Publiskās pārvaldes dokumentu pārvaldības sistēmu integrācijas vides izveide

V.

VRAA.DIV.INS.BIB.Bibliotēkas\_integrācijas\_instrukcija.1.3.3.2017.12.15

Apstiprinājumu lapa

|  |  |
| --- | --- |
| **Piegādātājs:** | **Pasūtītājs:** |
|  | A/S "RIX Technologies" |  |  | Valsts reģionālās attīstības aģentūra |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | paraksts |  |  | paraksts |  |
|  |  |  |  |  |  |
|  | datums |  |  | datums |  |

Dokumenta autortiesības

Šo dokumentu Valsts reģionālās attīstības aģentūras (VRAA) uzdevumā ir izstrādājis A/S "RIX Technologies" projekta „” ietvaros. Dokuments ir izmantojams saskaņā ar 2011. gada 22. marta iepirkuma līguma ID Nr. „” nosacījumiem un saskaņā ar 2015.gada 15.jūnija Vispārīgo vienošanos Nr. “13-7/15/33”.

Kontaktpersonas

R. Prikulis

A/S "RIX Technologies"

Projektu vadītājs

Blaumaņa iela 5a

Rīga, LV 1011

Tālr.: +371 67142990

Fakss: +371 67142991

E-pasts: ritvars.prikulis@rixtech.lv

WWW: [www.rixtech.lv](http://www.rixtech.lv/)

Valsts reģionālās attīstības aģentūras

Informācijas sistēmu attīstības departamenta

Integrāciju nodaļas vadītājs

Alberta iela 10, Rīga, LV 1010

Tālr.: +371 67079022,

Fakss: +371 67079001

WWW:

<http://www.vraa.gov.lv>

Izmaiņu vēsture

|  |  |  |  |
| --- | --- | --- | --- |
| Datums | Versija | Apraksts | Autors |
| 12.04.2012 | 1.0.0 | Dokumenta sākuma versija. | Gints PlivnaRitvars Prikulis |
| 16.04.2012 | 1.1.0 | Papildinājumi atbilstoši dokumenta iekšējai kvalitātes kontrolei. | R. Prikulis |
| 02.05.2012 | 1.1.1 | Papildinājumi atbilstoši CKS. | Gints PlivnaRitvars Prikulis |
| 09.05.2012 | 1.1.2 | Papildinājumi atbilstoši CKS.Pievienota informācija par TRACEINFO laukiem. | Gints PlivnaRitvars Prikulis |
| 10.09.2012 | 1.2.0 | Papildināti integrācijas piemēri, pēc Pasūtītāja sadarbības partneru jautājumu (neskaidrību) apkopošanas. | Ritvars Prikulis |
| 19.09.2012 | 1.2.1 | Ar Java koda piemēriem papildinātas nodaļas „Ziņojuma servera apliecinājuma izgūšana” un „Ziņojumu asinhrona apstiprināšana”.Pievienota nodaļa „Paziņojumu (notifikāciju) saņemšana, izmantojot tīmekļa pakalpi”. | Ritvars Prikulis |
| 10.08.2014 | 1.2.2 | Aktualizēta informācija par atbalstāmām Java versijām.Papildināta nodaļa „Bibliotēkas konfigurācija”.Pievienota nodaļa „Java bibliotēkas uzstādīšanas informācija”. | Ritvars Prikulis |
| 05.09.2014 | 1.2.3 | Pievienota informācija par JAVA kodēšanas algoritmiem. | Ritvars Prikulis |
| 12.12.2016 | 1.2.4 | Precizēta 2.nodaļa - .NET bibliotēka ar 12.12.2016(jauna bibliotēkas versija) nodrošina arī .NET4.0 64bit atbalstu. | Ritvars Prikulis |
| 01.09.2017 | 1.3.0 | Mainīta nodaļa 4.5. “Adresātu sinhronizācija”.Pievienota nodaļa “4.6 E-adreses pārbaude” | Gints PlivnaRitvars Prikulis |
| 27.09.2017 | 1.3.1 | Precizētas nodaļas:* “E-adreses pārbaude”
* “Adresātu sākotnējā sinhronizācija”
* Adresātu aktuālo izmaiņu sinhronizācija
 | Gints PlivnaRitvars Prikulis |
| 30.11.2017 | 1.3.2 | Papildinātas nodaļas (pievienoti JAVA piemēri):* “E-adreses pārbaude”
* “Adresātu sākotnējā sinhronizācija”
* Adresātu aktuālo izmaiņu sinhronizācija

Papildinātas nodaļas:* Apakšadresāciju vienību pārvaldība
* Prasības izpildes videi
 | Gints PlivnaRitvars Prikulis |
| 01.12.2017 | 1.3.3 | Pievineota nodaļa:* Bibliotēkas biznesa likumu paziņojumi;
* Kataloga e-adrešu aktualizēšana
 | Ritvars PrikulisSergejs Deņisovs |

Saturs

[1 Ievads 8](#_Toc501127987)

[1.1 Dokumenta nolūks 8](#_Toc501127988)

[1.2 Dokumenta struktūra 8](#_Toc501127989)

[1.3 Saīsinājumi 8](#_Toc501127990)

[1.4 Saistītie dokumenti 9](#_Toc501127991)

[2 Prasības izpildes videi 10](#_Toc501127992)

[3 Bibliotēkas konfigurācija 11](#_Toc501127993)

[3.1 .NET bibliotēkas konfigurācijas piemērs 11](#_Toc501127994)

[3.2 JAVA bibliotēkas konfigurācijas piemērs 12](#_Toc501127995)

[3.2.1 Java bibliotēkas uzstādīšanas informācija 13](#_Toc501127996)

[3.3 Bibliotēkas konfigurācijas parametru apraksts 14](#_Toc501127997)

[4 Izmantošanas scenāriji 15](#_Toc501127998)

[4.1 Bibliotēkas pieslēgšana izstrādes videi 15](#_Toc501127999)

[4.2 Bibliotēkas funkcionalitātes izmantošana 15](#_Toc501128000)

[4.3 Ziņojumu apmaiņas scenāriji 15](#_Toc501128001)

[4.3.1 Ziņojuma nosūtīšana bez datnēm 17](#_Toc501128002)

[4.3.2 Ziņojuma nosūtīšana ar datnēm 18](#_Toc501128003)

[4.3.3 Atbildes ziņojuma sūtīšana par saņemtu ziņojumu 19](#_Toc501128004)

[4.3.4 Ziņojuma sūtīšanas tehniskie parametri 20](#_Toc501128005)

[4.3.5 Ziņojumu specifisko metadatu uzstādīšana, izmantojot XML failu 21](#_Toc501128006)

[4.3.6 Ziņojumu specifisko metadatu uzstādīšana, izmantojot datu klases 23](#_Toc501128007)

[4.3.7 Ziņojuma specifisko metadatu izgūšana 26](#_Toc501128008)

[4.3.8 Paziņojumu (notifikāciju) saņemšana 27](#_Toc501128009)

[4.3.9 Paziņojumu (notifikāciju) apstiprināšana 28](#_Toc501128010)

[4.3.10 Paziņojumu (notifikāciju) saņemšana, izmantojot tīmekļa pakalpi 29](#_Toc501128011)

[4.3.11 Ziņojumu saņemšana bez datnēm, izgūstot ziņojuma galveni 29](#_Toc501128012)

[4.3.12 Ziņojumu saņemšana bez datnēm, izgūstot pēc kārtas 30](#_Toc501128013)

[4.3.13 Ziņojumu saņemšana ar datnēm 30](#_Toc501128014)

[4.3.14 Ziņojumu sinhrona apstiprināšana 32](#_Toc501128015)

[4.3.15 Ziņojumu asinhrona apstiprināšana 33](#_Toc501128016)

[4.3.16 Ziņojuma servera apliecinājuma izgūšana 34](#_Toc501128017)

[4.3.17 Ziņojuma informācijas apstrāde 36](#_Toc501128018)

[4.4 Apakšadresāciju vienību pārvaldība 36](#_Toc501128019)

[4.4.1 Adresācijas vienību izgūšana 36](#_Toc501128020)

[4.4.2 Adresācijas vienību reģistrēšana 37](#_Toc501128021)

[4.4.3 Adresācijas vienību atjaunināšana 37](#_Toc501128022)

[4.4.4 Adresācijas vienību dzēšana 38](#_Toc501128023)

[4.4.5 Adresācijas vienību meklēšana 38](#_Toc501128024)

[4.5 Adresātu sinhronizācija 39](#_Toc501128025)

[4.5.1 Adresātu sākotnējā sinhronizācija 40](#_Toc501128026)

[4.5.2 Adresātu aktuālo izmaiņu sinhronizācija 41](#_Toc501128027)

[4.6 E-adreses pārbaude 44](#_Toc501128028)

[4.6.1 Sinhrona E-adreses pārbaude 44](#_Toc501128029)

[4.6.2 Asinhrona E-adreses pārbaude 45](#_Toc501128030)

[5 Kataloga e-adrešu aktualizēšana 47](#_Toc501128031)

[5.1 Valsts iestādes izveidošana 47](#_Toc501128032)

[5.1.1 .NET bibliotēkas koda piemērs 47](#_Toc501128033)

[5.2 Reģistros reģistrētu un fizisku personu izveidošana 47](#_Toc501128034)

[5.2.1 .NET bibliotēkas koda piemērs 47](#_Toc501128035)

[5.3 Masveida statusa atjaunošana 48](#_Toc501128036)

[5.3.1 .NET bibliotēkas koda piemērs 48](#_Toc501128037)

[5.4 Masveida īpašieku datu atjaunošana 48](#_Toc501128038)

[5.4.1 .NET bibliotēkas koda piemērs 48](#_Toc501128039)

[5.5 Masveida operāciju rezultātu izgūšana 49](#_Toc501128040)

[5.5.1 .NET bibliotēkas koda piemērs 49](#_Toc501128041)

[6 Bibliotēkas biznesa likumu paziņojumi 50](#_Toc501128042)

# Ievads

## Dokumenta nolūks

Šis dokuments ir veidots „Publiskās pārvaldes dokumentu pārvaldības sistēmu integrācijas vides” (DIV) sistēmas izveides projektēšanas fāzes ietvaros. Dokuments satur integrācijas instrukcijas citu sistēmu izstrādātājiem, sniedzot priekšstatu par sadarbības iespējam ar DIV sistēmu.

Klientu DVS un citas programmatūras integrācija ar DIV notiks izmantojot DVS integrācijas bibliotēkas (Java) vai Portāla adaptera (.NET) komponentes. Šajā dokumentā aprakstīto scenāriju piemēri izmanto Portāla adapteru un .NET vidi.

## Dokumenta struktūra

Dokuments sastāv no ievada un 3 nodaļām:

* Prasības izpildes videi sniedz informāciju par bibliotēku darbināšanai nepieciešamu infrastruktūru un izpildes vides konfigurāciju.
* Bibliotēkas konfigurācija apraksta bibliotēku konfigurējamas vērtības un to nozīmi.
* Izmantošanas scenāriji satur informāciju par bibliotēkas pieslēgšanu izstrādes videi, ka arī tās funkcionālām iespējām.

## Saīsinājumi

Dokumentā izmantotie saīsinājumi ir paskaidroti tabulā Definīcijas un saīsinājumi.

