

class Profile implements XMLContentInterface

Konstruktorit :

- **Profile**(User user, String name)
  Luodaan käyttäjälle *user* uusi profiili nimeltä *name*.

Metodit :

- boolean **isActive**()

  Onko profiili aktiivinen.

- void **setActive**(boolean active)

  Asetetaan profiilin aktiivisuus parametrillä *active*.

- Rule **getRule**(String name)

  Palautetaan sääntö nimen *name* perusteella.

- void **setRule**(String name, Rule rule)

  Lisätään sääntö *rule* nimen *name* perusteella.

- void **deleteRule**(String name)

  Poistetaan sääntö nimeltä *name*.

- String[] **getRuleNames**()

  Palautetaan kaikkien profiilin sääntöjen nimet.

- void **addScheduledRule**(Scheduler rule)

  Lisätään ajastettu sääntö *rule*.

- void **removeAllScheduledRules**()

  Poistetaan kaikki ajastetut säännöt.

- List **getAllScheduledRules**()

  Palautetaan kaikki ajastetut säännöt listana.

- void **setContextModelValue**(String name, int value)

  Lisätään profiiliin uusi kontekstimallin kynnysarvo.

- void **removeContextModel**(String name)

  Poistetaan profiilista viittaus kontekstimalliin.

- Set **getContextModels**()

  Palautetaan profiilin viittaamat kontekstimallit.

- void **save**()

  Tallennetaan profiili. Tämä kutsuu User-olion save-metodia.

- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34.

5.1.20 Rule

Rule-luokalla kuvataan järjestelmässä käytettäviä sääntöjä, jotka ovat muodoltaan ehdot-toiminnot -pareja.

Määrittely :

class Rule implements XMLContentInterface

Konstruktorit :

- **Rule**(String name)
  Luodaan uusi sääntö, jonka nimi on *name*.

Kentät :

- static final int **EVENT_ADD**
  Uuden toiminnan lisäys.
- static final int **EVENT_REPLACE**
  Toiminnan korvaus.
- static final int **EVENT_DELETE**
  Toiminnan poisto.

Metodit :

- String **getName**()
  Palautetaan säännön nimi.
- void **addCondition**(String attributeName, String value, String expressionType)

Lisätään säännön ehtoon vertailu *attributeName*-attribuutin ja arvon *value* välillä *expressionType*:n mukaisesti esimerkiksi *Sender* = ”Ville Mäkelä, Maija Mäkelä”. *AttributeName* viittaa suoraan loogisiin attribuuttien nimiin .

- void **addContactListCondition**(String attributeName, String value, String expressionType)
  Lisätään sääntöön vertailu *attributeName*-attribuutin ja kontaktilistaviittauksen *value* välillä *expressionType*:n mukaisesti. Eli esimerkiksi addContactListCondition(”sender”, ”/”, ”in”) kysyy löytyykö käyttäjän kontaktilistalta attribuutin ”sender”-määrittelemä henkilö. (Tämä prototyyppi ei tule tukemaan muun tyyppisiä kysymyksiä kontaktilistaan liittyen, eli vain *attributeName* -parametriä voidaan muuttaa.)
- void **nextCondition**(String conditionalOperator)
  Jatketaan ilmentymän sisältämää ehtoa *conditionalOperator*:n mukaisesti.
- void **consequence**(int eventType, String eventname, String eventvalue)
  Säännön tyyppi on *eventType* ja toiminta määritellään avain *eventname* - arvo *eventvalue* pareina.
- void **consequence**(int eventType, String eventname, String oldvalue, String newvalue)
  Säännön tyyppi on *eventType* ja toiminta määritellään avain *eventname* - arvo *eventvalue* pareina, Mahdollinen vanha arvo *oldvalue* korvataan uudella.
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34.

5.1.21 Contact

Contact-luokka kuvaa yhtä kontaktilistassa olevaa tietoalkiota. Rakenteeltaan Contact on nimetty hajautustaulu, eli sillä on nimi ja muut tiedot esitetään avain-arvo -pareina.

Määrittely :

class Contact implements XMLContentInterface

Konstruktorit :

- **Contact**(String name)
  Luodaan uusi *name*-niminen Contact-olio.

Metodit :

- String **getName**()
  Palautetaan Contact-olion nimi.
- String **get**(String key)
  Palautetaan avaimeen *key* liitetty arvo.
- void **set**(String key, String value)
  Lisätään uusi tieto avaimella *key* ja arvolla *value*. Korvataan mahdollinen vanha arvo.
- Set **getKeys**()
  Palautetaan Contact-olion kaikkien avain-arvo -parien avaimet.
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

5.1.22 ContactList

Jokaisella järjestelmän käyttäjällä on yksi kontaktilista (ContactList), joka voi sisältää sekä varsinaisia kontaktitietoja (Contact) että toisia kontaktilistoja.

Määrittely :

class ContactList implements XMLContentInterface

Konstruktorit :

- **ContactList**(String name) Luodaan uusi kontaktilista.

Metodit :

- java.util.Set **getContactNames**()
  Palautetaan kaikkien ContactList-olion sisältämien Contact-olioiden nimet siten, että myös olion sisältämien muiden ContactList-olioiden getContactNames-metodia kutsutaan ja palautetaan niistä saadut arvot.
- Contact **getContact**(String name)
  Etsitään nimellä *name* kontakti tietoa. Myös alikontaktilistojen tiedot tutkitaan
- java.util.Set **getContactListNames**()
  Palautetaan kontaktilistan alikontaktilistat.
- ContactList **getContactList**(String name)
  Etsitään ContactList-olio nimen *name* perusteella.
- void **addContact**(Contact contact)
  Lisätään uusi kontaktitieto *contact*.
- void **addNewContactList**(ContactList clist)
  Lisätään uusi alikontaktilista *clist*
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.

- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

5.1.23 ScheduledRule

ScheduledRule muuttaa Rule-luokan määrityksiä siten, että sääntöjä voidaan käyttää myös ajastetuissa säännöissä. Tämä tarkoittaa erityisesti Droolsin käyttämän Message-olioviittauksen muuttamista User-tyyppiseksi.

Määrittely :

class ScheduledRule extends Rule

Metodit :

- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

5.1.24 ScheduleManager

ScheduleManager käy säännöllisin väliajoin läpi käyttäjien aktiivisten profiilien ajastetut säännöt ja mikäli jokin sääntö tulee laukaista, käytetään hyväksi Drools-Handlerin palveluita. Tämän jälkeen sääntöjen käsittelyssä syntynyt ActionEvent annetaan EventMonitorille, jolta EventManager sen myöhemmin pyytää ja suorittaa halutun toiminnon.

Määrittely :

class ScheduleManager extends Thread

Metodit :

- static ScheduleManager **getInstance**()
  Palautetaan järjestelmän ainoa ScheduleHandler-olio.

- void **run**()
  Säikeen toiminnallisuus on run-metodisssa.

5.1.25 Scheduler

Scheduler kuvaa aikaväliä tai tarkkaa aikaa, jolloin jonkin toiminnon pitäisi tapahtua.

Määrittely :

class Scheduler implements XMLContentInterface

Konstruktorit :

- **Scheduler**(Profile p, int type, int multiply)
  Luodaan uusi profiiliin *p* liitettävä ajastus tyypin *type* ja kertoimen *multiply* avulla.
- **Scheduler**(Profile p, Date date)
  Luodaan uusi profiiliin *p* liitettävä kerran suoritettava ajatus päivämääräolion *date* avulla.

