Valsts reģionālās attīstības aģentūra

----

versija

|  |
| --- |
|  |

Rīgā 2018

Dokumenta identifikācija

|  |  |
| --- | --- |
| Dokumenta ID: | -----V- |
| Dokumenta nosaukums: | .  . |
| 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 no Izpildītāja puses |  |  |
| SIA "ABC software" | , projekta vadītājs no Izpildītāja puses |  |  |

Izmaiņu vēsture

|  |  |  |  |
| --- | --- | --- | --- |
| Versija | Datums | Apraksts | Autors |
| 2.00 | 19.09.2017. | Izveidota dokumenta sākotnējā versija | V.Orlovs |
|  |  |  |  |
| 2.01 | 15.11.2017. | 2.2.3. nodaļas 4. scenarijs papildināts ar “appendContentStream” operāciju.  Atjaunoti 2.1.2. nodaļas piemeri | V.Orlovs |
| 2.02 | 03.10.2018. | Aktualizēta 2.1.2. sadaļa. | A.Mačiņš |

**Satura rādītājs**

[Attēlu saraksts 6](#_Toc3388198)

[1. Ievads 7](#_Toc3388199)

[1.1. Dokumenta nolūks 7](#_Toc3388200)

[1.2. Darbības sfēra 7](#_Toc3388201)

[1.3. Termini un pieņemtie apzīmējumi 7](#_Toc3388202)

[1.4. Saistītie dokumenti 7](#_Toc3388203)

[1.5. Dokumenta pārskats 8](#_Toc3388204)

[2. Integrācija 9](#_Toc3388205)

[2.1. Integrācijas apraksts .NET platformai 9](#_Toc3388206)

[2.1.1. Konfigurācijas apraksts 9](#_Toc3388207)

[2.1.1.1. Datnes konfigurēšana 9](#_Toc3388208)

[2.1.1.2. Savienošana ar sertifikātu 10](#_Toc3388209)

[2.1.2. Piemēri 13](#_Toc3388210)

[2.1.2.1. Mapes izveidošanas vai pārbaudes piemērs (CreateFolder) 15](#_Toc3388263)

[2.1.2.2. Datnes izveidošana (AddFile) 15](#_Toc3388264)

[2.1.2.3. Datnes izveidošana un dzēšana (AddRemoveFile) 16](#_Toc3388265)

[2.1.2.4. Mapes dzēšana (RemoveFolder) 17](#_Toc3388266)

[2.1.2.5. Dublēt datni citā mapē (CopyFileToFolder) 18](#_Toc3388267)

[2.1.2.6. Pārbaudīt mapi (CheckFolderExist) 19](#_Toc3388268)

[2.1.2.7. Izgūt mapes datnes (GetDirFiles) 20](#_Toc3388269)

[2.1.2.8. Izgūt datnes saturu (GetFileContent) 21](#_Toc3388270)

[2.1.2.9. Atrast nepieciešamās datnes mapē (SearchFile) 22](#_Toc3388271)

[2.1.2.10. Izgūt informāciju par repozitoriju un tā tipiem (RepositoryInformation) 24](#_Toc3388272)

[2.1.2.11. Navigācijas servisu izmantošana (NavigationServices) 26](#_Toc3388273)

[2.1.2.12. Kataloģizācijas izmantošana (Cataloging) 29](#_Toc3388274)

[2.1.2.13. Objektu pārvaldības servisu izmantošana (ObjectServices) 32](#_Toc3388275)

[2.1.2.14. Objektu īpašību izmaiņas (FileProperties) 35](#_Toc3388276)

[2.1.2.15. Objektu sasaistes servisu izmantošana (RelationshipServices) 38](#_Toc3388277)

[2.1.2.16. ACL servisu izmantošana (ACLServices) 40](#_Toc3388278)

[2.2. Integrācijas apraksts Java platformai 43](#_Toc3388279)

[2.2.1. Savienošana ar sertifikātu 43](#_Toc3388280)

[2.2.2. Wsdl-balstītās pirmkoda automātiska uzģenerēšana un atjaunošana 44](#_Toc3388281)

[2.2.3. EDK servera mijiedarbības demonstrācijas scenāriji 50](#_Toc3388282)

[2.2.4. Demonstrācijas projekta moduļu apraksts 53](#_Toc3388283)

[2.2.4.1. Cxf.xml 53](#_Toc3388284)

[2.2.5. Piemēri 57](#_Toc3388285)

[2.2.5.1. Mapes izveidošanas vai pārbaudes piemērs (CreateFolder) 57](#_Toc3388286)

[2.2.5.2. Datnes izveidošana (createDocument) 59](#_Toc3388287)

[2.2.5.3. Objekta dzēšana (deleteObject) 62](#_Toc3388288)

[2.2.5.4. Mapes dzēšana (deleteTree) 63](#_Toc3388289)

[2.2.5.5. Izgūt mapes struktūru (getFolderTree) 64](#_Toc3388290)

[2.2.5.6. Izgūt datnes saturu (getContentStream) 66](#_Toc3388291)

