Valsts reģionālās attīstības aģentūra

Valsts informācijas sistēmu savietotāja (VISS) un Vienotā valsts un pašvaldību pakalpojumu portāla www.latvija.lv pilnveidošana un uzturēšana

----

versija



Rīgā 2022

Dokumenta identifikācija

|  |  |
| --- | --- |
| Dokumenta ID: | -----V- |
| Dokumenta nosaukums: | Valsts informācijas sistēmu savietotāja (VISS) un Vienotā valsts un pašvaldību pakalpojumu portāla www.latvija.lv pilnveidošana un uzturēšana.  .  .  . |
| Dokumenta kods: | ---- |
| Versija: | Versija , Laidiens (saīsināti V ) |

Saskaņojumi

|  |  |  |  |
| --- | --- | --- | --- |
| Organizācija | Vārds, uzvārds, amats | Datums | Paraksts |
|  | Atbildīgā persona no Pasūtītāja puses |  |  |
| SIA "ABC software" | J.Korņijenko, projekta vadītājs par tehniskiem jautājumiem no Izpildītāja puses |  |  |
| SIA "ABC software" | A.Zeļikovičs, vecākais programmētājs |  |  |

Izmaiņu vēsture

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versija | Datums | Apraksts | Organizācija | Autors |
| 1.00 | 10.12.2012. | Izveidota dokumenta sākotnējā versija | SIA „ABC software” | A.Zeļikovičs, A.Pontags, J.Korņijenko,  M.Markovs |
| 1.01 | 25.02.2016 | Izņemta rekomendācija izmanto AlexFtps klientu. Mainīts piedāvājums uz System.Net.Ftps klientu:  Rediģēta: 1.1. Nodaļa, 3.2.2. sadaļa un 3.2.2.1. paragrāfs. | SIA „ABC software” | A.Zeļikovičs |
| 1.02 | 30.05.2017 | Aktualizēti .Net un Java koda piemēri:  Rediģēta: 3. un 4. Nodaļas, | SIA „ABC software” | V.Orlovs |
| 1.03 | 16.06.2017 | Labojums sadaļā 3.2.2. FTP piemēri.NET | SIA „ABC software” | V.Orlovs |
| 1.04 | 29.09.2017 | Labojumu sadaļa 5.1.1. atbilstoši jaunai DIT servisa versijai. | SIA “ABC software” | S. Degtjars |
| 1.05 | 11.11.2019 | Labojums sadaļā 3.2. Piemēri.NET | SIA „ABC software” | V.Orlovs |
| 1.06 | 02.12.2019 | Labojums sadaļā 3.2. Piemēri.NET un 4.2. Piemēri Java | SIA „ABC software” | V.Orlovs |
| 1.07 | 01.08.2022 | Labojums atbilstoši pieteikumam Nr.12648 | SIA „ABC software” | V.Orlovs |
| 1.08 | 02.08.2022 | Izņemtas lietotāju paroles | SIA „ABC software” | J.Korņijenko |

**Satura rādītājs**

[Attēlu saraksts 6](#_Toc110419674)

[1. Ievads 7](#_Toc110419675)

[1.1. Darbības sfēra 7](#_Toc110419676)

[1.2. Termini un pieņemtie apzīmējumi 7](#_Toc110419677)

[1.3. Saistītie dokumenti 7](#_Toc110419678)

[1.4. Dokumenta pārskats 7](#_Toc110419679)

[2. Vispārējais apraksts 8](#_Toc110419680)

[2.1. Biznesa ziņojums 9](#_Toc110419681)

[2.2. Transakcijas ziņojums 9](#_Toc110419682)

[2.3. Elektroniskā ziņojumu glabātava 9](#_Toc110419683)

[3. .Net 10](#_Toc110419684)

[3.1. Projekta izveidošana.Net vidē 10](#_Toc110419685)

[3.2. Piemēri .NET 10](#_Toc110419686)

[3.2.1. Tīkla servisu piemēri 11](#_Toc110419687)

[3.2.1.1. Savienošana ar sertifikātu (LoginCert) 11](#_Toc110419688)

[3.2.1.1.3. Koda piemērs (LoginCert) 14](#_Toc110419689)

[3.2.1.2. Savienošana ar lietotāja vārdu un paroli (LoginUserName) 15](#_Toc110419690)

[3.2.1.3. Kanāla pārbaude un izveidošana (SelectAndCreateChannel) 16](#_Toc110419691)

[3.2.1.4. Kanāla versijas pārbaude un izveidošana (SelectAndCreateChannelVersion) 19](#_Toc110419692)

[3.2.1.5. Ziņojuma nosūtīšana (SendMessageToChannelVersion) 23](#_Toc110419693)

[3.2.1.6. Ziņojumu dzēšana (MessageDelete) 27](#_Toc110419694)

[3.2.1.7. Tā, kas parakstījies pievienošana kanāla versijai (CreateSubscriberToChannelVersion) 29](#_Toc110419695)

[3.2.1.8. Ziņojuma nolasīšana (ReceiveMessage) 32](#_Toc110419696)

[3.2.1.9. Ziņojuma atjaunošana (Restore Messages) 34](#_Toc110419697)

[3.2.1.10. Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta 38](#_Toc110419698)

[3.2.1.11. Atbildes ziņojuma nosūtīšana 43](#_Toc110419699)

[3.2.1.12. ACK statusa maiņa 47](#_Toc110419700)

[3.2.1.13. Liela apjoma ziņojuma nosūtīšana 52](#_Toc110419701)

[3.2.2. FTP piemēri .NET 57](#_Toc110419702)

[3.2.2.1. Datnes pievienošana (FTPs\_SendMessage) 58](#_Toc110419703)

[3.2.2.2. Datnes nolasīšana un dzēšana (FTPs\_ReceiveMessage) 60](#_Toc110419704)

[3.2.2.3. Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta 62](#_Toc110419705)

[3.2.2.4. Atbildes ziņojuma nosūtīšana 64](#_Toc110419706)

[4. Java 67](#_Toc110419707)

[4.1. Projekta izveidošana Java vidē 67](#_Toc110419708)

[4.2. Piemēri Java 68](#_Toc110419709)

[4.2.1. Tīkla servisu piemēri 68](#_Toc110419710)

[4.2.1.1. Pieslēgšanās ar sertifikātu 69](#_Toc110419711)

[4.2.1.2. Pieslēgšanās ar paroli 70](#_Toc110419712)

[4.2.1.3. Kanāla pārbaude un izveidošana (SelectAndCreateChannel) 71](#_Toc110419713)

[4.2.1.4. Kanāla versijas pārbaude un izveidošana (SelectAndCreateChannelVersion) 73](#_Toc110419714)

[4.2.1.5. Ziņojuma nosūtīšana (SendMessageToChannelVersion) 76](#_Toc110419715)

[4.2.1.6. Ziņojumu dzēšana (MessageDelete) 80](#_Toc110419716)

[4.2.1.7. Pievienot pierakstītāju kanāla versijai (CreateSubscriberToChannelVersion) 81](#_Toc110419717)

[4.2.1.8. Ziņojuma nolasīšana (ReceiveMessage) 85](#_Toc110419718)

[4.2.1.9. Ziņojuma atjaunošana (Restore Messages) 86](#_Toc110419719)

[4.2.1.10. Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta 90](#_Toc110419720)

[4.2.1.11. Atbildes ziņojuma nosūtīšana 93](#_Toc110419721)

[4.2.1.12. ACK statusa maiņa 96](#_Toc110419722)

[4.2.1.13. Liela apjoma ziņojuma nosūtīšana 99](#_Toc110419723)

[4.2.2. FTP piemēri Java 102](#_Toc110419724)

[4.2.2.1. Datnes pievienošana (FTPSendMessage) 103](#_Toc110419725)

[4.2.2.2. Datnes nolasīšana un dzešana (FTPReceiveMessage) 104](#_Toc110419726)

[4.2.2.3. Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta 105](#_Toc110419727)

[4.2.2.4. Atbildes ziņojuma nosūtīšana 106](#_Toc110419728)

[5. PHP 107](#_Toc110419729)

[5.1. Projekta izveidošana PHP 107](#_Toc110419730)

[5.1.1. SOAP piemēri PHP 108](#_Toc110419731)

[5.1.1.1. Konfigurācijas datne Config.php 108](#_Toc110419732)

[5.1.1.2. Savienošana ar sertifikātu (LoginCert) 109](#_Toc110419733)

[5.1.1.3. Savienošana ar lietotāja vārdu un paroli (LoginUserName) 110](#_Toc110419734)

[5.1.1.4. Kanāla pārbaude un izveidošana (SelectAndCreateChannel) 112](#_Toc110419735)

[5.1.1.5. Kanāla versijas pārbaude un izveidošana (SelectAndCreateChannelVersion) 114](#_Toc110419736)

[5.1.1.6. Ziņojuma nosūtīšana (SendMessageToChannelVersion) 119](#_Toc110419737)

[5.1.1.7. Ziņojumu dzēšana (MessageDelete) 124](#_Toc110419738)

[5.1.1.8. Tā, kas parakstījies pievienošana kanāla versijai (CreateSubscriberToChannelVersion) 126](#_Toc110419739)

[5.1.1.9. Ziņojuma nolasīšana (ReceiveMessage) 131](#_Toc110419740)

[5.1.1.10. Ziņojuma atjaunošana (Restore Messages) 133](#_Toc110419741)

[5.1.1.11. Ziņojuma ACK statusa maiņa (ACKStatusChange) 138](#_Toc110419742)

[5.1.1.12. Ziņojuma nosutīšana vienam kanāla dalibniekam (SendMessageReceiver) 145](#_Toc110419743)

[5.1.1.13. Ziņojuma nosutīšana vienam kanāla dalibniekam (SendReplay) 150](#_Toc110419744)

[5.1.2. FTP piemēri PHP 154](#_Toc110419745)

[5.1.2.1. Datnes pievienošana (FTPs\_SendMessage) 155](#_Toc110419746)

[5.1.2.2. Datnes nolasīšana un dzēšana (FTPs\_ReceiveMessage) 156](#_Toc110419747)

[5.1.2.3. Ziņojuma sutīšana vienam kanāla dalibniekam (FTPSendMessageReceiver) 157](#_Toc110419748)

[5.1.2.4. Ziņojuma atbildēšana (FTPSendReply) 159](#_Toc110419749)

# Attēlu saraksts

[1.attēls. Datu izplatīšanas kanāls (DIK) 9](#_Toc110419750)

[2.attēls. Datu savākšanas kanāls (DSK) 9](#_Toc110419751)

[3.attēls. Projekta izveidošana 11](#_Toc110419752)

[4.attēls. Savienošana ar servisu 11](#_Toc110419753)

[5.attēls. Sertifikāta piemērs 12](#_Toc110419754)

[6.attēls. Konsoles atvēršana 12](#_Toc110419755)

[7.attēls. Sertifikāta pievienošana 12](#_Toc110419756)

[8.attēls. Sertifikāta pievienošana 12](#_Toc110419757)

[9.attēls. Importēšanas izvēlne 13](#_Toc110419758)

[10.attēls. Importēšanas izvēlne 13](#_Toc110419759)

[11.attēls. Sertifikātu izvēlne 13](#_Toc110419760)

[12.attēls. Sertifikāta dati 14](#_Toc110419761)

[13.attēls. Kanāla izveidošana 20](#_Toc110419762)

[14.attēls. Kanāla versijas izveidošana 24](#_Toc110419763)

[15.attēls. Izveidotais ziņojums 28](#_Toc110419764)

[16.attēls. Dzēstie ziņojumi 29](#_Toc110419765)

[17.attēls. Iestāde pieslēgta DIT 33](#_Toc110419766)

[18.attēls. DIK kanāla izmantošanas shēma 34](#_Toc110419767)

[19.attēls. DIK kanāla izmantošanas shēma, kas atjauno ziņojumu 36](#_Toc110419768)

[20.attēls. FileZilla FTP kanāla konfigurācija 58](#_Toc110419769)

[21.attēls. FTPs klienta pieslēgšanas piemērs 59](#_Toc110419770)

[22.attēls. Piemēru projektu struktūra Netbeans vidē 68](#_Toc110419771)

[23.attēls. Konfigurācijas datnes 69](#_Toc110419772)

[24.attēls. Kanāla izveidošana 74](#_Toc110419773)

[25.attēls. Kanāla versijas izveidošana 77](#_Toc110419774)

[26.attēls. Izveidotais ziņojums 81](#_Toc110419775)

[27.attēls. Dzēstie ziņojumi 82](#_Toc110419776)

[28.attēls. Iestāde pieslēgta DIT 86](#_Toc110419777)

[29.attēls. DIK kanāla izmantošanas shēma 86](#_Toc110419778)

[30.attēls. DIK kanāla izmantošanas shēma, par ziņojumu atjaunošanu 88](#_Toc110419779)

[31.attēls. FileZilla FTP kanāla konfigurācija 104](#_Toc110419780)

[32.attēls. Piemēru palaišanas saskarne 108](#_Toc110419781)

[33.attēls. Projekta struktūra 109](#_Toc110419782)

[34.attēls. Kanāla izveidošana 115](#_Toc110419783)

[35.attēls. Kanāla versijas izveidošana 120](#_Toc110419784)

[36.attēls. Izveidotais ziņojums 125](#_Toc110419785)

[37.attēls. Dzēstie ziņojumi 127](#_Toc110419786)

[38.attēls. Iestāde pieslēgta DIT 132](#_Toc110419787)

[39.attēls. DIK kanāla izmantošanas shēma 132](#_Toc110419788)

[40.attēls. DIK kanāla izmantošanas shēma, lai atjaunotu ziņojumu 135](#_Toc110419789)

[41.attēls. Iestāde pieslēgta DIT 146](#_Toc110419790)

# Ievads

Datu izplatīšanas tīkls (DIT) ir Valsts informācijas sistēmu savietotāja (VISS) infrastruktūras elements, kurš nodrošina virtuālo vidi, kurā iestādes var reglamentētā veidā veikt automātisku komunikāciju starp to informācijas sistēmām (IS).

## Darbības sfēra

Dokuments ir izmantojams DIT risinājuma integrācijai ar citām informācijas sistēmām.

Dokumentā aprakstītie piemēri ir izstrādāti ar mērķi demonstrēt pamatrisinājuma funkcionalitāti un tie nevar tikt izmantoti bez izmainām reālā pielietojumā.

## Termini un pieņemtie apzīmējumi

Apzīmējumu un terminu vārdnīca pieejama dokumentā [1].

## Saistītie dokumenti

Dokuments ir izstrādāts, balstoties uz šādiem dokumentiem:

1. "Valsts informācijas sistēmu savietotāju (VISS) un Vienotā valsts un pašvaldību pakalpojumu portāla www.latvija.lv pilnveidošana un uzturēšana". VISS infrastruktūra. Programmatūras projektējuma apraksts VRAA-13\_7\_17\_41-VISS\_2016-VISS\_INFR-PPA.
2. "Valsts informācijas sistēmu savietotāju (VISS) un Vienotā valsts un pašvaldību pakalpojumu portāla www.latvija.lv pilnveidošana un uzturēšana". Terminu un saīsinājumu indekss. VRAA-13\_7\_17\_41-VISS\_2016-TSI.

## Dokumenta pārskats

Dokumentu veido šādi nodalījumi:

* Dokumenta ievads – aprakstīts dokumenta nolūks, termini un pieņemtie apzīmējumi, kā arī norādīta saistība ar citiem dokumentiem un materiāliem;
* Vispārīgs apraksts – sniedz ieskatu par risinājumu funkcionalitātes izmantošanu.
* .Net – sniedz informāciju par .NET projekta izveidošanu un piemēriem.
* Java – sniedz informāciju par Java projekta izveidošanu un piemēriem.
* PHP – sniedz informāciju par PHP projekta izveidošanu un piemēriem.

# **Vispārējais apraksts**

Informācijas apmaiņa notiek, izmantojot kanālus, kur katrs kanāls nodrošina ziņojumu pārsūtīšanu no viena īpašnieka vairākiem dalībniekiem (DIK kanāls) vai no vairākiem dalībniekiem vienam īpašniekam (DSK kanāls):

* Datu izplatīšanas kanāls, skat. 1.attēlā. Kanāls, pa kuru ziņojumi plūst no viena sūtītāja, kas ir kanāla īpašnieks, vairākiem saņēmējiem, kas ir kanāla dalībnieki.



**1.attēls. Datu izplatīšanas kanāls (DIK)**

* Datu savākšanas kanāls. Kanāls, pa kuru ziņojumi plūst no vairākiem sūtītājiem, kas ir kanāla dalībnieki, vienam saņēmējam – kanāla īpašniekam.



.attēls. Datu savākšanas kanāls (DSK)

Katram kanālam ir reglamentēts, kāda tipa ziņojumus pa to var sūtīt. Katrs kanāls nodrošina Biznesa ziņojumu vai Transakcijas ziņojumu sūtīšanu.

## Biznesa ziņojums

Biznesa ziņojums – parasts ziņojums, kas nesaistās ar citiem ziņojumiem kanāla ietvaros.

## Transakcijas ziņojums

Transakcijas ziņojums – standarta (klasificētā) sakārtota operāciju kopa, kas tiek pielietota noteiktam datu masīvam tā aktualizācijai (replikācijai). Katrs transakcijas ziņojuma ieraksts parasti satur DML (*Data manipulation language*) operācijas kodu (*insert, update, delete*). Viena kanāla versija varēs nodrošināt darbu vienai datu replikācijai.

Definīcijā minētais datu masīvs jebkurā laika momentā ir aprēķināms pēc šādas formulas:

Aktuālais datu masīvs = Pielietot (Sākumdatu masīvam, Transakciju vēsturisko kopu).

Datu replikācijas sinhronitātes nodrošināšanai katram replikācijas dalībniekam pieejams vienāds sākumdatu masīvs un vienāda transakciju vēsture. Sākumdatu masīvs ir datu bāzes saturs, kas tika ņemts replikācijas uzsākšanas brīdī.

Kanāla (DIK) īpašnieks un dalībnieki uzturēs identiskas replicējamās datu bāzes datu masīva kopijas. Kad DIK īpašnieks veic kādas operācijas ar replicējamiem datiem, viņš arī aizsūta transakciju visiem DIK dalībniekiem – tie atjaunina savas datu masīva kopijas, un paralēlā transakciju vēsture tiek papildināta. Tādā veidā kanāla versijas ietvaros īpašniekam un dalībniekiem tiek nodrošināts sākumdatu masīvs un transakciju vēsture, kas dod rezultātā aktuālo replicējamas DB satura versiju.

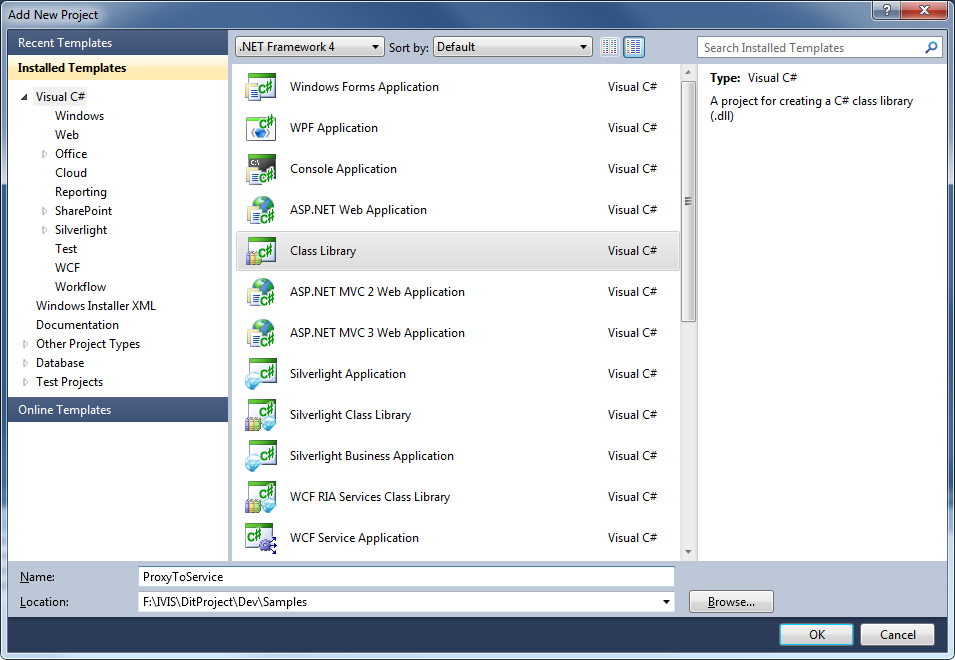
## Elektroniskā ziņojumu glabātava

Visi nosūtītie vai saņemti ziņojumi tiek saglabāti pasta kastītē (elektroniskā ziņojumu glabātava). Katrai iestādei, kas ir reģistrēta VISS, un kas lieto DIT ir piešķirta viena e-pastkastīte, kurā glabājas visi ienākošie vai izejošie ziņojumi. DIT vidē pastkastītes ir komunikatīvi savienojamas ar kanālu palīdzību.

# .Net

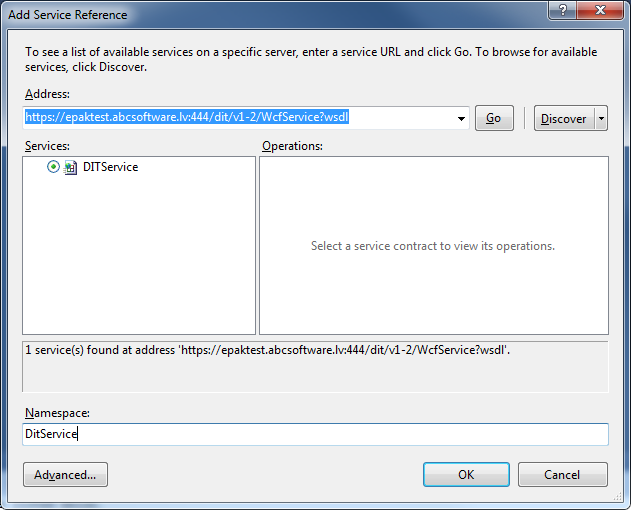
## Projekta izveidošana.Net vidē

Lai izveidotu savienojumu ar servisu, nepieciešams izveidot *Proxy* savienojumu:



.attēls. Projekta izveidošana

Projektā tiek izveidots savienojums ar servisu:



.attēls. Savienošana ar servisu

## Piemēri .NET

Nodaļā ir aprakstīti piemēri DIT saskarnes izmantošanai, tai skaita:

* DIT datu izgūšanas tīkla serviss;
* DIT iestādes pasta kastītes un kanālu konfigurācijas serviss;
* DIT FTP saskarnes.