Kentät :

- static final int **MINUTE**
  Tyyppi: minuutti.
- static final int **HOUR**
  Tyyppi: tunti.
- static final int **DAY**
  Tyyppi: vuorokausi.
- static final int **WEEK**
  Tyyppi: viikko.
- static final int **MONTH**
  Tyyppi: kuukausi.

Metodit :

- boolean **exact**()
  Onko ajastus luotu Date-olion avulla, jolloin tiedetään tarkasti milloin ajastuksen tulisi laueta.
- Date **getDate**()
  Palautetaan date-olio.
- long **getInterval**()
  Palautetaan ajastuksen laukeamisväli sekunneissa.
- void **addRule**(String name, Rule rule)
  Lisätään ajastukseen uusi sääntö *rule* nimeltään *name*.
- void **deleteRule**(String name)
  Poistetaan sääntö nimeltä *name*.
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

5.1.26 XMLUtil

XMLUtil-luokka tarjoaa XML:n käsittelyä helpottavia staattisia metodeja.

Määrittely :

class XMLUtil

Kentät :

- static final String **NS_DROOLS**
  Droolsin nimiavaruus.
- static final String **NS_DROOLS_JAVA**
  Droolsin vaatima Javan nimiavaruus.

Metodit :

- static Document **newXMLDocument**()
  Luodaan uusi w3c.org.dom.Document -tyyppinen olio.
- static Document **newDroolsDocument**()
  Luodaan uusi w3c.org.dom.Document -tyyppinen olio, joka pitää sisällään Droolsin tarvitsemia XML-elementtejä.

5.1.27 Log

Log-luokkaa käytetään järjestelmän lokitietojen tallennukseen.

Määrittely :

class Log

Kentät :

- static PrintStream **out**
  Järjestelmän oletustulostusvirta.
- static PrintStream **err**
  Järjestelmän virhetulostusvirta.

Metodit :

- static PrintStream **output**(String name)
  Palautetaan viittaus *name*-nimiseen tulostusvirtaan, jonka avulla tiedot tallennetaan tulostusvirran nimiseen tiedostoon.
- static synchronized void **flush**() Kutsutaan java.io.PrintStream-luokan flush-metodia kaikille järjestelmän loki-tulostusvirroille.

5.2 Rajapinnat

5.2.1 ActionInterface

ActionInterface:lla määritetään ne toiminnallisuudet, jotka jokaisen järjestelmään toteutettavan toiminnallisuusluokan on tarjottava. Toteutusvaiheessa tällaisia luokkia tullaan tekemään esimerkiksi viestien lähettämistä ja hakua varten.

Määrittely :

interface ActionEvent extends XMLContentInterface

Metodit :

- void **setValue**(User user, Object ob, String[] values)
  Asetetaan toiminnan suorittamiseksi arvoja, missä toiminta liitetään käyttäjään *user*, käyttäjän johonkin olioon *ob* (esimerkiksi Message) sekä arvotaulukon *values* sisältämien arvojen avulla. *Values*-arvotaulukon sisältö riippuu kutsutun toteutus luokan toiminnallisuuden ohjaamiseksi tarvittavista tiedoista, jotka voivat olla esimerkiksi joukko tietokantakyselyjä.
- void **doAction**(String action)
  Metodi laukaisee ilmentymän toiminnan, se mitä luokan tarjoamista toiminnalisuuksista annetuilla arvoilla halutaan laukaista määritetään *action*-parametrilla.
- String[] **supportedActions**()
  String-tyyppinen taulukko toiminnallisuuksista, joiden toteutuksen luokka tarjoaa. Taulukon sisältämät määritykset ovat *doAction*-metodin hyväksymiä syötteitä.

5.2.2 XMLContentInterface

XMLContentInterface-rajapinnan toteuttavien luokkien on osattava luoda omasta tietosisällöstään XML-kuvaus, jonka avulla ilmentymät voidaan myöhemmin pa-

lauttaa samaan tilaan kuin missä ne olivat kuvauksen luonnin yhteydessä.

Määrittely :

interface XMLContentInterface

Metodit :

- void **createXML**(Element parent)
  Parametrinä *parent* saadaan se XML-elementti, johon olion pitäisi liittää oma XML-kuvauksensa org.w3c.dom.Node -rajapinnan appendChild() -metodilla (lapsielementiksi).
- boolean **loadXML**(Element myContent)
  Oliota kutsuttaessa loadXML-metodilla, muuttaa se tilansa vastaamaan parametrinä saatua tietosisältöä, jonka kuvaus löytyy *myContent*-parametrista.

5.3 Attribuuttiluokat ja -rajapinnat

pakkaus converge.attribute

5.3.1 UserPropertyInterface

UserPropertyInterface:lla määritetään ne ominaisuudet, jotka jokaisen käyttäjäolioon liitettävien ominaisuuksien toteutusluokkien on tarjottava. Tällaisia ominaisuuksia ovat kaikki käyttäjän ulkomaailman kontekstia kuvaavat tiedot esimerkiksi käyttäjän paikkatieto (”paikannus”=”Helsinki,Kallio,Toinen Linja 10”) tai päätelaitteen saavutettavuus (”GSM-OnLine”=”false”).

Määrittely :

interface UserPropertyInterface

Metodit :

- boolean**compare**(String expressionType, Object value)
  Metodi jonka avulla voidaan vertailla annettavaa arvoa ilmentymän sisältämään arvotietosisältöön. Laskenta suoritetaan, ”Ilmentymän arvo” *expressionType value* esim. 100 < value. Epäselvän laskusäännön tapauksessa palautetaan*false*.
- void **setUser**(User user)
  Asettaa ilmentymään liitetyn käyttäjäolion
- void **setValue**(Object value)
  Ilmentymän arvotietosisältö alustetaan antamalla *value*.
- Object **getValue**(User user)
  Ilmentymän arvotietosisältö palautetaan Object-tyyppisenä oliona jonka todellinen tyyppi riippuu kutsutun ilmentymän toteutuksesta eli mitä ja miten luokka käsittelee omaa arvotietosisältöään.
- String[] **supportedAttributes**()
  String-tyyppinen taulukko niiden kaikkien attribuuttien loogisista nimistä joille toteutettu luokka tarjoaa käsittelyn toteutuksen.

5.3.2 AttributeInterface

AttributeInterface:lla määritetään ne ominaisuudet, jotka jokaisen attribuuttien käsittelyn toteuttavien luokkien on tarjottava.

Määrittely :

interface AttributeInterface extends XMLContentInterface

Metodit :

- void **setName**(String name)
  Asetetaan attribuutin nimi.
- String **getName**()
  Palauttaa ilmentymälle annetun nimen, joka vastaa loogista attribuutin

nimeä, jolle ilmentymä tarjoaa toteutuksen.

- void **setValue**(Object value)
  Ilmentymän arvotietosisältö alustetaan antamalla *value*. Toteutus itse käsittelee Object-tyypiä kuten se sitä tarvitsee.
- String[] **supportedAttributes**()
  Jokaisen attribuuttien käsittelyyn toteutetun luokan on pystyttävä kertomaan niiden kaikkien attribuuttien loogiset nimet, joiden käsittelyt se tarjoaa.