[2.2.5.7. Izgūt informāciju par repozitoriju (getRepositoryInfo) 67](#_Toc3388292)

[2.2.5.8. Vaicājumu izmantošana informācijas meklēšanai (DiscoveryServices) 72](#_Toc3388293)

[2.2.5.9. Navigācijas servisu izmantošana (NavigationServices) 73](#_Toc3388294)

[2.2.5.10. Kataloģizācijas izmantošana (Multi-filing) 76](#_Toc3388295)

[2.2.5.11. Objektu īpašību izmaiņas (FileProperties) 80](#_Toc3388296)

[2.2.5.12. Objektu sasaistes servisu izmantošana (RelationshipServices) 86](#_Toc3388297)

[2.2.5.13. ACL servisu izmantošana (ACL Services) 89](#_Toc3388298)

# Attēlu saraksts

[1.attēls. Sertifikātu pārvaldības logs 10](#_Toc3388299)

[2.attēls. Instalācijas sākums 11](#_Toc3388300)

[3.attēls. Sertifikāta izvēle 11](#_Toc3388301)

[4.attēls. Opcijas sertifikātam ar privāto atslēgu 12](#_Toc3388302)

[5.attēls. Sertifikāta krātuve 12](#_Toc3388303)

[6.attēls. Sertifikāta instalācijas pabeigšana 13](#_Toc3388304)

[7.attēls. Mapes, kas satur klases, interfeisus un fabrikas servisa piekļūšanai 50](#_Toc3388305)

# Ievads

## Dokumenta nolūks

Elektronisko dokumentu krātuve (saīsināti EDK) nodrošina vidi elektronisko dokumentu glabāšanai, kas pieejama VISS komponentēm, un nodrošina iespēju savstarpējā elektronisko dokumentu apmaiņā izmantot saglabāto elektronisko dokumentu atsauces, nevis to saturu.

Dokumenta mērķis ir aprakstīt saskarnes, kas nodrošina moduļa integrāciju.

## Darbības sfēra

Šajā dokumentā ir uzskaitītas EDK lietotnes integrācijas iespējas.

Šis dokuments ir paredzēts:

* projekta pasūtītāja (VRAA) pārstāvjiem, kuri ir atbildīgi par projekta nodevumu pieņemšanu un izvērtēšanu;
* projekta izpildītāja darbinieki, kuri ir atbildīgi par sistēmas projektēšanu, implementāciju un ieviešanu;
* trešo pušu izstrādātājiem, kuri izstrādās atsevišķas komponentes vai e-pakalpojumus, kuri tiks darbināti portālā.
* Dokumentā aprakstītie piemēri ir izstrādāti ar mērķi demonstrēt pamatrisinājuma funkcionalitāti un tie nevar tikt izmantoti bez izmaiņām reālā pielietojumā.

## Termini un pieņemtie apzīmējumi

Apzīmējumu un terminu vārdnīca pieejama dokumentā [2].

## 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". Elektronisko dokumentu krātuve. Programmatūras projektējuma apraksts. VRAA-13\_7\_17\_41-VISS\_2016-EDK-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.
3. "Valsts informācijas sistēmu savietotāju (VISS) un Vienotā valsts un pašvaldību pakalpojumu portāla www.latvija.lv pilnveidošana un uzturēšana". Elektronisko dokumentu krātuve. Programmatūras ārējo saskarņu projektējums. VRAA-13\_7\_17\_41-VISS\_2016-EDK-PPA\_AS.

## Dokumenta pārskats

Dokumentu veido divi nodalījumi:

* Dokumenta ievads – aprakstīts dokumenta nolūks, termini un pieņemtie apzīmējumi, kā arī norādīta saistība ar citiem dokumentiem un materiāliem;
* Integrācija – sniedz ieskatu par servisu izmantošanu .NET un Java vidēs.

# Integrācija

## Integrācijas apraksts .NET platformai

### Konfigurācijas apraksts

#### Datnes konfigurēšana

Piemēru palaišanai nepieciešams konfigurēt datni: “\Abc.Edk.Samples\Abc.Edk.Samples\  
App.config”, šī datne ir koplietota, tāpēc tās rediģēšana ir jāveic tikai vienu reizi.

##### Sekcijas “<appSettings>”konfigurēšana

Konfigurācijas datnes sekcijā “appSettings” jānomaina tālāk tabulā norādītās vērtības:

|  |  |  |
| --- | --- | --- |
| Vērtības nosaukums | Vērtība | Apraksts |
| RepositoryURN | URN:IVIS:100266:EDK-0000001 | Repozitorija identifikators |

##### Sekcijas “<system.serviceModel>”konfigurēšana

Konfigurācijas datnes sekcijā “bindings/ws2007FederationHttpBinding/binding/  
security/message/tokenRequestParameters” jānomaina tālāk tabulā norādītās vērtības:

|  |  |  |
| --- | --- | --- |
| Vērtības nosaukums | Vērtība | Apraksts |
| wsp:AppliesTo/EndpointReference/Address | https://epak2.abcsoftware.lv/EDK2 | EDK (realm) identifikators |
| issuer.address | https://epak2.abcsoftware.lv/PFAS/Pfas.STS/v1-3/STS/Issue.svc/trust/13/certificatemixed | STS adrese |
| issuerMetadata.address | https://epak2.abcsoftware.lv/PFAS/Pfas.STS/v1-3/STS/Issue.svc/mex | STS servera metadati |