Tabula 1: Definīcijas un saīsinājumi

|  |  |
| --- | --- |
| **Definīcija vai saīsinājums** | **Skaidrojums** |
| DIV | Dokumentu integrācijas vide. |
| Projekts | Publiskās pārvaldes dokumentu pārvaldības sistēmu integrācijas vides izveide. |
| Pasūtītājs, VRAA | Valsts reģionālās attīstības aģentūra. |
| Piegādātājs, RIX | A/S "RIX Technologies". |
| Sistēma | Publiskās pārvaldes dokumentu pārvaldības sistēmu integrācijas vide. |
| Klients | Ārēja informācijas sistēma – DVS, portāls vai cita, kas pieslēdzas DIV, izmantojot universālo vienoto programmatūras saskarni |
| Komponente | Relatīvi neatkarīga programmatūras, tās dokumentācijas vai projekta dokumentācijas daļa, kura tiek projektēta vai uzturēta. |
| DVS | Dokumentu vadības sistēma. |
| Ziņojums | Datu bloks, ar ko apmainās sistēmas klientu DVS sistēmas informācijas pārsūtīšanai, izmantojot DIV piedāvātas iespējas. Sastāv no dokumenta aploksnes XML dokumenta un, opcionāli, piesaistītām datnēm. |
| Dokumenta aploksne | Ziņojuma pamata sastāvdaļa, reprezentēta XML dokumenta veidā. Satur dokumenta metadatus (vispārīgos un specifiskos), transporta metadatus un elektroniskos parakstus. Dokumenta aploksnes struktūra ir aprakstīta saistītā dokumentā [3. Sākotnējie standarti]. |
| .NET | Microsoft programmatūras izstrādes un darbināšanas vide. |
| Java | Sun Microsystems (pašlaik Oracle apakšvienība) programmēšanas valoda, dokumenta kontekstā arī programmatūras platforma. |
| API | Application Program Interface – bibliotēkas vai citas programmatūras saskarne, kas definē piedāvātas funkcionālas iespējas.  |
| X509 | Publiskās atslēgas infrastruktūras standarts ciparsertifikātiem un to pārbaudei. |
| XML | Extensible Markup Language – tīmekļa dokumentu reprezentācijas veids. |
| WSDL | Web Services Description Language - tīmekļa pakalpju aprakstes valoda, definē tīmekļa pakalpes saskarni. |
| .cacerts | Java vides fails, kas satur sertificēšanas institūciju saknes ciparsertifikātus. |
| App.config/Web.config | .NET vides konfigurācijas faili attiecīgi izpildāmām lietotnēm un tīmekļa lietotnēm. |
| .properties | Faila paplašinājums Java konfigurācijas failiem. |

## Saistītie dokumenti

Šis dokuments ir lasāms kopā ar sekojošiem dokumentiem:

1. Tehniskā specifikācija. Publiskās pārvaldes dokumentu pārvaldības integrācijas vides izveide. Valsts reģionālās attīstības aģentūra, SIA “AA Projekts”, 2010.
2. Sistēmas konceptuālā arhitektūra. Publiskās pārvaldes dokumentu pārvaldības integrācijas vides izveide. Valsts reģionālās attīstības aģentūra, SIA „Rix Technologies”, 2011.
3. Sākotnējie standarti. Publiskās pārvaldes dokumentu pārvaldības sistēmu integrācijas vides izveide. V.2.0.2. Valsts reģionālās attīstības aģentūra, SIA „AA Projekts”, SIA „Evolution Consulting”, SIA „Rix Technologies”, 2011.
4. Sistēmas integrācijas instrukcija. Publiskās pārvaldes dokumentu pārvaldības sistēmu integrācijas vides izveide. Valsts reģionālās attīstības aģentūra, SIA „Rix Technologies”, 2017.

# Prasības izpildes videi

Portāla adaptera darbināšanai būs nepieciešama .NET Framework izpildes vides versijas 3.5 SP1 vai 4.

DVS integrācijas bibliotēka tiek atbalstīta:

* Java SE Runtime Environment 7 vides kontekstā (JAVA 1.7.60 (Version 7 Update 60)).
* .NET 3.5 SP1 vai .NET 4.

Komponenšu darbība tiks atbalstīta operētājsistēmās, kur ražotājs atbalsta atbilstošo izpildes vidi.

Bibliotēkas autentifikācijai izmantos X509 ciparsertifikātu. Klientu DVS programmatūrai jābūt pieejai ciparsertifikātam un tā privātai atslēgai. Ciparsertifikāta instalāciju Windows Certificate Store (.NET) vai Java Key Store (Java) veic klientu sistēmas administrators atbilstoši piegādātajām instrukcijām.

Korektai bibliotēku izpildei visiem izmantotiem ciparsertifikātiem, ieskaitot klientu autentifikācijas un DIV sistēmu ciparsertifikātus, jāvalidējas bez kļūdām. Tas nozīmē, ka būs nepieciešams nokonfigurēt visu šo ciparsertifikātu izdevēj institūcijas kā uzticamas. Šim nolūkam izdevēj institūcijas ciparsertifikātu publiskās daļas ir instalējamas Windows Trusted Root Certificate Store .NET vidē, un, atbilstoši, Java atslēgu glabātuvē (piemēram, cacerts. failā).

**Biznesa kļūdu paziņojumi tiek uzturēti Sistēmas integrācijas instrukcijā [4].**

# Bibliotēkas konfigurācija

Pamata bibliotēkas konfigurācijas scenārijs paredz .NET konfigurācijas faila izmantošanu iestatījumu pārvaldībai (App.config vai Web.config faili). Java bibliotēkai ir līdzīgas iespējas, kuras balstās uz App.config faila pielietošanu.

DIV bibliotēkās, lai nodrošinātu universālu pieeju, tiek izmantota standartizēta datņu saspiešanas metodika. Uzstādot, lai tiek saspiestas visas nosūtāmās datnes var novest pie tā, ka jau arhivēto datņu apjoms sūtīšanas laikā palielinās par aptuveni 30% (edoc, zip, docx u.c). Lai novērstu nevajadzīgas datu plūsmas, lūdzam rūpīgi iepazīties ar instrukcijas sadaļā aprakstītā konfigurēšanas iespējām un konfigurācijas datnē norādīt nepieciešamās bibliotēkas uzvedības vērtības. Rekomendējama šāda konfigurēšanas pieeja:

* Vērtību CompressByDefault uzstādīt uz False (skatīt zemāk piemēros);
* Datņu tipiem, kurus nepieciešams saspiest, norādīt compress uz true;

Ja integrējamai sistēmai, jau ir pieejami savas konfigurācijas datnes, tad līdzīgu loģiku iespējams nodrošināt arī „IntegrationMessage” un „IntegrationClient” klasēs.

## .NET bibliotēkas konfigurācijas piemērs

Minimālā .NET bibliotēkas konfigurācija ir sekojoša:

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

 <configSections>

 <section name="div.client" type="Vraa.Div.Client.Configuration.MainSection, Vraa.Div.Client"/>

 </configSections>

 <certificates>

 <add thumbprint="6562daf69228df50a7f15b84dbd585bc25fef6b0"/>

 </certificates>

 </div.client>

</configuration>

Konfigurācijas piemērs ar visu konfigurācijas parametru aizpildīšanu:

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

 <configSections>

 <section name="div.client" type="Vraa.Div.Client.Configuration.MainSection, Vraa.Div.Client"/>

 </configSections>

 <div.client

 serviceAddress="https://vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc"

 timeout="120"

 compressByDefault="false"

 encryptByDefault="true">

 <certificates>

 <add thumbprint="71 7c 2f 8f 70 04 63 49 eb c4 d2 c3 43 8e 41 3f b7 fc b0 88"/>

 <add thumbprint="A54303DEFDE7F678D9E105DB6804A38A74B4137D" isUserStore="true"/>

 </certificates>

 <fileTypes>

 <add extension="doc" compress="true" mimeType="application/msword"/>

 <add extension="zip" compress="false"/>

 </fileTypes>

 </div.client>

</configuration>

Aizpildot tikai minimālo konfigurācija datu komplektu pārējās vērtības tiek uzstādītas no noklusētajām bibliotēkas vērtībām.

Konfigurācijas faila izmantošana ļauj mainīt vērtības bez DVS sistēmas koda izmaiņām, bibliotēkas klienta objekta izveidošanai nebūs vajadzīga obligāto parametru norādīšana:

IntegrationClient client = new IntegrationClient();

//...darbs ar bibliotēku

Bibliotēka atbalsta arī programmējamu konfigurācijas uzstādīšanu. Koda piemērs, kur tiek uzstādīta konfigurācija inicializējot bibliotēku .NET kodā:

ClientConfiguration clientConfig = new ClientConfiguration();

// Konfigurācijas uzstādīšana

clientConfig.ServiceAddress = "https://div.vraa.gov.lv/vraa.div.webservice.unifiedinterface/unifiedservice.svc";

clientConfig.Certificates.Add("d9 94 3c 99 37 93 6b 7a 77 f1 d5 a8 dc d0 fe 4c e0 c0 aa 49");

clientConfig.Timeout = 100;

// Konfigurācijas nodošana bibliotēkai.

IntegrationClient client = new IntegrationClient(clientConfig);

//...darbs ar bibliotēku

## JAVA bibliotēkas konfigurācijas piemērs

Minimālā Java bibliotēkas konfigurācija ir sekojoša:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<div.client>

 <certificates>

 <certificate>

 <store>PKCS12</store>

 <storeFile>C:\temp\Janka\_ceturtais.pfx</storeFile>

 <password>12345678</password>

 <thumbprint>0a a4 ef c6 87 6e 09 70 40 57 2c cb e6 ee ca 04 ec 65 0c 8d</thumbprint>

 </certificate>

 </certificates>

</div.client>

Konfigurācijas piemērs ar visu konfigurācijas parametru aizpildīšanu:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<div.client>

 <serviceAddress>https://divtest.vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc</serviceAddress>

 <compressByDefault>true</compressByDefault>

 <encryptByDefault>false</encryptByDefault>

 <timeout>10</timeout>

 <certificates>

 <certificate>

 <store>PKCS12</store>

 <storeFile>C:\temp\sertifikats1.pfx</storeFile>

 <password>12345678</password>

 <thumbprint>2D 13 9D 32 C5 76 21 88 79 46 50 08 80 72 5F 2E 56 28 39 93</thumbprint>

 </certificate>

 <certificate>

 <store>PKCS12</store>

 <storeFile>C:\temp\sertifikats2.pfx</storeFile>

 <password>12345678</password>

 <thumbprint>0a a4 ef c6 87 6e 09 70 40 57 2c cb e6 ee ca 04 ec 65 0c 8d</thumbprint>

 </certificate>

 </certificates>

 <fileTypes>

 <fileType>

 <extension>txt</extension>

 <mimeType>text/plain</mimeType>

 <compress>true</compress>

 </fileType>

 <fileType>

 <extension>doc</extension>

 <mimeType>application/msword</mimeType>

 <compress>true</compress>

 </fileType>

 <fileType>

 <extension>zip</extension>

 <compress>false</compress>

 </fileType>

 </fileTypes>

</div.client>

Aizpildot tikai minimālo konfigurācija datu komplektu pārējās vērtības tiek uzstādītas no noklusētajām bibliotēkas vērtībām.

Konfigurācijas faila izmantošana ļauj mainīt vērtības bez DVS sistēmas koda izmaiņām, bibliotēkas klienta objekta izveidošanai nebūs vajadzīga obligāto parametru norādīšana:

IntegrationClient client = **new** IntegrationClient();

//...darbs ar bibliotēku

Bibliotēka atbalsta arī programmējamu konfigurācijas uzstādīšanu. Koda piemērs, kur tiek uzstādīta konfigurācija inicializējot bibliotēku Java kodā:

ClientConfiguration clientConfig = **new** ClientConfiguration();

// Konfigurācijas uzstādīšana

clientConfig.setServiceAddress("https://div.vraa.gov.lv/vraa.div.webservice.unifiedinterface/unifiedservice.svc");

clientConfig.setTimeout (100);

// Konfigurācijas nodošana bibliotēkai.

IntegrationClient client = **new** IntegrationClient(clientConfig);

//...darbs ar bibliotēku

### Java bibliotēkas uzstādīšanas informācija

Lai korekti varētu lietot DIV JAVA bibliotēkas API (VraaDivClient.jar), nepieciešams „JAVA CLASSPATH” norādīt sekojošas trešo pušu bibliotēkas:

* serializer-2.7.1.jar
* webservices-api.jar - jābūt definētai kā "endorsed" bibliotēkai (skat. http://docs.oracle.com/javase/1.5.0/docs/guide/standards/index.html)
* webservices-extra.jar
* webservices-extra-api.jar
* webservices-rt.jar
* webservices-tools.jar
* xalan-2.7.1.jar

Lai nodrošinātu DIV nepieciešamo kodēšanas algoritmus nepieciešams uzstādīt policy datnes, kas atļauj stiprākos kodēšanas algoritmus. Datnes jāuzstāda „$JRE/lib/security” mapē.

Java SE Runtime Environment 6 – jce\_policy-6.zip (http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html);

Java SE Runtime Environment 7 – UnlimitedJCEPolicyJDK7.zip (http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html).

## Bibliotēkas konfigurācijas parametru apraksts

Konfigurācijas parametri, kuri ir pieejami uzstādīšanai failā un programmatūras līmenī, būs ekvivalenti, un ir aprakstīti tabulā „Tabula 2”.

Tabula : Bibliotēkas konfigurācija

|  |  |  |
| --- | --- | --- |
| **Nosaukums** | **Tips** | **Apraksts** |
| ServiceAddress | string | DIV servera adrese. Pēc noklusējuma tiek uzstādīta DIV produkcijas vides adrese. |
| Timeout | int | Servisa atbildes noildzes laiks sekundēs.  |
| Certificates | ClientCertificateReferenceCollection | Klienta sertifikātu saraksts. Ciparsertifikāts tiks izmantots autentifikācijai, ziņojuma parakstīšanai, ienākošo ziņojumu atšifrēšanai. Izmantojamais sertifikāts atradīsies ciparsertifikātu glabātuvē. Ievērojot ciparsertifikāta potenciālo derīguma termiņa beigas, ir iespējams noradīt vairākus sertifikātus. |
| FileTypes | BaseCollection <FileTypeConfiguration> | Datņu tipu informācijas saraksts, kurā var tikt definēti datņu tipi ar sekojošu informāciju:* Compress - Neobligāts
* Extension - Obligāts
* MimeType - Obligāts

Datņu tipi nav obligāti jādefinē. Nedefinējot datņu tipu MimeType datnei tiks uzstādīts „application/octet-stream”, bet Compress tiek uzstādīta CompressByDefault vērtība. |
| CompressByDefault | bool | Vai pēc noklusējuma faili tiks saspiesti. Neaizpildot vērtību tiks ņemta Bibliotēkas noklusētā vērtība - True |
| EncryptByDefault | bool | Vai izejošie ziņojumi tiks šifrēti. Parametra vērtība tiek izmantota, ja ziņojuma nosūtīšanas brīdi šifrēšanas pielietojamība netiek norādīta. Neaizpildot vērtību tiks ņemta Bibliotēkas noklusētā vērtība - True |

# Izmantošanas scenāriji

## Bibliotēkas pieslēgšana izstrādes videi

Bibliotēka DVS izstrādātājam tiek piegādāta pakotnes veidā – Vraa.Div.Client.dll (.NET) un vraa.div.client.jar (Java). Pēc pieslēgšanas izstrādes projektam, bibliotēkas funkcionalitāte ir pieejama klientu DVS sistēmas izstrādātājiem.