5.3.3 MessageAttributeInterface

MessageAttributeInterface:lla määritetään kaikki ne ominaisuudet, jotka viestiolioon liitettävien attribuuttiolioiden toteutusluokkien on tarjottava.

Määrittely :

interface MessageAttributeInterface extends AttributeInterface

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Metodi jonka avulla voidaan vertailla annettavaa arvoa ilmentymän sisältämään arvotietosisältöön. Laskenta suoritetaan, ”Ilmentymän arvo” *expressionType value* esim. 100 < value. Epäselvän laskusäännön tapauksessa palautetaan *false*.
- Object **getValue**()
  Ilmentymän arvotietosisältö palautetaan Object-tyyppisenä oliona jonka todellinen tyyppi riippuu kutsutun ilmentymän toteutuksesta eli mitä ja miten luokka käsittelee omaa arvotietosisältöään.

5.3.4 ContextModelAttributeInterface

ContextModelAttributeInterface:lla määritetään ne ominaisuudet, jotka kontekstimalliolioon liitettävien attribuuttiolioiden toteutusluokkien on tarjottava.

Määrittely :

interface ContextModelAttributeInterface extends AttributeInterface

Metodit :

- boolean **compare**(Message msg)
  Suorittaa vertailun ilmentymän oman arvotietosisällön ja *msg*-olioon liitetyn samannimisen attribuutin arvotietosisällön välillä käyttämällä ilmentymään asetettua laskusääntöä. Laskenta suoritetaan ”Ilmentymän arvo” laskusääntö msg:n attribuutin arvo esim. 100 < msg.attr.value.
- void **setExpressionType**(String type)
  Ilmentymän compare-metodin laskennassa käytettävä laskusäännön asetus. *Type*-parametriksi hyväksyttäviä arvoja ”<”, ”>”, ”>=”, ”<=”, ”=” ja ”in”, joista viimeinen tarkoittaa attribuutista riippuen joko alimerkkijonoa tai alkiota taulukossa.
- Object **getValue**()
  Ilmentymän sisältämä arvotietosisältö palautetaan Object-tyyppisenä, jonka todellinen tyyppi riippuu luokan toteutuksesta.
- String **getExpressionType**()
  Palauttaa ilmentymän vertalulaskennassa käytettävän laskusäännön.

5.3.5 AttributeLoader

AttributeLoader-luokalla ladataan (tarvittaessa dynaamisesti) AttributeInterface -rajapinnan toteuttavia luokkia järjestelmän käyttöön.

Määrittely :

class AttributeLoader

Metodit :

- static UserPropertyInterface **newUserPropertyInstance**(String name)
  Luodaan uusi ilmentymä luokasta, joka toteuttaa käyttäjäominaisuudelta vaadittavat ominaisuudet (UserPropertyInterface-rajapinta) ja tarjoaa toteutuksen *name*-nimiselle attribuutille.
- static ContextModelAttributeInterface
  newContextModelAttributeInstance(String name)
  Luodaan uusi ilmentymä luokasta, joka toteuttaa kontekstimalliattribuutilta vaadittavat ominaisuudet (ContextModelAttributeInterface-rajapinta) ja tarjoaa toteutuksen *name*-nimiselle attribuutille.
- static MessageAttributeInterface
  newMessageAttributeInstance(String name)
  Luodaan uusi ilmentymä luokasta, joka toteuttaa viestiolioattribuutilta vaadittavat ominaisuudet (MessageAttributeInterface-rajapinta) ja tarjoaa toteutuksen *name*-nimiselle attribuutille.
- static synchronized void **loadClasses**()
  Luokka-metodi, jonka suorituksessa käydään kaikki attribuuttien toteutuksia tarjoavat luokat läpi. Tällöin päivitettään AttributeLoaderin tietosisältö siitä, minkä nimisille attribuuteille luokat tarjoavat toteutuksen.

5.4 Rajapinta tiedonhakumoduulille

Ytimen tiedonhakumoduulille tarjoamat rajapinnat löytyvät seuraavista luokista:

- 5.1.2 **ContextModelHandler**, s. 10
- 5.1.15 **User**, s. 19
- 5.3.2 **AttributeInterface**, s. 36
- 5.3.3 **MessageAttributeInterface**, s. 37

5.5 Rajapinta asiakasmoduulille

Ytimen asiakasmoduulille tarjoamat rajapinnat löytyvät seuraavista luokista:

- 5.1.16 **UserManager**, s. 21
- 5.1.15 **User**, s. 19
- 5.1.18 **GroupManager**, s. 23
- 5.1.17 **Group**, s. 22
- 5.1.19 **Profile**, s. 24
- 5.1.1 **ContextModel**, s. 9
- 5.1.20 **Rule**, s. 26
- 5.1.25 **Scheduler**, s. 31
- 5.1.10 **ViewMonitor**, s. 15
- 5.1.9 **View**, s. 15
- 5.1.21 **Contact**, s. 28
- 5.1.22 **ContactList**, s. 29
- 5.1.27 **Log**, s. 33

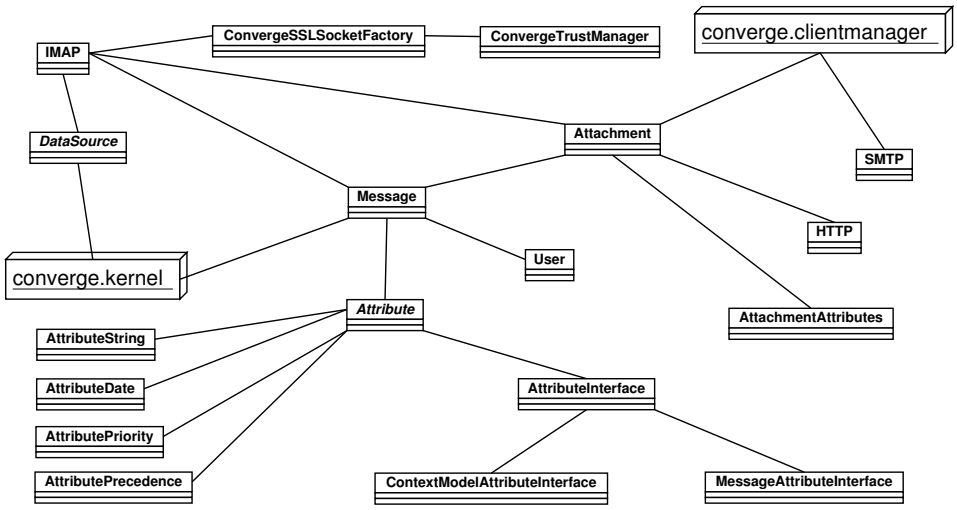
6 Tiedonhakumoduuli

Tiedonhakumoduuli huolehtii yhteyksistä muihin Internetin palvelimiin. Moduuli käyttää hyväkseen JavaMail- ja Java Secure Socket Extension -paketteja (katso http://java.sun.com/products/javamail/ ja http://java.sun.com/products/jsse/). Järjestelmän ydin kutsuu tiedonhakumoduulia kun käyttäjälle saapuneet uudet

viestit halutaan noutaa, ja viestit noudettuaan tiedonhakumoduuli muuttaa ne järjestelmän sisäiseen muotoon ja välittää ne järjestemän ytimelle jatkokäsittelyä varten.