##### Sekcijas “<client”> konfigurēšan

Konfigurācijas datnes sekcijā “<client”>, endpoint adresē jānomaina tālāk tabulā norādītās vērtības:

|  |  |  |
| --- | --- | --- |
| Vērtības nosaukums | Vērtība | Apraksts |
| MultiFilingService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/MultiFilingService.svc/ws2007FederationNoSctMtom | Tiek izmantots dokumentu ievietošanai katalogā / izņemšanai, bez fiziskās kopijas veidošanas. |
| ObjectService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ObjectService.svc/ws2007FederationNoSctMtom | Ļauj veikt standartu CRUD operāciju kopu ar repozitorija objektiem. |
| NavigationService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/NavigationService.svc/ws2007FederationNoSctMtom | Tiek izmantots mapju hierarhijas navigācijai un dokumentu izgūšanai. |
| RelationshipService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/RelationshipService.svc/ws2007FederationNoSctMtom | Tiek izmantots objektu savstarpējo saišu izgūšanai. |
| RepositoryService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/RepositoryService.svc/ws2007FederationNoSctMtom | Nodrošina vispārējās informācijas par repozitoriju izgūšanu, kā arī to objektu tipu pārvaldību. |
| VersioningService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/VersioningService.svc/ws2007FederationNoSctMtom | Nodrošina iespēju strādāt ar vairākām dokumentu versijām. |
| DiscoveryService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/DiscoveryService.svc/ws2007FederationNoSctMtom | Nodrošina iespēju meklēt informāciju, izmantojot vaicājumus (query). |
| ACLService | https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ACLService.svc/ws2007FederationNoSctMtom | Nodrošina repozitorija objektu piekļuves tiesību pārvaldību attiecīgi ACL mehānismam. |

#### Savienošana ar sertifikātu

Piemēru darbināšanai nepieciešams VRAA iestādes izsniegts sertifikāts, kas ir reģistrēts VRAA PFAS AUTH risinājumā ar EDK2.EDKAdministrator tiesībam. Šo sertifikātu nepieciešams importēt sertifikātu krātuvē.

Sertifikāta importēšana sertifikātu krātuvē:

Windows Server nospiediet pogu „Start” un pēc tam „Run”.

Logā „Open” ievadiet „mmc” un pēc tam noklikšķiniet „OK”.

Izvēlaties „File” izvēlni un „Add/Remove snap-in”.

Dialoga logā „Add/Remove snap-in” nospiediet „Add”.

Dialoga logā „Add Standalone Snap-in” izvēlieties „Certificates” un nospiediet „Add”.

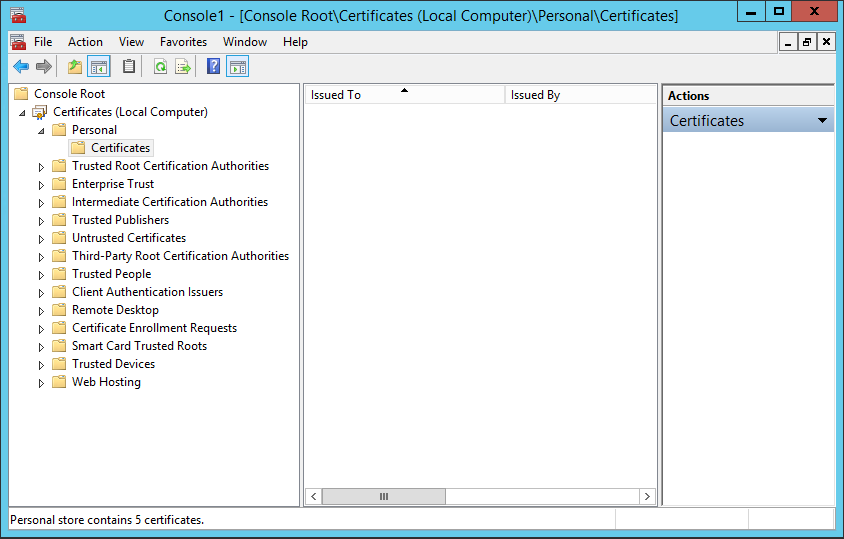
Dialoga logā „Certificates snap-in” izvēlieties „Computer account” un nospiediet „Next”.

Dialoga logā „Select Computer” izvēlieties „Local computer: (the computer this console is running on)” un nospiediet „Finish”.

Dialoga logā „Add Standalone Snap-in” nospiediet „Close”.

Dialoga logā „Add/Remove snap-in” nospiediet „OK”.