## Bibliotēkas funkcionalitātes izmantošana

Bibliotēkas funkcionalitāte ir pieejama izmantojot IntegrationClient klases metodes. Pēc nepieciešamās funkcionalitātes izpildes būs nepieciešams norādīt izpildes scenārija beigas, saucot Close vai Dispose (tikai .NET) metodes. Autentifikācija un konteksta uzturēšana tiks realizētas bibliotēkas iekšējā daļā. Izmantošanas piemērs .NET un Java bibliotēkām:

using (IntegrationClient client = new IntegrationClient())

{

 //... darbs ar bibliotēku

}

Alternatīva pieeja izmantošanas apgabala noteikšanai un izpildes scenārija pabeigšanai:

IntegrationClient client = new IntegrationClient();

try

{

 //... darbs ar bibliotēku

}

finally

{

 client.Close();

}

## Ziņojumu apmaiņas scenāriji

Ziņojuma pilnais apmaiņas scenārijs ir attēlots 1. attēlā.



1. attēls Ziņojuma nosūtīšana

1. Ziņojuma sagatavošana nosūtīšanai;
2. Nosūtītāja sistēma nosūta izveidotu ziņojumu DIV sistēmai;
3. Saņēmēja sistēma ir informēta, ka ir saņemts jauns ienākošais ziņojums. Šim nolūkam tiek atbalstītas divas iespējas:
	1. Paziņojumu saņemšana no DIV par ienākošiem ziņojumiem;
	2. Regulāra ziņojumu pastkastītes pārbaude;
4. Saņēmēja sistēma izgūst pilnu ziņojumu;
5. Saņēmēja sistēma piekļūst ziņojuma informācijai;
6. Saņēmēja sistēma apstiprina ziņojuma saņemšanu;
7. Nosūtītājs regulāri izgūst paziņojumus par nosūtīto ziņojumu statusu.

### Ziņojuma nosūtīšana bez datnēm

Ziņojuma nosūtīšanai bez datnēm ir nepieciešams:

1. izveidot IntegrationMessage klases instanci;
2. uzstādīt ziņojuma adresātu (s);
3. uzstādīt nosūtītāja e-adresi;
4. uzstādīt obligātos metadatus;
5. nosūtīt ziņojumu, saucot IntegrationClient klases SendMessage metodi.

Tehniskie ziņojuma metadati pēc iespējas tiek aizpildīti automātiski. Ziņojuma parakstīšanu pirms nosūtīšanas veic bibliotēka, izmantojot konfigurācijā noradīto sertifikātu.

#### .NET bibliotēkas koda piemērs

 static void SendMin()

 {

 IntegrationMessage message = new IntegrationMessage();

 //uzstāda ziņojuma nosūtītāju

 message.From = "mail@domain1.lv";

 //uzstāda adresātu

 message.To = "mail@domain2.lv";

 //uzstāda metadatus

 message.Document.Authors.AddInstitution("My institution name");

 message.Document.Kind.Code = "DOC\_EMPTY";

 message.Document.Title = "Document simple title";

 // Tiek izgūts Aploksnes XML

 // Aploksnes XML izgūšana nav obligāta

 string xml = message.Xml;

 using (IntegrationClient client = new IntegrationClient())

 {

 string id = client.SendMessage(message);

 }

 }

#### Java bibliotēkas koda piemērs

**public** **void** sendMin() {

 IntegrationMessage message = **new** IntegrationMessage();

 //uzstāda ziņojuma nosūtītāju

 message.setFrom("sets@rixtech.lv");

 //uzstāda adresātu

 message.setTo("sets@rixtech.lv");

 //uzstāda metadatus

 message.getDocument().getAuthors().addInstitution("My Institution");

 message.getDocument().getKind().setCode("DOC\_EMPTY");

 message.getDocument().setTitle("My document title");

 // Tiek izgūts Aploksnes XML

 // Aploksnes XML izgūšana nav obligāta

 String xml = message.getXmlString();

 IntegrationClient client = **new** IntegrationClient();

 String id = client.sendMessage(message);

}

### Ziņojuma nosūtīšana ar datnēm

Ziņojuma nosūtīšanai ar datnēm ir nepieciešams:

1. izveidot IntegrationMessage klases instanci;
2. uzstādīt ziņojuma adresātu(s);
3. uzstādīt nosūtītāja e-adresi;
4. uzstādīt obligātos metadatus;
5. pievienot datnes;
6. nosūtīt ziņojumu, saucot IntegrationClient klases SendMessage metodi.

Tehniskie ziņojuma metadati pēc iespējas tiek aizpildīti automātiski. Ziņojuma parakstīšanu pirms nosūtīšanas veic bibliotēka, izmantojot konfigurācijā norādīto sertifikātu.

#### .NET bibliotēkas koda piemērs

 static void SendMinWithFiles()

 {

 IntegrationMessage message = new IntegrationMessage();

 //uzstāda ziņojuma nosūtītāju

 message.From = "mail@domain1.lv";

 //uzstāda adresātus

 message.To = "mail@domain2.lv;mail@domain3.lv";

 //uzstāda metadatus

 message.Document.Authors.AddInstitution("My institution name");

 message.Document.Kind.Code = "DOC\_EMPTY";

 message.Document.Title = "Document simple title";

 //faila pievienošana no atmiņas

 byte[] file0 = new byte[] { 112 };

 message.Document.Files.Add(file0, "first file");

 //faila pievienošana no straumes

 using (FileStream stream = File.Open("delivery.zip", FileMode.Open))

 {

 MessageFile file = new MessageFile();

 file.SetContent(stream);

 file.Name = "delivery.zip";

 file.AppendixNumber = 1;

 message.Document.Files.Add(file);

 using (IntegrationClient client = new IntegrationClient())

 {

 string id = client.SendMessage(message);

 }

 }

 }

#### Java bibliotēkas koda piemērs

 **public** **void** SendMinWithFiles() **throws** FileNotFoundException,

 KeyStoreException, InterruptedException {

 IntegrationMessage message = **new** IntegrationMessage();

 // uzstāda ziņojuma nosūtītāju

 message.setFrom("mail@domain1.lv");

 // uzstāda adresātus

 message.setTo("mail@domain2.lv;mail@domain3.lv");

 // uzstāda metadatus

 message.getDocument().getAuthors().addInstitution("My Institution");

 message.getDocument().getKind().setCode("DOC\_EMPTY");

 message.getDocument().setTitle("My document title");

 // faila pievienošana no atmiņas

 **byte**[] file0 = **new** **byte**[] { 112 };

 message.getDocument().getFiles().add(file0, "first file");

 // faila pievienošana no straumes

 FileInputStream stream = **new** FileInputStream(

 "delivery.zip");

 MessageFile file1 = **new** MessageFile();

 file1.setContent(stream);

 file1.setName("delivery.zip");

 message.getDocument().getFiles().add(file1);

 IntegrationClient client = **new** IntegrationClient();

 String serverId = client.sendMessage(message);

 }

### Atbildes ziņojuma sūtīšana par saņemtu ziņojumu

Ja DVS sistēma atbild uz kādu saņemtu ziņojumu, tad ir nepieciešams aizpildīt trasējamības lauku struktūru (TraceInfo) ar Ziņojuma servera identifikatoru (MessageId).

Šo lauku mērķis ir nodrošināt ziņojumu un atbilžu izsekošanu. Lai to varētu nodrošināt, DVS, saņemot ziņojumu, šie lauki jāsaglabā, un nosūtot atbildes ziņojumu, attiecīgi jāieraksta trasējamības laukos tā ziņojuma attiecīgais identifikācijas numurs, uz kuru tiek sniegta atbilde.

### Ziņojuma sūtīšanas tehniskie parametri

Ziņojuma sūtīšanas tehniskie parametri ir definēti kā īpašības IntegrationMessage klasē. Parametri ir aprakstītai 3. Tabulā.

. tabula Ziņojuma sūtīšanas tehniskie parametri

|  |  |
| --- | --- |
| **Īpašība** | **Skaidrojums** |
| EnableEncryption | Ziņojuma piesaistīto datņu šifrēšanas pielietošanas noradīšana* True – šifrēt piesaistīto datņu saturu;
* False – pārsūtīt datnes nešifrēta veidā;
* Null – Tiek izmantota bibliotēkas konfigurācijas vērtība.
 |
| NotificationRequired | Vai sūtītājam būs iespējams saņemt paziņojumus par ziņojuma piegādes statusu. Tādu paziņojumu saņemšana aprakstīta nodaļā [4.3.8].* true – sistēma ģenerēs paziņojumus par ziņojuma saņemšanu/noraidīšanu.
* false – sistēma neveidos paziņojumus.

Paziņojumi par nokavētu piegādi tiek veidoti vienmēr. |
| ClientId | Klienta identifikators, pēc kā tiek identificēts no DVS izejošais sūtījums. Ja lauks netiek aizpildīts, tad Bibliotēka uzģenerē identifikatoru vērtību pati. |
| DeliveryBy | Sūtītāja uzstādīts ziņojuma piegādes laika ierobežojums. Pēc šī laika, ja ziņojums netiks apstiprināts, tika uzsākts nokavētas piegādes process un ziņojums izgūšanai vairs nebūs pieejams. Ja lauks nav aizpildīts, tad Serveris uzstāda vērtību no konfigurācijas. |

Ziņojumam piesaistīto datņu šifrēšanu nosaka bibliotēkas konfigurācija, vai arī individuālā ziņojuma sūtīšanas īpašība gadījumā, ja šifrēšana ir iespējota, bibliotēka automātiski vērsīsies pie Adresātu kataloga funkcionalitātes publiskās adresāta atslēgas izgūšanai, un veiks satura šifrēšanu.

Papildus ziņojuma sūtīšanas iestatījumi iekļaus pazīmi, vai sūtītājs vēlas saņemt paziņojumu par ziņojuma veiksmīgu pārsūtīšanu adresātam vai pārsūtīšanas kļūdu.

#### .NET bibliotēkas koda piemērs

 // Uzstāda šifrēšanu

 message.EnableEncryption = true;

 // Uzstāda paziņojumu saņemšanu

 message.EnableNotifications = true;

 // Uzstāda klienta identifikatoru

 message.ClientId = "AB123C";

 // Uzstāda ziņojuma piegādes ierobežojumu

 message.DeliveryBy = DateTime.Now.AddDays(1);

#### Java bibliotēkas koda piemērs

 // Uzstāda šifrēšanu

 message.setEnableEncryption(**true**);

 // Uzstāda paziņojumu saņemšanu

 message.setEnableNotifications(**true**);

 // Uzstāda klienta identifikatoru

 message.setClientId("ABC123");

 // Uzstāda ziņojuma piegādes ierobežojumu

 Calendar c1 = Calendar.*getInstance*();

 c1.add(Calendar.*DATE*, 1);

 message.setDeliveryBy(c1.getTime());

### Ziņojumu specifisko metadatu uzstādīšana, izmantojot XML failu

Bibliotēka nodrošina specifisko ziņojuma metadatu uzstādīšanu. Ziņojuma metadati un to struktūra ir aprakstīta Sākotnējo standartu dokumentā [3]. Ziņojuma specifiskie metadati tiek uzstādīti DocumentExtendedMetadata klasē.

Katram sistēmā definētajam dokumentu veidam, atbilstoši definētajam standartam ir piesaistīta shēma – uzstādītajiem metadatiem jāatbilst definētajai shēmai.