6.1 Luokat

Pakkaus converge.service



Kuva 3: Tiedonhakumoduulin luokkakaavio

6.1.1 DataSource

DataSource on abstrakti yläluokka kaikille tiedonlähteille. Luokka tarjoaa aliluokilleen joitakin tiettyjä yleisiä metodeja, mutta suurin osa metodeista pitää tehdä tiedonlähdekohtaisesti. DataSource-luokasta ei tehdä koskaan ilmentymiä, ainoastaan sen aliluokista.

Määrittely :

public abstract class DataSource

implements converge.kernel.XMLContentInterface

Kentät :

- private String dataSourceName
  Tiedonlähteen luonnollinen nimi, esim. "Yliopiston IMAP".
- private String latestErrorMessage
  Viimeisin tiedonlähdettä käytettäessä tullut virheilmoitus.

Metodit :

- abstract void **getNewData**(User user)
  Tätä kutsutaan ytimestä kun halutaan hakea uudet viestit.
- String **getName**()
  Palauttaa tiedonlähteen luonnollisen nimen.
- void **setName**(String newName)
  Asettaa tiedonlähteen luonnollisen nimen.
- abstract String **getType**()
  Palauttaa tiedonlähteen tyypin, esim. "IMAP"
- boolean **isValid**()
  Palauttaa tiedon siitä, onko tiedonlähde kunnossa eli onko tietoa haettaessa tullut ongelmia.
- String **getErrorMessage**()
  Palauttaa viimeksi tulleen tiedonlähteen virheilmoituksen, null jos ei virheitä.
- abstract void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- abstract boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34.

6.1.2 IMAP

IMAP on DataSource-luokan aliluokka ja laajentaa DataSourcea toteuttamalla tuen IMAP-postilaatikoille.

Määrittely :

public class IMAP
extends DataSource

Konstruktorit :

- **IMAP**(String dataSourceName, String serverAddress, String serverLogin, String serverPassword)

Kentät :

- private String serverAddress
  Palvelimen osoite, esim. ml.mappi.helsinki.fi
- private String serverLogin
  Käyttäjänimi.
- private String serverPassword
  Salasana.
- private Date lastchecked
  Milloin postit on viimeksi tarkistettu.

Metodit :

- void **getNewData**(User user)
  Tätä metodia kutsutaan ytimestä kun halutaan hakea uudet postit.
- String **getType**()
  Palauttaa tiedonlähteen tyypin "IMAP"
- void **getAttachment**(String URL, PipedOutputStream pipe)
  Tätä metodia kutsutaan Attachment-luokasta kun haetaan liitetiedosto

IMAP-postilaatikosta. Metodi hakee liitetiedostot *URL* ja kirjoittaa sen putkeen *pipe*.

- AttachmentAttributes **getAttachmentAttributes**(String URL)
  Tätä metodia kutsutaan Attachment-luokasta kun halutaan tietää liitetiedoston *URL* ominaisuudet.
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34.

6.1.3 HTTP

HTTP hakee tiedostoja HTTP-protokollan avulla, käytetään ensisijaisesti liitetiedostojen hakemiseen.

Määrittely :

class HTTP

Metodit :

- void **getAttachment**(String URL, PipedOutputStream pipe)
  Tätä metodia kutsutaan Attachment-luokasta kun haetaan liitetiedosto WWW:stä. Metodi hakee liitetiedostot *URL* ja kirjoittaa sen putkeen *pipe*.
- AttachmentAttributes **getAttachmentAttributes**(String URL)
  Tätä metodia kutsutaan Attachment-luokasta kun halutaan tietää liitetiedoston *URL* ominaisuudet.

6.1.4 SMTP

SMTP hoitaa sähköpostiviestien lähettämisen.

Määrittely :

public class SMTP

Metodit :

- static void **sendMail**(String from, String to, String subject, String message)
  Lähettää sähköpostiviestin annetuilla parametreilla.

6.1.5 Message

Message sisältää viestin järjestelmän sisäisessä muodossa.

Määrittely :

public class Message
implements converge.kernel.XMLContentInterface

Konstruktorit :

- public **Message**(User user, Document doc)
  Muodostaa uuden Message-olion XML-dokumentista *doc* ja asettaa käyttäjäksi *user*.
- protected **Message**(User user)
  Luo tyhjän Message-olion, luo uuden attribuutin Converge_msgid ja asettaa käyttäjäksi *user*.

Kentät :

- private User user
  Järjestelmän käyttäjä, jolle tämä viesti on tullut.
- private Attribute[] attr
  Viestiattribuuttien taulukko.
- private Attachment[] attachments
  Tiedot viestiin liitetyistä liitetiedostoista.

Metodit :

- User **getUser**()
  Palauttaa viestin sisältämän käyttäjäolion.
- MessageAttributeInterface **getMessageAttribute**(String name)
  Palauttaa *name*-nimisen viestiattribuutin.
- void **setMessageAttribute**(String name, Object value, boolean permanent)
  Asettaa *name*-nimisen viestiattribuutin arvoksi olion *value*, attribuutti tallennetaan myös kantaan jos *permanent* on true.
- Attachment[] **getAttachmentList**()
  Palauttaa listan viestin mukana tulleista liitetiedostoista
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34.

6.1.6 Email

Email on luokka, joka osaa muuntaa saamansa Internetin sähköpostiviestin järjestelmän sisäiseen muotoon.

Määrittely :

public class Email

Metodit :

- static Message **Email**(User user, MimeMessage mm)
  Muodostaa uuden Message-olion JavaMail-oliosta *mm* ja asettaa käyttäjäksi *user*. Palauttaa null jos viestin muodostaminen ei onnistu.

6.1.7 Attribute

Attribute on abstrakti yliluokka muille Attribute-sarjan luokille. Yksi Attribute-sarjan aliluokan ilmentymä pitää sisällään yhden viestin attribuutin (lähettäjä, otsikko, lähetysaika, tärkeys, viestityyppi). Erityyppisiä attribuutteja varten luodaan omia aliluokkia Attributesta, esim. merkkijonoja varten on luokka AttributeString.

Määrittely :

public abstract class Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Kentät :

- private Object value
  Viestiattribuutin tieto.
- private String name
  Viestiattribuutin nimi.
- private String expressionType
  Vertailutyyppi, katso 5.3.4 sivulla 38

Metodit :

- abstract boolean **compare**(String expressionType, Object value)
  Katso 5.3.3 sivulla 37
- abstract boolean **compare**(Message msg)
  Katso 5.3.4 sivulla 38
- Object **getValue**()
  Katso 5.3.3 sivulla 37
- void **setValue**(Object value)
  Katso 5.3.2 sivulla 36

- void **setExpressionType**(String expressionType)
  Katso 5.3.4 sivulla 38
- String **getExpressionType**()
  Katso 5.3.4 sivulla 38
- String **getName**()
  Katso 5.3.2 sivulla 36
- void **setName**(String name)
  Katso 5.3.2 sivulla 36
- abstract String[] **supportedAttributes**()
  Katso 5.3.2 sivulla 36
- abstract void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- abstract boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

6.1.8 AttributeString

Attribute-luokan aliluokka, joka osaa käsitellä merkkijonoja.

Määrittely :

public class AttributeString
extends Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributeString**(String name, String value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 5.3.3 sivulla 37
- boolean **compare**(Message msg)
  Katso 5.3.4 sivulla 38
- String[] **supportedAttributes**()
  Katso 5.3.2 sivulla 36
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

6.1.9 AttributeDate

Attribute-luokan aliluokka, joka osaa käsitellä päivämääriä (esim. viestin lähetysaika).