### Tīkla servisu piemēri

#### Savienošana ar sertifikātu (LoginCert)

##### Sertifikāta uzstādīšana

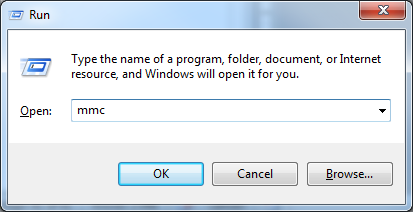
Izvēlamies sertifikātu DIT.pfx (piemēram).



.attēls. Sertifikāta piemērs

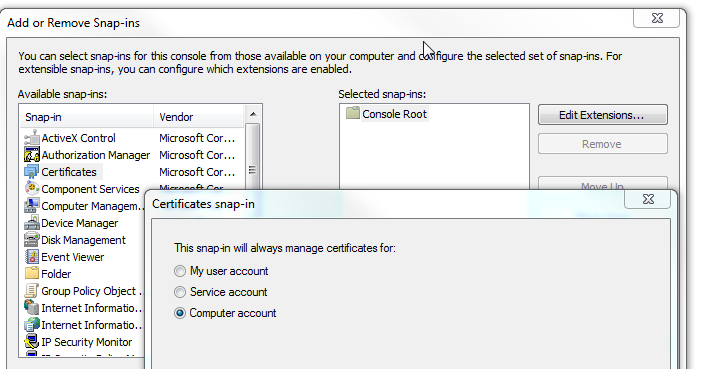
Sertifikāta uzstādīšana konsolē:

1. Atvēram konsoli;



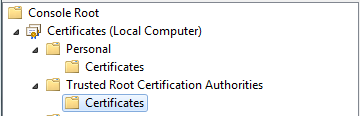
.attēls. Konsoles atvēršana

1. Pievienojam sertifikātu;



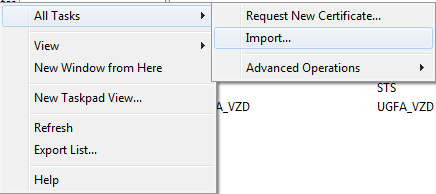
.attēls. Sertifikāta pievienošana

1. Atvēram *Trusted Root Sertificate Authorities* sertifikātus;



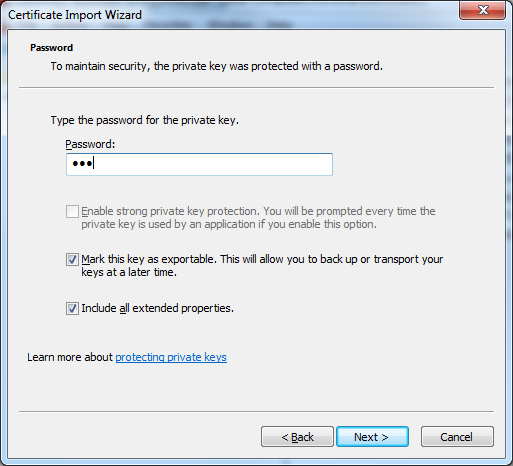
.attēls. Sertifikāta pievienošana

1. Importējam *DIT.pfx* personālos sertifikātos:



.attēls. Importēšanas izvēlne

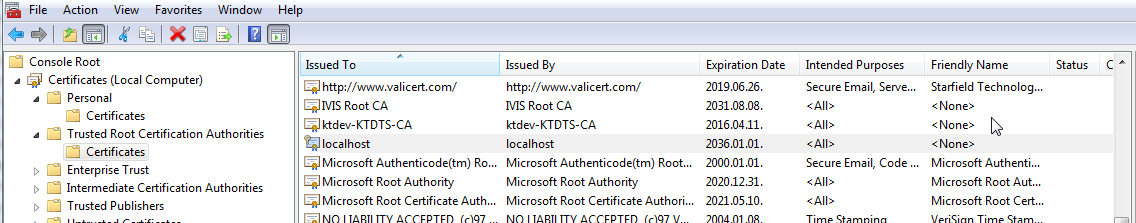
1. Izvelamies sertifikātu un atzīmējam, tā kā parādīts 10.attēlā ar paroli „*preved*”:
2. „*Mark this key as*….” jābūt atzīmētai;
3. „*Include all expanded properties*” jābūt atzīmētai.



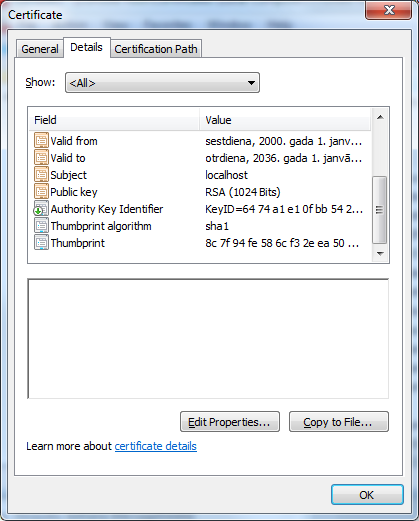
.attēls. Importēšanas izvēlne

###### Pārbaudīt sertifikāta thumbprint

1. Ar kreiso peles pogu noklikšķinām divas reizes uz sertifikāta;



.attēls. Sertifikātu izvēlne



.attēls. Sertifikāta dati

###### Konfigurācijas un sertifikāta sakritība

Konfigurācija un sertifikāta sakritībai jāizskatās, kā zemāk redzams. Turklāt tai jāsakrīt ar *DIT.pfx* *thumbprint*.

<behavior name="certificate">

<clientCredentials>

<clientCertificate findValue="8c 7f 94 fe 58 6c f3 2e ea 50 9a 32 a3 73 53 ba b4 e5 6c 6d" storeLocation="LocalMachine" x509FindType="FindByThumbprint"/>

</clientCredentials>

<!--<applyTo applyTo="https://ivis.eps.gov.lv/DIT.WebService"/>-->

</behavior>

##### Konfigurācija

Lai izveidotu savienojumu ar sertifikātu, ir jāizveido divi bindingi:

|  |  |  |
| --- | --- | --- |
| ServiceModel | Izvelētais nosaukums | Tiek Izmantots |
| ws2007FederationHttpBinding | ws2007FederationNoSct | ConfigurationService |
| ws2007HttpBinding | certificateMixed | ws2007FederationNoSct |
| endPointBehavior | certificate | ConfigurationService |
| Client endPoint | ConfigurationService | Kodā |



##### Koda piemērs (LoginCert)

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

var postBox = client.SelectPostBox("100000");

if (postBox != null)

{

Console.WriteLine(postBox.Authority.ShortName);

}

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

#### Savienošana ar lietotāja vārdu un paroli (LoginUserName)

##### Konfigurācija

Lai izveidotu savienojumu ar lietotāja vārdu un paroli, ir jāizveido divi bindingi:

|  |  |  |
| --- | --- | --- |
| ServiceModel | Izvelētais nosaukums | Tiek izmantots |
| ws2007FederationHttpBinding | ws2007FederationNoSct | ConfigurationService |
| ws2007HttpBinding | certificateMixed | ws2007FederationNoSct |
| Client endPoint | ConfigurationService | Kodā |
| UserName |  | Kodā |
| Password |  | Kodā |



##### Koda piemēri (LoginUserName)

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationServiceuser"))

{

try

{

client.ClientCredentials.UserName.UserName = System.Configuration.ConfigurationManager.AppSettings.Get("UserName");

client.ClientCredentials.UserName.Password = System.Configuration.ConfigurationManager.AppSettings.Get("Password");

ProxyToService.DitConfiguration.PostBoxInfoStructure postBox = client.SelectPostBox("100001");

if (postBox != null)

{

Console.WriteLine(postBox.PostBoxID);

}

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}

#### Kanāla pārbaude un izveidošana (SelectAndCreateChannel)

Piemērā tiek pārbaudīts vai eksistē kanāls. Ja kanāla nav, tad tiek izveidots jauns kanāls:

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

const string channelName = "SampleChannel";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

ProxyToService.DitConfiguration.ChannelSearchInfoStructure req = new ProxyToService.DitConfiguration.ChannelSearchInfoStructure

{

Channel = new ProxyToService.DitConfiguration.ChannelInfoStructure

{

ChannelURN = channelURN

},

OnlyHeader = true

};

ProxyToService.DitConfiguration.ChannelListStructure channels = Helper.InvokeIgnoreException<ChannelListStructure, CommunicationException>(() => client.SearchChannelsPaged(req, ChannelVersionStatusType.testing), "Argument exception: ChannelURN!");

bool channelExsists = channels != null && channels.Count > 0;

if (!channelExsists)

{

Console.WriteLine("Channel creation...");

ChannelInfoStructure channel = new ChannelInfoStructure();

channel.ShortName = channelName;

channel.ChannelURN = channelURN;

channel.ChannelType = ChannelType.dik;

channel.OwnerAuthority = new AuthorityBasicInfoStructure();

channel.OwnerAuthority.AuthorityID = "100000";

channel.OwnerAuthority.ShortName = "ABC";

channel.OwnerAuthority.Name = "ABC Software";

channel.ChannelName = channelName;

channel.ChannelTypeSpecified = true;

int channelId = client.InsertChannel(channel);

Console.WriteLine("Channel is created");

}

else

{

Console.WriteLine("Channel is found");

}

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}



.attēls. Kanāla izveidošana

#### Kanāla versijas pārbaude un izveidošana (SelectAndCreateChannelVersion)

Piemērā tiek pārbaudīts vai eksistē kanāls. Ja kanāla nav, tad tiek izveidots jauns kanāls un tiek pārbaudīts vai eksistē kanāla versija. Ja tādas nav, tad sāk tās veidošanu.

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

string channelName = "SampleChannel";

string chType = "DIK";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-" + chType;

ProxyToService.DitConfiguration.ChannelSearchInfoStructure req = new ProxyToService.DitConfiguration.ChannelSearchInfoStructure();

req.Channel = new ProxyToService.DitConfiguration.ChannelInfoStructure();

req.Channel.ChannelURN = channelURN;

req.OnlyHeader = true;

ProxyToService.DitConfiguration.ChannelListStructure channels = Helper.InvokeIgnoreException<ChannelListStructure, CommunicationException>(() => client.SearchChannelsPaged(req, ChannelVersionStatusType.testing), "Argument exception: ChannelURN!");

ChannelInfoStructure channel = new ChannelInfoStructure();

channel.ShortName = channelName;

channel.ChannelURN = channelURN;

channel.ChannelType = ChannelType.dik;

channel.OwnerAuthority = new AuthorityBasicInfoStructure();

channel.OwnerAuthority.AuthorityID = "100000";

channel.OwnerAuthority.ShortName = "ABC";

channel.OwnerAuthority.Name = "ABC Software";

channel.ChannelName = channelName;

channel.ChannelTypeSpecified = true;

bool channelExsists = channels != null && channels.Count > 0;

if (!channelExsists)

{

int channelId = client.InsertChannel(channel);

Console.WriteLine("Channel created");

}

else

{

Console.WriteLine("Channel found");

}

ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();

channelVersion.MessageType = new MessageTypeInfoStructure[3];

channelVersion.MessageType[0] = new MessageTypeInfoStructure

{

MessageClass = new MessageClassInfoStructure()

};

channelVersion.IsManualAckSpecified = true;

channelVersion.IsManualAck = false;

channelVersion.UseReceiverFilterSpecified = true;

channelVersion.UseReceiverFilter = true;

// 1 Biznesa ziņojums

// 2 Transakcijas ziņojums

// 3 error

channelVersion.MessageType[0].MessageClass.CodeListCodeValue = "2";

channelVersion.MessageType[0].XMLSchemaURN = "URN:IVIS:100001:XSD-Ditlog-DITLog-v1-0-TYPE-DITLog";

channelVersion.MessageType[1] = new MessageTypeInfoStructure

{

MessageClass = new MessageClassInfoStructure()

};

channelVersion.MessageType[1].MessageClass.CodeListCodeValue = "3";

channelVersion.MessageType[1].XMLSchemaURN = "URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError";

channelVersion.ChannelVersionStatus = ChannelVersionStatusType.active;

channelVersion.ChannelVersionStatusSpecified = true;

channelVersion.Channel = channel;

channelVersion.MinorVersionSpecified = true;

channelVersion.MinorVersion = 0;

channelVersion.MajorVersionSpecified = true;

channelVersion.MajorVersion = 1;

channelVersion.ChannelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v" + channelVersion.MajorVersion + "-" + channelVersion.MinorVersion + "-TYPE-" + chType;

ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();

channelVersionSearch.ChannelVersion = new ChannelVersionInfoStructure();

channelVersionSearch.ChannelVersion = channelVersion;

ChannelVersionListStructure channelVersions = client.SearchChannelVersionsPaged(channelVersionSearch);

if (channelVersions.Count == 0)

{

int channelId = client.InsertChannelVersion(channelVersion);

Console.WriteLine("Channel version created");

}

else

{

Console.WriteLine("Channel version found");

}

string channelVersionURN = channelVersion.ChannelVersionURN;

ChannelSubscriberInfoStructure channelSubscriber = new ChannelSubscriberInfoStructure();

channelSubscriber.ChannelVersionURN = channelVersionURN;

channelSubscriber.SubscriberAuthority = new AuthorityBasicInfoStructure();

channelSubscriber.SubscriberAuthority.AuthorityID = "100001";

channelSubscriber.SubscriberAuthority.ShortName = "VRAA";

channelSubscriber.SubscriberAuthority.Name = "VRAA";

ChannelSubscriberSearchInfoStructure channelSubscriberSearch = new ChannelSubscriberSearchInfoStructure();

channelSubscriberSearch.ChannelSubscriber = channelSubscriber;

ChannelSubscriberListStructure channelSubscriberList = client.SearchSubscribersToChannelPaged(channelSubscriberSearch);

if (channelSubscriberList.Count == 0)

{

int result = client.InsertSubscriberToChannel(channelSubscriber);

Console.WriteLine("Channel subscriber created");

}

else

{

Console.WriteLine("Channel subscriber found");

}

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}



.attēls. Kanāla versijas izveidošana

#### Ziņojuma nosūtīšana (SendMessageToChannelVersion)

Tiek pārbaudīts, vai eksistē kanāla versija. Ja tā ir, tad uz to tiek nosūtīts ziņojums.

string channelName = "SampleChannel";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

string channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel-v1-0-TYPE-DIK";

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

ProxyToService.DitConfiguration.ChannelSearchInfoStructure req = new ProxyToService.DitConfiguration.ChannelSearchInfoStructure();

req.Channel = new ProxyToService.DitConfiguration.ChannelInfoStructure();

req.Channel.ChannelURN = channelURN;

req.OnlyHeader = true;

ProxyToService.DitConfiguration.ChannelListStructure channels = Helper.InvokeIgnoreException<ChannelListStructure, CommunicationException>(() => client.SearchChannelsPaged(req, ChannelVersionStatusType.testing), "Argument exception: ChannelURN!");

ChannelInfoStructure channel = new ChannelInfoStructure();

channel.ShortName = channelName;

channel.ChannelURN = channelURN;

channel.ChannelType = ChannelType.dik;

channel.OwnerAuthority = new AuthorityBasicInfoStructure();

channel.OwnerAuthority.AuthorityID = "100000";

channel.OwnerAuthority.ShortName = "ABC";

channel.OwnerAuthority.Name = "ABC Software";

channel.ChannelName = channelName;

channel.ChannelTypeSpecified = true;

bool channelExsists = channels != null && channels.Count > 0;

if (!channelExsists)

{

int channelId = client.InsertChannel(channel);

Console.WriteLine("Channel created");

}

else

{

Console.WriteLine("Channel found");

}

ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();

channelVersion.MessageType = new ProxyToService.DitConfiguration.MessageTypeInfoStructure[2];

channelVersion.MessageType[0] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[0].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

// 1 Biznesa ziņojums

// 2 Transakcijas ziņojums

// 3 error

channelVersion.MessageType[0].MessageClass.CodeListCodeValue = "1";

channelVersion.MessageType[0].XMLSchemaURN = "URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName";

channelVersion.MessageType[1] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[1].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

channelVersion.MessageType[1].MessageClass.CodeListCodeValue = "3";

channelVersion.MessageType[1].XMLSchemaURN = "URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError";

channelVersion.ChannelVersionStatus = ChannelVersionStatusType.active;

channelVersion.ChannelVersionStatusSpecified = true;

channelVersion.Channel = channel;

channelVersion.MinorVersionSpecified = true;

channelVersion.MinorVersion = 0;

channelVersion.MajorVersionSpecified = true;

channelVersion.MajorVersion = 1;

ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();

channelVersionSearch.ChannelVersion = new ChannelVersionInfoStructure();

channelVersionSearch.ChannelVersion = channelVersion;

ChannelVersionListStructure channelVersions = client.SearchChannelVersionsPaged(channelVersionSearch);

if (channelVersions.Count == 0)

{

int channelId = client.InsertChannelVersion(channelVersion);

Console.WriteLine("Channel version created");

}

else

{

Console.WriteLine("Channel version found");

}

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageService"))

{

var applyTo = new Uri("https://ivis.eps.gov.lv/DIT.WebService");

var factory = client.ChannelFactory;

var other = factory.Endpoint.Behaviors.Find<ClientCredentials>();

factory.Endpoint.Behaviors.Remove(other.GetType());

var item = new ApplyToClientCredentials(other) { ApplyTo = applyTo };

factory.Endpoint.Behaviors.Add(item);

try

{

ProxyToService.DitService.MessageSaveStructure2 message = new ProxyToService.DitService.MessageSaveStructure2();

message.ChannelVersionURN = channelVersionURN;

BussinesMessageType bussinesMessage = new BussinesMessageType();

XmlDocument xml = new XmlDocument();

xml.Load("../../../data.xml");

bussinesMessage.Any = xml.DocumentElement;

message.Item = bussinesMessage;

bool resp = client.SendMessage2(message);

Console.WriteLine("message was sent");

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}



.attēls. Izveidotais ziņojums

#### Ziņojumu dzēšana (MessageDelete)

No kanāla tiek dzēsti ziņojumi:

string channelName = "SampleChannel";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

//string channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageService"))

{

try

{

MessageRequestStructure messDelete = new MessageRequestStructure();

messDelete.Item = channelURN;

bool resp = client.DeleteMessage(messDelete);

Console.WriteLine("message was deleted");

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}



.attēls. Dzēstie ziņojumi

#### Tā, kas parakstījies pievienošana kanāla versijai (CreateSubscriberToChannelVersion)

Tiek pārbaudīts, vai uz kanālu kāds ir parakstījies. Ja nav, tad pievieno kanāla dalībnieku.

string channelName = "SampleChannel";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

string channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel-v1-0-TYPE-DIK";

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

ProxyToService.DitConfiguration.ChannelSearchInfoStructure req = new ProxyToService.DitConfiguration.ChannelSearchInfoStructure

{

Channel = new ProxyToService.DitConfiguration.ChannelInfoStructure()

};

req.Channel.ChannelURN = channelURN;

req.OnlyHeader = true;

ProxyToService.DitConfiguration.ChannelListStructure channels = Helper.InvokeIgnoreException<ChannelListStructure, CommunicationException>(() => client.SearchChannelsPaged(req, ChannelVersionStatusType.testing), "Argument exception: ChannelURN!");

ChannelInfoStructure channel = new ChannelInfoStructure();

channel.ShortName = channelName;

channel.ChannelURN = channelURN;

channel.ChannelType = ChannelType.dik;

channel.OwnerAuthority = new AuthorityBasicInfoStructure();

channel.OwnerAuthority.AuthorityID = "100000";

channel.OwnerAuthority.ShortName = "ABC";

channel.OwnerAuthority.Name = "ABC Software";

channel.ChannelName = channelName;

channel.ChannelTypeSpecified = true;

bool channelExsists = channels != null && channels.Count > 0;

if (!channelExsists)

{

int channelId = client.InsertChannel(channel);

Console.WriteLine("Channel created");

}

else

{

Console.WriteLine("Channel found");

}

ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure

{

MessageType = new ProxyToService.DitConfiguration.MessageTypeInfoStructure[2]

};

channelVersion.MessageType[0] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[0].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

// 1 Biznesa ziņojums

// 2 Transakcijas ziņojums

// 3 error

channelVersion.MessageType[0].MessageClass.CodeListCodeValue = "1";

channelVersion.MessageType[0].XMLSchemaURN = "URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName";

channelVersion.MessageType[1] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[1].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

channelVersion.MessageType[1].MessageClass.CodeListCodeValue = "3";

channelVersion.MessageType[1].XMLSchemaURN = "URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError";

channelVersion.ChannelVersionStatus = ChannelVersionStatusType.active;

channelVersion.ChannelVersionStatusSpecified = true;

channelVersion.Channel = channel;

channelVersion.MinorVersionSpecified = true;

channelVersion.MinorVersion = 0;

channelVersion.MajorVersionSpecified = true;

channelVersion.MajorVersion = 1;

channelVersion.UseReceiverFilterSpecified = true;

channelVersion.UseReceiverFilter = true;

ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();

channelVersionSearch.ChannelVersion = new ChannelVersionInfoStructure();

channelVersionSearch.ChannelVersion = channelVersion;

ChannelVersionListStructure channelVersions = client.SearchChannelVersionsPaged(channelVersionSearch);

if (channelVersions.Count == 0)

{

int channelId = client.InsertChannelVersion(channelVersion);

Console.WriteLine("Channel version created");

}

else

{

Console.WriteLine("Channel version found");

}

ChannelSubscriberInfoStructure channelSubscriber = new ChannelSubscriberInfoStructure();

channelSubscriber.ChannelVersionURN = channelVersionURN;

channelSubscriber.SubscriberAuthority = new AuthorityBasicInfoStructure

{

AuthorityID = "100001",

ShortName = "VRAA",

Name = "VRAA"

};

ChannelSubscriberSearchInfoStructure channelSubscriberSearch = new ChannelSubscriberSearchInfoStructure

{

ChannelSubscriber = channelSubscriber

};

ChannelSubscriberListStructure channelSubscriberList = client.SearchSubscribersToChannelPaged(channelSubscriberSearch);

if (channelSubscriberList.Count == 0)

{

int result = client.InsertSubscriberToChannel(channelSubscriber);

Console.WriteLine("Channel subscriber created");

}

else

{

Console.WriteLine("Channel subscriber found");

}

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}



.attēls. Iestāde pieslēgta DIT

#### Ziņojuma nolasīšana (ReceiveMessage)

Pēc DIT kanāla izveidošanas, iespējams saņemt ziņojumus par lietotāju, kurš pierakstījies pieslēgumam.