#### .NET bibliotēkas koda piemērs

 static void SendMinExtended()

 {

 IntegrationMessage message = new IntegrationMessage();

 //uzstāda ziņojuma nosūtītāju

 message.From = "mail@domain1.lv";

 //uzstāda adresātus

 message.To = "mail@domain2.lv; mail2@domain3.lv";

 //Uzstāda obligātos metadatus

 message.Document.Authors.AddInstitution("My institution name");

 message.Document.Title = "Document simple title";

 // Uzstāda dokumenta veidu, kam piesaistīti paplašinātie meta dati.

 message.Document.Kind.Code = "DOC\_APPLICATION";

 // Uzstāda dokumenta veida versiju. Ja lauks netiek aizpildīts,

 //pēc noklusējuma tiek uzstādīta versija 1.0.

 message.Document.Kind.Version = "1.0";

 // Tiek ielasīts XML fails, kurā ir aizpildīti dati,

 // atbilstoši dokumenta veida shēmai.

 string docAppl =

 System.IO.File.ReadAllText("extended.xml");

 // No faila ielasītie metadati tiek sagatavoti sūtīšanai.

 message.Document.ExtendedMetadata.Xml = docAppl;

 using (IntegrationClient client = new IntegrationClient())

 {

 string id = client.SendMessage(message);

 }

 }

#### Java bibliotēkas koda piemērs

**public** **void** SendMinExtended() **throws** KeyStoreException, InterruptedException, IOException {

 IntegrationMessage message = **new** IntegrationMessage();

 //uzstāda ziņojuma nosūtītāju

 message.setFrom("mail@domain1.lv");

 //uzstāda adresātus

 message.setTo("mail@domain2.lv; mail@domain3.lv");

 //Uzstāda obligātos metadatus

 message.getDocument().getAuthors().addInstitution("My Institution");

 message.getDocument().setTitle("Document simple titl");

 // Uzstāda dokumenta veidu, kam piesaistīti paplašinātie meta dati.

 message.getDocument().getKind().setCode("DOC\_APPLICATION");

 // Uzstāda dokumenta veida versiju. Ja lauks netiek aizpildīts,

 //pēc noklusējuma tiek uzstādīta versija 1.0.

 message.getDocument().getKind().setVersion("1.0");

 // Tiek ielasīts XML fails, kurā ir aizpildīti dati,

 // atbilstoši dokumenta veida shēmai.

 StreamToString sToS = **new** StreamToString();

 InputStream in = getClass().getResourceAsStream(

 "recources/extended.xml");

 String xml = sToS.convertStreamToString(in);

 // No faila ielasītie metadati tiek sagatavoti sūtīšanai.

 message.getDocument().getExtendedMetadata().setXml(xml);

 IntegrationClient client = **new** IntegrationClient();

 String serverId = client.sendMessage(message);

}

#### Iesnieguma paplašināto metadatu XML datnes piemērs

<q1:Application DocumentKind="IESNIEGUMS" p2:schemaLocation="http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0/ExtendedMetadata-Application.xsd" xmlns:q1="http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0" xmlns:p2="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/v1-0">

 <q1:AdditionalContactInfo>Papildus kontaktinformācija</q1:AdditionalContactInfo>

 <q1:ApplicationKind>

 <DocumentKindCode>CITS</DocumentKindCode>

 <DocumentKindVersion>001</DocumentKindVersion>

 <DocumentKindName>Iesniegums</DocumentKindName>

 </q1:ApplicationKind>

 <q1:HasElectronicSignature>false</q1:HasElectronicSignature>

 <q1:AnswerOptional>false</q1:AnswerOptional>

</q1:Application>

### Ziņojumu specifisko metadatu uzstādīšana, izmantojot datu klases

Ziņojuma specifiskos meta datus ir iespējams uzstādīta izmantojot XML datnes, kā tas aprakstīts nodaļā [4.3.5]. Lai atvieglotu sistēmu integrāciju, tiek piedāvāta ziņojuma specifisko meta datu pievienošana, izmantojot datu klases.

#### .NET bibliotēkas koda piemērs

static void SendMinExtendedObject()

{

 IntegrationMessage message = new IntegrationMessage();

 //uzstāda ziņojuma nosūtītāju

 message.From = "mail@domain1.lv";

 //uzstāda adresātus

 message.To = "mail@domain2.lv; mail2@domain3.lv";

 //Uzstāda obligātos metadatus

 message.Document.Authors.AddInstitution("My institution name");

 message.Document.Title = "Document simple title";

 // Uzstāda dokumenta veidu, kam piesaistīti paplašinātie meta dati.

 message.Document.Kind.Code = "DOC\_APPLICATION";

 // Uzstāda dokumenta veida versiju. Ja lauks netiek aizpildīts,

 //pēc noklusējuma tiek uzstādīta versija 1.0.

 message.Document.Kind.Version = "1.0";

 // Datu klases tiek aizpildītas ar datiem

 ExtendedApplication ap = new ExtendedApplication()

 {

 AdditionalContactInfo = "Papildus kontaktinformācija",

 AnswerOptional = false,

 HasElectronicSignature = false,

 ApplicationKind = new ExtendedApplicationKind()

 {

 DocumentKindCode = "CITS",

 DocumentKindName = "Iesniegums",

 DocumentKindVersion = "001"

 }

 };

 message.Document.ExtendedMetadata.SetObject<ExtendedApplication>(ap);

 using (IntegrationClient client = new IntegrationClient())

 {

 string id = client.SendMessage(message);

 }

}

#### .NET datu klases piemērs

Piemērā tiek demonstrēta manuāli veidota .NET datu klases, kas atbilst nodaļā [4.3.6.1] aprakstītā piemēra datu klasei. Automātiskai datu klašu veidošanai no XSD datnes priekš .NET, ieteicams izmantot XSD rīku (XML Shēmu definēšanas rīku).

[Serializable]

[XmlRootAttribute(Namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/v1-0")]

public class ExtendedApplicationKind

{

 public string DocumentKindCode { get; set; }

 public string DocumentKindVersion { get; set; }

 public string DocumentKindName { get; set; }

}

[Serializable]

[XmlRootAttribute("Application",

 Namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0",

 IsNullable = false)]

public class ExtendedApplication

{

 [XmlAttribute]

 public string DocumentKind = "IESNIEGUMS";

 [XmlAttributeAttribute("schemaLocation",

 Namespace = "http://www.w3.org/2001/XMLSchema-instance")]

 public string schemaLocation =

 "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0/ExtendedMetadata-Application.xsd";

 public string AdditionalContactInfo { get; set; }

 public ExtendedApplicationKind ApplicationKind { get; set; }

 public bool? HasElectronicSignature { get; set; }

 /\* Datu klases konstrukcijas, kas jālieto, ja kāds lauks nav obligāts.\*/

 public bool ShouldSerializeHasElectronicSignature()

 {

 return HasElectronicSignature.HasValue;

 }

 public bool AnswerOptional { get; set; }

}

#### Java bibliotēkas koda piemērs

**public** **void** SendMinExtendedObject() **throws** FileNotFoundException,

 KeyStoreException, InterruptedException {

 IntegrationMessage message = **new** IntegrationMessage();

 // uzstāda ziņojuma nosūtītāju

 message.setFrom("sets@domain1.lv");

 // uzstāda adresātus

 message.setTo("sets@domain2.lv");

 // Uzstāda obligātos metadatus

 message.getDocument().getAuthors().addInstitution("My Institution");

 message.getDocument().setTitle("Document simple titl");

 // Uzstāda dokumenta veidu, kam piesaistīti paplašinātie meta dati.

 message.getDocument().getKind().setCode("DOC\_APPLICATION");

 // Uzstāda dokumenta veida versiju. Ja lauks netiek aizpildīts,

 // pēc noklusējuma tiek uzstādīta versija 1.0.

 message.getDocument().getKind().setVersion("1.0");

 // Datu klases tiek aizpildītas ar datiem

 DocumentKindStructure ApplicationKind = **new** DocumentKindStructure();

 ApplicationKind.setDocumentKindCode("CITS");

 ApplicationKind.setDocumentKindName("Iesniegums");

 ApplicationKind.setDocumentKindVersion("001");

 ExtendedApplication ap = **new** ExtendedApplication();

 ap.setAdditionalContactInfo("Papildus kontaktinformācija");

 ap.setApplicationKind(ApplicationKind);

 ap.setAnswerOptional(**false**);

 ap.setHasElectronicSignature(**false**);

 // Datu klases tiek pievienota ziņojumam kā paplašinātie meta dati

 message.getDocument().getExtendedMetadata()

 .setObject(

 "Application", // XML saknes elementa nosaukums

 // XML vārdu telpa (namespace) specifiskajiem metadatiem

 "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0",

 // Specifisko metadatu datu klase

 ExtendedApplication.**class**,

 // Specifisko metadatu datu objekts

 ap);

 // Ziņojuma sūtīšana

 IntegrationClient client = **new** IntegrationClient();

 String id = client.sendMessage(message);

}

#### Java datu klases piemērs

Piemērā tiek demonstrēta manuāli veidota Java datu klases, kas atbilst nodaļā [4.3.6.3] aprakstītā piemēra datu klasei. Automātiskai datu klašu veidošanai no XSD datnes priekš Java, ieteicams izmantot JAXB bibliotēku.

 @XmlRootElement(namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/v1-0")

 @XmlAccessorType(XmlAccessType.*FIELD*)

 **static** **class** DocumentKindStructure {

 @XmlElement(required = **true**, namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/v1-0")

 **protected** String DocumentKindCode;

 @XmlElement(required = **true**,namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/v1-0")

 **protected** String DocumentKindVersion;

 @XmlElement(namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/v1-0")

 **protected** String DocumentKindName;

 **public** String getDocumentKindCode() {

 **return** DocumentKindCode;

 }

 **public** **void** setDocumentKindCode(String value) {

 **this**.DocumentKindCode = value;

 }

 **public** String getDocumentKindVersion() {

 **return** DocumentKindVersion;

 }

 **public** **void** setDocumentKindVersion(String value) {

 **this**.DocumentKindVersion = value;

 }

 **public** String getDocumentKindName() {

 **return** DocumentKindName;

 }

 **public** **void** setDocumentKindName(String value) {

 **this**.DocumentKindName = value;

 }

 }

 @XmlRootElement(namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0", name = "Application")

 @XmlAccessorType(XmlAccessType.*FIELD*)

 **public** **static** **class** ExtendedApplication {

 @XmlAttribute (name = "DocumentKind")

 **protected** String DocumentKind = "IESNIEGUMS";

 @XmlAttribute(name = "schemaLocation", namespace = "http://www.w3.org/2001/XMLSchema-instance")

 **protected** String schemaLocation = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0/ExtendedMetadata-Application.xsd";

 @XmlElement(namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0")

 **protected** String AdditionalContactInfo;

 @XmlElement(namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0")

 **protected** DocumentKindStructure ApplicationKind;

 @XmlElement(namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0")

 **protected** Boolean HasElectronicSignature;

 @XmlElement(namespace = "http://ivis.eps.gov.lv/XMLSchemas/100001/DIV/Application/v1-0")

 **protected** Boolean AnswerOptional;

 **public** String getAdditionalContactInfo() {

 **return** AdditionalContactInfo;

 }

 **public** **void** setAdditionalContactInfo(String additionalContactInfo) {

 AdditionalContactInfo = additionalContactInfo;

 }

 **public** DocumentKindStructure getApplicationKind() {

 **return** ApplicationKind;

 }

 **public** **void** setApplicationKind(DocumentKindStructure applicationKind) {

 ApplicationKind = applicationKind;

 }

 **public** Boolean getHasElectronicSignature() {

 **return** HasElectronicSignature;

 }

 **public** **void** setHasElectronicSignature(Boolean hasElectronicSignature) {

 HasElectronicSignature = hasElectronicSignature;

 }

 **public** Boolean getAnswerOptional() {

 **return** AnswerOptional;

 }

 **public** **void** setAnswerOptional(Boolean answerOptional) {

 AnswerOptional = answerOptional;

 }

 }

### Ziņojuma specifisko metadatu izgūšana

Bibliotēka nodrošina specifisko ziņojuma metadatu izgūšanu. Ziņojuma metadati un to struktūra ir aprakstīta Sākotnējo standartu dokumentā [3]. Ziņojuma specifiskie metadati tiek izgūti DocumentExtendedMetadata klasē.

#### .NET bibliotēkas koda piemērs

static void GetMinExtended()

{

 // Tiek inicializēta bibliotēka

 using (IntegrationClient client = new IntegrationClient())

 {

 // Tiek izgūts ziņojums

 var msg = client.GetMessage();

 // Tiek izgūts ziņojuma veida kods

 String code = msg.Document.Kind.Code;

 // Tiek izgūts ziņojuma veida versija

 String version = msg.Document.Kind.Version;

 // Tiek pārbaudīts, vai ziņojuma veids ir "Iesniegums".

 // Ja ziņojuma veids ir "Iesniegums", tad tiek izgūti ziņojumam

 // piesaistītie meta dati.

 if (code == "DOC\_APPLICATION" && version == "1.0")

 {

 // Paplašināto meta datu ielasīšana datu objektā

 ExtendedApplication ap =

 msg.Document.ExtendedMetadata.GetObject<ExtendedApplication>();

 // Paplašināto meta datu ielasīšana XML

 String xml = msg.Document.ExtendedMetadata.Xml;

 }

 // Tiek veikta ziņojuma apstiprināšana

 client.ConfirmMessage(msg);

 }

}

#### Java bibliotēkas koda piemērs

**public** **void** getMinExtended() **throws** KeyStoreException,

 InterruptedException, IOException {

 // Tiek inicializēta bibliotēka

 IntegrationClient client = **new** IntegrationClient();

 // Tiek izgūts ziņojums

 IntegrationMessage msg = client.getMessage();

 // Tiek izgūts ziņojuma veida kods

 String code = msg.getDocument().getKind().getCode();

 // Tiek izgūts ziņojuma veida versija

 String version = msg.getDocument().getKind().getVersion();

 // Tiek pārbaudīts, vai ziņojuma veids ir "Iesniegums".