Määrittely :

public class AttributeDate
extends Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributeDate**(String name, Date value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 5.3.3 sivulla 37
- boolean **compare**(Message msg)
  Katso 5.3.4 sivulla 38

- String[] **supportedAttributes**()
  Katso 5.3.2 sivulla 36
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

6.1.10 AttributePriority

Attribute-luokan aliluokka, joka osaa käsitellä viestin mukana tullutta Priority-arvoa (normal, high, low).

Määrittely :

public class AttributePriority
extends Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributePriority**(String name, String value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 5.3.3 sivulla 37
- boolean **compare**(Message msg)
  Katso 5.3.4 sivulla 38
- String[] **supportedAttributes**()
  Katso 5.3.2 sivulla 36
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.

- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

6.1.11 AttributePrecedence

Attribute-luokan aliluokka, joka osaa käsitellä viestin mukana tullutta Precedence-arvoa (bulk, normal, priority ym.)

Määrittely :

public class AttributePrecedence
extends Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributePrecedence**(String name, String value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 5.3.3 sivulla 37
- boolean **compare**(Message msg)
  Katso 5.3.4 sivulla 38
- String[] **supportedAttributes**()
  Katso 5.3.2 sivulla 36
- void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

6.1.12 Attachment

Kaikista liitetiedostoista tallennetaan tietokantaan ainoastaan liitetiedoston osoite. Jos viestin mukana tulee perinteinen liitetiedosto, se korvataan pointterilla IMAP-postilaatikkoon, jolloin URLiksi tulee esim. imap://kayttaja@palvelin/msgid/liitenro.

Jos viestin liitetiedostona tulee määrätynmuotoinen osoitteen sisältämä tekstitiedosto, asetetaan viestin liitetiedoston osoitteeksi saapunut osoite. Tekstitiedosto noudattaa mm. Internet Explorerin käyttämää url-tiedostomuotoa.

```
[InternetShortcut]
URL=http://miuku.net/
Modified=10DA2746B38FC2015C
```

Määrittely :

public class Attachment
extends Thread
implements converge.kernel.XMLContentInterface

Konstruktorit :

- public **Attachment**(String URL)
  Muodostaa uuden *Attachment*-olion, jonka liitetiedosto on *URL*.

Kentät :

- private AttachmentAttributes attributes
  Viestin liitetiedoston ominaisuudet. Arvo on null kunnes ominaisuuksia kysytään, tätä tietoa ei tallenneta kantaan.
- private String URL
  Liitetiedoston osoite.

Metodit :

- public PipedOutputStream **getAttachment**(User user)
  Käynnistää uuden säikeen joka lukee liitetiedoston palauttamaansa *PipedOutputStream*iin.
- public AttachmentAttributes **getAttachmentAttributes**(User user)
  Palauttaa liitetiedoston ominaisuudet *AttachmentAttributes* -oliona. Jos ominaisuudet eivät ole tiedossa, ne käydään katsomassa liitetiedoston palvelimelta.
- public boolean **refreshAttachmentAttributes**(User user)
  Tarkistaa ovatko liitetiedoston tiedot muuttuneet ja päivittää attributes-arvot.
- public String **getPointerAddress**()
  Palauttaa liitetiedoston osoitteen.
- public void **createXML**(Element parent)
  Katso 5.2.2 sivulla 34.
- public boolean **loadXML**(Element myContent)
  Katso 5.2.2 sivulla 34

6.1.13 AttachmentAttributes

Tietovarasto liitetiedoston ominaisuuksille.

Määrittely :

public class AttachmentAttributes

Konstruktorit :

- public **AttachmentAttributes**()

Kentät :

- private available boolean
  Onko liitetiedosto saatavilla (jos esim. palvelinongelmia).

- private long size

  Liitetiedoston koko, -1 jos tuntematon.

- private Date date

  Liitetiedoston viimeisimmän muokkauksen päivämäärä.

- private String MIMEtype

  Liitetiedoston MIME-tyyppi.

- private String description

  Liitetiedoston sanallinen kuvaus.

- private String filename

  Liitetiedoston tiedostonimi.

Metodit :

- public boolean **getAvailable**()

  Palauttaa liitetiedoston saatavuustiedon.

- public void **setAvailable**(boolean available)

  Asettaa liitetiedoston saatavuustiedon.

- public long **getSize**()

  Palauttaa liitetiedoston koon.

- public void **setSize**(long size)

  Asettaa liitetiedoston koon.

- public Date **getDate**()

  Palauttaa liitetiedoston viimeisimmän muokkauspäivämäärän.

- public void **setDate**(Date date)

  Asettaa liitetiedoston viimeisimmän muokkauspäivämäärän.

- public String **getMIMEType**()

  Palauttaa liitetiedoston MIME-tyypin.

- public void **setMIMEType**(String MIMEtype)

  Asettaa liitetiedoston MIME-tyypin.

- public String **getDescription**()
  Palauttaa liitetiedoston sanallisen kuvauksen.
- public void **setDescription**(String description)
  Asettaa liitetiedoston sanallisen kuvauksen.
- public String **getFilename**()
  Palauttaa liitetiedoston tiedostonimen.
- public void **setFilename**(String filename)
  Asettaa liitetiedoston tiedostonimen.

6.1.14 ConvergeSSLSocketFactory

Käytetään SSL-IMAP -yhteyksissä.

Määrittely :

class ConvergeSSLSocketFactory

Konstruktorit :

- static **ConvergeSSLSocketFactory**()

Metodit :

- SocketFactory **getDefault**()
- Socket **createSocket**(Socket socket, String s, int i, boolean flag)
- Socket **createSocket**(InetAddress inaddr, int i, InetAddress inaddr1, int j)
- Socket **createSocket**(InetAddress inaddr, int i)
- Socket **createSocket**(String s, int i, InetAddress inaddr, int j)
- Socket **createSocket**(String s, int i)
- String[] **getDefaultCipherSuites**()
- String[] **getSupportedCipherSuites**()

6.1.15 ConvergeTrustManager

Käytetään SSL-IMAP -yhteyksissä.

Määrittely :

class ConvergeTrustManager implements X509TrustManager

Metodit :

- boolean **isClientTrusted**(X509Certificate[] cert)
- boolean **isServerTrusted**(X509Certificate[] cert)
- X509Certificate[] **getAcceptedIssuers**()

6.2 Rajapinta järjestelmän ytimelle

Tiedonhakumoduulin järjestelmän ytimelle tarjoamat rajapinnat löytyvät seuraavista luokista:

- 6.1.5 **Message**, s. 45
- 6.1.12 **Attachment**, s. 52
- 6.1.1 **DataSource**, s. 41

6.3 Rajapinta asiakasmoduulille

Tiedonhakumoduulin asiakasmoduulille tarjoamat rajapinnat löytyvät seuraavista luokista:

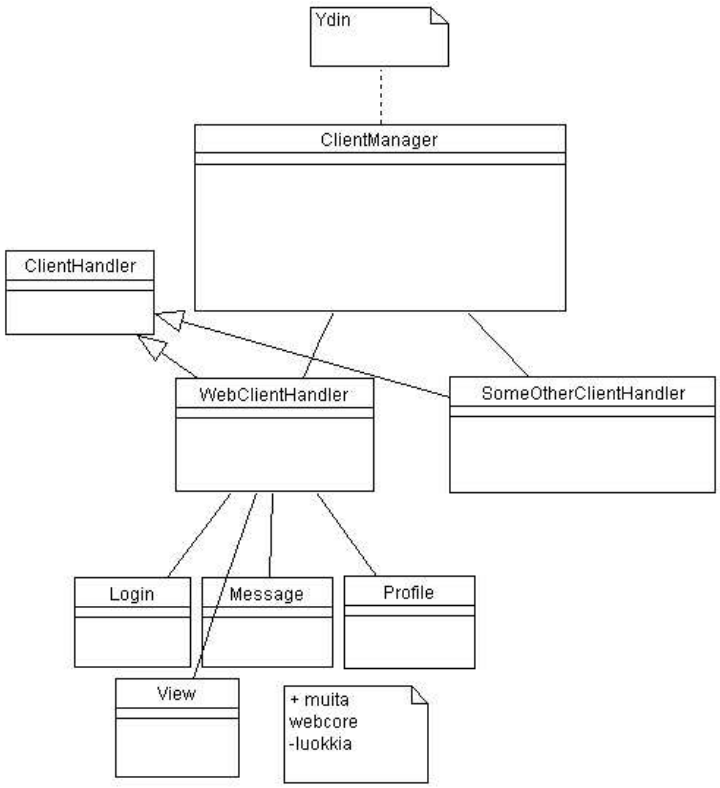
- 6.1.5 **Message**, s. 45
- 6.1.12 **Attachment**, s. 52
- 6.1.4 **SMTP**, s. 44

7 Asiakasmoduuli

Asiakasmoduuli toimii välittäjänä järjestelmän ytimen ja erilaisten asiakasohjelmien tai päätelaitteiden välillä. Jokaista tuettua asiakasohjelmatyyppiä kohden on asiakasmoduulissa oma komponentti, joka osaa kaikki tarvittavat protokollat joiden avulla kyseisen ohjelman tai päätelaitteen kanssa voidaan kommunikoida.

Ohjelmointirajapinta jota vasten uudentyyppisiä asiakasohjelmaluokkia voidaan rakentaa, koostuu ClientHandler -yläluokasta ja ClientManager-luokan tarjoamista metodeista.

Huomautus: asiakasmoduulin ne osat jotka toteuttavat tämän prototyypin ainoan asiakasohjelmatyypin, WWW-asiakasohjelman, on kuvattu seuraavassa luvussa.



Kuva 4: Asiakasmoduuli

7.1 Luokat

Pakkaus converge.clientmanager

7.1.1 ClientManager

Järjestelmässä on yksi ClientManager-ilmentymä, jonka tehtävä (tässä prototyypissä) on mm. pitää kirjaa aktiivisista käyttäjistä, hoitaa sessionhallintaa ja välittää User-olioita asiakasohjelmaluokille.

Ytimelle tarjottavat palvelut

- void **sendMessage**(Message message)
  Parametrina saatu *message* lähetetään sille päätelaitteelle jota vastaanottaja sillä hetkellä käyttää. Päätelaitteiden / asiakasohjelmien erilaisista luonteista johtuen tämän metodin kutsuminen ei takaa että viesti menee perille. (Vastaanottaja tunnetaan, sillä Message-olio sisältää viitteen User-olioon.)

- void **start**()
  Aiheuttaa erilaisten asiakasrajapintojen käynnistymisen (toteutettavassa prototyypissä koskee HTTP-palvelinta)

Erityyppisille asiakasohjelmaluokille tarjottavat palvelut

- User **getUserBySessionId**(long sessionId)
  Palauttaa annettua *sessionId*:tä vastaavan User-olion. Jos *sessionId* ei ole validi (esimerkiksi istunto on vanhentunut), metodi palauttaa null-arvon.

- long **login**(String username, String password)
  Tätä metodia kutsutaan sisäänkirjautumisen yhteydessä. Jos annetut *username* ja *password* ovat validit, palautetaan long-tyyppiä oleva session-id. Tämä metodi toteutetaan niin, että se kutsuu ytimen UserManagerin palveluita ja

(jos tunnus ja salasana ovat oikein) saa sieltä User-olion, jota vastaamaan se luo uuden session-id:n.

- void **logout**(long sessionId)

  Asiakasohjelmaluokat kutsuvat tätä metodia, jos käyttäjä haluaa eksplisiittisesti kirjautua ulos palvelusta. Parametrina annettu *sessionId* ja sitä vastaava User-olio poistetaan ClientManagerin sisäisestä tietorakeenteesta. (Eli jos samalla *sessionId*:llä kutsuttaisiin tämän jälkeen getUserBySessionId-metodia, palauttaisi se arvon null.)

Muut metodit

- static ClientManager **getInstance**()
  Palauttaa järjestelmän ainoan ClientManager-olion.

7.1.2 ClientHandler

ClientHandler on abstrakti yläluokka kaikentyyppisten asiakasohjelmien käsittelijöille. Siinä on määritelty joitakin metodeja ja muuttujia, jotka ovat yhteisiä kaikille aliluokille, mutta jokainen aliluokka totetuttaa omalla tavallaan.

Metodit

- void **start**()
  Käynnistää kyseiseen asiakasohjelmatyyppiin mahdollisesti liittyvän palvelimen, kuuntelijan tms.

- void **sendMessage**(Message msg)
  Viestin lähetys kyseisentyyppiseen asiakasohjelmaan / päätelaitteeseen

8 Asiakasohjelma

Järjestelmän prototyyppiä varten tehdään WWW:ssä toimiva asiakasohjelma. Tämä ohjelma sisältää varsin rajoitetun määrän toiminnallisuutta, eikä esimerkiksi käyttöliittymän suunnittelussa ole nähty siinä määrin vaivaa kuin ”oikean” asiakasohjelman kohdalla pitäisi tehdä. Sen avulla on kuitenkin mahdollista käyttää järjestelmän yleisimpiä toimintoja.

Asiakasmoduuliin kuuluu Javalla tehty palvelin, joka kuuntelee jotakin TCP/IP -porttia ja ymmärtää HTTP-protokollaa. Asiakasohjelmana toimii siis oikeastaan WWW-selain, jolla käyttäjä voi suoraan ottaa yhteyden järjestelmään. Tässä luvussa ei käsitellä varsinaista asiakasohjelmaa (WWW-selainta), vaan esitellään ne järjestelmän (asiakasmoduulin) osat, jotka mahdollistavat WWW-selaimen käyttämisen asiakasohjelmana.

HTTP-palvelimen toiminnallisuutta toteuteta itse, vaan prototyypissä käytetään valmista **webcore**[1]-nimistä open source Java-palvelinta. Kaikki dynaamista HTML:ää tuottavat luokat toteuttavat webcore-järjestelmään kuuluvan WebRequestable -rajapinnan. Webcoren toimintaa ei tässä dokumentissa kuitenkaan tarkemmin selitetä, sillä yksittäisen asiakasohjelmatyypin toteutus ei ole koko järjestelmän kannalta kovin oleellista.

8.1 Luokat

8.1.1 WebClientHandler

WebClientHandler on ClientHandler-luokan aliluokka. Tämä on prototyypin ainoa luokka joka mahdollistaa järjestelmän käytön jollain tietyllä päätelaiteella / asiakasohjelmalla, tässä tapauksessa WWW-selaimella.