.attēls. DIK kanāla izmantošanas shēma

Pēc nolasīšanas ziņojumu nepieciešams dzēst.

string channelName = "SampleChannel";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

//string channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageServiceUser"))

{

try

{

client.ClientCredentials.UserName.UserName = System.Configuration.ConfigurationManager.AppSettings.Get("VraaUserName");

client.ClientCredentials.UserName.Password = System.Configuration.ConfigurationManager.AppSettings.Get("VraaPassword");

MessageRequestStructure messageRequest = new MessageRequestStructure();

messageRequest.Item = channelURN;

MessageResponseStructure2 resp = client.ReceiveMessage2(messageRequest);

if (resp != null && resp.Items != null)

{

foreach (var item in resp.Items)

{

Console.WriteLine("message receved id:" + item.MessageID);

MessageRequestStructure messageRequestDelete = new MessageRequestStructure();

messageRequestDelete.Item = item.MessageID;

messageRequestDelete.ItemElementName = ItemChoiceType.MessageID;

client.DeleteMessage(messageRequestDelete);

Console.WriteLine("message deleted id:" + item.MessageID);

}

}

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}

#### Ziņojuma atjaunošana (Restore Messages)

Pēc DIT kanāla struktūras izveidošanas ar lietotāju, kurš parakstījies uz pieslēgumu, iespējams saņemt un atjaunot ziņojumus.

Gadījumā, ja ziņojuma saņemšana bija nekorekta, tad to var atjaunot, lietojot metodes noteiktajā secībā:

1. ReceiveMessageList;
2. RestoreMessageList (ReadMessage=true);
3. ReceveMessage;
4. DeleteMessage.



.attēls. DIK kanāla izmantošanas shēma, kas atjauno ziņojumu

string channelName = "SampleChannel7";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

string channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageService"))

{

try

{

ProxyToService.DitService.MessageSaveStructure2 message = new ProxyToService.DitService.MessageSaveStructure2();

message.ChannelVersionURN = channelVersionURN;

BussinesMessageType bussinesMessage = new BussinesMessageType();

XmlDocument xml = new XmlDocument();

xml.Load("../../../data.xml");

bussinesMessage.Any = xml.DocumentElement;

message.Item = bussinesMessage;

message.ReceiverFilter = new ReceiverFilterArrayStructure()

{

Authority = new[] { "100001" }

};

bool resp = client.SendMessage2(message);

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageServiceUser"))

{

var applyTo = new Uri("https://ivis.eps.gov.lv/DIT.WebService");

var factory = client.ChannelFactory;

var other = factory.Endpoint.Behaviors.Find<ClientCredentials>();

factory.Endpoint.Behaviors.Remove(other.GetType());

var item1 = new ApplyToClientCredentials(other) { ApplyTo = applyTo };

factory.Endpoint.Behaviors.Add(item1);

try

{

client.ClientCredentials.UserName.UserName = System.Configuration.ConfigurationManager.AppSettings.Get("VraaUserName");

client.ClientCredentials.UserName.Password = System.Configuration.ConfigurationManager.AppSettings.Get("VraaPassword");

MessageRequestStructure messageRequest = new MessageRequestStructure();

messageRequest.Item = channelVersionURN;

messageRequest.ItemElementName = ItemChoiceType.ChannelVersionURN;

MessageResponseStructure2 resp = client.ReceiveMessage2(messageRequest);

Console.WriteLine("ReceiveMessage");

MessageListRequestStructure messageListRequest = new MessageListRequestStructure();

messageListRequest.Item = channelVersionURN;

messageListRequest.ItemElementName = ItemChoiceType.ChannelVersionURN;

messageListRequest.ReadMessage = true;

messageListRequest.ReadMessageSpecified = true;

MessageBasicInfoListStructure receiveMessageList = client.ReceiveMessageList(messageListRequest);

Console.WriteLine("ReceiveMessageList");

if (receiveMessageList != null && receiveMessageList.MessageBasicInfo != null)

{

MessageBasicInfoListStructure messageBasicInfoList = new MessageBasicInfoListStructure();

messageBasicInfoList.MessageBasicInfo = new MessageBasicInfoStructure[receiveMessageList.MessageBasicInfo.Length];

int i = 0;

foreach (var item in receiveMessageList.MessageBasicInfo)

{

messageBasicInfoList.MessageBasicInfo[i] = item;

}

bool data = Helper.InvokeIgnoreException<bool, FaultException>(() => client.RestoreMessageList(messageBasicInfoList), "statuss \"Pegādāts\" vairs nevar tikt mainīts uz statusu \"Pegādāts\"!");

}

MessageResponseStructure2 messageResponse = client.ReceiveMessage2(messageRequest);

if (messageResponse != null && messageResponse.Items != null)

{

foreach (var item in messageResponse.Items)

{

Console.WriteLine("message receved id:" + item.MessageID);

MessageRequestStructure messageRequestDelete = new MessageRequestStructure();

messageRequestDelete.Item = item.MessageID;

messageRequestDelete.ItemElementName = ItemChoiceType.MessageID;

client.DeleteMessage(messageRequestDelete);

Console.WriteLine("message deleted id:" + item.MessageID);

}

}

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}

#### Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta

Tiek pārbaudīts, vai eksistē kanāla versija. Ja tā ir, tad uz to tiek nosūtīts ziņojums saņēmējam.

string channelName = "SampleChannel578";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

string channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel578-v1-0-TYPE-DIK";

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

ProxyToService.DitConfiguration.ChannelSearchInfoStructure req = new ProxyToService.DitConfiguration.ChannelSearchInfoStructure();

req.Channel = new ProxyToService.DitConfiguration.ChannelInfoStructure();

req.Channel.ChannelURN = channelURN;

req.OnlyHeader = true;

ProxyToService.DitConfiguration.ChannelListStructure channels = Helper.InvokeIgnoreException<ChannelListStructure, CommunicationException>(() => client.SearchChannelsPaged(req, ChannelVersionStatusType.testing), "Argument exception: ChannelURN!");

ChannelInfoStructure channel = new ChannelInfoStructure();

channel.ShortName = channelName;

channel.ChannelURN = channelURN;

channel.ChannelType = ChannelType.dik;

channel.OwnerAuthority = new AuthorityBasicInfoStructure();

channel.OwnerAuthority.AuthorityID = "100000";

channel.OwnerAuthority.ShortName = "ABC";

channel.OwnerAuthority.Name = "ABC Software";

channel.ChannelName = channelName;

channel.ChannelTypeSpecified = true;

bool channelExsists = channels != null && channels.Count > 0;

if (!channelExsists)

{

int channelId = client.InsertChannel(channel);

Console.WriteLine("Channel created");

}

else

{

Console.WriteLine("Channel found");

}

ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();

channelVersion.UseReceiverFilter = true;

channelVersion.UseReceiverFilterSpecified = true;

channelVersion.MessageType = new ProxyToService.DitConfiguration.MessageTypeInfoStructure[2];

channelVersion.MessageType[0] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[0].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

// 1 Biznesa ziņojums

// 2 Transakcijas ziņojums

// 3 error

channelVersion.MessageType[0].MessageClass.CodeListCodeValue = "1";

channelVersion.MessageType[0].XMLSchemaURN = "URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName";

channelVersion.MessageType[1] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[1].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

channelVersion.MessageType[1].MessageClass.CodeListCodeValue = "3";

channelVersion.MessageType[1].XMLSchemaURN = "URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError";

channelVersion.ChannelVersionStatus = ChannelVersionStatusType.active;

channelVersion.ChannelVersionStatusSpecified = true;

channelVersion.Channel = channel;

channelVersion.MinorVersionSpecified = true;

channelVersion.MinorVersion = 0;

channelVersion.MajorVersionSpecified = true;

channelVersion.MajorVersion = 1;

ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();

channelVersionSearch.ChannelVersion = new ChannelVersionInfoStructure();

channelVersionSearch.ChannelVersion = channelVersion;

ChannelVersionListStructure channelVersions = client.SearchChannelVersionsPaged(channelVersionSearch);

if (channelVersions.Count == 0)

{

int channelId = client.InsertChannelVersion(channelVersion);

Console.WriteLine("Channel version created");

}

else

{

Console.WriteLine("Channel version found");

}

ChannelSubscriberInfoStructure channelSubscriber = new ChannelSubscriberInfoStructure();

channelSubscriber.ChannelVersionURN = channelVersionURN;

channelSubscriber.SubscriberAuthority = new AuthorityBasicInfoStructure();

channelSubscriber.SubscriberAuthority.AuthorityID = "100001";

channelSubscriber.SubscriberAuthority.ShortName = "VRAA";

channelSubscriber.SubscriberAuthority.Name = "VRAA";

ChannelSubscriberSearchInfoStructure channelSubscriberSearch = new ChannelSubscriberSearchInfoStructure();

channelSubscriberSearch.ChannelSubscriber = channelSubscriber;

ChannelSubscriberListStructure channelSubscriberList = client.SearchSubscribersToChannelPaged(channelSubscriberSearch);

if (channelSubscriberList.Count == 0)

{

int result = client.InsertSubscriberToChannel(channelSubscriber);

Console.WriteLine("Channel subscriber created");

}

else

{

Console.WriteLine("Channel subscriber found");

}

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageService"))

{

var applyTo = new Uri("https://ivis.eps.gov.lv/DIT.WebService");

var factory = client.ChannelFactory;

var other = factory.Endpoint.Behaviors.Find<ClientCredentials>();

factory.Endpoint.Behaviors.Remove(other.GetType());

var item = new ApplyToClientCredentials(other) { ApplyTo = applyTo };

factory.Endpoint.Behaviors.Add(item);

try

{

ProxyToService.DitService.MessageSaveStructure2 message = new ProxyToService.DitService.MessageSaveStructure2();

message.ChannelVersionURN = channelVersionURN;

BussinesMessageType bussinesMessage = new BussinesMessageType();

XmlDocument xml = new XmlDocument();

xml.Load("../../../data.xml");

bussinesMessage.Any = xml.DocumentElement;

message.Item = bussinesMessage;

message.ReceiverFilter = new ReceiverFilterArrayStructure()

{

Authority = new[] { "100001" }

};

bool resp = client.SendMessage2(message);

Console.WriteLine("message was sent");

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}

#### Atbildes ziņojuma nosūtīšana

string channelName = "SampleChannel7";

string channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";

string messageId = string.Empty;

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageService"))

{

try

{

ProxyToService.DitService.MessageSaveStructure2 message = new ProxyToService.DitService.MessageSaveStructure2();

message.ChannelVersionURN = channelVersionURN;

BussinesMessageType bussinesMessage = new BussinesMessageType();

XmlDocument xml = new XmlDocument();

xml.Load("../../../data.xml");

bussinesMessage.Any = xml.DocumentElement;

message.Item = bussinesMessage;

message.ReceiverFilter = new ReceiverFilterArrayStructure() { Authority = new string[] { "100001" } };

bool resp = client.SendMessage2(message);

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageServiceUser"))

{

var applyTo = new Uri("https://ivis.eps.gov.lv/DIT.WebService");

var factory = client.ChannelFactory;

var other = factory.Endpoint.Behaviors.Find<ClientCredentials>();

factory.Endpoint.Behaviors.Remove(other.GetType());

var item1 = new ApplyToClientCredentials(other) { ApplyTo = applyTo };

factory.Endpoint.Behaviors.Add(item1);

try

{

client.ClientCredentials.UserName.UserName = System.Configuration.ConfigurationManager.AppSettings.Get("VraaUserName");

client.ClientCredentials.UserName.Password = System.Configuration.ConfigurationManager.AppSettings.Get("VraaPassword");

MessageRequestStructure messageRequest = new MessageRequestStructure();

messageRequest.Item = channelVersionURN;

messageRequest.ItemElementName = ItemChoiceType.ChannelVersionURN;

MessageResponseStructure2 resp = client.ReceiveMessage2(messageRequest);

messageId = resp.Items?.FirstOrDefault()?.MessageID;

Console.WriteLine("ReceiveMessage");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageService"))

{

var applyTo = new Uri("https://ivis.eps.gov.lv/DIT.WebService");

var factory = client.ChannelFactory;

var other = factory.Endpoint.Behaviors.Find<ClientCredentials>();

factory.Endpoint.Behaviors.Remove(other.GetType());

var item = new ApplyToClientCredentials(other) { ApplyTo = applyTo };

factory.Endpoint.Behaviors.Add(item);

try

{

ProxyToService.DitService.MessageSaveStructure2 message = new ProxyToService.DitService.MessageSaveStructure2();

message.ChannelVersionURN = channelVersionURN;

EventMessageType eventMessage = new EventMessageType();

eventMessage.Name = "Error!";

eventMessage.RelatedMessageID = messageId;

XmlDocument xml = new XmlDocument();

xml.Load("../../../kluda.xml");

eventMessage.Any = xml.DocumentElement;

message.Item = eventMessage;

bool resp2 = client.SendMessage2(message);

Console.WriteLine("message was sent");

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

Console.ReadLine();

}

#### ACK statusa maiņa

string messageId = string.Empty;

string channelName = "SampleChannel455";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

string channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel455-v1-0-TYPE-DIK";

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

ProxyToService.DitConfiguration.ChannelSearchInfoStructure req = new ProxyToService.DitConfiguration.ChannelSearchInfoStructure();

req.Channel = new ProxyToService.DitConfiguration.ChannelInfoStructure();

req.Channel.ChannelURN = channelURN;

req.OnlyHeader = true;

ProxyToService.DitConfiguration.ChannelListStructure channels = Helper.InvokeIgnoreException<ChannelListStructure, CommunicationException>(() => client.SearchChannelsPaged(req, ChannelVersionStatusType.testing), "Argument exception: ChannelURN!");

ChannelInfoStructure channel = new ChannelInfoStructure();

channel.ShortName = channelName;

channel.ChannelURN = channelURN;

channel.ChannelType = ChannelType.dik;

channel.OwnerAuthority = new AuthorityBasicInfoStructure();

channel.OwnerAuthority.AuthorityID = "100000";

channel.OwnerAuthority.ShortName = "ABC";

channel.OwnerAuthority.Name = "ABC Software";

channel.ChannelName = channelName;

channel.ChannelTypeSpecified = true;

bool channelExsists = channels != null && channels.Count > 0;

if (!channelExsists)

{

int channelId = client.InsertChannel(channel);

Console.WriteLine("Channel created");

}

else

{

Console.WriteLine("Channel found");

}

ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();

channelVersion.UseReceiverFilter = true;

channelVersion.UseReceiverFilterSpecified = true;

channelVersion.MessageType = new ProxyToService.DitConfiguration.MessageTypeInfoStructure[2];

channelVersion.MessageType[0] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[0].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

// 1 Biznesa ziņojums

// 2 Transakcijas ziņojums

// 3 error

channelVersion.MessageType[0].MessageClass.CodeListCodeValue = "1";

channelVersion.MessageType[0].XMLSchemaURN = "URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName";

channelVersion.MessageType[1] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[1].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

channelVersion.MessageType[1].MessageClass.CodeListCodeValue = "3";

channelVersion.MessageType[1].XMLSchemaURN = "URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError";

channelVersion.ChannelVersionStatus = ChannelVersionStatusType.active;

channelVersion.ChannelVersionStatusSpecified = true;

channelVersion.Channel = channel;

channelVersion.MinorVersionSpecified = true;

channelVersion.MinorVersion = 0;

channelVersion.MajorVersionSpecified = true;

channelVersion.MajorVersion = 1;

channelVersion.IsManualAckSpecified = true;

channelVersion.IsManualAck = true;

ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();

channelVersionSearch.ChannelVersion = new ChannelVersionInfoStructure();

channelVersionSearch.ChannelVersion = channelVersion;

ChannelVersionListStructure channelVersions = client.SearchChannelVersionsPaged(channelVersionSearch);

if (channelVersions.Count == 0)

{

int channelId = client.InsertChannelVersion(channelVersion);

Console.WriteLine("Channel version created");

}

else

{

Console.WriteLine("Channel version found");

}

ChannelSubscriberInfoStructure channelSubscriber = new ChannelSubscriberInfoStructure();

channelSubscriber.ChannelVersionURN = channelVersionURN;

channelSubscriber.SubscriberAuthority = new AuthorityBasicInfoStructure();

channelSubscriber.SubscriberAuthority.AuthorityID = "100001";

channelSubscriber.SubscriberAuthority.ShortName = "VRAA";

channelSubscriber.SubscriberAuthority.Name = "VRAA";

ChannelSubscriberSearchInfoStructure channelSubscriberSearch = new ChannelSubscriberSearchInfoStructure();

channelSubscriberSearch.ChannelSubscriber = channelSubscriber;

ChannelSubscriberListStructure channelSubscriberList = client.SearchSubscribersToChannelPaged(channelSubscriberSearch);

if (channelSubscriberList.Count == 0)

{

int result = client.InsertSubscriberToChannel(channelSubscriber);

Console.WriteLine("Channel subscriber created");

}

else

{

Console.WriteLine("Channel subscriber found");

}

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageService"))

{

try

{

ProxyToService.DitService.MessageSaveStructure2 message = new ProxyToService.DitService.MessageSaveStructure2();

message.ChannelVersionURN = channelVersionURN;

BussinesMessageType bussinesMessage = new BussinesMessageType();

XmlDocument xml = new XmlDocument();

xml.Load("../../../data.xml");

bussinesMessage.Any = xml.DocumentElement;

message.Item = bussinesMessage;

message.ReceiverFilter = new ReceiverFilterArrayStructure() { Authority = new string[] { "100001" } };

bool resp = client.SendMessage2(message);

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageServiceUser"))

{

var applyTo = new Uri("https://ivis.eps.gov.lv/DIT.WebService");

var factory = client.ChannelFactory;

var other = factory.Endpoint.Behaviors.Find<ClientCredentials>();

factory.Endpoint.Behaviors.Remove(other.GetType());

var item1 = new ApplyToClientCredentials(other) { ApplyTo = applyTo };

factory.Endpoint.Behaviors.Add(item1);

try

{

client.ClientCredentials.UserName.UserName = System.Configuration.ConfigurationManager.AppSettings.Get("VraaUserName");

client.ClientCredentials.UserName.Password = System.Configuration.ConfigurationManager.AppSettings.Get("VraaPassword");

ProxyToService.DitService.MessageRequestStructure messageRequest = new ProxyToService.DitService.MessageRequestStructure();

messageRequest.Item = channelVersionURN;

messageRequest.ItemElementName = ProxyToService.DitService.ItemChoiceType.ChannelVersionURN;

MessageResponseStructure2 resp = client.ReceiveMessage2(messageRequest);

messageId = resp.Items?.FirstOrDefault()?.MessageID;

client.ConfirmationAck(Guid.Parse(messageId));

Console.WriteLine("ReceiveMessage");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

#### Liela apjoma ziņojuma nosūtīšana

Tiek pārbaudīts, vai eksistē kanāla versija. Ja tā ir, tad uz to tiek nosūtīts ziņojums.

string channelName = "SampleChannel578";

string channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";

string channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel578-v1-0-TYPE-DIK";

using (ProxyToService.DitConfiguration.DITConfigurationServiceContractClient client = new ProxyToService.DitConfiguration.DITConfigurationServiceContractClient("ConfigurationService"))

{

try

{

ProxyToService.DitConfiguration.ChannelSearchInfoStructure req = new ProxyToService.DitConfiguration.ChannelSearchInfoStructure();

req.Channel = new ProxyToService.DitConfiguration.ChannelInfoStructure();

req.Channel.ChannelURN = channelURN;

req.OnlyHeader = true;

ProxyToService.DitConfiguration.ChannelListStructure channels = Helper.InvokeIgnoreException<ChannelListStructure, CommunicationException>(() => client.SearchChannelsPaged(req, ChannelVersionStatusType.testing), "Argument exception: ChannelURN!");

ChannelInfoStructure channel = new ChannelInfoStructure();

channel.ShortName = channelName;

channel.ChannelURN = channelURN;

channel.ChannelType = ChannelType.dik;

channel.OwnerAuthority = new AuthorityBasicInfoStructure();

channel.OwnerAuthority.AuthorityID = "100000";

channel.OwnerAuthority.ShortName = "ABC";

channel.OwnerAuthority.Name = "ABC Software";

channel.ChannelName = channelName;

channel.ChannelTypeSpecified = true;

bool channelExsists = channels != null && channels.Count > 0;

if (!channelExsists)

{

int channelId = client.InsertChannel(channel);

Console.WriteLine("Channel created");

}

else

{

Console.WriteLine("Channel found");

}

ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();

channelVersion.UseReceiverFilter = true;

channelVersion.UseReceiverFilterSpecified = true;

channelVersion.MessageType = new ProxyToService.DitConfiguration.MessageTypeInfoStructure[2];

channelVersion.MessageType[0] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[0].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

// 1 Biznesa ziņojums

// 2 Transakcijas ziņojums

// 3 error

channelVersion.MessageType[0].MessageClass.CodeListCodeValue = "1";

channelVersion.MessageType[0].XMLSchemaURN = "URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName";

channelVersion.MessageType[1] = new ProxyToService.DitConfiguration.MessageTypeInfoStructure();

channelVersion.MessageType[1].MessageClass = new ProxyToService.DitConfiguration.MessageClassInfoStructure();

channelVersion.MessageType[1].MessageClass.CodeListCodeValue = "3";

channelVersion.MessageType[1].XMLSchemaURN = "URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError";

channelVersion.ChannelVersionStatus = ChannelVersionStatusType.active;

channelVersion.ChannelVersionStatusSpecified = true;

channelVersion.Channel = channel;

channelVersion.MinorVersionSpecified = true;

channelVersion.MinorVersion = 0;

channelVersion.MajorVersionSpecified = true;

channelVersion.MajorVersion = 1;

ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();

channelVersionSearch.ChannelVersion = new ChannelVersionInfoStructure();

channelVersionSearch.ChannelVersion = channelVersion;

ChannelVersionListStructure channelVersions = client.SearchChannelVersionsPaged(channelVersionSearch);

if (channelVersions.Count == 0)

{

int channelId = client.InsertChannelVersion(channelVersion);

Console.WriteLine("Channel version created");

}

else

{

Console.WriteLine("Channel version found");

}

ChannelSubscriberInfoStructure channelSubscriber = new ChannelSubscriberInfoStructure();

channelSubscriber.ChannelVersionURN = channelVersionURN;

channelSubscriber.SubscriberAuthority = new AuthorityBasicInfoStructure();

channelSubscriber.SubscriberAuthority.AuthorityID = "100001";

channelSubscriber.SubscriberAuthority.ShortName = "VRAA";

channelSubscriber.SubscriberAuthority.Name = "VRAA";

ChannelSubscriberSearchInfoStructure channelSubscriberSearch = new ChannelSubscriberSearchInfoStructure();

channelSubscriberSearch.ChannelSubscriber = channelSubscriber;

ChannelSubscriberListStructure channelSubscriberList = client.SearchSubscribersToChannelPaged(channelSubscriberSearch);

if (channelSubscriberList.Count == 0)

{

int result = client.InsertSubscriberToChannel(channelSubscriber);

Console.WriteLine("Channel subscriber created");

}

else

{

Console.WriteLine("Channel subscriber found");

}

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

}

}

using (ProxyToService.DitService.DITMessageServiceContractClient client = new ProxyToService.DitService.DITMessageServiceContractClient("MessageServiceMtom"))

{

var applyTo = new Uri("https://ivis.eps.gov.lv/DIT.WebService");

var factory = client.ChannelFactory;

var other = factory.Endpoint.Behaviors.Find<ClientCredentials>();

factory.Endpoint.Behaviors.Remove(other.GetType());

var item = new ApplyToClientCredentials(other) { ApplyTo = applyTo };

factory.Endpoint.Behaviors.Add(item);

try

{

ProxyToService.DitService.MessageSaveStructure2 message = new ProxyToService.DitService.MessageSaveStructure2();

message.ChannelVersionURN = channelVersionURN;

BussinesMessageType bussinesMessage = new BussinesMessageType();

XmlDocument xml = new XmlDocument();

xml.Load("../../../largeData.xml");

bussinesMessage.Any = xml.DocumentElement;

message.Item = bussinesMessage;

message.ReceiverFilter = new ReceiverFilterArrayStructure()

{

Authority = new[] { "100001" }

};

bool resp = client.SendMessage2(message);

Console.WriteLine("message was sent");

Console.WriteLine("Press key...");

}

catch (CommunicationException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (TimeoutException e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

}

catch (Exception e)

{

string mess = e.Message;

Console.WriteLine(mess);

client.Abort();

throw;

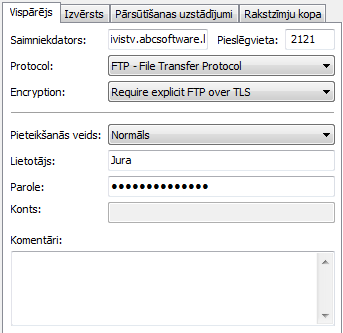
}

Console.ReadLine();

}

### FTP piemēri .NET

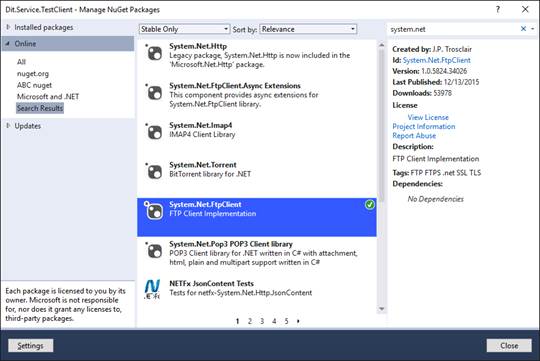
DIT FTP pieslēgumu var pārbaudīt, izmantojot FTP klientu, piemēram, *FileZilla. FileZilla ftp* kanāla konfigurācija redzama 20.attēlā.