 // Ja ziņojuma veids ir "Iesniegums", tad tiek izgūti ziņojumam

 // piesaistītie meta dati.

 **if** (code.equals("DOC\_APPLICATION") && version.equals("1.0")) {

 ExtendedApplication ap = **new** ExtendedApplication();

 // Paplašināto meta datu ielasīšana datu objektā

 ap = msg.getDocument().getExtendedMetadata()

 .getObject(ExtendedApplication.**class**);

 // Paplašināto meta datu ielasīšana XML

 String xml = msg.getDocument().getExtendedMetadata().getXml();

 }

 // Tiek veikta ziņojuma apstiprināšana

 client.confirmMessage(msg);

}

### Paziņojumu (notifikāciju) saņemšana

Klienta DVS sistēmai ir iespējams saņemt un apstrādāt notifikācijas par nosūtītajiem ziņojumiem. Notifikāciju saņemšanai klienta programmatūra sauc GetNotifications metodi. Notifikācijas informē par jauniem ziņojumiem pastkastītē, par veiksmīgu ziņojumu apstrādi DIV serverī, par ziņojuma nogādāšanu adresāta sistēmā, vai arī saturēs kļūdas kodu un problēmas aprakstu.

#### .NET bibliotēkas koda piemērs

 static void GetSenderNotifications()

 {

 using (IntegrationClient client = new IntegrationClient())

 {

 // tiek izgūts notifikāciju saraksts

 var notifications = client.GetNotifications();

 // tiek izveidots aptiprināmo notifikāciju objekts

 List<Notification> toConfirm = new List<Notification>();

 // tiek apstrādātas DIV servera sūtītās notifikācijas

 var msgProccessedNotifications =

 notifications.Where(n => n.NotificationType ==

 NotificationType.MessageProcessed);

 foreach (var notifyData in msgProccessedNotifications)

 {

 // Tiek apstrādātas norifikācijas, kas informē par ziņojumiem,

 // kurus serveris ir apstiprinājis.

 if (notifyData.ServerStatus == MessageStatus.Accepted)

 {

 // Tiek izgūts Ziņojuma DIV apliecinājums

 var serverConfirm =

 client.GetServerConfirmation(notifyData.MessageId);

 // Servera apliecinājums tiek sagatavots XML veidā

 string xml = serverConfirm.Xml;

 // Notifikācija tiek pievienota pie apstiprināmo

 // notifikāciju saraksta

 toConfirm.Add(notifyData);

 }

 }

 if (toConfirm.Count > 0)

 {

 // Tiek veikta notifikāciju apstiprināšana

 client.ConfirmNotifications(toConfirm);

 }

 }

 }

#### Java bibliotēkas koda piemērs

IntegrationClient client = **new** IntegrationClient();

**try** {

 // tiek izgūts notifikāciju saraksts

 List<Notification> notifications = client.getNotifications();

 // tiek izveidots aptiprināmo notifikāciju objekts

 List<Notification> toConfirm = **new** ArrayList<Notification>();

 **for** (Notification notifyData : notifications) {

 // tiek apstrādātas DIV servera sūtītās notifikācijas

 **if** (notifyData.getNotificationType() != NotificationType.*MessageProcessed*)

 **continue**;

 // Tiek apstrādātas norifikācijas, kas informē par ziņojumiem,

 // kurus serveris ir apstiprinājis.

 **if** (notifyData.getServerStatus() == MessageStatus.*Accepted*) {

 // var aplūkot DIV apliecinājumu

 MessageServerConfirmation serverConfirm = client

 .getServerConfirmation(notifyData.getMessageId());

 // Servera apliecinājums tiek sagatavots XML veidā

 Element xml = serverConfirm.getXml();

 // Notifikācija tiek pievienota pie apstiprināmo

 // notifikāciju saraksta

 toConfirm.add(notifyData);

 }

 }

 // Tiek veikta notifikāciju apstiprināšana

 **if** (!toConfirm.isEmpty())

 client.confirmNotifications(toConfirm);

} **finally** {

 client.close();

}

### Paziņojumu (notifikāciju) apstiprināšana

Lai piekļūtu nākamajām notifikācijām, ir jāveic notifikāciju apstiprināšana. Nodaļā [4.3.8] apskatītā piemēra beigās tiek parādīts metodes ConfirmNotifications izsaukums.

### Paziņojumu (notifikāciju) saņemšana, izmantojot tīmekļa pakalpi

Klienta sistēmas opcionāli varēs saņemt un apstrādāt DIV paziņojumus sinhroni, izmantojot tīmekļa pakalpi. Lai saņemtu paziņojumus šādā veidā, DVS izstrādātājiem būs nepieciešamas izstrādāt un izvērst tīmekļa pakalpi (web service) atbilstoši WSDL definīcijai, kura tiks noteikta DIV sistēmas izstrādes laikā. Izvērstās tīmekļa pakalpes adresei jābūt pierēģestrētai DIV sistēmas klienta pieslēguma konfigurācijā (DIV administēšanas vietnē). DIV sauks noteiktu tīmekļa pakalpes metodi, sniedzot informāciju par ziņojumu metodes parametru datu struktūrā.

Paziņojumi, kas saņemti izmantojot tīmekļa pakalpi, arī ir jāapstiprina atbilstoši nodaļai [4.3.9].

### Ziņojumu saņemšana bez datnēm, izgūstot ziņojuma galveni

Bibliotēkas funkcionalitāte iekļauj ziņojumu galvenes saraksta izgūšanu no klienta DVS pastkastītes. Ziņojuma galvenes informācija iekļauj ziņojuma nosūtītāju, identifikatorus un citu pamata informāciju, bet nesatur ziņojuma rumpi un piesaistītās datnes. Ziņojuma galvenes saraksta atlasīšanai tiek izmantota GetMessageHeaders metode, kurai ir vairākas saukšanas iespējas dažādu scenāriju atbalstam:

* ziņojumu informācijas izgūšana;
* ziņojumu atlase pēc adresācijas vienības;
* maksimālā atlasīto ziņojuma informācijas vienumu skaita norādīšana (saraksta izgūšana pa daļām).

Ziņojuma galvenes informācija, kura tiek reprezentēta ar IntegrationMessageHeader klasi, var tikt izmantota visa ziņojuma satura izgūšanai, nododot to GetMessage metodei. Sekojošais piemērs izgūst jaunu ziņojumu sarakstu, pēc tam lejupielādējot pirmo ziņojumu:

#### .NET bibliotēkas koda piemērs

 static void GetMin()

 {

 using (IntegrationClient client = new IntegrationClient())

 {

 // Tiek izgūts ziņojumu saraksts

 var msgList = client.GetMessageHeaders();

 if (msgList.Count > 0)

 {

 // Tiek izgūts ziņojums

 var msg = client.GetMessage(msgList[0].MessageId);

 string from = msg.From;

 string id = msg.MessageId;

 // Tiek veikta ziņojuma apstiprināšana

 client.ConfirmMessage(msg);

 }

 }

 }

#### Java bibliotēkas koda piemērs

**public** **void** getMin() **throws** KeyStoreException, InterruptedException,

IOException{

 IntegrationClient client = **new** IntegrationClient();

 // Tiek izgūts ziņojumu saraksts

 List<IntegrationMessageHeader> head =

 client.getMessageHeaders();

 **if** (head.size()>0){

 // Tiek izgūts ziņojums

 IntegrationMessage msg =

 client.getMessage(head.get(0).getMessageId());

 String from = msg.getFrom();

 String to = msg.getTo();

 // Tiek veikta ziņojuma apstiprināšana

 client.confirmMessage(msg);

 }

}

### Ziņojumu saņemšana bez datnēm, izgūstot pēc kārtas

 Jaunus ziņojumus ir iespējams saņemt arī pa vienam, bez saraksta izgūšanas, saucot GetMessage metodi bez parametriem. Tādējādi tiek izgūts vecākais neapstiprinātais ziņojums.

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 IntegrationMessage message;

 while ((message = client.GetMessage()) != null)

 {

 //sauc dvs funkcionalitāti ziņojuma apstrādei

 ProcessMessage(message);

 // Tiek veikta ziņojuma apstiprināšana

 client.ConfirmMessage(message);

 }

 }

#### Java bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 IntegrationMessage msg = **new** IntegrationMessage();

 **while** ((msg = client.getMessage()) != **null**){

 // sauc dvs funkcionalitāti ziņojuma apstrādei

 ProcessMessage(msg);

 // Tiek veikta ziņojuma apstiprināšana

 client.confirmMessage(msg);

 }

### Ziņojumu saņemšana ar datnēm

Ziņojumu saņemšanu ar datnēm veic līdzīgi kā aprakstīts nodaļā „4.3.11 Ziņojumu saņemšana bez datnēm”. Galvenā atšķirība ir, ka pēc ziņojuma izgūšanas ir jāizgūst katra datne.

#### .NET bibliotēkas koda piemērs

static void GetWithFiles()

{

 using (IntegrationClient client = new IntegrationClient())

 {

 var msg = client.GetMessage();

 var path = "C:\\temp\\data\\get\\";

 if (msg != null)

 {

 if (msg.Document.Files.Count > 0)

 {

 using (var fileContext = msg.CreateFileContext())

 {

 foreach (var file in msg.Document.Files)

 {

 //tikai piemēra nolūkos demonstrē darbu

 //gan ar atmiņas buferi, gan straumi

 if (file.Size > 2097152)

 {

 using (var fileOut =

 File.Open(path + "inc\_" + file.Name, FileMode.Create))

 {

 // Piemērs scenārijam ar straumi

 client.SaveMessageFileContent(file,

 fileOut, fileContext);

 }

 }

 else

 {

 // Piemērs scenārijam ar atmiņas buferi

 byte[] content =

 client.GetMessageFileContent(file, fileContext);

 File.WriteAllBytes(path + "inc\_" + file.Name, content);

 }

 }

 }

 }

 // Ziņojuma apstiprināšana

 client.ConfirmMessage(msg);

 }

 }

}

#### Java bibliotēkas koda piemērs

**public** **void** GetWithFiles() **throws** KeyStoreException, InterruptedException,

 IOException {

 IntegrationClient client = **new** IntegrationClient();

 **try** {

 IntegrationMessage message = client.getMessage();

 String path = "C:\\temp\\data\\get\\";

 **if** (message != **null**) {

 **if** (message.getDocument().getFiles().size() > 0) {

 FileReceiveContext fileContext = message

 .createFileContext();

 MessageFileCollection myFiles = message.getDocument()

 .getFiles();

 **for** (MessageFile file : myFiles) {

 FileOutputStream fos = **new** FileOutputStream(path

 + "inc\_" + file.getName());

 // tikai piemēra nolūkos demonstrē darbu

 // gan ar atmiņas buferi, gan straumi

 **if** (file.getSize() > 2097152) {

 // Piemērs scenārijam ar straumi

 client.saveMessageFileContent(file, fos,

 fileContext);

 } **else** {

 // Piemērs scenārijam ar atmiņas buferi

 **byte**[] content = client.getMessageFileContent(file,

 fileContext);

 fos.write(content);

 }

 }

 }

 }

 // Ziņojuma apstiprināšana

 client.confirmMessage(message);

 } **finally** {

 client.close();

 }

### Ziņojumu sinhrona apstiprināšana

 Pēc ziņojuma lejupielādes klienta DVS sistēma sūta apstiprinājumu DIV, ka ziņojums ir saņemts, izmantojot ConfirmMessage metodi, vai arī noraida ziņojuma saņemšanu ar accept = false parametra palīdzību. Papildus ir iespēja uzstādīt noraidīšanas iemeslu un iemesla kodu (text un code).

#### .NET bibliotēkas koda piemērs

 static void GetMinConfirmAll()

 {

 using (IntegrationClient client = new IntegrationClient())

 {

 IntegrationMessage msg;

 //tiek izgūts ziņojums pa vienam

 while ((msg = client.GetMessage()) != null)

 {

 //dvs veic specifiskās pārbaudes

 bool isValid = ValidateMetadata(message);

 if (isValid)

 {

 // Ziņojumu akceptē

 client.ConfirmMessage(msg);

 }

 else

 {

 // Ziņojumu noraida

 client.ConfirmMessage(msg, false, "ERR1", "Error");

 }

 }

 }

 }

#### Java bibliotēkas koda piemērs

**public** **void** GetMinConfirmAll() **throws** FileNotFoundException,

 KeyStoreException, InterruptedException {

 IntegrationClient client = **new** IntegrationClient();

 **try** {

 IntegrationMessage msg = **new** IntegrationMessage();

 // tiek izgūts ziņojums pa vienam

 **while** ((msg = client.getMessage()) != **null**) {

 // dvs veic specifiskās pārbaudes

 **boolean** isValid = ValidateMetadata(msg);

 // Tiek veikta ziņojuma apstiprināšana

 **if** (isValid) {

 // Ziņojumu akceptē

 client.confirmMessage(msg);

 } **else** {

 // Ziņojumu noraida

 client.confirmMessage(msg, **false**, "ERR1", "Error");

 }

 }

 } **finally** {

 client.close();

 }

}

### Ziņojumu asinhrona apstiprināšana

Par ziņojumu asinhronu apstiprināšanu tiek uzskatīts process, kur ziņojumu izgūšana un apstiprināšana notiek dažādos apstrādes procesos.