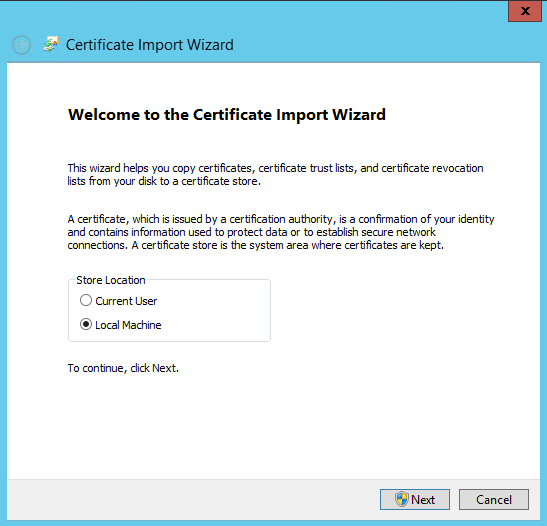
Loga kreisajā panelī izvēlieties „Certificates (Local Computer)”.



1.attēls. Sertifikātu pārvaldības logs

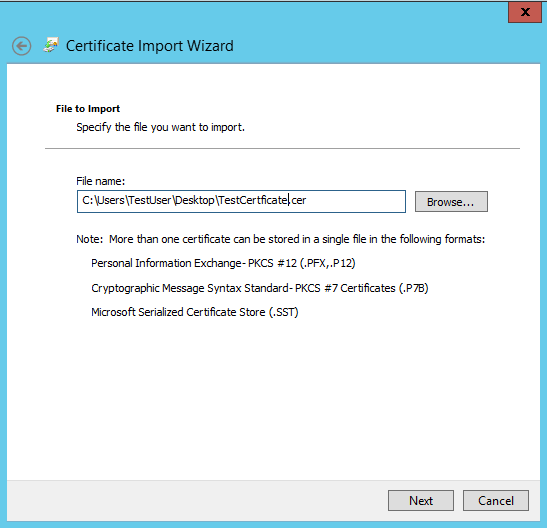
Noklikšķiniet uz “Personal”, ar peles pogu jāizvēlas “All Tasks” sertifikātu, noklikšķiniet “Import”.

Dialoga logā “Welcome to the Certificate Wizard” nospiediet “Next”.



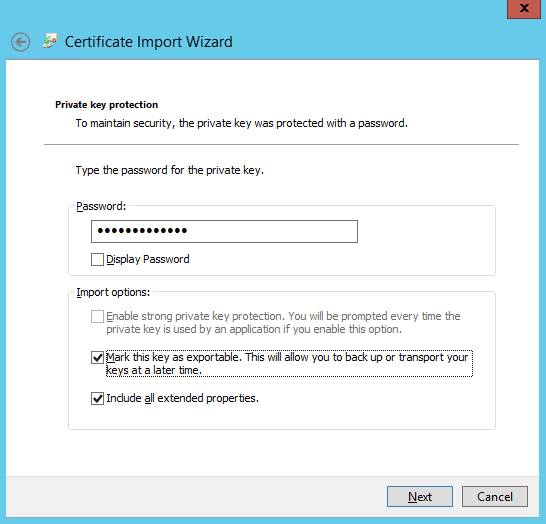
2.attēls. Instalācijas sākums

Dialoga lapā „File to Import” nospiediet „Browse” un atrodiet sertifikāta datni, nospiediet „Next”.



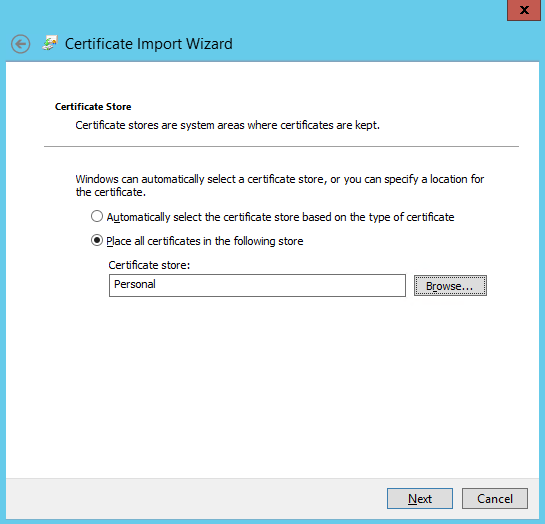
3.attēls. Sertifikāta izvēle

Ja sertifikātam ir parole, ievadiet to logā „Password” un nospiediet „Next”.



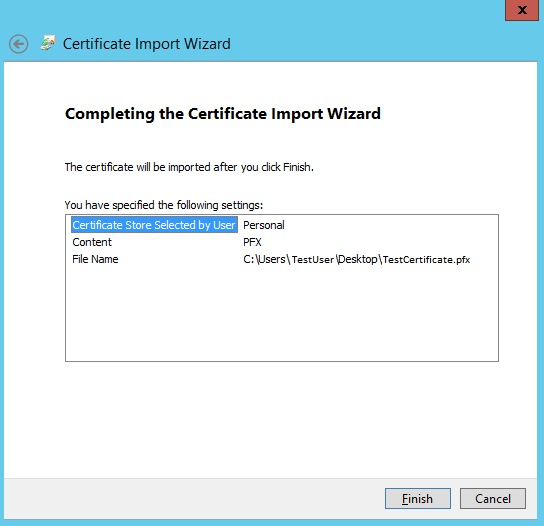
4.attēls. Opcijas sertifikātam ar privāto atslēgu

Dialoga lapā „Certificate Store” izvēlieties „Place all certificates in the following store” un nospiediet „Next”.