.attēls. FileZilla FTP kanāla konfigurācija

Lai izveidotu pieslēgumu FTPs, var izmantot FTPs klientu System.Net.FTPClient, jo .net uz izstrādāšanas brīdi neļauj pilnīgi izmantot FTPs funkcijas.

FTPs pieslēgumu var izveidot, izmantojot trešās puses bibliotēkas, piemēram, System.Net.FTPClient, kas ir pieejama no NuGet ( vai no https://netftp.codeplex.com/):



.attēls. FTPs klienta pieslēgšanas piemērs

DIT FTPs klientā var izmantot arī citas bibliotēkas, atkarībā no projekta mērķiem un izstrādes metodoloģijas.

#### Datnes pievienošana (FTPs\_SendMessage)

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam:

public static class Program

{

/// <summary>

/// Mains the specified args.

/// </summary>

/// <param name="args"> The args. </param>

public static void Main(string[] args)

{

Dictionary<string, int> fileSendDic = new Dictionary<string, int>();

fileSendDic.Add("../../../DIT10MB.xml", 2);

string channelName = "SampleChannel";

foreach (var item in fileSendDic)

{

for (int i = 0; i < item.Value; i++)

{

UploadFileToFtp(new FileInfo(item.Key), channelName, i);

Console.WriteLine(i);

}

}

Console.ReadLine();

}

public static void UploadFileToFtp(FileInfo fileInfo, string channelName,int i)

{

if (fileInfo == null)

{

throw new ArgumentException("fileInfo is null");

}

var ftpAdress = System.Configuration.ConfigurationManager.AppSettings.Get("FtpAdress");

var port = System.Configuration.ConfigurationManager.AppSettings.Get("FtpPort");

var userName = System.Configuration.ConfigurationManager.AppSettings.Get("UserName");

var password = System.Configuration.ConfigurationManager.AppSettings.Get("Password");

using (var ftpClient = new FtpClient())

{

string write = "Write";

string folderName = $"{channelName}/DIK-v1-0/{write}/{i}.xml";

ftpClient.ValidateCertificate += new FtpSslValidation(myCertificateValidation);

ftpClient.Host = ftpAdress;

ftpClient.EncryptionMode = FtpEncryptionMode.Explicit;

ftpClient.Port = Convert.ToInt32(port, CultureInfo.InvariantCulture);

ftpClient.Credentials = new NetworkCredential(userName, password);

ftpClient.EnableThreadSafeDataConnections = true;

// do not open a new connection for each file

ftpClient.DataConnectionType = FtpDataConnectionType.EPSV;

ftpClient.SocketKeepAlive = true;

System.Net.ServicePointManager.Expect100Continue = false;

System.Net.ServicePointManager.SecurityProtocol = System.Net.SecurityProtocolType.Ssl3 | System.Net.SecurityProtocolType.Tls;

ftpClient.Connect();

using (var fileStream = File.OpenRead(fileInfo.FullName))

{

using (var ftpStream = ftpClient.OpenWrite(folderName, FtpDataType.ASCII))

{

var buffer = new byte[8 \* 1024];

int count;

while ((count = fileStream.Read(buffer, 0, buffer.Length)) > 0)

{

ftpStream.Write(buffer, 0, count);

}

}

}

ftpClient.Disconnect();

}

}

private static void myCertificateValidation(FtpClient control, FtpSslValidationEventArgs e)

{

e.Accept = true;

}

}

#### Datnes nolasīšana un dzēšana (FTPs\_ReceiveMessage)

Tiek izveidots pieslēgums FTP, un tiek nolasīta datne no kanāla mapes, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam:

public static class Program

{

/// <summary>

/// Mains the specified args.

/// </summary>

/// <param name="args"> The args. </param>

public static void Main(string[] args)

{

var interval = int.Parse(System.Configuration.ConfigurationManager.AppSettings.Get("Interaval"));

string ftpAdress = System.Configuration.ConfigurationManager.AppSettings.Get("FtpAdress");

int port = int.Parse(System.Configuration.ConfigurationManager.AppSettings.Get("FtpPort"));

var userName = System.Configuration.ConfigurationManager.AppSettings.Get("UserName");

var password = System.Configuration.ConfigurationManager.AppSettings.Get("Password");

var folder = System.Configuration.ConfigurationManager.AppSettings.Get("FolderTemp");

long ss = 0;

Console.WriteLine("Press Enter");

while (true)

{

try

{

Thread.Sleep(interval);

var t = Task.Factory.StartNew(() =>

{

using (var ftpClient = new System.Net.FtpClient.FtpClient())

{

ftpClient.Host = ftpAdress;

ftpClient.EncryptionMode = System.Net.FtpClient.FtpEncryptionMode.Explicit;

ftpClient.Port = Convert.ToInt32(port, CultureInfo.InvariantCulture);

ftpClient.Credentials = new NetworkCredential(userName, password);

ftpClient.ReadTimeout = 5000;

string folderName = "/SampleChannel/DIK-v1-0/ReadError";

DirectoryInfo dir = new DirectoryInfo(folder);

var ss1 = dir.GetFiles();

if (ss1 != null)

{

foreach (var item in ss1)

{

item.Delete();

}

}

ftpClient.SetWorkingDirectory(folderName);

ftpClient.ValidateCertificate += new System.Net.FtpClient.FtpSslValidation(myCertificateValidation);

foreach (var item in ftpClient.GetListing(ftpClient.GetWorkingDirectory()))

{

switch (item.Type)

{

case FtpFileSystemObjectType.Directory:

Console.WriteLine("Folder:" + item.Name);

break;

case FtpFileSystemObjectType.File:

GetFile(ftpClient, item.FullName, item.Name, dir);

Console.WriteLine("File:" + item.Name);

break;

}

}

FileInfo[] fileInfo = dir.GetFiles();

foreach (var item in fileInfo)

{

Console.WriteLine(item.Name + " received");

}

ss++;

Console.WriteLine(ss);

}

});

t.Wait();

}

catch (Exception e)

{

Thread.Sleep(3000);

string mess = e.Message;

Console.WriteLine(mess);

}

}

}

/// <summary>

/// Gets the file.

/// </summary>

/// <param name="ftpClient"> The FTP client. </param>

/// <param name="fileName"> Name of the file. </param>

/// <param name="dirToSave"> The dir to save. </param>

private static void GetFile(System.Net.FtpClient.FtpClient ftpClient, string fileNameFullName, string fileName, DirectoryInfo dirToSave)

{

string userID = WindowsIdentity.GetCurrent().Name;

using (var ftpStream = ftpClient.OpenRead(fileNameFullName))

using (var fileStream = new FileStream(new FileInfo(dirToSave.FullName + fileName).FullName, FileMode.OpenOrCreate))

{

var buffer = new byte[8 \* 1024];

int count;

while ((count = ftpStream.Read(buffer, 0, buffer.Length)) > 0)

{

fileStream.Write(buffer, 0, count);

}

}

}

// callback used to validate the certificate in an SSL conversation

private static void myCertificateValidation(System.Net.FtpClient.FtpClient control, System.Net.FtpClient.FtpSslValidationEventArgs e)

{

e.Accept = true;

}

}

#### Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam konkrētajam saņēmējam:

public static class Program

{

/// <summary>

/// Mains the specified args.

/// </summary>

/// <param name="args"> The args. </param>

public static void Main(string[] args)

{

Dictionary<string, int> fileSendDic = new Dictionary<string, int>();

fileSendDic.Add("../../../DIT10MB.xml", 2);

string authority = "100001";

string channelName = "SampleChannel7";

foreach (var item in fileSendDic)

{

for (int i = 0; i < item.Value; i++)

{

UploadFileToFtp(new FileInfo(item.Key), channelName, i, authority);

Console.WriteLine(i);

}

}

Console.ReadLine();

}

public static void UploadFileToFtp(FileInfo fileInfo, string channelName, int i, string authority)

{

if (fileInfo == null)

{

throw new ArgumentException("fileInfo is null");

}

var ftpAdress = System.Configuration.ConfigurationManager.AppSettings.Get("FtpAdress");

var port = System.Configuration.ConfigurationManager.AppSettings.Get("FtpPort");

var userName = System.Configuration.ConfigurationManager.AppSettings.Get("UserName");

var password = System.Configuration.ConfigurationManager.AppSettings.Get("Password");

using (var ftpClient = new FtpClient())

{

string write = "Write";

string folderName = $"{channelName}/DIK-v1-0/{write}/{authority}/{i}.xml";

ftpClient.ValidateCertificate += new FtpSslValidation(myCertificateValidation);

ftpClient.Host = ftpAdress;

ftpClient.EncryptionMode = FtpEncryptionMode.Explicit;

ftpClient.Port = Convert.ToInt32(port, CultureInfo.InvariantCulture);

ftpClient.Credentials = new NetworkCredential(userName, password);

ftpClient.EnableThreadSafeDataConnections = true;

// do not open a new connection for each file

ftpClient.DataConnectionType = FtpDataConnectionType.EPSV;

ftpClient.SocketKeepAlive = true;

System.Net.ServicePointManager.Expect100Continue = false;

System.Net.ServicePointManager.SecurityProtocol = System.Net.SecurityProtocolType.Ssl3 | System.Net.SecurityProtocolType.Tls;

ftpClient.Connect();

using (var fileStream = File.OpenRead(fileInfo.FullName))

{

using (var ftpStream = ftpClient.OpenWrite(folderName, FtpDataType.ASCII))

{

var buffer = new byte[8 \* 1024];

int count;

while ((count = fileStream.Read(buffer, 0, buffer.Length)) > 0)

{

ftpStream.Write(buffer, 0, count);

}

}

}

ftpClient.Disconnect();

}

}

private static void myCertificateValidation(FtpClient control, FtpSslValidationEventArgs e)

{

e.Accept = true;

}

}

#### Atbildes ziņojuma nosūtīšana

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam zem WriteEventError:

public class Program

{

/// <summary>

/// Mains the specified args.

/// </summary>

/// <param name="args"> The args. </param>

public static void Main(string[] args)

{

Dictionary<string, int> fileSendDic = new Dictionary<string, int>();

fileSendDic.Add("../../../kluda.xml", 2);

string authority = "100001";

string channelName = "SampleChannel4";

foreach (var item in fileSendDic)

{

for (int i = 0; i < item.Value; i++)

{

UploadFileToFtp(new FileInfo(item.Key), channelName, i, authority);

Console.WriteLine(i);

}

}

Console.ReadLine();

}

public static void UploadFileToFtp(FileInfo fileInfo, string channelName, int i, string authority)

{

if (fileInfo == null)

{

throw new ArgumentException("fileInfo is null");

}

var ftpAdress = System.Configuration.ConfigurationManager.AppSettings.Get("FtpAdress");

var port = System.Configuration.ConfigurationManager.AppSettings.Get("FtpPort");

var userName = System.Configuration.ConfigurationManager.AppSettings.Get("VraaUserName");

var password = System.Configuration.ConfigurationManager.AppSettings.Get("VraaPassword");

using (var ftpClient = new FtpClient())

{

string writeEvent = "WriteEvent";

string writeEventError = "WriteEventError";

string folderName = $"{channelName}/DIK-v1-0/{writeEvent}/{writeEventError}/{i}.xml";

ftpClient.ValidateCertificate += new FtpSslValidation(myCertificateValidation);

ftpClient.Host = ftpAdress;

ftpClient.EncryptionMode = FtpEncryptionMode.Explicit;

ftpClient.Port = Convert.ToInt32(port, CultureInfo.InvariantCulture);

ftpClient.Credentials = new NetworkCredential(userName, password);

ftpClient.EnableThreadSafeDataConnections = true;

// do not open a new connection for each file

ftpClient.DataConnectionType = FtpDataConnectionType.EPSV;

ftpClient.SocketKeepAlive = true;

System.Net.ServicePointManager.Expect100Continue = false;

System.Net.ServicePointManager.SecurityProtocol = System.Net.SecurityProtocolType.Ssl3 | System.Net.SecurityProtocolType.Tls;

ftpClient.Connect();

using (var fileStream = File.OpenRead(fileInfo.FullName))

{

using (var ftpStream = ftpClient.OpenWrite(folderName, FtpDataType.ASCII))

{

var buffer = new byte[8 \* 1024];

int count;

while ((count = fileStream.Read(buffer, 0, buffer.Length)) > 0)

{

ftpStream.Write(buffer, 0, count);

}

}

}

ftpClient.Disconnect();

}

}

private static void myCertificateValidation(FtpClient control, FtpSslValidationEventArgs e)

{

e.Accept = true;

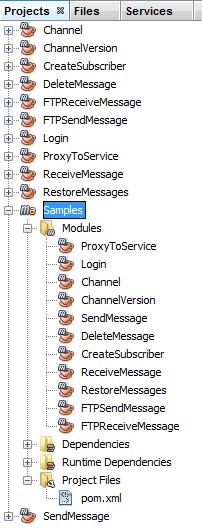
}

}

# Java

## Projekta izveidošana Java vidē

Projekts ir izveidots kā *Maven* daudzmoduļu projekts, tāpēc izstrādes videi ir jābūt *Maven* atbalstam. Galvenais projekts ir „Samples”, kurš satur visus pārējos piemēru projektus.



.attēls. Piemēru projektu struktūra Netbeans vidē

Savukārt projekts *ProxyToService* satur konfigurāciju, kas ir kopīga visiem piemēriem, tajā skaitā DIT servisu klientu. Projekta POM.xml konfigurācijas datnē tiek norādītas WSDL datņu atrašanās vietas, tiem servisiem, kuriem ir jāģenerē klienta kods.

<plugin>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-codegen-plugin</artifactId>

<version>${cxf.version}</version>

<executions>

<execution>

<id>generate-sources</id>

<phase>generate-sources</phase>

<configuration>

<fork>true</fork>

<wsdlOptions>

<wsdlOption> <wsdl>src/main/resources/files/ConfigurationService.wsdl</wsdl>

</wsdlOption>

<wsdlOption>

<wsdl>src/main/resources/files/WcfService.wsdl</wsdl>

</wsdlOption>

</wsdlOptions>

</configuration>

<goals>

<goal>wsdl2java</goal>

</goals>

</execution>

</executions>

</plugin>

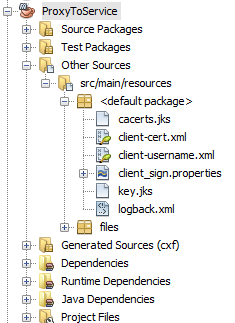
## Piemēri Java

Nodaļā ir aprakstīti piemēri DIT saskarnes izmantošanai, tai skaita:

* DIT datu izgūšanas tīkla serviss;
* DIT iestādes pasta kastītes un kanālu konfigurācijas serviss;
* DIT FTP saskarnes.

### Tīkla servisu piemēri

*Projekts ProxyToService* satur kopīgu piemēru konfigurāciju, kuru veido šādas datnes:



.attēls. Konfigurācijas datnes

1. cacerts.jks – Java atslēgu krātuve (*keystore*), kas satur servera sertifikātus;
2. client-cert.xml – konfigurācijas datne savienojumam ar klienta sertifikātu;
3. client-username.xml – konfigurācijas datne savienojumam ar lietotājvārdu un paroli;
4. client\_sign.properties – klienta sertifikāta konfigurācijas datne;
5. vissTest.pfx – satur klienta sertifikātu (privāto atslēgu);
6. logback.xml – SL4J žurnalēšanas konfigurācijas datne.

DIT servisa klients tiek realizēts, izmantojot *Apache CXF* bibliotēkas.

#### Pieslēgšanās ar sertifikātu

##### Klienta sertifikāta konfigurācija

Pieslēgšanās ar sertifikātu konfigurācija ir aprakstīta datnē *client-cert.xml* un *client\_sign.properties*.

Datnes *client\_sign.properties* saturs:

org.apache.ws.security.crypto.provider=org.apache.ws.security.components.crypto.Merlin

org.apache.ws.security.crypto.merlin.keystore.type=pkcs12

org.apache.ws.security.crypto.merlin.keystore.password=**viss**

org.apache.ws.security.crypto.merlin.keystore.alias=**f0288df6-5cfb-433b-95ca-6c4f29b4f550**

org.apache.ws.security.crypto.merlin.file=**vissTest.pfx**

Trīs parametri:

1. org.apache.ws.security.crypto.merlin.keystore.password – atslēgu krātuves parole;
2. org.apache.ws.security.crypto.merlin.keystore.alias – privātās atslēgas nosaukums atslēgu krātuvē (alias);
3. org.apache.ws.security.crypto.merlin.file – atslēgu krātuves datnes nosaukums

##### CallbackHandler klase

*CallbackHandler* klase ir nepieciešama, lai varētu servisam nodot klienta atslēgas paroli. Tālāk redzamā piemērā atslēgas parole tiek saņemta no CXF konfigurācijas datnes:

**package** lv.abcsoftware.eves.samples.sts;

**import** java.io.IOException;

**import** javax.security.auth.callback.Callback;

**import** javax.security.auth.callback.CallbackHandler;

**import** javax.security.auth.callback.UnsupportedCallbackException;

**import** org.apache.ws.security.WSPasswordCallback;

**public** **class** KeystorePasswordCallback **implements** CallbackHandler{

**private** String keyPassword;

**public** **void** setKeyPassword(String keyPassword) {

**this**.keyPassword = keyPassword;

}

@Override

**public** **void** handle(Callback[] callbacks) **throws** IOException, UnsupportedCallbackException {

**for** (**int** i = 0; i < callbacks.length; i++) {

WSPasswordCallback pc = (WSPasswordCallback)callbacks[i];

pc.setPassword(**this**.keyPassword);

}

}

}

##### Kods

**package** lv.abcsoftware.eves.samples.login;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SelectPostBoxRequest;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SelectPostBoxResponse;

**import** org.apache.cxf.BusFactory;

**import** org.apache.cxf.bus.spring.SpringBusFactory;

**public** **class** LoginCert {

**public** **static** **void** main(String[] args) {

//Izveido CXF klientu

SpringBusFactory bf = **new** SpringBusFactory();

BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));

//Izveido DIT konfigurācijas servisa klientu

DITConfigurationService service = **new** DITConfigurationService();

IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();

SelectPostBoxRequest request = **new** SelectPostBoxRequest();

request.setAuthorityId("100001");

SelectPostBoxResponse response = port.selectPostBox(request);

System.*out*.println(response.getPostBox().getPostBoxID());

}

}

#### Pieslēgšanās ar paroli

Pieslēgšanās ar sertifikātu konfigurācija ir aprakstīta datnē *client-username.xml*

##### Kods

**package** lv.abcsoftware.eves.samples.login;

**import** java.util.Map;

**import** javax.xml.ws.BindingProvider;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SelectPostBoxRequest;

**import** lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SelectPostBoxResponse;

**import** org.apache.cxf.BusFactory;

**import** org.apache.cxf.bus.spring.SpringBusFactory;

**public** **class** LoginUserName {

**public** **static** **void** main(String[] args) {

SpringBusFactory bf = **new** SpringBusFactory();

BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-username.xml"));

DITConfigurationService service = **new** DITConfigurationService();

IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();

Map<String, Object> ctx = ((BindingProvider) port).getRequestContext();

ctx.put("ws-security.username", "piemeru\_tests");

ctx.put("ws-security.password", "password");

SelectPostBoxRequest request = **new** SelectPostBoxRequest();

request.setAuthorityId("100001");