Tässä toteutuksessa WebClientHandlerilla on vain muutama tehtävä; se huolehtii webcore-palvelimen käynnistyksestä ja lisäksi tarjoaa ClientManagerille (ja sitä

[1] Lisätietoja osoitteesta http://www.hunter-lovell.org/webcore/doc/overview.html

kautta ytimelle) alkeellisen toteutuksen viestin lähettämisestä. (WWW-asiakkaalle ei varsinaisesti voida suoraan lähettää viestiä, vaan kaikki toiminta tapahtuu asiakasohjelman aloitteesta. WebClientHandler tarjoaa tässä metodin, joka kertoo onko kyseisellä käyttäjälle uusia viestejä; HTML:ää tuottavat luokat voivat lisätä tämän tiedon selaimelle palautettavaan HTML-sivuun.)

ClientManagerille tarjottavat palvelut

- void **start**()
  Käynnistää webcore-palvelimen
- void **sendMessage**(Message msg)
  Toteuttaa eräänlaisen viestin lähettämisen, eli tässä tapauksessa kirjaa muistiin, että tietylle käyttäjälle on uusia viestejä.

Webcore-luokille tarjottavat palvelut

- static boolean **hasPendingMessages**(User user)
  Palauttaa true, jos annetulle käyttäjälle on uusia viestejä, muuten palauttaa false. Tämä tieto voidaan välittää käyttäjälle WWW-käyttöliittymän kautta.

Muut metodit

- static WebClientHandler **getInstance**()
  Palauttaa järjestelmän ainoan WebClientHandler-olion.

8.1.2 HTML:ää tuottavat webcore-luokat

Nämä luokat huolehtivat tietynlaisten sivujen luomisesta (generoivat siis HTML:ää) ja palauttamisesta varsinaiselle asiakasohjelmalle eli WWW-selaimelle. Muutenkin suurin osa asiakasmoduulin toiminnallisuudesta löytyy näistä luokista.

Sessionhallinta hoidetaan HTTP-”keksien” (cookie) ja yksilöllisten session-id -tunnusten avulla. Sisäänkirjautumisen jälkeen jokainen selaimelta tuleva HTTP-pyyntö sisältää session-id:n, ja pyynnön käsittevä webcore-luokka kutsuu ClientManagerin getUserBySessionId -metodia, joka palauttaa session-id:tä vastaavan User-olion. Tämän jälkeen kyseinen webcore-luokka voi suoraan käyttää User-oliota esimerkiksi listatakseen käyttäjän tietyn näkymän viestit, tai tehdäkseen käyttäjän pyytämiä muutoksia tiettyyn profiiliin tai kontekstimalliin.

Lista webcore-luokista

Login - sisään- ja uloskirjautumiseen liittyvät asiat, etusivu
Profile - profiilien luominen, muokkaaminen ja poistaminen
ContextModel - kontekstimallien luominen, muokkaaminen ja poistaminen
Main - sisäänkirjautumisen jälkeinen pääsivu, lista kaikista näkymistä
View - yhden näkymän sisältö
Message - viestin näyttäminen
Send - viestin kirjoittaminen ja lähetys
Settings - asetusten muokkaaminen (sisältää tässä prototyypissä myös ainoan tiedonlähteen, eli IMAP-palvelimen, asetukset)
ContactList - kontaktilistan muokkaaminen
HtmlUtils - tänne on koottu useissa luokissa tarvittavia metodeita, esimerkiksi sellaisen HTML:n tuottaminen, joka lisätään useille tai kaikille tuotettaville sivuille

Periaattessa kaiken HTML:n voisi generoida esimerkiksi vain yhdessä luokassa; tämä toiminnallisuus on jaettu useampaan luokkaan lähinnä selkeyden vuoksi.

8.2 Käyttöliittymä

8.2.1 Sisäänkirjautuminen

Jotta käyttäjä voi kirjautua sisään järjestelmään, tulee hänen olla kirjautunut järjestelmän käyttäjäksi.
Sisäänkirjautuminen tapahtuu menemällä järjestelmän aloitussivulle. Aloitussivulla käyttäjä syöttää olemassa olevan käyttäjätunnuksen Name-kenttään sekä käyttäjätunnusta vastaavan salasanan Password-kenttään. Jos käyttäjätunnus ja salasana ovat oikeat, käyttäjä siirtyy Converge-systeemin pääsivulle. Jos taas käyttäjätunnus tai sitä vastaava salasana on väärin, kehotetaan käyttäjää tarkastamaan käyttäjätunnuksensa ja syöttämään salasanansa uudelleen.

8.2.2 Uloskirjautuminen

Käyttäjä kirjautuu järjestelmästä ulos painamalla käyttöliittymän logout-linkkiä. Tämän jälkeen järjestelmä kirjaa käyttäjän ulos ja käyttäjälle avautuu palvelun aloitussivu.

8.2.3 Vastaanotettujen viestien käsittely

Jotta käyttäjä pääsee käsittelemään vastaanottamiaan viestejään, on hänen ensin kirjauduttava sisään järjestelmään.

Viestin lukeminen :
Käyttäjä pääsee lukemaan viestejään valitsemalla ensin näkymän, josta haluaa viestit lukea. Näkymä valitaan aloitussivun pudotusvalikosta (views). Tämän jälkeen painamalla Go to -nappulaa käyttäjälle avautuu näyttö, jossa kyseisen näkymän sisältämät viestit ovat. Näistä viesteistä käyttäjä voi valita viestin luettavaksi painamalla kyseistä viestin otsikkoa.

Viestin poistaminen :
Kun käyttäjä haluaa poistaa tietyn viestin tietystä näkymästä, menee hän ensin siihen näkymään, josta haluaa viestin poistaa. Tämän jälkeen käyttäjä painaa Delete-painiketta viestin otsikon vieressä, jolloin viite viestiin poistetaan tästä näkymästä.
Jos viestiin ei jää enää yhtään viitettä mihinkään näkymään, alkuperäinen viesti poistetaan lopullisesti.

8.2.4 Liitetiedostojen käsittely

Koska käyttäjälle toimitettavat viestit eivät sisällä itse liitetiedostoa, vaan pointterin siihen, tarjoaa järjestelmä käyttäjälle seuraavia keinoja liitetiedostojen käsittelyyn.

Liitetiedoston noutaminen :
Käyttäjä voi lukea viestin sisältämän liitetiedosta avaamalla sen viestin, joka sisältää halutun liitetiedoston. Tässä viestissä on linkki, jota painamalla kyseinen liitetiedosto noudetaan käyttäjälle sen alkuperäisestä sijainnista.

Liitetiedoston poistaminen :
Liitetiedostojen erillistä poistamista ei toteuteta, vaan pointterit liitetiedostoihin ovat olemassa niin kauan, kuin alkuperäinen, liitetiedoston sisältänyt viesti on olemassa.

8.2.5 Profiilien käsittely

Jotta käyttäjä pääsee muokkaamaan profiileitaan, on hänen ensin kirjauduttava sisään järjestelmään.

Profiilin luominen :
Käyttäjä luo uuden profiilin siirtymällä Converge Systemin pääsivulta Profiles-sivulle, josta painaa Add New Profile -painiketta. Tämän jälkeen käyttäjälle avautuu lomake, jossa hän voi valita jo olemassaolevia ja luoda uusia sääntöjä liitettäväksi luotavaan profiiliin tai määrittää, milloin profiili on aktiivinen. Lopuksi käyttäjä tallentaa uuden profiilinsa.