Veicot ziņojumu, apstiprināšanu asinhroni ir svarīgi ievērot, ka jābūt aizpildītai ConfirmationInfo struktūrai, ko sinhronas apstiprināšanas gadījumā bibliotēka aizpilda pēc metodes GetMessage izsaukšanas.

#### .NET bibliotēkas koda piemērs

static void AsyncConfirm()

{

 using (IntegrationClient client = new IntegrationClient())

 {

 String msgXml;

 MessageRecipientConfirmationInfo confirmInfo;

 // Tiek izgūts ziņojums.

 IntegrationMessage msg = client.GetMessage();

 if (msg != null)

 {

 // Ziņojums tiek saglabāts XML veidā.

 msgXml = msg.Xml;

 // Tiek saglabāta ziņojuma apstiprināšanas informācija.

 confirmInfo = msg.ConfirmationInfo;

 // Tiek veidots jauns ziņojuma objekts,

 // tādējādi demonstrējot darbības, kas veicamas asinhronas apstiprināšanas

 // gadījumā. Ziņojuma objekts tiek veidots no saglabātā Ziņojuma XML,

 // tādējādi aizpildot biznesa datus.

 IntegrationMessage message = new IntegrationMessage(msgXml);

 // Ziņijuma objekts tiek papildināts ar apstiprinājuma informāciju.

 message.ConfirmationInfo = confirmInfo;

 client.ConfirmMessage(message);

 }

 }

}

#### Java bibliotēkas koda piemērs

**static** **void** AsyncConfirm() {

 IntegrationClient client = **new** IntegrationClient();

 String msgXml;

 MessageRecipientConfirmationInfo confirmInfo;

 // Tiek izgūts ziņojums.

 IntegrationMessage msg = client.getMessage();

 **if** (msg != **null**) {

 // Ziņojums tiek saglabāts XML veidā.

 msgXml = msg.getXmlString();

 // Tiek saglabāta ziņojuma apstiprināšanas informācija.

 confirmInfo = msg.getConfirmationInfo();

 // Tiek veidots jauns ziņojuma objekts,

 // tādējādi demonstrējot darbības, kas veicamas asinhronas

 // apstiprināšanas

 // gadījumā. Ziņojuma objekts tiek veidots no saglabātā Ziņojuma

 // XML,

 // tādējādi aizpildot biznesa datus.

 IntegrationMessage message = **new** IntegrationMessage(msgXml);

 // Ziņijuma objekts tiek papildināts ar apstiprinājuma informāciju.

 message.setConfirmationInfo(confirmInfo);

 client.confirmMessage(message);

 }

}

### Ziņojuma servera apliecinājuma izgūšana

Ziņojuma servera apliecinājuma izgūšana ir process, kas pieejams ziņojuma nosūtītājām, lai izgūtu DIV parakstītu XML objektu ar laika zīmogu. Servera apliecinājumu var izgūt asinhronā procesā pret ziņojuma nosūtīšanas procesu, jeb brīdī, kad DIV ir veicis ziņojuma apstrādi. Koda piemērā ziņojuma sūtīšanas process un apliecinājuma izgūšana ērtākai attēlošanai tiek demonstrēti, kā viens process. Veicot integrāciju, vēlams procesus atdalīt, jo lielāku datu apjoma apstrādes gadījumā ziņojuma apstrāde var būt ilgāka par 2 sekundēm.

#### .NET bibliotēkas koda piemērs

static void GetServerConfirmation()

{

 IntegrationMessage message = new IntegrationMessage();

 // Ziņojuma obligāto datu aizpildīšana

 message.Document.Authors.AddInstitution("My institution name");

 message.Document.Kind.Code = "DOC\_EMPTY";

 message.Document.Title = "Document simple title";

 message.From = "mail@rixtech.lv";

 message.To = "mail@rixtech.lv";

 using (IntegrationClient client = new IntegrationClient())

 {

 // Tiek nosūtīts ziņojums

 string id = client.SendMessage(message);

 // Ziņojuma objekts tiek sagatavos XML veidā priekš saglabāšanas.

 string xml = message.Xml;

 // Tālāk tiek parādīts process, kas pēc notifikācijas saņemšanas

 // ziņojumam pievieno servera apliecinājumu.

 System.Threading.Thread.Sleep(2 \* 1000);

 // tiek izgūts notifikāciju saraksts

 var notifications = client.GetNotifications();

 // tiek apstrādātas DIV servera sūtītās notifikācijas

 var msgProccessedNotifications =

 notifications.Where(n => n.NotificationType ==

 NotificationType.MessageProcessed);

 foreach (var notifyData in msgProccessedNotifications)

 {

 // Tiek apstrādātas norifikācijas, kas informē par ziņojumiem,

 // kurus serveris ir apstiprinājis.

 if (notifyData.ServerStatus == MessageStatus.Accepted)

 {

 if (notifyData.MessageId == id)

 {

 // Tiek izgūts Ziņojuma DIV apliecinājums

 // ziņojumam, kas tika nosūtīts

 var serverConfirm =

 client.GetServerConfirmation(notifyData.MessageId);

 // Tiek izveidots ziņojuma objekts no ziņojuma XML dokumenta,

 // kas saglabāts DVS

 IntegrationMessage message2 = new IntegrationMessage(xml);

 // Ziņojuma objektam tiek pievienota servera apstiprinājuma

 // informācija.

 message2.Append(serverConfirm);

 // Ziņojuma objekts ar pievienoto servera apstiptinājumu

 // tiek sagatavots XML veidā.

 string fullXml = message2.Xml;

 // Notifikācijas apstiprināšana

 client.ConfirmNotifications(notifyData);

 }

 }

 }

 }

}

#### Java bibliotēkas koda piemērs

**public** **void** getConfirmation() **throws** InterruptedException {

 IntegrationMessage message = **new** IntegrationMessage();

 // Ziņojuma obligāto datu aizpildīšana

 message.getDocument().getAuthors().addInstitution("My Institution");

 message.getDocument().getKind().setCode("DOC\_EMPTY");

 message.getDocument().setTitle("My document title");

 message.setFrom("sets@rixtech.lv");

 message.setTo("sets@rixtech.lv");

 IntegrationClient client = **new** IntegrationClient();

 // Tiek nosūtīts ziņojums

 String id = client.sendMessage(message);

 // Ziņojuma objekts tiek sagatavos XML veidā priekš saglabāšanas.

 String xml = message.getXmlString();

 // Tālāk tiek parādīts process, kas pēc notifikācijas saņemšanas

 // ziņojumam pievieno servera apliecinājumu.

 Thread.*sleep*(5 \* 1000);

 // tiek izgūts notifikāciju saraksts

 List<Notification> notifications = client.getNotifications();

 List<Notification> toConfirm = **new** ArrayList<Notification>();

 **for** (Notification notifyData : notifications) {

 // tiek apstrādātas DIV servera sūtītās notifikācijas

 **if** (notifyData.getNotificationType() !=

 NotificationType.*MessageProcessed*)

 **continue**;

 // Tiek apstrādātas norifikācijas, kas informē par ziņojumiem,

 // kurus serveris ir apstiprinājis.

 **if** (notifyData.getServerStatus() == MessageStatus.*Accepted*) {

 **if** (notifyData.getMessageId().equals(id)) {

 // Tiek izgūts Ziņojuma DIV apliecinājums

 // ziņojumam, kas tika nosūtīts

 MessageServerConfirmation serverConfirm = client

 .getServerConfirmation(notifyData.getMessageId());

 // Tiek izveidots ziņojuma objekts no ziņojuma XML

 // dokumenta,

 // kas saglabāts DVS

 IntegrationMessage message2 =

 **new** IntegrationMessage(xml);

 // Ziņojuma objektam tiek pievienota servera

 // apstiprinājuma informācija.

 message2.append(serverConfirm);

 // Ziņojuma objekts ar pievienoto servera

 // apstiptinājumu tiek sagatavots XML veidā.

 String fullXml = message2.getXmlString();

 toConfirm.add(notifyData);

 // Notifikācijas apstiprināšana

 }

 }

 }

 **if** (!toConfirm.isEmpty())

 client.confirmNotifications(toConfirm);

}

### Ziņojuma informācijas apstrāde

Klientu DVS sistēmas piekļūst saņemtā ziņojuma informācijai, saucot IntegrationMessage klases īpašības un metodes.

## Apakšadresāciju vienību pārvaldība

Bibliotēka no DVS puses varēs veikt Adresācijas vienību administrēšanu. Alternatīva ir veikt administrēšanu, izmantojot DIV administrēšanas formas.

### Adresācijas vienību izgūšana

Adresācijas vienība tiks reprezentēta ar AddresseeUnit klasi. Klienta DVS sistēmai ir iespējams izgūt katras adresācijas vienības visus atribūtus.

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Tiek izgūta adresācijas vienība pēc tās identifikatora

 var adresseUnit = client.GetAddresseeUnit(11303);

 }

#### Java bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 // Tiek izgūta adresācijas vienība pēc tās identifikatora

 AddresseeUnit adresseUnit = client.getAddresseeUnit(11303);

### Adresācijas vienību reģistrēšana

Adresācijas vienība tiks reprezentēta ar AddresseeUnit klasi. Klienta DVS sistēmai ir iespējams reģistrēt sava pieslēguma apakšadresācijas vienības, izmantojot CreateAddresseeUnit metodi.

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 var adresse = new AddresseeUnit();

 // Adresācijas vienības E-adrese

 adresse.EAddress = "ManaE-adrese";

 // Adresācijas vienības nosaukums

 adresse.Name = "Mans adresācijas vienības nosaukums";

 // Adresāta identifikators, kuram piesaistīta adresācijas vienība.

 adresse.AddresseeId = 10000;

 // Adresācijas vienības tipa identifikators

 adresse.TypeId = 10004;

 // Laiks kad adresācijas vienība tiek aktivizēta

 adresse.ActiveFrom = DateTime.Today;

 // Adresācijas vienības izveidošana

 client.CreateAddresseeUnit(adresse);

 }

#### Java bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 AddresseeUnit adresse = **new** AddresseeUnit();

 // Adresācijas vienības E-adrese

 adresse.setEAddress("NEWADRR");

 // Adresācijas vienības nosaukums

 // Adresāta identifikators, kuram piesaistīta adresācijas vienība.

 adresse.setAddresseeId(10616);

 // Adresācijas vienības tipa identifikators

 adresse.setTypeId(10112);

 adresse.setAddresseeUnitId((**long**) 18306);

 // Laiks kad adresācijas vienība tiek aktivizēta

 Calendar c1 = Calendar.*getInstance*();

 adresse.setActiveFrom(c1.getTime());

 // Uzstāda Adresācijas vienības īpašnieka struktūru

 Owner owner = **new** Owner();

 // Uzstāda nosaukumu

 owner.setTitle("Nosaukums");

 adresse.setOwner(owner);

 // Adresācijas vienības izveidošana

 client.createAddresseeUnit(adresse);

### Adresācijas vienību atjaunināšana

Adresācijas vienības atribūtu rediģēšanai tiek izmantota UpdateAddresseeUnit metode.

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Tiek izgūta adresācijas vienības ar

 // identifikatoru 10001 aktuālā versija

 var adresseUnit = client.GetAddresseeUnit(10001);

 // Uzstādam jaunu e-adresi. EAdress laukā glabājās

 // E-adreses vērtība līdz @ simbolam.

 adresseUnit.EAddress = "JaunaE-adrese";

 // Veicam ieraksta atjaunināšanau

 client.UpdateAddresseeUnit(adresseUnit);

 }

#### Java bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 **try** {

 // Tiek izgūta adresācijas vienības ar

 // identifikatoru 10001 aktuālā versija

 AddresseeUnit adresseUnit = client.getAddresseeUnit(10001);

 // Uzstādam jaunu e-adresi. EAdress laukā glabājās

 adresseUnit.setEAddress("JaunaE-adrese");

 // Veicam ieraksta atjaunināšanau

 client.updateAddresseeUnit(adresseUnit);

 } **finally** {

 client.close();

 }

### Adresācijas vienību dzēšana

Adresācijas vienību ir iespējams atzīmēt kā nodzēstu, saucot DeleteAddresseeUnit metodi.

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Tiek izgūta adresācijas vienības ar

 // identifikatoru 11400, kuru vēlamies dzēst

 var adresseUnit = client.GetAddresseeUnit(11400);

 // Adresācijas vienības dzēšana

 client.DeleteAddresseeUnit(adresseUnit);

 }

#### Java bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 // Tiek izgūta adresācijas vienības ar

 // identifikatoru 11400, kuru vēlamies dzēst

 AddresseeUnit adresseUnit = client.getAddresseeUnit(11400);

 // Adresācijas vienības dzēšana

 client.deleteAddresseeUnit(adresseUnit);

###  Adresācijas vienību meklēšana

Klienta DVS sistēmai būs iespēja meklēt DIV adresātu katalogā, saucot SearchAddresseeUnits metodi.