SelectPostBoxResponse response = port.selectPostBox(request);

System.*out*.println(response.getPostBox().getPostBoxID());

}

}

#### Kanāla pārbaude un izveidošana (SelectAndCreateChannel)

Piemērā tiek pārbaudīts, vai eksistē kanāls. Ja kanāla nav, tad tiek izveidots jauns:

package lv.abcsoftware.eves.samples.channel;  
  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannel;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SendMessageResponse;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelType;  
import lv.gov.eps.ivis.xmlschemas.\_100001.ivis.v1\_0.AuthorityBasicInfoStructure;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
  
public class SelectAndCreateChannel {  
  
 public static void main(String[] args) {  
 SpringBusFactory bf = new SpringBusFactory();   
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITConfigurationService service = new DITConfigurationService();  
 IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 ChannelSearchInfoStructure req = new ChannelSearchInfoStructure();  
  
 req.setChannel(new ChannelInfoStructure());  
 req.getChannel().setChannelURN(channelURN);  
 req.setOnlyHeader(true);  
  
 SearchChannelsPaged searchChannelsPaged = new SearchChannelsPaged();  
 searchChannelsPaged.setChannelSearch(req);  
 SendMessageResponse sendMessageResponse = port.searchChannelsPaged(searchChannelsPaged);  
   
 if (sendMessageResponse.getChannelList().getCount() == 0) {  
 System.out.println("Channel creation..");  
  
 ChannelInfoStructure channel = new ChannelInfoStructure();  
 channel.setShortName(channelName);  
 channel.setChannelURN(channelURN);  
 channel.setChannelType(ChannelType.DIK);  
 channel.setOwnerAuthority(new AuthorityBasicInfoStructure());  
 channel.getOwnerAuthority().setAuthorityID("100000");  
 channel.setChannelName(channelName);  
   
 InsertChannel insertChannel = new InsertChannel();  
 insertChannel.setChannel(channel);  
 InsertChannelResponse insertChannelResponse = port.insertChannel(insertChannel);  
   
 int id = insertChannelResponse.getInsertChannelResponse();   
  
 System.out.println("Channel created " + id);  
 } else {  
 System.out.println("Channel found");  
 }  
  
 }  
}



.attēls. Kanāla izveidošana

#### Kanāla versijas pārbaude un izveidošana (SelectAndCreateChannelVersion)

Piemērā tiek pārbaudīts, vai eksistē kanāls. Ja tā nav, tad tiek izveidots jauns kanāls. Kad ir izveidots kanāls, tiek pārbaudīts vai eksistē kanāla versija; ja tās nav, tad sāk tās veidošanu.

package lv.abcsoftware.eves.samples.channelversion;  
  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannel;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelVersion;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelVersionResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPagedResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SendMessageResponse;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelType;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageClassInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageTypeInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100001.ivis.v1\_0.AuthorityBasicInfoStructure;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
  
public class SelectAndCreateChannelVersion {  
  
 public static void main(String[] args) {  
 SpringBusFactory bf = new SpringBusFactory();   
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITConfigurationService service = new DITConfigurationService();  
 IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 ChannelSearchInfoStructure req = new ChannelSearchInfoStructure();  
  
 req.setChannel(new ChannelInfoStructure());  
 req.getChannel().setChannelURN(channelURN);  
 req.setOnlyHeader(true);  
  
 SearchChannelsPaged searchChannelsPaged = new SearchChannelsPaged();  
 searchChannelsPaged.setChannelSearch(req);  
 SendMessageResponse sendMessageResponse = port.searchChannelsPaged(searchChannelsPaged);  
  
 ChannelInfoStructure channel = new ChannelInfoStructure();  
 channel.setShortName(channelName);  
 channel.setChannelURN(channelURN);  
 channel.setChannelType(ChannelType.DIK);  
 channel.setOwnerAuthority(new AuthorityBasicInfoStructure());  
 channel.getOwnerAuthority().setAuthorityID("100000");  
 channel.setChannelName(channelName);  
  
 if (sendMessageResponse.getChannelList().getCount() == 0) {  
 InsertChannel insertChannel = new InsertChannel();  
 insertChannel.setChannel(channel);  
 InsertChannelResponse insertChannelResponse = port.insertChannel(insertChannel);  
  
 int id = insertChannelResponse.getInsertChannelResponse();  
 System.out.println("Channel created " + id);  
 } else {  
 System.out.println("Channel found");  
 }  
  
 ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();  
  
 // 1 Biznesa ziņojums  
 // 2 Transakcijas ziņojums   
 // 3 error  
  
 MessageTypeInfoStructure messageType = new MessageTypeInfoStructure();  
 messageType.setMessageClass(new MessageClassInfoStructure());  
 messageType.getMessageClass().setCodeListCodeValue("1");  
 messageType.setXMLSchemaURN("URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName");  
  
 MessageTypeInfoStructure messageType1 = new MessageTypeInfoStructure();  
 messageType1.setMessageClass(new MessageClassInfoStructure());  
 messageType1.getMessageClass().setCodeListCodeValue("3");  
 messageType1.setXMLSchemaURN("URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError");  
  
 channelVersion.getMessageType().add(messageType);  
 channelVersion.getMessageType().add(messageType1);  
  
 channelVersion.getChannelVersionStatus().add("active");  
 // channelVersion.setChannelVersionStatus();  
 // channelVersion.ChannelVersionStatusSpecified = true;  
  
 channelVersion.setChannel(channel);  
 channelVersion.setMinorVersion((short) 0);  
 channelVersion.setMajorVersion((short) 1);  
  
 ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();  
 channelVersionSearch.setChannelVersion(channelVersion);  
  
 SearchChannelVersionsPaged searchChannelVersionsPaged = new SearchChannelVersionsPaged();  
 searchChannelVersionsPaged.setChannelVersionSearch(channelVersionSearch);  
  
 SearchChannelVersionsPagedResponse channelVersions = port.searchChannelVersionsPaged(searchChannelVersionsPaged);  
  
 if (channelVersions.getChannelVersionsList().getCount() == 0) {  
 InsertChannelVersion insertChannelVersion = new InsertChannelVersion();  
 insertChannelVersion.setChannelVersionData(channelVersion);  
 InsertChannelVersionResponse insertChannelVersionResponse = port.insertChannelVersion(insertChannelVersion);  
 int channelId = insertChannelVersionResponse.getInsertChannelVersionResponse();  
 System.out.println("Channel version created");  
 } else {  
 System.out.println("Channel version found");  
 }  
  
 }  
}



.attēls. Kanāla versijas izveidošana

#### Ziņojuma nosūtīšana (SendMessageToChannelVersion)

Tiek pārbaudīts, vai eksistē kanāla versija, ja ir, tad tiek nosūtīts ziņojums uz to.

package lv.abcsoftware.eves.samples.sendmessage;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.ws.BindingProvider;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannel;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelVersion;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPagedResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SendMessageResponse;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.IDITMessageServiceContract;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessage;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DITMessageService;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.BussinesMessageType;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelType;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageClassInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageSaveStructure2;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageTypeInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100001.ivis.v1\_0.AuthorityBasicInfoStructure;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.xml.sax.SAXException;  
  
public class SendMessageToChannelVersion {  
  
 public static void main(String[] args) {  
  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITConfigurationService service = new DITConfigurationService();  
 IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();  
 Map<String, Object> ctx = ((BindingProvider) port).getRequestContext();  
 ctx.put("ws-security.username", "DitAdmin");  
 ctx.put("ws-security.password", "Password");  
  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 String channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel-v1-0-TYPE-DIK";  
  
 SearchChannelsPaged searchChannelsPaged = new SearchChannelsPaged();  
 ChannelSearchInfoStructure request = new ChannelSearchInfoStructure();  
 searchChannelsPaged.setChannelSearch(request);  
  
 ChannelInfoStructure channel = new ChannelInfoStructure();  
 request.setChannel(channel);  
  
 channel.setChannelURN(channelURN);  
 request.setOnlyHeader(true);  
  
 SendMessageResponse sendMessageResponse = port.searchChannelsPaged(searchChannelsPaged);  
  
 ChannelInfoStructure newChannel = new ChannelInfoStructure();  
 newChannel.setShortName(channelName);  
 newChannel.setChannelURN(channelURN);  
 newChannel.setChannelType(ChannelType.DIK);  
 newChannel.setOwnerAuthority(new AuthorityBasicInfoStructure());  
 newChannel.getOwnerAuthority().setAuthorityID("100000");  
 newChannel.setChannelName(channelName);  
  
 if (sendMessageResponse.getChannelList().getCount() == 0) {  
 InsertChannel insertChannel = new InsertChannel();  
 insertChannel.setChannel(channel);  
 int channelId = port.insertChannel(insertChannel).getInsertChannelResponse();  
 System.out.println("Channel created");  
 } else {  
 System.out.println("Channel found");  
 }  
  
 ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();  
  
 MessageTypeInfoStructure messageType = new MessageTypeInfoStructure();  
 messageType.setMessageClass(new MessageClassInfoStructure());  
 messageType.getMessageClass().setCodeListCodeValue("1");  
 messageType.setXMLSchemaURN("URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName");  
  
 MessageTypeInfoStructure messageType1 = new MessageTypeInfoStructure();  
 messageType1.setMessageClass(new MessageClassInfoStructure());  
 messageType1.getMessageClass().setCodeListCodeValue("3");  
 messageType1.setXMLSchemaURN("URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError");  
  
 channelVersion.getMessageType().add(messageType);  
 channelVersion.getMessageType().add(messageType1);  
  
  
 channelVersion.getChannelVersionStatus().add("active");  
  
 channelVersion.setChannel(channel);  
 channelVersion.setMinorVersion((short) 0);  
 channelVersion.setMajorVersion((short) 1);  
  
 ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();  
 channelVersionSearch.setChannelVersion(channelVersion);  
  
 SearchChannelVersionsPaged searchChannelVersionsPaged = new SearchChannelVersionsPaged();  
 searchChannelVersionsPaged.setChannelVersionSearch(channelVersionSearch);  
 SearchChannelVersionsPagedResponse channelVersions = port.searchChannelVersionsPaged(searchChannelVersionsPaged);  
  
 if (channelVersions.getChannelVersionsList().getCount() == 0) {  
 InsertChannelVersion insertChannelVersion = new InsertChannelVersion();  
 insertChannelVersion.setChannelVersionData(channelVersion);  
 int channelId = port.insertChannelVersion(insertChannelVersion).getInsertChannelVersionResponse();  
 System.out.println("Channel version created");  
 } else {  
 System.out.println("Channel version found");  
 }  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 /\*ditCtx.put("ws-security.username", "DitAdmin");  
 ditCtx.put("ws-security.password", "password");\*/  
  
  
 MessageSaveStructure2 message = new MessageSaveStructure2();  
 message.setChannelVersionURN(channelVersionURN);  
 BussinesMessageType bussinesMessage = new BussinesMessageType();  
 message.setBussinesMessage(bussinesMessage);  
  
 Object obj = getObject();  
 bussinesMessage.setAny(obj);  
  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.setMessageSave2(message);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessageResponse sendMessageResponse1 = ditClient.sendMessage2(sendMessage);  
 System.out.println("Message Sent : " + sendMessageResponse1.isSendMessageResult());  
  
 }  
  
 static Object getObject() {  
 try {  
 File fXmlFile = new File("data.xml");  
  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(fXmlFile);  
 doc.getDocumentElement().normalize();  
  
 Element e = doc.getDocumentElement();  
 return e;  
  
 } catch (SAXException ex) {  
 Logger.getLogger(SendMessageToChannelVersion.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(SendMessageToChannelVersion.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (ParserConfigurationException ex) {  
 Logger.getLogger(SendMessageToChannelVersion.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 return null;  
 }  
}



.attēls. Izveidotais ziņojums

#### Ziņojumu dzēšana (MessageDelete)

No kanāla tiek dzēsti ziņojumi:

package lv.abcsoftware.eves.samples.deletemessage;  
  
import java.util.Map;  
import javax.xml.ws.BindingProvider;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DeleteMessageResponse;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.IDITMessageServiceContract;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DITMessageService;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageRequestStructure;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
  
public class DeleteMessage {  
  
 public static void main(String[] args) {  
 SpringBusFactory bf = new SpringBusFactory();   
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 String channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 /\*ditCtx.put("ws-security.username", "username3");  
 ditCtx.put("ws-security.password", "password");\*/  
  
 MessageRequestStructure messDelete = new MessageRequestStructure();  
 messDelete.setChannelURN(channelURN);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.DeleteMessage deleteRequest = new lv.gov.eps.ivis.iss.ditservice.v1\_0.DeleteMessage();  
 deleteRequest.setMessageRequest(messDelete);  
  
  
 DeleteMessageResponse deleteMessageResponse = ditClient.deleteMessage(deleteRequest);  
 System.out.println("Message deleted result = " + deleteMessageResponse.isDeleteMessageResult());   
 }  
}



.attēls. Dzēstie ziņojumi

#### Pievienot pierakstītāju kanāla versijai (CreateSubscriberToChannelVersion)

Tiek pārbaudīts vai eksistē pierakstītājs kanāla; ja tā nav, tad pievieno pierakstītāju.

package lv.abcsoftware.eves.samples.createsubscriber;  
  
import java.util.Map;  
import javax.xml.ws.BindingProvider;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannel;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelVersion;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelVersionResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertSubscriberToChannel;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPagedResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchSubscribersToChannelPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchSubscribersToChannelPagedResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SendMessageResponse;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelSubscriberInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelSubscriberSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelType;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageClassInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageTypeInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100001.ivis.v1\_0.AuthorityBasicInfoStructure;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
  
public class CreateSubscriberToChannelVersion {  
  
 public static void main(String[] args) {  
 SpringBusFactory bf = new SpringBusFactory();   
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 String channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel-v1-0-TYPE-DIK";  
  
 DITConfigurationService service = new DITConfigurationService();  
 IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();  
  
 Map<String, Object> ctx = ((BindingProvider) port).getRequestContext();  
 /\*ctx.put("ws-security.username", "DitAdmin");  
 ctx.put("ws-security.password", "password");\*/  
  
 ChannelSearchInfoStructure req = new ChannelSearchInfoStructure();  
  
 req.setChannel(new ChannelInfoStructure());  
 req.getChannel().setChannelURN(channelURN);  
 req.setOnlyHeader(true);  
  
 SearchChannelsPaged searchChannelsPaged = new SearchChannelsPaged();  
 searchChannelsPaged.setChannelSearch(req);  
 SendMessageResponse sendMessageResponse = port.searchChannelsPaged(searchChannelsPaged);  
  
 ChannelInfoStructure channel = new ChannelInfoStructure();  
 channel.setShortName(channelName);  
 channel.setChannelURN(channelURN);  
 channel.setChannelType(ChannelType.DIK);  
 channel.setOwnerAuthority(new AuthorityBasicInfoStructure());  
 channel.getOwnerAuthority().setAuthorityID("100000");  
 channel.setChannelName(channelName);  
  
 if (sendMessageResponse.getChannelList().getCount() == 0) {  
 InsertChannel insertChannel = new InsertChannel();  
 insertChannel.setChannel(channel);  
 InsertChannelResponse insertChannelResponse = port.insertChannel(insertChannel);  
  
 int id = insertChannelResponse.getInsertChannelResponse();  
 System.out.println("Channel created " + id);  
 } else {  
 System.out.println("Channel found");  
 }  
  
 ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();  
  
 // 1 Biznesa ziņojums  
 // 2 Transakcijas ziņojums   
 // 3 error  
  
 MessageTypeInfoStructure messageType = new MessageTypeInfoStructure();  
 messageType.setMessageClass(new MessageClassInfoStructure());  
 messageType.getMessageClass().setCodeListCodeValue("1");  
 messageType.setXMLSchemaURN("URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName");  
  
 MessageTypeInfoStructure messageType1 = new MessageTypeInfoStructure();  
 messageType1.setMessageClass(new MessageClassInfoStructure());  
 messageType1.getMessageClass().setCodeListCodeValue("3");  
 messageType1.setXMLSchemaURN("URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError");  
  
 channelVersion.getMessageType().add(messageType);  
 channelVersion.getMessageType().add(messageType1);  
  
 channelVersion.getChannelVersionStatus().add("active");  
 // channelVersion.setChannelVersionStatus();  
 // channelVersion.ChannelVersionStatusSpecified = true;  
  
 channelVersion.setChannel(channel);  
 channelVersion.setMinorVersion((short) 0);  
 channelVersion.setMajorVersion((short) 1);  
  
 ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();  
 channelVersionSearch.setChannelVersion(channelVersion);  
  
 SearchChannelVersionsPaged searchChannelVersionsPaged = new SearchChannelVersionsPaged();  
 searchChannelVersionsPaged.setChannelVersionSearch(channelVersionSearch);  
  
 SearchChannelVersionsPagedResponse channelVersions = port.searchChannelVersionsPaged(searchChannelVersionsPaged);  
  
 if (channelVersions.getChannelVersionsList().getCount() == 0) {  
 InsertChannelVersion insertChannelVersion = new InsertChannelVersion();  
 insertChannelVersion.setChannelVersionData(channelVersion);  
 InsertChannelVersionResponse insertChannelVersionResponse = port.insertChannelVersion(insertChannelVersion);  
 int channelId = insertChannelVersionResponse.getInsertChannelVersionResponse();  
 System.out.println("Channel version created");  
 } else {  
 System.out.println("Channel version found");  
 }  
  
 ChannelSubscriberInfoStructure channelSubscriber = new ChannelSubscriberInfoStructure();  
 channelSubscriber.setChannelVersionURN(channelVersionURN);  
 channelSubscriber.setSubscriberAuthority(new AuthorityBasicInfoStructure());  
 channelSubscriber.getSubscriberAuthority().setAuthorityID("100001");  
 ChannelSubscriberSearchInfoStructure channelSubscriberSearch = new ChannelSubscriberSearchInfoStructure();  
 channelSubscriberSearch.setChannelSubscriber(channelSubscriber);  
  
 SearchSubscribersToChannelPaged request = new SearchSubscribersToChannelPaged();  
 request.setChannelSubscriberSearch(channelSubscriberSearch);  
 SearchSubscribersToChannelPagedResponse searchSubscribersToChannelPagedResponse = port.searchSubscribersToChannelPaged(request);  
  
 if (searchSubscribersToChannelPagedResponse.getChannelSubscriberList().getCount() == 0) {  
 InsertSubscriberToChannel insertRequest = new InsertSubscriberToChannel();  
 insertRequest.setChannelSubscriber(channelSubscriber);  
 int result = port.insertSubscriberToChannel(insertRequest).getInsertSubscriberToChannelResponse();  
 System.out.println("Channel subscriber created status: " + result );  
 } else {  
 System.out.println("Channel subscriber found");  
 }  
 }  
}



.attēls. Iestāde pieslēgta DIT

#### Ziņojuma nolasīšana (ReceiveMessage)

Izveidojot DIT kanāla struktūru, var saņemt ziņojumus ar lietotāju, kuram būtu pierakstītāja pieslēgums.



.attēls. DIK kanāla izmantošanas shēma

Pēc nolasīšanas ziņojumu vajag dzēst.

package lv.abcsoftware.eves.samples.receivemessage;  
  
import java.util.Map;  
import javax.xml.ws.BindingProvider;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DeleteMessage;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.IDITMessageServiceContract;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessageResponse;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DITMessageService;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageRequestStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageType;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
  
public class ReceiveMessage {  
  
 public static void main(String[] args) {  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 String channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 /\*ditCtx.put("ws-security.username", "username3");  
 ditCtx.put("ws-security.password", "password");\*/  
  
 MessageRequestStructure messageRequest = new MessageRequestStructure();  
 messageRequest.setChannelURN(channelURN);  
 messageRequest.setChannelVersionURN(channelVersionURN);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage request = new lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage();  
 request.setMessageRequest(messageRequest);  
  
 ReceiveMessageResponse response = ditClient.receiveMessage2(request);  
 System.out.println("Message received: " + response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage());  
  
 if (response != null && !response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage().isEmpty()) {  
 for (MessageType mt : response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage()) {  
 System.out.println("message receved id: " + mt.getMessageID());  
 MessageRequestStructure messageRequestDelete = new MessageRequestStructure();  
 messageRequestDelete.setMessageID(mt.getMessageID());  
 DeleteMessage deleteMessageRequest = new DeleteMessage();  
 deleteMessageRequest.setMessageRequest(messageRequest);  
 ditClient.deleteMessage(deleteMessageRequest);  
 System.out.println("message deleted id:" + mt.getMessageID());  
 }  
 }  
 }  
}

#### Ziņojuma atjaunošana (Restore Messages)

Izveidojot DIT kanāla struktūru, var saņemt ziņojumus ar lietotāju, kuram būtu pierakstītāja pieslēgums.