5.attēls. Sertifikāta krātuve

Nospiediet „Finish” un, tad „OK”, apstiprinot sertifikāta importēšanu.



6.attēls. Sertifikāta instalācijas pabeigšana

### Piemēri

Piemēros ir izmantota Abc.Edk.WcfBinding bibliotēka, kas ir programmatūras abstrakcijas līmenis (wrapper) virs EDK servisu saskarnes, kas nav obligāts nosacījums servisu izsaukšanai un izstrādātājs var to neizmantot, tomēr bibliotēka nodrošina iespējas, kuras nav pieejamas, izmantojot, piemēram, no WSDL noģenēto klientu (bibliotēka neizmanto EDK servisu WSDL):

* iespēja izsaukt EDK, izmantojot deleģētu identitāti (ActAs);
* iespēja uz kādu laikua pieglabāt un atkārtoti nepieparasīt drošības talonu no STS pie katra izsaukuma (samazināt noslodzi uz infrastruktūru).

Koda fragments, kurā ir redzamas piemēros atkārtoti izmantotās metodes:

|  |
| --- |
| public static class ObjectServiceExtension {  public static string GetOrCreateFolderPath(this Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService objectService, string repositoryId, string edkFolderPath) {  var obj = objectService.GetObjectByPath(  repositoryId: repositoryId,  path: edkFolderPath,  createMissingFolders: true,  filter: CmisProperties.ObjectId,  includeAllowableActions: null,  includeAcl: null,  includePolicyIds: null,  includeRelationships: null,  renditionFilter: null).Result;  return obj?.Id;  }  public static (DocumentId Id, string Name, string FolderId, uint Size) CreateFile(this Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService objectService, string repositoryId, string edkFolderPath, string filePath) {  FileInfo file = new FileInfo(filePath);  if (!file.Exists) {  throw new ArgumentException("This file does not Exist");  }  var folderId = objectService.GetOrCreateFolderPath(repositoryId, edkFolderPath);  var fileByteArray = File.ReadAllBytes(filePath);  var properties = new PropertyDictionary {  [CmisProperties.Name] = new PropertyString(new PropertyStringDefinition { Id = CmisProperties.Name }, new[] { file.Name }),  [CmisProperties.Description] = new PropertyString(new PropertyStringDefinition { Id = CmisProperties.Description }, new[] { file.Name }),  [CmisProperties.ObjectTypeId] = new PropertyId(new PropertyIdDefinition { Id = CmisProperties.ObjectTypeId }, new[] { EdkDocumentTypes.Base }),  [EdkProperties.RetentionGroup] = new PropertyString(new PropertyStringDefinition { Id = EdkProperties.RetentionGroup }, new[] { "nenoteikta" }),  [EdkProperties.Owner] = new PropertyString(new PropertyStringDefinition { Id = EdkProperties.Owner }, new[] { "TestUser" }),  };  var urn = objectService.CreateDocument(repositoryId, properties, folderId, new ContentStream(fileByteArray, file.Name, length:null, mimeType: null, extension: null), VersioningState.Major, policies: null, addAces: null, removeAces: null).Result;  return (urn, file.Name, folderId, (uint)fileByteArray.Length);  }  public static string CreateRelationship(this Abc.Cmis.Abstractions.Bindings.Services.IObjectService objectService, string repositoryId, string sourceFileId, string targetFileId, string relationshipName) {  var properties = new PropertyDictionary {  [CmisProperties.ObjectTypeId] = new PropertyId(new PropertyIdDefinition { Id = CmisProperties.ObjectTypeId }, new[] { CmisRelationshipTypes.Base }),  [CmisProperties.SourceId] = new PropertyId(new PropertyIdDefinition { Id = CmisProperties.SourceId }, new[] { sourceFileId }),  [CmisProperties.TargetId] = new PropertyId(new PropertyIdDefinition { Id = CmisProperties.TargetId }, new[] { targetFileId }),  [CmisProperties.Name] = new PropertyString(new PropertyStringDefinition { Id = CmisProperties.Name }, new[] { relationshipName })  };  var objectId = objectService.CreateRelationship(repositoryId: repositoryId, properties: properties, policies: null, addAces: null, removeAces: null).Result;  return objectId.Id;  }  } |

#### Mapes izveidošanas vai pārbaudes piemērs (CreateFolder)

Ar metodes palīdzību var izveidot mapes. Ja ceļā (parametra), kuru padod metodē, mapes neeksistē EDK, tad metode pati izveidos šo mapi un atgriezīs izveidotās vai eksistējošās gala mapes EDK identifikatoru.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string path = "/Samples";  var edkObjectService = new EdkWcfBinding(new EdkConfiguration()).GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, path);  tring folderId = edkManager.GetOrCreateFolderPath(repositoryId, path); |

#### Datnes izveidošana (AddFile)

Datnes izveidošanas metodes piemērā tiek izveidota mape vai iegūts mapes identifikators.