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 bool hasMoreResults = false;

 var Criteria = new AddresseeUnitSearchCriteria();

 // Meklēšanas kritēriju uzstādīšana

 Criteria.EAddress = "KANCELEJA";

 // Adresācijas vienību meklēšana

 List<AddresseeUnit> unitList =

 client.SearchAddresseeUnits(Criteria, out hasMoreResults);

 }

#### Java bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 **try** {

 AddresseeUnitSearchCriteria myCriteria =

 **new** AddresseeUnitSearchCriteria();

 // Meklēšanas kritēriju uzstādīšana

 myCriteria.setEAddress("KANCELEJA");

 // Adresācijas vienību meklēšana

 ListToContinue<AddresseeUnit> unitList =

 client.searchAddresseeUnits(myCriteria);

 // Parametrs, kas norāda vai tika atgriezti visi rezultāti

 unitList.isMoreResults();

 } **finally** {

 client.close();

 }

## Adresātu sinhronizācija

Klienta DVS sistēmai būs iespēja veikt sinhronu adresāta kataloga ierakstu sinhronizāciju. Lai veiktu sinhronizāciju, ir pieejamas sekojošas bibliotēkas metodes:

* GetInitialAddresseeRecords;
* GetChangedAddresseeRecords;
* GetChangedAddresseeRecordsAsyncStart;
* GetChangedAddresseeRecordsAsyncResult;
* GetChangedAddresseeRecordsAsyncConfirm;

Sinhronizācijas rezultātā tiek atgriests saraksts ar adresāta ierakstiem –“AddresseeRecord”. Klase AddresseeRecord satur šādus atribūtus:

DateTime? ActiveFrom – ieraksta izveides/aktivizēšanas laiks. Var būt nākotnē.

Classifier AddreseeRecordType – Adresāta ierakstu tips (piemēram, Struktūrvienība, Privātpersona, kas pilda valsts funkciju).

Classifier AddreseeType – Adresāta tips (piemēram, Valsts iestāde, Reģistros reģistrēta iestāde).

DateTime? BlockedFrom – ieraksta deaktivizēšana laiks. Var būt nākotnē.

string Code – Adresātu Kods (Reģistrācijas numurs, personas kods).

string EAddress – Tehniskā Eadrese. Sastāv no eadreses un pazīmes/iestādes norādītā elementa (pazīme@eadrese).

string FirstName – Personas vārds.

string LastName – Personas uzvārds.

long Id – Ieraksta unikālais identifikators.

bool IsDeleted – Pazīme vai ieraksts ir dzēsts.

string Name – Adresāta nosaukums.

long? ParentId – Vecāka ieraksta ID. (Koka struktūras veidošanai).

long Version – Versijas identifikators.

Asinhroni izgūstāmie dati tiek sagatavoti kā JSON datne atbilstoši AddreseeRecord klasei.

### Adresātu sākotnējā sinhronizācija

Metode GetInitialAddresseeRecords nodrošina visu kataloga adresātu sinhronizāciju līdz konkrētās dienas izmaiņām. Nenorādot ieejas parametrus lapošanas mehānismu nodrošina pati bibliotēka.

#### .NET bibliotēkas koda piemērs

 long MaxVersion = 0;

 using (IntegrationClient client = new IntegrationClient())

 {

 string token = null;

 do

 {

 var list = client.GetInitialAddresseeRecords(token, out token);

 if (list != null)

 {

 // Apstrādā katru saņemto adresātu

 foreach (var record in list)

 {

 // Ierakstu versijas identifikatori tiek atgreisti secīgi.

 // Tomēr tiek rekomendēts pārbaudīt katru versijas identifikatoru.

 // Maksimālais versijas identifikators ir nepieicešams

 // izmaiņu izgūšanai.

 if (record.Version > MaxVersion)

 {

 // Tiek uzstādīts maksimālās versijas identifikators

 MaxVersion = record.Version;

 }

 // Tiek pārbaudīts vai E-adrese ir dzēsta

 // (vairāk aktuāls apakšadresācijas vienībām)

 if (record.IsDeleted != true)

 {

 // Veic atribūtu apstrādi pēc DVS biznesa procesa

 }

 else

 {

 // Dzēst e-adresi no DVS sistēmas;

 }

 }

 }

 }

 while (token != null);

 // DVS saglabā Versijas numuru pēc kura nākamajā reizē veikt sinhronizāciju.

 var MaxVersionSave = MaxVersion;

 }

#### JAVA bibliotēkas koda piemērs

 **long** MaxVersion = 0;

 IntegrationClient client = **new** IntegrationClient();

 String token = **null**;

 **do** {

 ListWithContinuationToken<AddresseeRecord> list

 = client.getInitialAddresseeRecords(token);

 **if** (list != **null**) {

 // Apstrādā katru saņemto adresātu

 **for** (AddresseeRecord record : list) {

 // Ierakstu versijas identifikatori tiek atgreisti secīgi.

 // Tomēr tiek rekomendēts pārbaudīt katru versijas

 // identifikatoru.

 // Maksimālais versijas identifikators ir nepieicešams

 // izmaiņu izgūšanai.

 **if** (record.getVersion() > MaxVersion) {

 // Tiek uzstādīts maksimālās versijas identifikators

 MaxVersion = record.getVersion();

 }

 // Tiek pārbaudīts vai E-adrese ir dzēsta

 // (vairāk aktuāls apakšadresācijas vienībām)

 **if** (record.isDeleted() != **true**) {

 // Veic atribūtu apstrādi pēc DVS biznesa procesa

 } **else** {

 // Dzēst e-adresi no DVS sistēmas;

 }

 }

 }

 } **while** (token != **null**);

 // DVS saglabā Versijas numuru pēc kura nākamajā reizē veikt

 // sinhronizāciju.

 **long** MaxVersionSave = MaxVersion;

### Adresātu aktuālo izmaiņu sinhronizācija

Datu izmaiņas iespējams izgūt sinhroni, kā arī asinhroni.

Sinhrono izgūšanu nodrošina metode GetChangedAddresseeRecords, bet asinhrono metodes GetChangedAddresseeRecordsAsyncStart, GetChangedAddresseeRecordsAsyncResult, GetChangedAddresseeRecordsAsyncConfirm. Datu kopa (apjoms) abiem izmaiņu izgūšanas veidiem ir identiska – atšķiras tikai datu formāts, kā arī metožu izmantošanas ierobežojumi.

***Svarīgi***

Sinhronā sinhronizācija tiek ierobežota ar laika periodu, cik bieži sinhronizāciju var veikt, piemēram, reizi 12 stundās, bet toties tiek nodrošināts, ka atbilde tiks sniegta maksimāli ātri (webservisa izsaukuma laikā).

Asinhrono sinhronizāciju var lietot neierobežoti, bet atbilde tiks sniegta ilgākā laika periodā, atkarībā no pieprasījumu rindas.

#### Sinhrona adresātu aktuālo izmaiņu sinhronizācija

Metode GetChangedAddresseeRecords nodrošina kataloga izmaiņu atgriešanu no konkrētas versijās līdz pēdējām izmaiņām (metode atgriež izmaiņas par konfigurācijā norādīto laika periodu, piemēram pa pēdējo nedēļu) – ja tiks pieprasītas izmaiņas par vecāku periodu tiks saņemta kļūda un būs jāveic atkārtoti pilna sinhronizācija.

##### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Pazīme vai eksistē meklēšanas rezultāti, kuri netika atgriezti.

 var hasMoreRecords = true;

 while (hasMoreRecords)

 {

 // Veicot sinhronizāciju jānorāda maksimālais versijas

 // numurs no jau nosinhronizētajiem ierakstiem.

 var data = client.GetChangedAddresseeRecords(MaxVersion, out hasMoreRecords);

 // Tiek izgūts saraksts ar adresātu identifikatoriem, kuriem ir izmaiņas

 // pret sinhronizējamo versiju.

 if (data != null)

 {

 // Apstrādā katru saņemto adresātu

 foreach (var record in data)

 {

 // Ierakstu versijas identifikatori tiek atgriezti secīgi.

 // Tomēr tiek rekomendēts pārbaudīt katru versijas identifikatoru.

 // Maksimālais versijas identifikators ir nepieicešams

 // izmaiņu izgūšanai.

 if (record.Version > MaxVersion)

 {

 // Tiek uzstādīts maksimālās versijas identifikators

 MaxVersion = record.Version;

 }

 // Tiek pārbaudīts vai E-adrese ir dzēsta

 // (vairāk aktuāls apakšadresācijas vienībām)

 if (record.IsDeleted != true)

 {

 // Veic atribūtu apstrādi pēc DVS biznesa procesa

 }

 else

 {

 // Dzēst e-adresi no DVS sistēmas;

 }

 }

 }

 }

 }

 // DVS saglabā Versijas numuru pēc kura nākamajā reizē veikt sinhronizāciju.

 var MaxVersionSave = MaxVersion;

##### JAVA bibliotēkas koda piemērs

 **long** MaxVersion = 58000;

 IntegrationClient client = **new** IntegrationClient();

 Boolean hasMoreRecords = **true**;

 **while** (hasMoreRecords) {

 // Veicot sinhronizāciju jānorāda maksimālais versijas

 // numurs no jau nosinhronizētajiem ierakstiem.

 ListToContinue<AddresseeRecord> data

 = client.getChangedAddresseeRecords(MaxVersion);

 // Tiek izgūts saraksts ar adresātu identifikatoriem, kuriem ir

 // izmaiņas

 // pret sinhronizējamo versiju.

 **if** (data != **null**) {

 // Apstrādā katru saņemto adresātu

 **for** (AddresseeRecord record : data) {

 {

 // Ierakstu versijas identifikatori tiek atgriezti

 // secīgi.

 // Tomēr tiek rekomendēts pārbaudīt katru versijas

 // identifikatoru.

 // Maksimālais versijas identifikators ir nepieicešams

 // izmaiņu izgūšanai.

 **if** (record.getVersion() > MaxVersion) {

 // Tiek uzstādīts maksimālās versijas identifikators

 MaxVersion = record.getVersion();

 }

 // Tiek pārbaudīts vai E-adrese ir dzēsta

 // (vairāk aktuāls apakšadresācijas vienībām)

 **if** (record.isDeleted() != **true**) {

 // Veic atribūtu apstrādi pēc DVS biznesa procesa

 } **else** {

 // Dzēst e-adresi no DVS sistēmas;

 }

 }

 }

 }

 hasMoreRecords=data.isMoreResults();

 }

 // DVS saglabā Versijas numuru pēc kura nākamajā reizē veikt

 // sinhronizāciju.

 Long MaxVersionSave = MaxVersion;

#### Asinhronā adresātu sinhronizācija

Klienta DVS sistēmai būs iespēja veikt asinhronu adresāta kataloga ierakstu sinhronizāciju. Lai veiktu sinhronizāciju, ir pieejamas sekojošas bibliotēkas metodes:

##### Adresātu asinhronās sinhronizācijas uzsākšana

Metode GetChangedAddresseeRecordsAsyncStart nodrošina kataloga izmaiņu izgūšanas uzsākšanu no konkrētas versijās līdz pēdējām izmaiņām. Pēc metodes izsaukšanas asinhroni tiek uzsākts izmaiņu sagatavošanas process.

###### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 long MaxVersion = x;

 // Veicot sinhronizāciju jānorāda maksimālais versijas

 // numurs no jau nosinhronizētajiem ierakstiem. Metode iniciē

 // sinhronizācijas uzsākšanu.

 string RequestId = client.GetChangedAddresseeRecordsAsyncStart(MaxVersion);

 // Metode atgriež pieprasījuma identifikatoru,

 //ko tālāk jālieto datu izgūšanai.

 return RequestId;

 }

###### JAVA bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

**long** MaxVersion = 58000;

 // Veicot sinhronizāciju jānorāda maksimālais versijas

 // numurs no jau nosinhronizētajiem ierakstiem. Metode iniciē

 // sinhronizācijas uzsākšanu.

 String RequestId = client.getChangedAddresseeRecordsAsyncStart(MaxVersion);

 // Metode atgriež pieprasījuma identifikatoru,

 //ko tālāk jālieto datu izgūšanai.

 **return** RequestId;

##### Adresātu asinhronās sinhronizācijas datu izgūšana

Metode GetChangedAddresseeRecordsAsyncResult nodrošina asinhroni sagatavoto kataloga izmaiņu izgūšanu.

###### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Pēc pieprasijuma identifikatora tiek izgūta informācija, kur

 // piekļūt sinhronizācijas datiem

 var data = client.GetChangedAddresseeRecordsAsyncResult(RequestId);

 var ftpfile = data.FtpFileName;

 // Izmantojot ftpfile jāpieslēdzas ar ftp klientu ftp serverim

 // un jāizgūst datne, kuru var apstrādāt atbilstoši nepieciešamajam.