Gadījumā, ja ziņojuma saņemšana bija nekorekta, tad to var atjaunot, lietojot metodes secībā:

1. ReceiveMessageList;
2. RestoreMessageList (ReadMessage=true);
3. ReceveMessage;
4. DeleteMessage.



.attēls. DIK kanāla izmantošanas shēma, par ziņojumu atjaunošanu

package lv.abcsoftware.eves.samples.restoremessages;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.ws.BindingProvider;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DeleteMessage;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.IDITMessageServiceContract;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessageList;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessageListResponse;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessageResponse;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.RestoreMessageList;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessage;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DITMessageService;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.BussinesMessageType;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageListRequestStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageRequestStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageSaveStructure2;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageType;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.xml.sax.SAXException;  
  
public class RestoreMessages {  
  
 public static void main(String[] args) {  
 String channelName = "SampleChannel7";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 String channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";  
  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("../ProxyToService/src/main/resources/client-username.xml"));  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 ditCtx.put("ws-security.username", "DitAdmin");  
 ditCtx.put("ws-security.password", "password");  
  
  
 MessageSaveStructure2 message = new MessageSaveStructure2();  
 message.setChannelVersionURN(channelVersionURN);  
 BussinesMessageType bussinesMessage = new BussinesMessageType();  
 message.setBussinesMessage(bussinesMessage);  
  
 Object obj = getObject();  
 bussinesMessage.setAny(obj);  
  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.setMessageSave2(message);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessageResponse sendMessageResponse1 = ditClient.sendMessage2(sendMessage);  
 System.out.println("Message Sent : " + sendMessageResponse1.isSendMessageResult());  
  
 ((BindingProvider) ditClient).getRequestContext().put("ws-security.username", "username3");  
 ((BindingProvider) ditClient).getRequestContext().put("ws-security.password", "password");  
  
 MessageRequestStructure messageRequest = new MessageRequestStructure();  
 messageRequest.setChannelURN(channelURN);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage request = new lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage();  
 request.setMessageRequest(messageRequest);  
  
 ReceiveMessageResponse response = ditClient.receiveMessage2(request);  
 System.out.println("Message received: " + response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage());  
  
 MessageListRequestStructure messageListRequest = new MessageListRequestStructure();  
 messageListRequest.setChannelVersionURN(channelVersionURN);  
 messageListRequest.setReadMessage(true);  
  
 ReceiveMessageList receiveMessageList = new ReceiveMessageList();  
 receiveMessageList.setMessageListRequest(messageListRequest);  
  
 ReceiveMessageListResponse receiveMessageListResponse = ditClient.receiveMessageList(receiveMessageList);  
 System.out.println("ReceiveMessageList");  
  
 if (receiveMessageListResponse.getMessageBasicInfoList() != null) {   
 RestoreMessageList restoreMessageList = new RestoreMessageList();  
 restoreMessageList.setMessageBasicInfoList(receiveMessageListResponse.getMessageBasicInfoList());  
 ditClient.restoreMessageList(restoreMessageList);  
 }  
   
 response = ditClient.receiveMessage2(request);  
  
 if (response != null && !response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage().isEmpty()) {  
 for (MessageType mt : response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage()) {  
 System.out.println("message receved id: " + mt.getMessageID());  
 MessageRequestStructure messageRequestDelete = new MessageRequestStructure();  
 messageRequestDelete.setMessageID(mt.getMessageID());  
 DeleteMessage deleteMessageRequest = new DeleteMessage();  
 deleteMessageRequest.setMessageRequest(messageRequest);  
 ditClient.deleteMessage(deleteMessageRequest);  
 System.out.println("message deleted id:" + mt.getMessageID());  
 }  
 }  
 }  
  
 static Object getObject() {  
 try {  
 File fXmlFile = new File("../data.xml");  
  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(fXmlFile);  
 doc.getDocumentElement().normalize();  
  
 Element e = doc.getDocumentElement();  
 return e;  
  
 } catch (SAXException ex) {  
 Logger.getLogger(RestoreMessages.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(RestoreMessages.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (ParserConfigurationException ex) {  
 Logger.getLogger(RestoreMessages.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 return null;  
 }  
}

#### Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta

Tiek pārbaudīts, vai eksistē kanāla versija. Ja tā ir, tad uz to tiek nosūtīts ziņojums saņēmējam.

package lv.abcsoftware.eves.samples.sendmessagereceiver;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.List;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.ws.BindingProvider;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannel;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelVersion;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPagedResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SendMessageResponse;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.IDITMessageServiceContract;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessage;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DITMessageService;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.\*;  
import lv.gov.eps.ivis.xmlschemas.\_100001.ivis.v1\_0.AuthorityBasicInfoStructure;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.xml.sax.SAXException;  
  
public class SendMessageReceiver {  
 public static void main(String[] args) {  
  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITConfigurationService service = new DITConfigurationService();  
 IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();  
 Map<String, Object> ctx = ((BindingProvider) port).getRequestContext();  
 ctx.put("ws-security.username", "DitAdmin");  
 ctx.put("ws-security.password", "Password");  
  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 String channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel-v1-0-TYPE-DIK";  
  
 SearchChannelsPaged searchChannelsPaged = new SearchChannelsPaged();  
 ChannelSearchInfoStructure request = new ChannelSearchInfoStructure();  
 searchChannelsPaged.setChannelSearch(request);  
  
 ChannelInfoStructure channel = new ChannelInfoStructure();  
 request.setChannel(channel);  
  
 channel.setChannelURN(channelURN);  
 request.setOnlyHeader(true);  
  
 SendMessageResponse sendMessageResponse = port.searchChannelsPaged(searchChannelsPaged);  
  
 ChannelInfoStructure newChannel = new ChannelInfoStructure();  
 newChannel.setShortName(channelName);  
 newChannel.setChannelURN(channelURN);  
 newChannel.setChannelType(ChannelType.DIK);  
 newChannel.setOwnerAuthority(new AuthorityBasicInfoStructure());  
 newChannel.getOwnerAuthority().setAuthorityID("100000");  
 newChannel.setChannelName(channelName);  
  
 if (sendMessageResponse.getChannelList().getCount() == 0) {  
 InsertChannel insertChannel = new InsertChannel();  
 insertChannel.setChannel(channel);  
 int channelId = port.insertChannel(insertChannel).getInsertChannelResponse();  
 System.out.println("Channel created");  
 } else {  
 System.out.println("Channel found");  
 }  
  
 ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();  
  
 MessageTypeInfoStructure messageType = new MessageTypeInfoStructure();  
 messageType.setMessageClass(new MessageClassInfoStructure());  
 messageType.getMessageClass().setCodeListCodeValue("1");  
 messageType.setXMLSchemaURN("URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName");  
  
 MessageTypeInfoStructure messageType1 = new MessageTypeInfoStructure();  
 messageType1.setMessageClass(new MessageClassInfoStructure());  
 messageType1.getMessageClass().setCodeListCodeValue("3");  
 messageType1.setXMLSchemaURN("URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError");  
  
 channelVersion.getMessageType().add(messageType);  
 channelVersion.getMessageType().add(messageType1);  
  
  
 channelVersion.getChannelVersionStatus().add("active");  
  
 channelVersion.setChannel(channel);  
 channelVersion.setMinorVersion((short) 0);  
 channelVersion.setMajorVersion((short) 1);  
  
 ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();  
 channelVersionSearch.setChannelVersion(channelVersion);  
  
 SearchChannelVersionsPaged searchChannelVersionsPaged = new SearchChannelVersionsPaged();  
 searchChannelVersionsPaged.setChannelVersionSearch(channelVersionSearch);  
 SearchChannelVersionsPagedResponse channelVersions = port.searchChannelVersionsPaged(searchChannelVersionsPaged);  
  
 if (channelVersions.getChannelVersionsList().getCount() == 0) {  
 InsertChannelVersion insertChannelVersion = new InsertChannelVersion();  
 insertChannelVersion.setChannelVersionData(channelVersion);  
 int channelId = port.insertChannelVersion(insertChannelVersion).getInsertChannelVersionResponse();  
 System.out.println("Channel version created");  
 } else {  
 System.out.println("Channel version found");  
 }  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 /\*ditCtx.put("ws-security.username", "DitAdmin");  
 ditCtx.put("ws-security.password", "password");\*/  
  
  
 MessageSaveStructure2 message = new MessageSaveStructure2();  
 message.setChannelVersionURN(channelVersionURN);  
 BussinesMessageType bussinesMessage = new BussinesMessageType();  
 message.setBussinesMessage(bussinesMessage);  
 ReceiverFilterArrayStructure receiverFilterArray = new ReceiverFilterArrayStructure();  
 List<String> authority = receiverFilterArray.getAuthority();  
 authority.add("100001");  
 message.setReceiverFilter(new ReceiverFilterArrayStructure());  
  
 Object obj = getObject();  
 bussinesMessage.setAny(obj);  
  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.setMessageSave2(message);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessageResponse sendMessageResponse1 = ditClient.sendMessage2(sendMessage);  
 System.out.println("Message Sent : " + sendMessageResponse1.isSendMessageResult());  
 }  
  
 static Object getObject() {  
 try {  
 File fXmlFile = new File("data.xml");  
  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(fXmlFile);  
 doc.getDocumentElement().normalize();  
  
 Element e = doc.getDocumentElement();  
 return e;  
  
 } catch (SAXException ex) {  
 Logger.getLogger(SendMessageReceiver.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(SendMessageReceiver.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (ParserConfigurationException ex) {  
 Logger.getLogger(SendMessageReceiver.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 return null;  
 }  
}

#### Atbildes ziņojuma nosūtīšana

package lv.abcsoftware.eves.samples.replay;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.ws.BindingProvider;  
  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.\*;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.\*;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.xml.sax.SAXException;  
  
public class Replay {  
 public static void main(String[] args) {  
 String channelName = "SampleChannel";  
 String channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";  
 sendMessage2(channelVersionURN);  
 String messageId = receiveMessage2(channelName, channelVersionURN);  
 sendMessage2EventMessage(messageId, channelVersionURN);  
 }  
  
 private static void sendMessage2EventMessage(String messageId, String channelVersionURN) {  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("../ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
  
 MessageSaveStructure2 message = new MessageSaveStructure2();  
 message.setChannelVersionURN(channelVersionURN);  
 EventMessageType eventMessageType = new EventMessageType();  
 eventMessageType.setName("Error!");  
 eventMessageType.setRelatedMessageID(messageId);  
 message.setEventMessage(eventMessageType);  
  
 Object obj = getObject("kluda.xml");  
 eventMessageType.setAny(obj);  
  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.setMessageSave2(message);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessageResponse sendMessageResponse1 = ditClient.sendMessage2(sendMessage);  
 System.out.println("Message Sent : " + sendMessageResponse1.isSendMessageResult());  
 }  
  
 private static void sendMessage2(String channelVersionURN){  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("../ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
  
 MessageSaveStructure2 message = new MessageSaveStructure2();  
 message.setChannelVersionURN(channelVersionURN);  
 BussinesMessageType bussinesMessage = new BussinesMessageType();  
 message.setBussinesMessage(bussinesMessage);  
  
 Object obj = getObject("data.xml");  
 bussinesMessage.setAny(obj);  
  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.setMessageSave2(message);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessageResponse sendMessageResponse1 = ditClient.sendMessage2(sendMessage);  
 System.out.println("Message Sent : " + sendMessageResponse1.isSendMessageResult());  
 }  
  
 private static String receiveMessage2(String channelName, String channelVersionURN) {  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-username.xml"));  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 ditCtx.put("ws-security.username", "username3");  
 ditCtx.put("ws-security.password", "password");  
  
 MessageRequestStructure messageRequest = new MessageRequestStructure();  
 messageRequest.setChannelVersionURN(channelVersionURN);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage request = new lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage();  
 request.setMessageRequest(messageRequest);  
  
 ReceiveMessageResponse response = ditClient.receiveMessage2(request);  
 System.out.println("Message received: " + response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage());  
 return response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage().get(0).getMessageID();  
 }  
  
 static Object getObject(String pathName) {  
 try {  
 File fXmlFile = new File(pathName);  
  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(fXmlFile);  
 doc.getDocumentElement().normalize();  
  
 Element e = doc.getDocumentElement();  
 return e;  
  
 } catch (SAXException ex) {  
 Logger.getLogger(Replay.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(Replay.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (ParserConfigurationException ex) {  
 Logger.getLogger(Replay.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 return null;  
 }  
}

#### ACK statusa maiņa

package lv.abcsoftware.eves.samples.ackstatuschange;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.ws.BindingProvider;  
  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.\*;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.\*;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.xml.sax.SAXException;  
  
public class ACKStatusChange {  
  
 public static void main(String[] args) {  
 String channelName = "SampleChannel";  
 String channelVersionURN = "URN:IVIS:100001:CHA-" + channelName + "-v1-0-TYPE-DIK";  
 sendMessage2(channelVersionURN);  
 receiveMessage2(channelName, channelVersionURN);  
 }  
  
 private static void receiveMessage2(String channelName, String channelVersionURN) {  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-username.xml"));  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 ditCtx.put("ws-security.username", "username3");  
 ditCtx.put("ws-security.password", "password");  
  
 MessageRequestStructure messageRequest = new MessageRequestStructure();  
 messageRequest.setChannelVersionURN(channelVersionURN);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage request = new lv.gov.eps.ivis.iss.ditservice.v1\_0.ReceiveMessage();  
 request.setMessageRequest(messageRequest);  
  
 ReceiveMessageResponse response = ditClient.receiveMessage2(request);  
 System.out.println("Message received: " + response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage());  
 String messageId = response.getMessageResponse2().getBussinesMessageOrEventMessageTypeOrMessage().get(0).getMessageID();  
 ConfirmationAck confirmationAck = new ConfirmationAck();  
 confirmationAck.setMessageSampleId(messageId);  
 ditClient.confirmationAck(confirmationAck);  
 }  
  
 private static void sendMessage2(String channelVersionURN){  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("../ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSct();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
  
 MessageSaveStructure2 message = new MessageSaveStructure2();  
 message.setChannelVersionURN(channelVersionURN);  
 BussinesMessageType bussinesMessage = new BussinesMessageType();  
 message.setBussinesMessage(bussinesMessage);  
  
 Object obj = getObject("data.xml");  
 bussinesMessage.setAny(obj);  
  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.setMessageSave2(message);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessageResponse sendMessageResponse1 = ditClient.sendMessage2(sendMessage);  
 System.out.println("Message Sent : " + sendMessageResponse1.isSendMessageResult());  
 }  
  
 static Object getObject(String pathName) {  
 try {  
 File fXmlFile = new File(pathName);  
  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(fXmlFile);  
 doc.getDocumentElement().normalize();  
  
 Element e = doc.getDocumentElement();  
 return e;  
  
 } catch (SAXException ex) {  
 Logger.getLogger(ACKStatusChange.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(ACKStatusChange.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (ParserConfigurationException ex) {  
 Logger.getLogger(ACKStatusChange.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 return null;  
 }  
}

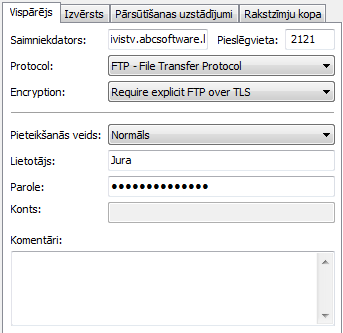
#### Liela apjoma ziņojuma nosūtīšana

Tiek pārbaudīts, vai eksistē kanāla versija. Ja tā ir, tad uz to tiek nosūtīts ziņojums.

package lv.abcsoftware.eves.samples.sendlargemessage;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.ws.BindingProvider;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.DITConfigurationService;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.IDITConfigurationServiceContract;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannel;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.InsertChannelVersion;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelVersionsPagedResponse;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SearchChannelsPaged;  
import lv.gov.eps.ivis.iss.ditconfigurationservice.v1\_2.SendMessageResponse;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.IDITMessageServiceContract;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessage;  
import lv.gov.eps.ivis.iss.ditservice.v1\_0.DITMessageService;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.BussinesMessageType;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelType;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.ChannelVersionSearchInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageClassInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageSaveStructure2;  
import lv.gov.eps.ivis.xmlschemas.\_100000.dit.v1\_0.MessageTypeInfoStructure;  
import lv.gov.eps.ivis.xmlschemas.\_100001.ivis.v1\_0.AuthorityBasicInfoStructure;  
import org.apache.cxf.BusFactory;  
import org.apache.cxf.bus.spring.SpringBusFactory;  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.xml.sax.SAXException;  
  
public class SendLargeMessage {  
 public static void main(String[] args) {  
  
 SpringBusFactory bf = new SpringBusFactory();  
 BusFactory.setDefaultBus(bf.createBus("ProxyToService/src/main/resources/client-cert.xml"));  
  
 DITConfigurationService service = new DITConfigurationService();  
 IDITConfigurationServiceContract port = service.getWs2007FederationNoSct();  
 Map<String, Object> ctx = ((BindingProvider) port).getRequestContext();  
 ctx.put("ws-security.username", "DitAdmin");  
 ctx.put("ws-security.password", "Password");  
  
  
 String channelName = "SampleChannel";  
 String channelURN = "URN:IVIS:100001:CHA-" + channelName + "-TYPE-DIK";  
 String channelVersionURN = "URN:IVIS:100001:CHA-SampleChannel-v1-0-TYPE-DIK";  
  
 SearchChannelsPaged searchChannelsPaged = new SearchChannelsPaged();  
 ChannelSearchInfoStructure request = new ChannelSearchInfoStructure();  
 searchChannelsPaged.setChannelSearch(request);  
  
 ChannelInfoStructure channel = new ChannelInfoStructure();  
 request.setChannel(channel);  
  
 channel.setChannelURN(channelURN);  
 request.setOnlyHeader(true);  
  
 SendMessageResponse sendMessageResponse = port.searchChannelsPaged(searchChannelsPaged);  
  
 ChannelInfoStructure newChannel = new ChannelInfoStructure();  
 newChannel.setShortName(channelName);  
 newChannel.setChannelURN(channelURN);  
 newChannel.setChannelType(ChannelType.DIK);  
 newChannel.setOwnerAuthority(new AuthorityBasicInfoStructure());  
 newChannel.getOwnerAuthority().setAuthorityID("100000");  
 newChannel.setChannelName(channelName);  
  
 if (sendMessageResponse.getChannelList().getCount() == 0) {  
 InsertChannel insertChannel = new InsertChannel();  
 insertChannel.setChannel(channel);  
 int channelId = port.insertChannel(insertChannel).getInsertChannelResponse();  
 System.out.println("Channel created");  
 } else {  
 System.out.println("Channel found");  
 }  
  
 ChannelVersionInfoStructure channelVersion = new ChannelVersionInfoStructure();  
  
 MessageTypeInfoStructure messageType = new MessageTypeInfoStructure();  
 messageType.setMessageClass(new MessageClassInfoStructure());  
 messageType.getMessageClass().setCodeListCodeValue("1");  
 messageType.setXMLSchemaURN("URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName");  
  
 MessageTypeInfoStructure messageType1 = new MessageTypeInfoStructure();  
 messageType1.setMessageClass(new MessageClassInfoStructure());  
 messageType1.getMessageClass().setCodeListCodeValue("3");  
 messageType1.setXMLSchemaURN("URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError");  
  
 channelVersion.getMessageType().add(messageType);  
 channelVersion.getMessageType().add(messageType1);  
  
  
 channelVersion.getChannelVersionStatus().add("active");  
  
 channelVersion.setChannel(channel);  
 channelVersion.setMinorVersion((short) 0);  
 channelVersion.setMajorVersion((short) 1);  
  
 ChannelVersionSearchInfoStructure channelVersionSearch = new ChannelVersionSearchInfoStructure();  
 channelVersionSearch.setChannelVersion(channelVersion);  
  
 SearchChannelVersionsPaged searchChannelVersionsPaged = new SearchChannelVersionsPaged();  
 searchChannelVersionsPaged.setChannelVersionSearch(channelVersionSearch);  
 SearchChannelVersionsPagedResponse channelVersions = port.searchChannelVersionsPaged(searchChannelVersionsPaged);  
  
 if (channelVersions.getChannelVersionsList().getCount() == 0) {  
 InsertChannelVersion insertChannelVersion = new InsertChannelVersion();  
 insertChannelVersion.setChannelVersionData(channelVersion);  
 int channelId = port.insertChannelVersion(insertChannelVersion).getInsertChannelVersionResponse();  
 System.out.println("Channel version created");  
 } else {  
 System.out.println("Channel version found");  
 }  
  
 DITMessageService dITService = new DITMessageService();  
  
 IDITMessageServiceContract ditClient = dITService.getWs2007FederationNoSctMtom();  
 Map<String, Object> ditCtx = ((BindingProvider) ditClient).getRequestContext();  
 /\*ditCtx.put("ws-security.username", "DitAdmin");  
 ditCtx.put("ws-security.password", "password");\*/  
  
  
 MessageSaveStructure2 message = new MessageSaveStructure2();  
 message.setChannelVersionURN(channelVersionURN);  
 BussinesMessageType bussinesMessage = new BussinesMessageType();  
 message.setBussinesMessage(bussinesMessage);  
  
 Object obj = getObject();  
 bussinesMessage.setAny(obj);  
  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.setMessageSave2(message);  
  
 lv.gov.eps.ivis.iss.ditservice.v1\_0.SendMessageResponse sendMessageResponse1 = ditClient.sendMessage2(sendMessage);  
 System.out.println("Message Sent : " + sendMessageResponse1.isSendMessageResult());  
  
 }  
  
 static Object getObject() {  
 try {  
 File fXmlFile = new File("data.xml");  
  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(fXmlFile);  
 doc.getDocumentElement().normalize();  
  
 Element e = doc.getDocumentElement();  
 return e;  
  
 } catch (SAXException ex) {  
 Logger.getLogger(SendLargeMessage.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(SendLargeMessage.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (ParserConfigurationException ex) {  
 Logger.getLogger(SendLargeMessage.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 return null;  
 }  
}

### FTP piemēri Java

DIT FTP pieslēgumu var pārbaudīt, izmantojot parasto FTP klientu, piemēram, *FileZilla. FileZilla ftp* kanāla konfigurācija ir pieejama 31.attēlā.