Izmantojot metodi CreateFile, tiek padots baitu masīvs, mapes identifikators un datnes nosaukums.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkObjectService = new EdkWcfBinding(new EdkConfiguration()).GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  void CleanUp() {  edkObjectService.DeleteTree(repositoryId: repositoryId, folderId: edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), allVersions: true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  byte[] fileBytes = File.ReadAllBytes(localFilePath);  var file = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  Console.WriteLine("File Id: {0}.", file.Id);  Console.ReadKey();  }  finally {  CleanUp();  }ring repositoryPath = "/Samples"; |

#### Datnes izveidošana un dzēšana (AddRemoveFile)

Lai dzēstu datni, ir nepieciešams padot datnes identifikatoru.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkObjectService = new EdkWcfBinding(new EdkConfiguration()).GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  void CleanUp() {  edkObjectService.DeleteTree(repositoryId: repositoryId, folderId: edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), allVersions: true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  var file = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  if (file.Id != null) {  edkObjectService.DeleteObject(repositoryId, file.Id, allVersions: false).Wait();  }  Console.WriteLine("Object: {0} is deleted.", file.Id);  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Mapes dzēšana (RemoveFolder)

Izmantojot metodi DeleteFolderWithFiles, tiks dzēstas visas datnes no mapes, un pēc tam dzēsta arī pati mape.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkObjectService = new EdkWcfBinding(new EdkConfiguration()).GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  void CleanUp() {  edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  Console.WriteLine("Folder {0} with path {1} existed or was created.", folderId, repositoryPath);  var file = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  Console.WriteLine("File {0} was created in {1}.", file.Id.Id, repositoryPath);  if (file.Id != null) {  edkObjectService.DeleteTree(repositoryId, folderId, true, unfileObjects: null, continueOnFailure: true).Wait();  Console.WriteLine("Folder {0} with path {1} was deleted with all it's conent.", folderId, repositoryPath);  }  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Dublēt datni citā mapē (CopyFileToFolder)

Var dublēt datni citā mapē, norādot datnes identifikatoru un mapes identifikatoru.

Izmantojot šo metodi, datne būs abstrakti dublēta.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples/Nested1";  string repositoryCopyPath = "/Samples/Nested2";  string rootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFile = Path.Combine(rootFolder, "Data.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var edkMultifilingService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.IMultiFilingService>();  // Clean up  void CleanUp() {  edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  string copyFolderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryCopyPath);  var file = edkObjectService.CreateFile(repositoryId, repositoryPath, localFile);  Console.WriteLine("{0} created in to {1}.", file.Id, repositoryPath);  if (file.Id != null) {  edkMultifilingService.AddObjectToFolder(repositoryId, file.Id, folderId: copyFolderId, allVersions: null).Wait();  Console.WriteLine("{0} copied to {1}.", file.Id, repositoryCopyPath);  }  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Pārbaudīt mapi (CheckFolderExist)

Metode, nepievienojot datni, pārbauda vai tā eksistē.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  var edkObjectService = new EdkWcfBinding(new EdkConfiguration()).GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  IObjectInfo response = null;  try {  response = edkObjectService.GetObjectByPath(  repositoryId: repositoryId,  path: repositoryPath,  filter: CmisProperties.ObjectId,  includeAcl: null,  renditionFilter: null,  includeRelationships: null,  includePolicyIds: null,  includeAllowableActions: null).Result;  }  catch (Exception ex) when (ex.InnerException is CmisObjectNotFoundException && string.Equals((ex.InnerException as CmisObjectNotFoundException).Message, repositoryPath, StringComparison.Ordinal)) {  // hide object not found exception.  }  bool isFolderExists = response != null;  Console.WriteLine("isFolderExists: {0}", isFolderExists);  Console.ReadKey(); |

#### Izgūt mapes datnes (GetDirFiles)

Piemērā tiks izgūtas visas datnes no mapes, kura tika norādīta parametrā.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string path = "/";  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var edkNavigationService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.INavigationService>();  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, path);  var response = edkNavigationService.GetChildren(  repositoryId,  folderId,  filter: CmisProperties.ObjectId,  orderBy: null,  includeAllowableActions: null,  includePathSegment: true,  includeRelationships: null,  renditionFilter: null,  maxItems: null,  skipCount: null).Result;  foreach (var obj in response) {  var objectName = obj.PathSegment;  Console.WriteLine(objectName);  }  Console.ReadKey(); |