 }

###### JAVA bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 // Pēc pieprasijuma identifikatora tiek izgūta informācija, kur

 // piekļūt sinhronizācijas datiem

 ChangedAddresseeRecordsAsyncResult data =

 client.getChangedAddresseeRecordsAsyncResult(RequestId);

 String ftpfile = data.getFtpFileName();

 // Izmantojot ftpfile jāpieslēdzas ar ftp klientu ftp serverim

 // un jāizgūst datne, kuru var apstrādāt atbilstoši nepieciešamajam.

##### Adresātu asinhronās sinhronizācijas pabeigšana

Metode GetChangedAddresseeRecordsAsyncConfirm nodrošina sinhronizācijas pabeigšanu – apstiprinājumu, ka dati ir saņemti.

###### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Tiek apstirpināta datu saņemšana.

 client.GetChangedAddresseeRecordsAsyncConfirm(RequestId);

 }

###### JAVA bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 // Tiek apstirpināta datu saņemšana.

 client.getChangedAddresseeRecordsAsyncConfirm(RequestId);

## E-adreses pārbaude

Klienta DVS sistēmai būs iespēja veikt E-adreses pārbaudi. Pārbaudi var viekt sinhroni un asinhroni. Sinhronai pārbaudei pieejama metode “ValidateEAddress”. Asinhronai pārbaudei pieejamas metodes:

* ValidateAddresseesAsyncStart
* ValidateAddresseesAsyncResult
* ValidateAddresseesAsyncConfirm

### Sinhrona E-adreses pārbaude

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 IEnumerable<string> eAddresses = new string[] {"12345678902","12345678901" };

 // Pārbauda eadreses

 var result = client.ValidateEAddress(eAddresses, AddreesseePersonType.NaturalPerson);

 foreach (var record in result)

 {

 if (record.IsValid)

 {

 // Eadrese ir Aktīva

 }

 else

 {

 // Eadrese nav derīga

 }

 }

 }

#### JAVA bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 List<String> eAddresses = **new** ArrayList<String>();

 eAddresses.add("12345678902");

 eAddresses.add("12345678901");

 // Pārbauda eadreses

 List<EAddressValidationResult> result =

 client.validateEAddress(eAddresses,

 AddreesseePersonType.***NaturalPerson***);

 **for** (EAddressValidationResult record : result)

 {

 **if** (record.getIsValid())

 {

 // Eadrese ir Aktīva

 }

 **else**

 {

 // Eadrese nav derīga

 }

 }

### Asinhrona E-adreses pārbaude

#### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 IEnumerable<string> eAddresses = new string[] { "30128211323", "12345678901" };

 // Uzsāk eadreses validāciju.

 // Sistēmas pusē jasaglabā pieprasījuma identifikators.

 var requestId =

 client.ValidateAddresseesAsyncStart(eAddresses);

 // Saņem sarakstu ar derīgām e-adresēm.

 // Datu saņemšana jāveic ar saglabāto pieprašijuma identifikatoru.

 // Dati tiks atgriezti, kad sistēma rindas kārtībā tos būs sagatavojusi

 var result = client.ValidateAddresseesAsyncResult(requestId);

 // foreach (var record in result)

 {

 // Apstrādā derīgās eadrese

 }

 // Apstiprina rezultātu saņemšanu.

 client.ValidateAddresseesAsyncConfirm(requestId);

 }

#### JAVA bibliotēkas koda piemērs

 IntegrationClient client = **new** IntegrationClient();

 List<String> eAddresses = **new** ArrayList<String>();

 eAddresses.add("12345678902");

 eAddresses.add("12345678901");

 // Uzsāk eadreses validāciju.

 // Sistēmas pusē jasaglabā pieprasījuma identifikators.

 String requestId =

 client.validateAddresseesAsyncStart(eAddresses);

 // Saņem sarakstu ar derīgām e-adresēm.

 // Datu saņemšana jāveic ar saglabāto pieprašijuma identifikatoru.

 // Dati tiks atgriezti, kad sistēma rindas kārtībā tos būs sagatavojusi

 List<String> result = client.validateAddresseesAsyncResult(requestId);

 **for** (String record : result)

 {

 // Apstrādā derīgās eadrese

 }

 // Apstiprina rezultātu saņemšanu.

 client.validateAddresseesAsyncConfirm(requestId);

# Kataloga e-adrešu aktualizēšana

Metodes pieejamas, lai veiktu Eadrešu kataloga aktualizēšanu. Metodes pieejamas ārējiem datu reģistriem. Datu aktualizēšani pieejamas šādas metodes:

* CreateInstitution – Valsts iestāžu izviedošanai.
* ActivatePersonAccount – Reģistros reģistrētu un fizisku personu izveidošanai.
* AddresseeStatusUpdateAsyncStart – Masveida statusa atjaunošanai.
* AddresseeOwnerUpdateAsyncStart – Masveida īpašieku datu atjaunošanai.
* GetResultJournal – Masveida operāciju rezultātu izgūšanai.

Visas metodes pieejamas tikai .NET bibliotēkā.

## Valsts iestādes izveidošana

### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Izveido iestādes ierakstu, kas jāreģistrē

 var OwnerRecord = new Owner();

 OwnerRecord.Code = "XXXXXXXXXXX";

 OwnerRecord.Title = "Iestāde 1";

 //OwnerRecord.FirstName

 // Izveido Valsts iestāde

 client.CreateInstitution("TPI", OwnerRecord);

 }

## Reģistros reģistrētu un fizisku personu izveidošana

### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Izveido iestādes ierakstu, kas jāreģistrē

 var OwnerRecord = new Owner();

 OwnerRecord.Code = "XXXXXXXXXX";

 OwnerRecord.FirstName = "FirstName";

 OwnerRecord.Surname = "Surname";

 // Izveido Privātpersonu

 client.ActivatePersonAccount(

 AddreesseePersonType.NaturalPerson, OwnerRecord);

 }

 using (IntegrationClient client = new IntegrationClient())

 {

 // Izveido iestādes ierakstu, kas jāreģistrē

 var OwnerRecord = new Owner();

 OwnerRecord.Code = "XXXXXXXXXXX";

 OwnerRecord.Title = "SIA QWERER";

 // Izveido Reģistros reģistrētu personu

 client.ActivatePersonAccount(

 AddreesseePersonType.RegisteredEntity, OwnerRecord);

 }

## Masveida statusa atjaunošana

### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 // Uzsāk datu masveida statusaatjaunošanas procesu.

 // Atbildē tiek saņemts pieprasījuma identifikators

 IEnumerable<String> eAddresses =

 new string[] { "40003249152", "40000000003" };

 var requestId = client.AddresseeStatusUpdateAsyncStart(

 AddresseeStatusUpdatePersonType.Institution,

 AddresseeStatusUpdateStatusType.Annulled,

 eAddresses);

 // Tiek izsaugta rezultātu izgūšanas metode.

 // Datu apstrāde ir asinhrons process un var prasīt ilgu laiku.

 var response = client.GetResultJournal(requestId);

 }

## Masveida īpašieku datu atjaunošana

### .NET bibliotēkas koda piemērs

 using (IntegrationClient client = new IntegrationClient())

 {

 List<AddresseeOwnerUpdateRecord> instOwners = new List<AddresseeOwnerUpdateRecord>();

 // Izveido pirmo atjaunojamo ierakstu

 var OwnerRecord = new AddresseeOwnerUpdateRecord();

 OwnerRecord.IdentificationNumber = "400000000001";

 OwnerRecord.Title = "AAA, SIA";

 instOwners.Add(OwnerRecord);

 // Izveido otro atjaunojamo ierakstu

 var OwnerRecord2 = new AddresseeOwnerUpdateRecord();

 OwnerRecord2.IdentificationNumber = "40000000002";

 OwnerRecord2.Title = "DEPARTMENT";

 instOwners.Add(OwnerRecord2);

 // Izveido X atjaunojamo ierakstu

 var OwnerRecordx = new AddresseeOwnerUpdateRecord();

 OwnerRecordx.IdentificationNumber = "40000000XXX";

 OwnerRecordx.Title = "DEPARTMENT2";

 instOwners.Add(OwnerRecordx);

 // Uzsāk datu masveida atjaunošanas procesu.

 // Atbildē tiek saņemts pieprasījuma identifikators

 var requestId =

 client.AddresseeOwnerUpdateAsyncStart(

 AddresseeOwnerUpdatePersonType.RegisteredEntity,

 instOwners);

 // Tiek izsaugta rezultātu izgūšanas metode.

 // Datu apstrāde ir asinhrons process un var prasīt ilgu laiku.

 var response = client.GetResultJournal(requestId);

 }

## Masveida operāciju rezultātu izgūšana

### .NET bibliotēkas koda piemērs

 // Lai izgūtu Masveida operāciju rezultātus kā ievadlauks ir jāpadod pieprasījuma

// identifikatos, kas tika lietots veidotjot pieprasījumu.

 var response = client.GetResultJournal(requestId);

Masveida operācijām par katru ierakstu var tikt atgriezta šāda statusa informācija:

* 'Not\_Found' – ieraksts nav atrasts;
* 'Not\_Unique' – pieprasījuma iervaris ieraksts nav unikāls;
* 'Invalid\_Attributes' – ievadparametri neatbilsts specifikācijai;
* 'Invalid\_Institution\_Type' – nederīga tipa vērtība;
* 'Invalid\_Type\_Change' – nepieļaujama statusa maiņa;
* 'Invalid\_Status\_Code' – nederīga statusa koda vērtība;
* 'Status\_Already\_Set' – Ierakstam jau bija uzstādāmais statuss;
* 'Addressee\_Is\_Anulled' – Eadrese ir anulēta;
* 'Record\_Processing\_Error' – Apstŗades kļūda;

# Bibliotēkas biznesa likumu paziņojumi

|  |  |  |
| --- | --- | --- |
| **Kods** | **Tips** | **Teksts latviešu valodā** |
| BIB.001 | string | <Standarta .net/java uzstādīts kļūdas teksts> |
| BIB.002 | string | Konfigurācija '{0}' nav atrasta. |
| BIB.003 | string | Servisa konfigurētā adrese '{0}' nav pareiza absolūta URI adrese. |
| BIB.004 | string | Servisa konfigurētās adreses '{0}' shēma nav pareiza. Jābūt https. |
| BIB.005 | string | Nav norādīts neviens klienta sertifikāts. |
| BIB.006 | string | Sertifikātam ar indeksu '{0}' nav norādīts izvilkums (thumbprint). |
| BIB.007 | string | Sertifikātam ar izvilkumu (thumbprint) '{0}' reģistrēts vairāk kā vienu reizi. |
| BIB.008 | string | Sertifikātam ar izvilkumu (thumbprint) '{0}' nav atrasts sertifikātu glabātuvē '{1}'. |
| BIB.009 | string | Glabātuvē '{1}' atrasti vairāki sertifikāti ar izvilkumu (thumbprint) '{0}'. |
| BIB.010 | string | Sertifikātam ar izvilkumu (thumbprint) '{0}' nav privātās atslēgas. |
| BIB.011 | string | Konfigurācijā nav neviena derīga sertifikāta. |
| BIB.012 | string | Failu tipam ar indeksu '{0}' nav norādīt paplašinājums. |
| BIB.013 | string | Failu tipas ar paplašinājumu '{0}' jau ir reģistrēts. |
| BIB.014 | string | {0}' paraksta nav ziņojuma objektā. |
| BIB.015 | string | Ziņojumam jābūt nosūtītam, t.i. ziņojumam jābūt izveidotam klienta parakstam. |
| BIB.016 | string | Ziņojumam jau ir pievienots DIV servera apliecinājums. |
| BIB.017 | string | Lai nosūtītu ziņojumu, ir jādefinē vismaz viens adresāts. |
| BIB.018 | string | Ziņojuma adresātam ar indeksu '{0}' ir tukša e-adrese. |
| BIB.019 | string | Datnei ar indeksu '{0}' nav satura. |
| BIB.020 | string | Ziņojuma saņēmējam ar e-adresi '{0}' nevar atrast publisko sertifikāta atslēgu šifrēšanai. |
| BIB.021 | string | Datnes jaukšanas algoritmam jābūt SHA-512 |
| BIB.022 | string | Datnei jābūt definētai jaukšanas funkcijas vērtībai (hash). |
| BIB.024 | string | Datne ir tukša. |
| BIB.025 | string | Nav iespējams noteikt vai datne ir šifrēta. |
| BIB.026 | string | Šifrētas datnes izgūšanai attiecīgam sertifikātam jābūt pieejamam konfigurācijā. |
| BIB.027 | string | Datnes jaucējfunkcijas vērtība nesakrīt ar aploksnē norādīto vērtību. |
| BIB.028 | string | Ziņojumam jābūt piesaistītam apstiprināšanas informācijai, lai veiktu apstiprinājumu. |
| BIB.029 | string | Konfigurācijā norādītajai komunikācijas noildzei jābūt robežās starp 0 un 3600 sekundēm. |