.attēls. FileZilla FTP kanāla konfigurācija

#### Datnes pievienošana (FTPSendMessage)

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam:

package lv.abcsoftware.eves.samples.ftpsendmessage;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.net.SocketException;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import org.apache.commons.net.ftp.FTPSClient;  
  
public class FTPSendMessage {  
  
 public static void main(String[] args) {  
 FTPSClient ftp = new FTPSClient();  
 String host = "dittest.vraa.gov.lv";  
 int port = 2121;  
 String channelName = "SampleChannel";  
 String write = "Write";  
 String folderName = channelName + "/DIK-v1-0/" + write;  
 String username = "DitAdmin";  
 String password = "Password";  
 try {  
 ftp.connect(host, port);  
 ftp.login(username, password);  
 InputStream fis = new FileInputStream("../data.xml");  
  
 ftp.storeFile("/" + folderName + "/data1.xml", fis);  
 fis.close();  
  
 ftp.logout();  
 ftp.disconnect();  
 } catch (SocketException ex) {  
 Logger.getLogger(FTPSendMessage.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(FTPSendMessage.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 }  
}

#### Datnes nolasīšana un dzešana (FTPReceiveMessage)

Tiek izveidots pieslēgums FTP, un tiek nolasīta datne no kanāla mapes, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam:

package lv.abcsoftware.eves.samples.ftpreceivemessage;  
  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.net.SocketException;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import org.apache.commons.net.ftp.FTPFile;  
import org.apache.commons.net.ftp.FTPSClient;  
  
public class FTPReceiveMessage {  
  
 public static void main(String[] args) {  
 FTPSClient ftp = new FTPSClient();  
 String host = "dittest.vraa.gov.lv";  
 int port = 2121;  
 String channelName = "SampleChannel";  
 String write = "ReadDIK";  
 String folderName = write + "-" + channelName + "/v1-0";  
 String username = "usernamevraa";  
 String password = "Password";  
  
 try {  
 ftp.connect(host, port);  
 ftp.login(username, password);  
  
 ftp.changeWorkingDirectory(folderName);  
 FTPFile[] files = ftp.listFiles();  
  
 for (FTPFile file : files) {  
 FileOutputStream fos = new FileOutputStream("target/"+file.getName());  
   
 ftp.retrieveFile(file.getName(), fos);  
 System.out.println(file.getName() + " downloaded");  
 ftp.deleteFile(file.getName());  
 System.out.println(file.getName() + " deleted");  
 fos.close();  
 }  
 ftp.logout();  
 ftp.disconnect();  
 } catch (SocketException ex) {  
 Logger.getLogger(FTPReceiveMessage.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(FTPReceiveMessage.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 }  
}

#### Ziņojuma nosūtīšana vienam vai vairākiem saņēmējiem DIT no pilnā pierakstītāju saraksta

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam konkrētajam saņēmējam:

package lv.abcsoftware.eves.samples.ftpsendmessagereceiver;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.net.SocketException;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import org.apache.commons.net.ftp.FTPSClient;  
  
public class FTPSendMessageReceiver {  
 public static void main(String[] args) {  
 FTPSClient ftp = new FTPSClient();  
 String host = "dittest.vraa.gov.lv";  
 int port = 2121;  
 String username = "DitAdmin";  
 String password = "Password";  
 String channelName = "SampleChannel7";  
 String authority = "100001";  
 String write = "write";  
 String folderName = channelName + "/DIK-v1-0/" + write + "/" + authority;  
  
 try {  
 ftp.connect(host, port);  
 ftp.login(username, password);  
 InputStream fis = new FileInputStream("../data.xml");  
  
 ftp.storeFile("/" + folderName + "/data1.xml", fis);  
 fis.close();  
  
 ftp.logout();  
 ftp.disconnect();  
 } catch (SocketException ex) {  
 Logger.getLogger(FTPSendMessageReceiver.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(FTPSendMessageReceiver.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 }  
}

#### Atbildes ziņojuma nosūtīšana

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam zem WriteEventError:

package lv.abcsoftware.eves.samples.ftpreplay;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.net.SocketException;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import org.apache.commons.net.ftp.FTPSClient;  
  
public class FTPReplay {  
 public static void main(String[] args) {  
 FTPSClient ftp = new FTPSClient();  
 String host = "dittest.vraa.gov.lv";  
 int port = 2121;  
 String username = "usernamevraa";  
 String password = "Password";  
 String channelName = "SampleChannel4";  
 String write = "WriteEvent/WriteEventError";  
 String folderName = channelName + "/DIK-v1-0/" + write;  
  
 try {  
 ftp.connect(host, port);  
 ftp.login(username, password);  
 InputStream fis = new FileInputStream("../kluda.xml");  
  
 ftp.storeFile("/" + folderName + "/kluda.xml", fis);  
 fis.close();  
  
 ftp.logout();  
 ftp.disconnect();  
 } catch (SocketException ex) {  
 Logger.getLogger(FTPReplay.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(FTPReplay.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 }  
}

# PHP

## Projekta izveidošana PHP

Piemēri ir izveidoti, un pārbaudīti izmantojot šādu programmatūru:

* Apache HTTP serveris: Apache/2.2.22 (Win32);
* PHP versija: 5.4.11;
* PHP izmantotie paplašinājumi (php.ini):
  + php\_openssl.dll;
  + php\_soap.dll.

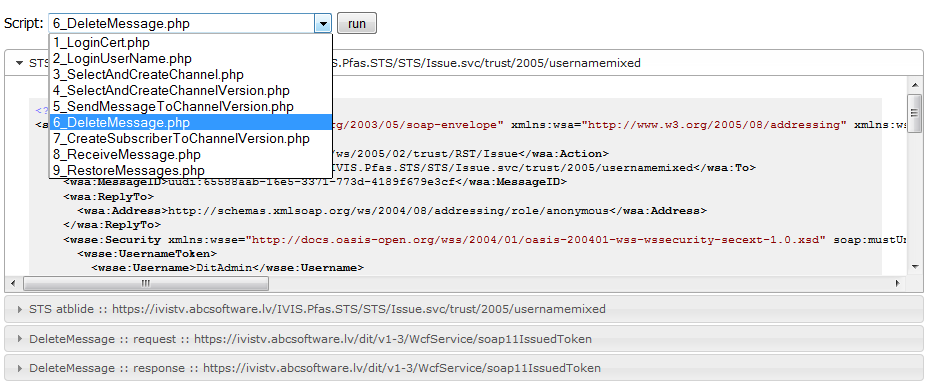
FTP PHP piemēriem jāizmanto PHP versija ar iekompilētiem FTP un OpenSSL moduļiem.

php.ini ir jānokonfigurē SOAP paplašinājums ar šādām parametru vērtībām:

* soap.wsdl\_cache\_enabled=1;
* soap.wsdl\_cache\_dir="c:\wsdlc\";
* soap.wsdl\_cache\_ttl=86400.

Parametru skaidrojums atrodams PHP dokumentācijā vai php.ini datnes komentāros.

Piemēru ērtākai apskatei un izpildei izveidota palaišanas saskarne:

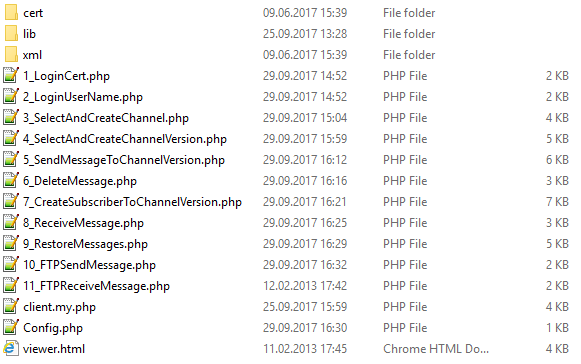


.attēls. Piemēru palaišanas saskarne

Piemēru palaišanas saskarne tiek atvērta, ielādējot datni „viewer.html” jebkurā pārlūkprogrammā (testēts IE9, FF18 un Chrome24). Saskarne ir implementēta, izmantojot HTML un JavaScript un tai ir šādas atkarības:

* jQuery 1.8.3;
* jQueryUI 1.10.0;
* knockout 2.2.0;
* highlightjs 7.3.

Visas minētās JavaScript bibliotēkas tiek ielādētas no CDN (*Content delivery network*), tāpēc piemēru palaišanas saskarnes izmantošanai nepieciešamais interneta pieslēgums.



33.attēls. Projekta struktūra

Projekta struktūra ir redzama 33.attēlā, tas saturs:

1. „cert” – publiska un privāta atslēgas .pem formātā;
2. „lib” – PHP bibliotēkas darbam ar WS-Security un STS;
3. „xml” – testa ziņojumu glabāšanas mape;
4. „\*.php” – piemēri;
5. „client.my.php” – PHP SoapClient klases paplašinājums darbam ar WS-Security;
6. “Config.php” – Konfigurācijas datnē ar servisu adresēm;
7. „vievier.html” – piemēru palaišanas saskarne.

### SOAP piemēri PHP

#### Konfigurācijas datne Config.php

Konfigurācijās datnē jānorāda STS un DIT adreses, un DIT servisa realm.

<?php

$sts\_address = 'https://epakvisstv.vraa.gov.lv/STS/Viss.Pfas.STS/sts/Issue.svc';

$apply\_to =  'https://ivis.eps.gov.lv/DIT.WebService';

$DIT\_config\_service\_addr = 'https://dittest.vraa.gov.lv/dit/v1-3/ConfigurationService';

$DIT\_WCF\_service\_addr = 'https://dittest.vraa.gov.lv/dit/v1-2/MessageService';

?>

#### Savienošana ar sertifikātu (LoginCert)

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    //$service->customtoken = $tokens[0];

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    //var\_dump($fault);

    echo $fault;

}

} catch (Exception $ex)  {

    echo $ex->getMessage();

    //var\_dump($ex);

}

?>

#### Savienošana ar lietotāja vārdu un paroli (LoginUserName)

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/usernamemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$ms = Milestone::getInstance();

try {

try {

    // set UsernameToken variables and request for tokens

    $username = 'DitAdmin';

    $password = 'Password';

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $tokens = $sts->issueToken($username, $password, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    //$service->customtoken = $tokens[0];

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    //var\_dump($fault);

    echo $fault;

}

} catch (Exception $ex)  {

    echo $ex->getMessage();

    //var\_dump($ex);

}

?>

#### Kanāla pārbaude un izveidošana (SelectAndCreateChannel)

Piemērā tiek pārbaudīts vai eksistē kanāls. Ja kanāla nav, tad tiek izveidots jauns kanāls:

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$service = new mySOAP($DIT\_config\_service\_addr.'?wsdl', array(

                                         'exceptions' => TRUE,

                                         'soap\_version' => SOAP\_1\_2,

                                         'location' => $DIT\_config\_service\_addr.'/ws2007FederationNoSct',

                                         'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $req =  array(

        "ChannelSearch"  => array(

            "OnlyHeader" => true,

            "Channel" =>  array(

                "ChannelURN" => $channel\_urn

            )

        ),

        'ChannelVersionStatus' => array('creating', 'testing', 'active', 'tehnicalbreak', 'activedepricated', 'deleted')

    );

    $method\_name = 'SearchChannelsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    //if channel not found creating new one

    if($result->ChannelList->Count == 0) {

        $req2 =  array(

            'Channel' => array(

                'ShortName' => $channel\_name,

                'ChannelURN' => $channel\_urn,

                'ChannelType' => 'dik',

                'OwnerAuthority' => array(

                    'AuthorityID' => '100000',

                    'ShortName' => 'test',

                ),

                'ChannelName' => $channel\_name

            )

        );

        $method2\_name = 'InsertChannel';

        $result2 = $service->\_\_soapCall($method2\_name, array($req2) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method2\_name);

    } else {

        // channel found...

    }

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    echo $fault;

}

} catch (Exception $ex)  {

    echo $ex->getMessage();

}

?>



.attēls. Kanāla izveidošana

#### Kanāla versijas pārbaude un izveidošana (SelectAndCreateChannelVersion)

Piemērā tiek pārbaudīts vai eksistē kanāls. Ja kanāla nav, tad tiek izveidots jauns kanāls un tiek pārbaudīts vai eksistē kanāla versija; ja tādas nav, tad sāk tās veidošanu.

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$service = new mySOAP($DIT\_config\_service\_addr.'?wsdl', array(

                                         'exceptions' => TRUE,

                                         'soap\_version' => SOAP\_1\_2,

                                         'location' => $DIT\_config\_service\_addr.'/ws2007FederationNoSct',

                                         'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $req =  array(

        "ChannelSearch"        => array(

            /\*"OnlyHeader" => true,\*/

            "Channel" =>  array(

                "ChannelURN" => $channel\_urn

            )

        ),

        'ChannelVersionStatus' => array('creating', 'testing', 'active', 'tehnicalbreak', 'activedepricated', 'deleted')

    );

    $method\_name = 'SearchChannelsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    // setting channel data

    $channel = array(

        'ShortName' => $channel\_name,

        'ChannelURN' => $channel\_urn,

        'ChannelType' => 'dik',

        'OwnerAuthority' => array(

            'AuthorityID' => '100000',

            'ShortName' => 'ABC software',

        ),

        'ChannelName' => $channel\_name

    );

    //if channel not found creating new one

    if($result->ChannelList->Count == 0) {

        // adding channel data to request

        $req2 =  array(

            'Channel' => $channel

        );

        $method2\_name = 'InsertChannel';

        $result2 = $service->\_\_soapCall($method2\_name, array($req2) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method2\_name);

        $result2->InsertChannelResponse;

    } else {

        //channel found

    }

    $minor\_v = 0;

    $major\_v = 2;

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $channel\_version = array(

        'MessageType' => array(

            0 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '1'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName'

            ),

            1 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '3'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError',

            )

        ),

        'ChannelVersionStatus' => 'active',

        'Channel' => $channel,

        'MinorVersion' => $minor\_v,

        'MajorVersion' => $major\_v,

        'ChannelVersionURN' => $channel\_version\_URN

    );

    $req3 =  array(

        'ChannelVersionSearch' => array(

        'ChannelVersion' => $channel\_version

        )

    );

    $method\_name = 'SearchChannelVersionsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req3) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    //if channel version not found creating new one

    if($result->ChannelVersionsList->Count == 0) {

        // adding channel version data to request

        $req =  array(

            'ChannelVersionData' => $channel\_version

        );

        $method\_name = 'InsertChannelVersion';

        $result = $service->\_\_soapCall($method\_name, array($req) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        //channel version found

    }

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    //var\_dump($fault);

    echo $fault;

}

} catch (Exception $ex)  {

    echo $ex->getMessage();

    //var\_dump($ex);

}

?>



.attēls. Kanāla versijas izveidošana

#### Ziņojuma nosūtīšana (SendMessageToChannelVersion)

Tiek pārbaudīts ,vai eksistē kanāla versija. Ja tā ir, tad uz to tiek nosūtīts ziņojums.

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$service = new mySOAP($DIT\_config\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_config\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$service2 = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $service2->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $req =  array(

        'ChannelSearch'        => array(

            'OnlyHeader' => true,

            'Channel' =>  array(

                'ChannelURN' => $channel\_urn

            )

        ),

        'ChannelVersionStatus' => array('creating', 'testing', 'active', 'tehnicalbreak', 'activedepricated', 'deleted')

    );

    $method\_name = 'SearchChannelsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    // setting channel data

    $channel = array(

        'ShortName' => $channel\_name,

        'ChannelURN' => $channel\_urn,

        'ChannelType' => 'dik',

        'OwnerAuthority' => array(

            'AuthorityID' => '100000',

            'ShortName' => 'ABC software',

        ),

        'ChannelName' => $channel\_name

    );

    //if channel not found creating new one

    if($result->ChannelList->Count == 0) {

        // adding channel data to request

        $req2 =  array(

            'Channel' => $channel

        );

        $method2\_name = 'InsertChannel';

        $result2 = $service->\_\_soapCall($method2\_name, array($req2) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method2\_name);

        $result2->InsertChannelResponse;

    } else {

        //channel found

    }

    $channel\_version = array(

        'MessageType' => array(

            0 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '1'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName'

            ),

            1 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '3'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError',

            )

        ),

        'ChannelVersionStatus' => 'active',

        'Channel' => $channel,

        'MinorVersion' => $minor\_v,

        'MajorVersion' => $major\_v,

        'ChannelVersionURN' => $channel\_version\_URN

    );

    $req3 =  array(

        'ChannelVersionSearch' => array(

        'ChannelVersion' => $channel\_version

        )

    );

    $method\_name = 'SearchChannelVersionsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req3) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    //if channel version not found creating new one

    if($result->ChannelVersionsList->Count == 0) {

        // adding channel version data to request

        $req =  array(

            'ChannelVersionData' => $channel\_version

        );

        $method\_name = 'InsertChannelVersion';

        $result = $service->\_\_soapCall($method\_name, array($req) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        //channel version found

    }

    $dom = new DOMDocument('1.0', 'UTF-8');

    $dom->loadXML(file\_get\_contents('xml/data.xml'));

    $req =  array(

        'MessageSave2' => array(

            'ReceiverFilter' => array(

                'Authority' => array('100001')

            ),

            'ChannelVersionURN' => $channel\_version\_URN,

            'BussinesMessage' => array(

                'any' => array(

                    $dom->saveXML($dom->documentElement)

                )

            )

        )

    );

    $method\_name = 'SendMessage2';

    $result2 = $service2->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service2, $DIT\_WCF\_service\_addr, $method\_name);

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    echo $fault;

}

} catch (Exception $ex)  {

    var\_dump($ex);

}

?>



.attēls. Izveidotais ziņojums

#### Ziņojumu dzēšana (MessageDelete)

No kanāla tiek dzēsti ziņojumi:

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/usernamemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

                            $service2 = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    // set UsernameToken variables and request for tokens

    $username = 'DitAdmin';

    $password = 'Password';

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $tokens = $sts->issueToken($username, $password, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service2->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel2';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $req =  array(

        'MessageRequest' => array(

            'ChannelURN' => $channel\_urn

        )

    );

    $method\_name = 'DeleteMessage';

    $result = $service2->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service2, $DIT\_WCF\_service\_addr, $method\_name);

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    //var\_dump($fault);

    echo $fault;

}

} catch (Exception $ex)  {

    //echo $ex->getMessage();

    var\_dump($ex);

}

?>



.attēls. Dzēstie ziņojumi

#### Tā, kas parakstījies pievienošana kanāla versijai (CreateSubscriberToChannelVersion)

Tiek pārbaudīts, vai uz kanālu kāds ir parakstījies, ja nav, tad pievieno.

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$service = new mySOAP($DIT\_config\_service\_addr.'?wsdl', array(

                                         'exceptions' => TRUE,

                                         'soap\_version' => SOAP\_1\_2,

                                         'location' => $DIT\_config\_service\_addr.'/ws2007FederationNoSct',

                                         'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    //$tokens = $sts->issueToken($username, $password, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel2';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $req =  array(

        'ChannelSearch'        => array(

            'OnlyHeader' => true,

            'Channel' =>  array(

                'ChannelURN' => $channel\_urn

            )

        ),

        'ChannelVersionStatus' => array('creating', 'testing', 'active', 'tehnicalbreak', 'activedepricated', 'deleted')

    );

    $method\_name = 'SearchChannelsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    // setting channel data

    $channel = array(

        'ShortName' => $channel\_name,

        'ChannelURN' => $channel\_urn,

        'ChannelType' => 'dik',

        'OwnerAuthority' => array(

            'AuthorityID' => '100000',

            'ShortName' => 'ABC software',

        ),

        'ChannelName' => $channel\_name

    );

    //if channel not found creating new one

    if($result->ChannelList->Count == 0) {

        // adding channel data to request

        $req2 =  array(

            'Channel' => $channel

        );

        $method2\_name = 'InsertChannel';

        $result2 = $service->\_\_soapCall($method2\_name, array($req2) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method2\_name);

        $result2->InsertChannelResponse;

    } else {

        //channel found

    }

    $channel\_version = array(

        'MessageType' => array(

            0 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '1'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName'

            ),

            1 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '3'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError',

            )

        ),

        'ChannelVersionStatus' => 'active',

        'Channel' => $channel,

        'MinorVersion' => $minor\_v,

        'MajorVersion' => $major\_v,

        'ChannelVersionURN' => $channel\_version\_URN

    );

    $req3 =  array(

        'ChannelVersionSearch' => array(

        'ChannelVersion' => $channel\_version

        )

    );

    $method\_name = 'SearchChannelVersionsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req3) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    //if channel version not found creating new one

    if($result->ChannelVersionsList->Count == 0) {

        // adding channel version data to request

        $req =  array(

            'ChannelVersionData' => $channel\_version

        );

        $method\_name = 'InsertChannelVersion';

        $result = $service->\_\_soapCall($method\_name, array($req) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        //channel version found

    }

    $channel\_subscriber = array(

        'ChannelSubscriber' => array(

            'ChannelVersionURN' => $channel\_version\_URN,

            'SubscriberAuthority' => array(

                'AuthorityID' => '100001',

                'ShortName' => 'ABC software'

            )

        )

    );

    $req = array(

        'ChannelSubscriberSearch' => $channel\_subscriber

    );

    $method\_name = 'SearchSubscribersToChannelPaged';

    $result = $service->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    //if subscrition to the channel version not found creating new one

    if($result->ChannelSubscriberList->Count == 0) {

        $method\_name = 'InsertSubscriberToChannel';

        $result = $service->\_\_soapCall($method\_name, array($channel\_subscriber));

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        //subscrition to the channel version found

    }

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    //var\_dump($fault);

    echo $fault;

}

} catch (Exception $ex)  {

    //echo $ex->getMessage();

    var\_dump($ex);

}

?>



.attēls. Iestāde pieslēgta DIT

#### Ziņojuma nolasīšana (ReceiveMessage)

Pēc DIT kanāla izveidošanas iespējams saņemt ziņojumus par lietotāju, kurš pierakstījies pieslēgumam.



.attēls. DIK kanāla izmantošanas shēma

Pēc nolasīšanas ziņojumu nepieciešams dzēst.

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/usernamemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$service2 = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    // set UsernameToken variables and request for tokens

    $username2 = 'usernamevraa';

    $password2 = 'Password';

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $tokens = $sts->issueToken($username2, $password2, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service2->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $req = array(

        'MessageRequest' => array(

            'ChannelURN' => $channel\_urn

        )

    );

    $method\_name = 'ReceiveMessage2';

    $result = $service2->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service2, $DIT\_WCF\_service\_addr, $method\_name);

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    echo $fault;

}

} catch (Exception $ex)  {

    var\_dump($ex);

}

?>

#### Ziņojuma atjaunošana (Restore Messages)

Pēc DIT kanāla struktūras izveidošanas ar lietotāju, kurš parakstījies uz pieslēgumu, iespējams saņemt un atjaunot ziņojumus.