#### Izgūt datnes saturu (GetFileContent)

Piemērā tiek izgūts datnes saturs.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var edkNavigationService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.INavigationService>();  void CleanUp() {  edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  var file = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  if (file.Id != null) {  var files = edkNavigationService.GetChildren(repositoryId, file.FolderId, filter: CmisProperties.ObjectId, orderBy: null, includeAllowableActions: null, includePathSegment: null, includeRelationships: null, renditionFilter: null, maxItems: null, skipCount: null).Result;  foreach (var obj in files) {  var content = edkObjectService.GetContentStream(repositoryId: repositoryId, objectId: obj.Object.Id, streamId: null, offset: null, length: null).Result;  Console.WriteLine("{0} content of size {1} bytes retrieved.", obj.Object.Id, content.Length);  }  }  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Atrast nepieciešamās datnes mapē (SearchFile)

Piemērā tiek izgūta informācija uz SQL vaicājuma bāzes, izmantojot metodi Query.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var edkDiscoveryService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.IDiscoveryService>();  void CleanUp() {  edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  Console.WriteLine("Folder {0} with path {1} existed or was created.", folderId, repositoryPath);  var file = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  Console.WriteLine("File {0} was created in {1}.", file.Id.Id, repositoryPath);  var queryResponse = edkDiscoveryService.Query(  repositoryId: repositoryId,  statement: string.Format(  "SELECT \* FROM edk:d:customDocument WHERE cmis:name = '{0}' And IN\_TREE ('{1}')",  "Data.xml", folderId),  searchAllVersions: null,  includeAllowableActions: null,  includeRelationships: null,  renditionFilter: null,  maxItems: null,  skipCount: null).Result;  foreach (var item in queryResponse) {  Console.WriteLine("Item {0} was found by search query.", item.Id);  }  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Izgūt informāciju par repozitoriju un tā tipiem (RepositoryInformation)

Piemērā tiek izgūta repozitorija informācija un tā tipu metadati.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| var edkRepositoryService = new EdkWcfBinding(new EdkConfiguration()).GetService<Abc.Cmis.Abstractions.Bindings.Services.IRepositoryService>();  var repositories = edkRepositoryService.GetRepositories().Result;  foreach (var repo in repositories) {  var repoInfo = edkRepositoryService.GetRepositoryInfo(repositoryId: repo.Id).Result;  Console.WriteLine("RepositoryName: {0}", repoInfo.Name);  var typeChildren = edkRepositoryService.GetTypeChildren(  repositoryId: repo.Id,  typeId: CmisDocumentTypes.Base,  includePropertyDefinitions: null,  maxItems: null,  skipCount: null).Result;  Console.WriteLine("{1}{0} children:", nameof(CmisDocumentTypes.Base), Environment.NewLine);  foreach (var obj in typeChildren) {  if (!string.IsNullOrWhiteSpace(obj.Id))  Console.WriteLine("{0}", obj.Id);  }  string typeId;  if (string.Equals(repo.Id, EdkRepositories.Base, StringComparison.OrdinalIgnoreCase)) {  typeId = EdkDocumentTypes.Edoc;  }  else if (string.Equals(repo.Id, EdkRepositories.EAddress, StringComparison.OrdinalIgnoreCase)) {  typeId = EdkDocumentTypes.EaddressEnvelope;  }  else {  typeId = null;  }  if (typeId == null) {  Console.WriteLine("Unknown reposiotry. {0}", Environment.NewLine);  }  else {  var typeDefinition = edkRepositoryService.GetTypeDefinition(repositoryId: repo.Id, typeId: typeId).Result;  var typeDefinitionString = new StringBuilder();  typeDefinitionString.AppendLine();  typeDefinitionString.AppendLine($"{CmisProperties.ObjectId}: {typeDefinition.Id}");  typeDefinitionString.AppendLine($"{CmisProperties.BaseTypeId}: {typeDefinition.BaseTypeId}");  typeDefinitionString.AppendLine($"{CmisProperties.ParentTypeId}: {typeDefinition.ParentTypeId}");  typeDefinitionString.AppendLine($"{CmisProperties.Description}: {typeDefinition.Description}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.DisplayName)}: {typeDefinition.DisplayName}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.LocalName)}: {typeDefinition.LocalName}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.LocalNamespace)}: {typeDefinition.LocalNamespace}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.QueryName)}: {typeDefinition.QueryName}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.IsControllableAcl)}: {typeDefinition.IsControllableAcl}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.IsCreatable)}: {typeDefinition.IsCreatable}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.IsFileable)}: {typeDefinition.IsFileable}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.IsFulltextIndexed)}: {typeDefinition.IsFulltextIndexed}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.IsIncludedInSupertypeQuery)}: {typeDefinition.IsIncludedInSupertypeQuery}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.IsQueryable)}: {typeDefinition.IsQueryable}");  typeDefinitionString.AppendLine($"{nameof(typeDefinition.PropertyDefinitions)}: {typeDefinition.PropertyDefinitions.Count}");  Console.WriteLine("{2}{0} type definition: {1}{2}", typeId, typeDefinitionString.ToString(), Environment.NewLine);  }  }  Console.ReadKey(); |