Profiilin muokkaaminen :
Käyttäjä muokkaa olemassa olevia profiilejaan siirtymällä Converge Systemin pääsivulta Profiles-sivulle ja valitsemalla sieltä pudotusvalikosta, profiilin, jota haluaa muokata. Tämän jälkeen käyttäjälle avautuu lomake, jossa ovat profiiliin liittyvät säännöt. Tämän jälkeen käyttäjä voi muokata tai poistaa sääntöjä, lisätä kokonaan uusia sääntöjä tai muuttaa aikaa, jolloin profiili on aktiivinen. Tehtyään haluamansa muutokset profiiliin, käyttäjä tallentaa profiilin.

Profiilin poistaminen :
Profiilien poistaminen tapahtuu siirtymällä Converge Systemin pääsivulta Profiles-sivulle, josta käyttäjä valitsee profiilin, jonka haluaa poistaa. Tämän jälkeen käyttäjä painaa Delete-painiketta, jolloin profiili poistetaan lopullisesti järjestelmästä. Käyttäjä voi poistaa vain olemassaolevia profiileita, jotka ovat hänen omiaan.

8.2.6 Kontekstimallien käsittely

Jotta käyttäjä pääsee käsittelemään kontekstimallejaan, on hänen ensin kirjauduttava sisään järjestelmään.

Kontekstimallin luominen :
Käyttäjä luo uuden kontekstimallin siirtymällä Converge Systemin pääsivulta Contextmodels-sivulle, josta painaa Add New Contextmodel -painiketta. Tämän jälkeen käyttäjälle avautuu lomake, jossa hän voi valita haluamiaan attribuutteja ku-

vaamaan uuttaa kontekstimalliaan. Valitessaan haluamansa attribuutin, käyttäjä määrittää sille arvon tai arvovälin, joka kuuluu luotavaan kontekstimalliin. Valittuaan tarvittavat attribuutit kuvaamaan kontekstimallia, käyttäjä tallentaa kontekstimallin.

Kontekstimalli muokkaaminen :
Kontekstimallin muokkaaminen tapahtuu valitsemalla Converge-systemin pääsivulta valikosta Contextmodels muokattavaksi haluttu kontekstimalli ja painamalla Edit-painiketta. Tämän jälkeen käyttäjälle avautuu vastaava lomake, kuin kontekstimallia luotaessa, mutta johon on valmiiksi täytetty käyttäjän kyseistä profiilia kuvaavat attribuutit. Käyttäjä voi muokata näiden attribuuttien arvoja tai arvovälejä sekä poistaa jo olemassaolevia tai lisätä uusia attribuutteja. Muokattuaan kontekstimallin mieleisekseen käyttäjä tallentaa kontekstimallin.

Kontekstimallin poistaminen :
Käyttäjä poistaa olemassa olevan kontekstimallin siirtymällä Converge Systemin pääsivulta Contextmodels-sivulle ja sieltä valitsemalla poistettavan kontekstimallin ja painamalla Delete-painiketta.

9 Muuta huomioitavaa

9.1 Koodin ulkoasu

Koodin nimeämisessä, kommentoinnissa sekä muissa ohjelmointiteknisissä asioissa noudatetaan pääsääntöisesti Catharina Candolinin *Java Style Guide* -dokumenttia[2].

Muutama erityishuomio:

Kieli Koodin kieli on englanti, kommentit kirjoitetaan kuitenkin suomeksi.

[2]http://www.cs.hut.fi/%7Ecandolin/java/styleguide.html

Sisennys Ei käytetä tabia. Sisennys tapahtuu 4:llä välilyönnillä.

Kommentointi Käytetään Javadoc-tyylistä kommentointia tavalliseen tapaan jokaisen luokan, metodin ja muuttujan kohdalla. Myös muita selventäviä kommentteja lisätään koodiin runsaasti. Java-kielen normaaleja ominaisuuksia *ei* kuitenkaan kommentoida:

```
i++;  // Lisätään i:hin 1
```

Varataan `/* ...*/` -kommentit koodin väliaikaista poistamista varten; varsinaiseen kommentointiin käytetään siis `//` -merkintää.

Aaltosulkeet Aaltosulkeita käytetään näin:

```
public void example(int i){
    ...
}
```

A AttributeInterface -rajapinnan toteutusesimerkkejä

A.1 ContextModelValueAttribute

ContextModelValueAttribute on attribuutin toteutusluokka, joka on tarkoitettu viestiolioon liitettäväksi (implements MessageAttributeInterface) ja sen toiminnallisuus on tarkoitettu käsittelemään kontekstimalliolion itselleen määrittelemää arvoa, joka kuvastaa prosentuaalisesti (kokonaislukuna 0-100) vertailun onnistumista.

Määrittely :

class ContextModelValueAttribute implements MessageAttributeInterface

Konstruktorit :

- **ContextModelValueAttribute**(int value)

Metodit :

- void **setName**(String name)
  Katso 5.3.2, sivulla 36.
- String **getName**()
  Katso 5.3.2, sivulla 36.
- void **setValue**(Object value)
  Katso 5.3.2, sivulla 36. Luokka hyväksyy syötteekseen (Integer) kokonaislukuarvoja väliltä 0 - 100, millä ilmaistaan kontekstimallin itselleen saamaa arvo prosenteina.
- String[] **supportedAttributes**()
  Katso 5.3.2, sivulla 36. Luokan toteutus on käyttökelpoinen vain kontekstimallien luodessa itsestää arvoilmauksen, ja koska käyttäjän kontekstimalleille antamia nimiä ei voida tietää etukäteen, palauttaa tämä metodi tyhjän String-taulukon.

- boolean **compare**(Object value, String expressionType)
  Katso 5.3.3, sivulla 37. *Value*-parametrina hyväksytään vain kokonaislukuarvoja (Integer) väliltä 0-100.
- Object **getValue**()
  Katso 5.3.3, sivulla 37. Palauttaa ilmentymän arvotietosisällön Integer-tyyppisenä kokonaislukuarvona.
- void **createXML**(Element parent)
  Katso 5.2.2, sivulla 34.
- boolean **loadXML**(Element myContent)
  Katso 5.2.2, sivulla 34.

A.2 AttributeStringArray

AttributeStringArray-luokka on toteutus luokka viestiolioon liitettävillä (implements MessageAttributeInterface) attribuuteille, joiden arvotietosisältö voi olla moniosainen tekstiarvo.

Määrittely :

class AttributeStringArray implements MessageAttributeInterface

Konstruktorit :

- **StringArrayImplementation**(String[] value)

Metodit :

- void **setName**(String name)
  Katso 5.3.2, sivulla 36.
- String **getName**()
  Katso 5.3.2, sivulla 36.
- void **setValue**(Object value)
  Katso 5.3.2, sivulla 36. Hyväksyttä syöte on String-tyyppinen lista.

- String[] **supportedAttributes**()

  Katso 5.3.2, sivulla 36.

- boolean **compare**(Object value, String expressionType)

  Katso 5.3.3, sivulla 37. *Value*-parametrina hyväksytään String-tyyppinen taulukko tai yksi String-tyyppinen olio.

- Object **getValue**()

  Katso 5.3.3, sivulla 37. Palauttaa ilmentymän tietosisällön String-tyyppisenä taulukkona.

- void **createXML**(Element parent)

  Katso 5.2.2, sivulla 34.

- boolean **loadXML**(Element myContent)

  Katso 5.2.2, sivulla 34.