Gadījumā, ja ziņojuma saņemšana bija nekorekta, tad to var atjaunot, lietojot metodes noteiktajā secībā:

1. ReceiveMessageList;
2. RestoreMessageList (ReadMessage=true);
3. ReceveMessage;
4. DeleteMessage.



.attēls. DIK kanāla izmantošanas shēma, lai atjaunotu ziņojumu

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/usernamemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$sts2 = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$service = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $username = 'usernamevraa';

    $password = 'Password';

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts2->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts2, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $dom = new DOMDocument('1.0', 'UTF-8');

    $dom->loadXML(file\_get\_contents('xml/data.xml'));

    $req =  array(

        'MessageSave2' => array(

            'ReceiverFilter' => array(

                'Authority' => array('100001')

            ),

            'ChannelVersionURN' => $channel\_version\_URN,

            'BussinesMessage' => array(

                'any' => array(

                    $dom->saveXML($dom->documentElement)

                )

            )

        )

    );

    //sending message

    $method\_name = 'SendMessage2';

    $result = $service->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

    if($result->SendMessageResult == true) {

        //authentication with another user credentials

        $tokens = $sts->issueToken($username, $password, $apply\_to, $claims);

        $service->customtoken = $tokens[0];

        $req\_message = array(

            'MessageRequest' => array(

                'ChannelVersionURN' => $channel\_version\_URN

            )

        );

        //receiving messages

        $method\_name = 'ReceiveMessage2';

        $result = $service->\_\_soapCall($method\_name, array($req\_message));

        $ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

        $req = array(

            'MessageListRequest' => array(

                'ReadMessage' => true,

                'ChannelVersionURN' => $channel\_version\_URN

            )

        );

        //receiving read messages

        $method\_name = 'ReceiveMessageList';

        $result = $service->\_\_soapCall($method\_name, array($req));

        $ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

        //if there are read messages

        if($result->MessageBasicInfoList != null) {

            $req = array(

                'MessageBasicInfoList' => $result->MessageBasicInfoList

            );

            //restoring messages

            $method\_name = 'RestoreMessageList';

            $result = $service->\_\_soapCall($method\_name, array($req));

            $ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

        }

        //receiving messages again

        $method\_name = 'ReceiveMessage2';

        $result2 = $service->\_\_soapCall($method\_name, array($req\_message));

        $ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

        //if there are messages

        if($result2->MessageResponse != null && sizeof($result2->MessageResponse) > 0) {

            //iterating over message list

            $i = 0;

            foreach ($result2->MessageResponse as $message) {

                //deleting only 5 first messages if more exist

                $i++;

                if($i > 5) break;

                $req = array(

                    'MessageRequest' => array(

                        'MessageID' => $message->MessageID

                    )

                );

                //deleting messages

                $method\_name = 'DeleteMessage';

                $result2 = $service->\_\_soapCall($method\_name, array($req));

                $ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

            }

        }

    }

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    echo $fault;

}

} catch (Exception $ex)  {

    var\_dump($ex);

}

?>

#### Ziņojuma ACK statusa maiņa (ACKStatusChange)

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$sts2 = new STSClient(NULL, array(

                                'trace' => 1,

                                'location' => $sts\_address.'/trust/13/usernamemixed',

                                'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                                'exceptions' => TRUE,

                                'style'    => SOAP\_DOCUMENT,

                                'use'      => SOAP\_LITERAL,

                                'soap\_version' => SOAP\_1\_2

                                ));

$service = new mySOAP($DIT\_config\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_config\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$service2 = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$service3 = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                                'exceptions' => TRUE,

                                'soap\_version' => SOAP\_1\_2,

                                'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                                'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $service2->customtoken = $tokens[0];

    // set UsernameToken variables and request for tokens

    $username = 'usernamevraa';

    $password = 'Password';

    $tokens = $sts2->issueToken($username, $password, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts2, $sts\_address);

    $service3->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel579';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $req =  array(

        'ChannelSearch'        => array(

            'OnlyHeader' => true,

            'Channel' =>  array(

                'ChannelURN' => $channel\_urn

            )

        ),

        'ChannelVersionStatus' => array('creating', 'testing', 'active', 'tehnicalbreak', 'activedepricated', 'deleted')

    );

    $method\_name = 'SearchChannelsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    // setting channel data

    $channel = array(

        'ShortName' => $channel\_name,

        'ChannelURN' => $channel\_urn,

        'ChannelType' => 'dik',

        'OwnerAuthority' => array(

            'AuthorityID' => '100000',

            'ShortName' => 'ABC software',

        ),

        'ChannelName' => $channel\_name

    );

    //if channel not found creating new one

    if($result->ChannelList->Count == 0) {

        // adding channel data to request

        $req2 =  array(

            'Channel' => $channel

        );

        $method2\_name = 'InsertChannel';

        $result2 = $service->\_\_soapCall($method2\_name, array($req2) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method2\_name);

        $result2->InsertChannelResponse;

    } else {

        //channel found

    }

    $channel\_version = array(

        'MessageType' => array(

            0 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '1'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName'

            ),

            1 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '3'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError',

            )

        ),

        'ChannelVersionStatus' => 'active',

        'Channel' => $channel,

        'MinorVersion' => $minor\_v,

        'MajorVersion' => $major\_v,

        'ChannelVersionURN' => $channel\_version\_URN

    );

    $req3 =  array(

        'ChannelVersionSearch' => array(

            'ChannelVersion' => $channel\_version

        )

    );

    $method\_name = 'SearchChannelVersionsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req3) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    //if channel version not found creating new one

    if($result->ChannelVersionsList->Count == 0) {

        // adding channel version data to request

        $req =  array(

            'ChannelVersionData' => $channel\_version

        );

        $method\_name = 'InsertChannelVersion';

        $result = $service->\_\_soapCall($method\_name, array($req) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        //channel version found

    }

    $req = array(

        'ChannelSubscriberSearch' => array(

            'ChannelSubscriber' => array(

                'ChannelVersionURN' => $channel\_version\_URN,

                'SubscriberAuthority' => array(

                    'AuthorityID' => '100001',

                    'ShortName' => 'VRAA',

                    'Name' => 'VRAA'

                )

            )

        )

    );

    $method\_name = 'SearchSubscribersToChannelPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    if($result->ChannelSubscriberList->Count == 0) {

        // Channel subscriber created

        $req = array(

            'ChannelSubscriber' => array(

                'ChannelVersionURN' => $channel\_version\_URN,

                'SubscriberAuthority' => array(

                    'AuthorityID' => '100001',

                    'ShortName' => 'VRAA',

                    'Name' => 'VRAA'

                )

            )

        );

        $method\_name = 'InsertSubscriberToChannel';

        $result = $service->\_\_soapCall($method\_name, array($req) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        // Channel subscriber found

    }

    $dom = new DOMDocument('1.0', 'UTF-8');

    $dom->loadXML(file\_get\_contents('xml/data.xml'));

    $req =  array(

        'MessageSave2' => array(

            'ReceiverFilter' => array(

                'Authority' => array('100001')

            ),

            'ChannelVersionURN' => $channel\_version\_URN,

            'BussinesMessage' => array(

                'any' => array(

                    $dom->saveXML($dom->documentElement)

                )

            )

        )

    );

    $method\_name = 'SendMessage2';

    $result2 = $service2->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service2, $DIT\_WCF\_service\_addr, $method\_name);

    $req = array(

        'MessageRequest' => array(

            'ChannelVersionURN' => $channel\_version\_URN

        )

   );

   $method\_name = 'ReceiveMessage2';

   $result = $service3->\_\_soapCall($method\_name, array($req));

   $ms->addServiceCallMilestone($service3, $DIT\_WCF\_service\_addr, $method\_name);

   $messageId = $result->MessageResponse2->Items[0]->MessageID;

   $req = array(

    'MessageSampleId' => $messageId

    );

    $method\_name = 'ConfirmationAck';

    $result = $service3->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service3, $DIT\_WCF\_service\_addr, $method\_name);

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    echo $fault;

}

} catch (Exception $ex)  {

    var\_dump($ex);

}

?>



.attēls. Iestāde pieslēgta DIT

#### Ziņojuma nosutīšana vienam kanāla dalibniekam (SendMessageReceiver)

Tiek pārbaudīts, vai eksistē kanāla versija. Ja tā ir, tad uz to tiek nosūtīts ziņojums saņēmējam.

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$service = new mySOAP($DIT\_config\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_config\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$service2 = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $service2->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel578';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $req =  array(

        'ChannelSearch'        => array(

            'OnlyHeader' => true,

            'Channel' =>  array(

                'ChannelURN' => $channel\_urn

            )

        ),

        'ChannelVersionStatus' => array('creating', 'testing', 'active', 'tehnicalbreak', 'activedepricated', 'deleted')

    );

    $method\_name = 'SearchChannelsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    // setting channel data

    $channel = array(

        'ShortName' => $channel\_name,

        'ChannelURN' => $channel\_urn,

        'ChannelType' => 'dik',

        'OwnerAuthority' => array(

            'AuthorityID' => '100000',

            'ShortName' => 'ABC software',

        ),

        'ChannelName' => $channel\_name

    );

    //if channel not found creating new one

    if($result->ChannelList->Count == 0) {

        // adding channel data to request

        $req2 =  array(

            'Channel' => $channel

        );

        $method2\_name = 'InsertChannel';

        $result2 = $service->\_\_soapCall($method2\_name, array($req2) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method2\_name);

        $result2->InsertChannelResponse;

    } else {

        //channel found

    }

    $channel\_version = array(

        'MessageType' => array(

            0 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '1'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-Person-FullName-v1-0-TYPE-PersonFullName'

            ),

            1 =>  array(

                'MessageClass' => array(

                    'CodeListCodeValue' => '3'),

                'XMLSchemaURN' => 'URN:IVIS:100001:XSD-DIT-DITMessage-v1-0-TYPE-DefaultMessageError',

            )

        ),

        'ChannelVersionStatus' => 'active',

        'Channel' => $channel,

        'MinorVersion' => $minor\_v,

        'MajorVersion' => $major\_v,

        'ChannelVersionURN' => $channel\_version\_URN

    );

    $req3 =  array(

        'ChannelVersionSearch' => array(

            'ChannelVersion' => $channel\_version

        )

    );

    $method\_name = 'SearchChannelVersionsPaged';

    $result = $service->\_\_soapCall($method\_name, array($req3) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    //if channel version not found creating new one

    if($result->ChannelVersionsList->Count == 0) {

        // adding channel version data to request

        $req =  array(

            'ChannelVersionData' => $channel\_version

        );

        $method\_name = 'InsertChannelVersion';

        $result = $service->\_\_soapCall($method\_name, array($req) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        //channel version found

    }

    $req = array(

        'ChannelSubscriberSearch' => array(

            'ChannelSubscriber' => array(

                'ChannelVersionURN' => $channel\_version\_URN,

                'SubscriberAuthority' => array(

                    'AuthorityID' => '100001',

                    'ShortName' => 'VRAA',

                    'Name' => 'VRAA'

                )

            )

        )

    );

    $method\_name = 'SearchSubscribersToChannelPaged';

    $result = $service->\_\_soapCall($method\_name, array($req) );

    $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    if($result->ChannelSubscriberList->Count == 0) {

        // Channel subscriber created

        $req = array(

            'ChannelSubscriber' => array(

                'ChannelVersionURN' => $channel\_version\_URN,

                'SubscriberAuthority' => array(

                    'AuthorityID' => '100001',

                    'ShortName' => 'VRAA',

                    'Name' => 'VRAA'

                )

            )

        );

        $method\_name = 'InsertSubscriberToChannel';

        $result = $service->\_\_soapCall($method\_name, array($req) );

        $ms->addServiceCallMilestone($service, $DIT\_config\_service\_addr, $method\_name);

    } else {

        // Channel subscriber found

    }

    $dom = new DOMDocument('1.0', 'UTF-8');

    $dom->loadXML(file\_get\_contents('xml/data.xml'));

    $req =  array(

        'MessageSave2' => array(

            'ReceiverFilter' => array(

                'Authority' => array('100001')

            ),

            'ChannelVersionURN' => $channel\_version\_URN,

            'BussinesMessage' => array(

                'any' => array(

                    $dom->saveXML($dom->documentElement)

                )

            )

        )

    );

    $method\_name = 'SendMessage2';

    $result2 = $service2->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service2, $DIT\_WCF\_service\_addr, $method\_name);

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    echo $fault;

}

} catch (Exception $ex)  {

    var\_dump($ex);

}

?>

#### Ziņojuma nosutīšana vienam kanāla dalibniekam (SendReplay)

<?php

header("Content-Type: application/json; charset=UTF-8");

require\_once('lib/milestone.php');

require\_once('client.my.php');

require\_once('config.php');

$sts = new STSClient(NULL, array(

                            'trace' => 1,

                            'location' => $sts\_address.'/trust/13/certificatemixed',

                            'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                            'exceptions' => TRUE,

                            'style'    => SOAP\_DOCUMENT,

                            'use'      => SOAP\_LITERAL,

                            'soap\_version' => SOAP\_1\_2

                            ));

$sts2 = new STSClient(NULL, array(

                                'trace' => 1,

                                'location' => $sts\_address.'/trust/13/usernamemixed',

                                'uri' => 'http://docs.oasis-open.org/ws-sx/ws-trust/200512',

                                'exceptions' => TRUE,

                                'style'    => SOAP\_DOCUMENT,

                                'use'      => SOAP\_LITERAL,

                                'soap\_version' => SOAP\_1\_2

                                ));

$service = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                            'exceptions' => TRUE,

                            'soap\_version' => SOAP\_1\_2,

                            'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                            'trace' => 1));

$service2 = new mySOAP($DIT\_WCF\_service\_addr.'?wsdl', array(

                                'exceptions' => TRUE,

                                'soap\_version' => SOAP\_1\_2,

                                'location' => $DIT\_WCF\_service\_addr.'/ws2007FederationNoSct',

                                'trace' => 1));

$ms = Milestone::getInstance();

try {

try {

    $claims[] = array('uri' => "http://www.oasis-open.org/RSA2004/attributes/AUTHORITY");

    $claims[] = array('uri' => "http://docs.oasis-open.org/wsfed/authorization/200706/claims/action");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid");

    $claims[] = array('uri' => "http://ivis.eps.gov.lv/schema/identity/claims/employeegroup");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname");

    $claims[] = array('uri' => "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name");

    $claims[] = array('uri' => "urn:ivis:100001:name.id-viss");

    $cert = file\_get\_contents('cert/public.pem');

    $private\_key = file\_get\_contents('cert/not\_enc.pem');

    $tokens = $sts->issueTokenCert($cert, $private\_key, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts, $sts\_address);

    // set Token for service call - we use only first token from collection if many returned

    $service->customtoken = $tokens[0];

    $channel\_name = 'SampleChannel';

    $channel\_urn = 'URN:IVIS:100001:CHA-'.$channel\_name.'-TYPE-DIK';

    $minor\_v = 0; //version minor value

    $major\_v = 1; //version major value

    $channel\_version\_URN = 'URN:IVIS:100001:CHA-'  .$channel\_name. '-v' . $major\_v . "-" . $minor\_v . '-TYPE-DIK';

    $dom = new DOMDocument('1.0', 'UTF-8');

    $dom->loadXML(file\_get\_contents('xml/data.xml'));

    $req =  array(

        'MessageSave2' => array(

            'ReceiverFilter' => array(

                'Authority' => array('100001')

            ),

            'ChannelVersionURN' => $channel\_version\_URN,

            'BussinesMessage' => array(

                'any' => array(

                    $dom->saveXML($dom->documentElement)

                )

            )

        )

    );

    $method\_name = 'SendMessage2';

    $result2 = $service->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

    // set UsernameToken variables and request for tokens

    $username = 'usernamevraa';

    $password = 'Password';

    $tokens = $sts2->issueToken($username, $password, $apply\_to, $claims);

    $ms->addSTSCallmilestone($sts2, $sts\_address);

     // set Token for service call - we use only first token from collection if many returned

    $service2->customtoken = $tokens[0];

    $req = array(

         'MessageRequest' => array(

             'ChannelURN' => $channel\_urn

        )

    );

    $method\_name = 'ReceiveMessage2';

    $result = $service2->\_\_soapCall($method\_name, array($req));

    $ms->addServiceCallMilestone($service2, $DIT\_WCF\_service\_addr, $method\_name);

    $messageId = $result->MessageResponse2->Items[0]->MessageID;

    $dom = new DOMDocument('1.0', 'UTF-8');

    $dom->loadXML(file\_get\_contents('xml/kluda.xml'));

    $req =  array(

        'MessageSave2' => array(

            'ChannelVersionURN' => $channel\_version\_URN,

            'EventMessage' => array(

                'any' => array(

                    $dom->saveXML($dom->documentElement)

                ),

                'Name' => 'Error!',

                'RelatedMessageID' => $messageId

            )

        )

    );

    $method\_name = 'SendMessage2';

    //$result2 = $service->\_\_soapCall($method\_name, array($req));

    //$ms->addServiceCallMilestone($service, $DIT\_WCF\_service\_addr, $method\_name);

    $ms->fromatXML();

    echo $ms->getJSON();

} catch (SoapFault $fault) {

    echo $fault;

}

} catch (Exception $ex)  {

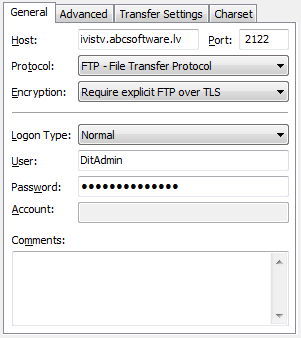
    var\_dump($ex);

}

?>

### FTP piemēri PHP

DIT FTP pieslēgumu var pārbaudīt, izmantojot FTP klientu, piemēram, *FileZilla. FileZilla ftp* kanāla konfigurācija ir redzama 20.attēlā.



42.attēls. FileZilla FTPs kanāla konfigurācija

#### Datnes pievienošana (FTPs\_SendMessage)

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA-SampleChannel2-v1-0-TYPE-DIK” kanālam:

<?php

try {

    // set UsernameToken variables and request for ftp

    $username = 'DitAdmin';

    $password = 'Password';

    $ftp\_host = 'dittest.vraa.gov.lv';

    $ftp\_port = 2121;

    $channel\_name = 'AAPE';

    // set up basic connection

    $conn\_id = ftp\_ssl\_connect($ftp\_host, $ftp\_port);

    // login with username and password

    $login\_result = ftp\_login($conn\_id, $username, $password);

    // check connection

    if ((!$conn\_id) || (!$login\_result)) {

        echo "FTP connection has failed!";

        echo "Attempted to connect to $ftp\_host for user $username\n";

        exit;

    } else {

        echo "Connected to $ftp\_host, for user $username\n";

    }

    // upload the file

    $dest\_folder = $channel\_name . '/DIK-v2-3/Write/';

    $upload = ftp\_put($conn\_id, $dest\_folder.'message2.xml', 'xml/data.xml', FTP\_ASCII);

    // check upload status

    if (!$upload) {

        echo "FTP upload has failed!";

    } else {

        echo "Uploaded succesfull";

    }

    // close the FTP stream

    ftp\_close($conn\_id);

} catch (Exception $ex)  {

    //echo $ex->getMessage();

    var\_dump($ex);

}

?>

#### Datnes nolasīšana un dzēšana (FTPs\_ReceiveMessage)

Tiek izveidots pieslēgums FTP, un tiek nolasīta datne no kanāla mapes, kura atbilst „URN:IVIS:100001:CHA-SampleChannel2-v1-0-TYPE-DIK” kanālam:

<?php

try {

    // set UsernameToken variables and request for ftp

    $username = 'DitAdmin';

    $password = 'Password';

    $ftp\_host = 'dittest.vraa.gov.lv';

    $ftp\_port = 2121;

    $channel\_name = 'SampleChannel';

    // set up basic connection

    $conn\_id = ftp\_ssl\_connect($ftp\_host, $ftp\_port);

    // login with username and password

    $login\_result = ftp\_login($conn\_id, $username, $password);

    // check connection

    if ((!$conn\_id) || (!$login\_result)) {

        echo "FTP connection has failed!";

        echo "Attempted to connect to $ftp\_host for user $username\n";

        exit;

    } else {

        echo "Connected to $ftp\_host, for user $username\n";

    }

    // get ftp folder file list

    $dest\_folder = $channel\_name . '/DIK-v1-0/ReadError';

    $file\_list = ftp\_nlist($conn\_id, $dest\_folder);

    foreach($file\_list as $file\_name) {

        if($file\_name != '.' && $file\_name != '..' && strstr($file\_name, '.xml'))

        {

            $file\_name = substr($file\_name, strlen($dest\_folder) + 1);

            // open some file to write to

            $handle = fopen('xml/'.$file\_name, 'w');

            // try to download $file\_name and save it to $handle

            if (ftp\_fget($conn\_id, $handle, $dest\_folder.'/'.$file\_name, FTP\_ASCII, 0)) {

                echo "successfully written to xml/$file\_name\n";

                // if file is read, then delete it

                if (ftp\_delete($conn\_id, $dest\_folder.'/'.$file\_name)) {

                    echo "file deleted successfully\n";

                } else {

                    echo "could not delete file\ n";

                }

            } else {

                echo "There was a problem while downloading remote file \n";

            }

        }

    }

    // close the FTP stream

    ftp\_close($conn\_id);

} catch (Exception $ex)  {

    //echo $ex->getMessage();

    var\_dump($ex);

}

?>

#### Ziņojuma sutīšana vienam kanāla dalibniekam (FTPSendMessageReceiver)

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA-SampleChannel2-v1-0-TYPE-DIK” kanālam:

<?php

try {

    // set UsernameToken variables and request for ftp

    $username = 'DitAdmin';

    $password = 'Password';

    $ftp\_host = 'dittest.vraa.gov.lv';

    $ftp\_port = 2121;

    $channel\_name = 'SampleChannel';

    $authority = '100001';

    // set up basic connection

    $conn\_id = ftp\_ssl\_connect($ftp\_host, $ftp\_port);

    // login with username and password

    $login\_result = ftp\_login($conn\_id, $username, $password);

    // check connection

    if ((!$conn\_id) || (!$login\_result)) {

        echo "FTP connection has failed!";

        echo "Attempted to connect to $ftp\_host for user $username\n";

        exit;

    } else {

        echo "Connected to $ftp\_host, for user $username\n";

    }

    // upload the file

    $dest\_folder = $channel\_name . '/DIK-v1-0/Write/'.$authority.'/';

    $upload = ftp\_put($conn\_id, $dest\_folder.'message2.xml', 'xml/data.xml', FTP\_ASCII);

    // check upload status

    if (!$upload) {

        echo "FTP upload has failed!";

    } else {

        echo "Uploaded succesfull";

    }

    // close the FTP stream

    ftp\_close($conn\_id);

} catch (Exception $ex)  {

    //echo $ex->getMessage();

    var\_dump($ex);

}

?>

#### Ziņojuma atbildēšana (FTPSendReply)

Tiek izveidots pieslēgums FTP, un tiek pievienota datne kanāla mapei, kura atbilst „URN:IVIS:100001:CHA- SampleChannel-v1-0-TYPE-DIK” kanālam zem WriteEventError:

<?php

try {

    // set UsernameToken variables and request for ftp

    $username = 'usernamevraa';

    $password = 'Password';

    $ftp\_host = 'dittest.vraa.gov.lv';

    $ftp\_port = 2121;

    $channel\_name = 'SampleChannel';

    // set up basic connection

    $conn\_id = ftp\_ssl\_connect($ftp\_host, $ftp\_port);

    // login with username and password

    $login\_result = ftp\_login($conn\_id, $username, $password);

    // check connection

    if ((!$conn\_id) || (!$login\_result)) {

        echo "FTP connection has failed!";

        echo "Attempted to connect to $ftp\_host for user $username\n";

        exit;

    } else {

        echo "Connected to $ftp\_host, for user $username\n";

    }

    // upload the file

    $dest\_folder = $channel\_name . '/DIK-v1-0/WriteEvent/WriteEventError/';

    $upload = ftp\_put($conn\_id, $dest\_folder.'2.xml', 'xml/kluda.xml', FTP\_ASCII);

    // check upload status

    if (!$upload) {

        echo "FTP upload has failed!";

    } else {

        echo "Uploaded succesfull";

    }

    // close the FTP stream

    ftp\_close($conn\_id);

} catch (Exception $ex)  {

    //echo $ex->getMessage();

    var\_dump($ex);

}

?>