#### Navigācijas servisu izmantošana (NavigationServices)

Piemērā attēlota EDK navigācijas servisu darbības demonstrācija.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var edkNavigationService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.INavigationService>();  var edkDiscoveryService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.IDiscoveryService>();  // Clean up  void CleanUp() {  edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  string folder0 = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  Console.WriteLine("Folder0: {0}", folder0);  string folder1 = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath + "/Folder1");  Console.WriteLine("Folder1: {0}", folder1);  string folder2 = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath + "/Folder1/Folder2");  Console.WriteLine("Folder2: {0}", folder2);  var file1 = edkObjectService.CreateFile(repositoryId, repositoryPath + "/Folder1", localFilePath);  Console.WriteLine("File1: {0}", file1.Id);  var file2 = edkObjectService.CreateFile(repositoryId, repositoryPath + "/Folder1/Folder2", localFilePath);  Console.WriteLine("File2: {0}", file2.Id);  var folderTree = edkNavigationService.GetFolderTree(repositoryId, folder0, 3, filter: CmisProperties.ObjectId, includePathSegment: null, includeAllowableActions: null, renditionFilter: null, includeRelationships: null).Result;  Console.WriteLine();  Console.WriteLine("Folder tree for root folder ({0}):", folder0);  var treeBuilder = new StringBuilder();  foreach (var elem in folderTree) {  treeBuilder.AppendFormat("-- {0}", elem.Object.Id);  foreach (var childrenElem in elem.Children) {  treeBuilder.AppendLine();  treeBuilder.AppendFormat("---- {0}", childrenElem.Object.Id);  }  treeBuilder.AppendLine();  }  Console.WriteLine(treeBuilder.ToString());  var descendants = edkNavigationService.GetChildren(  repositoryId,  folder0,  filter: string.Concat(CmisProperties.ObjectId, CmisProperties.Path),  orderBy: null,  includeRelationships: null,  includeAllowableActions: null,  renditionFilter: null,  includePathSegment: null,  maxItems: null,  skipCount: null).Result;  var objectResponse = edkObjectService.GetObjectByPath(  repositoryId,  repositoryPath + "/Folder1/Data.xml",  filter: string.Concat(CmisProperties.ObjectId, CmisProperties.Path),  includeAllowableActions: null,  includeRelationships: null,  renditionFilter: null,  includePolicyIds: null,  includeAcl: null).Result;  Console.WriteLine();  Console.WriteLine("File1 ObjectInfo: {0}:{1}", CmisProperties.ObjectId, objectResponse.Id);  var parent = edkNavigationService.GetFolderParent(repositoryId, folder2, filter: CmisProperties.ObjectId).Result;  Console.WriteLine();  Console.WriteLine("Folder2 parent: {0}", parent.Id);  var queryResponse = edkDiscoveryService.Query(  repositoryId: repositoryId,  statement: string.Format(  "SELECT \* FROM cmis:document WHERE cmis:name = '{0}' And IN\_TREE ('{1}')",  "Data.xml", folder2),  searchAllVersions: null,  includeAllowableActions: null,  includeRelationships: null,  renditionFilter: null,  maxItems: null,  skipCount: null).Result;  Console.WriteLine();  Console.WriteLine("File2 ObjectInfo: {0}:{1}", CmisProperties.ObjectId, queryResponse.First().Id);  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Kataloģizācijas izmantošana (Cataloging)

Piemērā attēlota EDK kataloģizācijas servisu darbības demonstrācija.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var edkNavigationService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.INavigationService>();  var edkDiscoveryService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.IDiscoveryService>();  // Clean up  void CleanUp() {  edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  string folder0 = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  Console.WriteLine("Folder0: {0}", folder0);  string folder1 = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath + "/Folder1");  Console.WriteLine("Folder1: {0}", folder1);  string folder2 = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath + "/Folder1/Folder2");  Console.WriteLine("Folder2: {0}", folder2);  var file1 = edkObjectService.CreateFile(repositoryId, repositoryPath + "/Folder1", localFilePath);  Console.WriteLine("File1: {0}", file1.Id);  var file2 = edkObjectService.CreateFile(repositoryId, repositoryPath + "/Folder1/Folder2", localFilePath);  Console.WriteLine("File2: {0}", file2.Id);  var folderTree = edkNavigationService.GetFolderTree(repositoryId, folder0, 3, filter: CmisProperties.ObjectId, includePathSegment: null, includeAllowableActions: null, renditionFilter: null, includeRelationships: null).Result;  Console.WriteLine();  Console.WriteLine("Folder tree for root folder ({0}):", folder0);  var treeBuilder = new StringBuilder();  foreach (var elem in folderTree) {  treeBuilder.AppendFormat("-- {0}", elem.Object.Id);  foreach (var childrenElem in elem.Children) {  treeBuilder.AppendLine();  treeBuilder.AppendFormat("---- {0}", childrenElem.Object.Id);  }  treeBuilder.AppendLine();  }  Console.WriteLine(treeBuilder.ToString());  var descendants = edkNavigationService.GetChildren(  repositoryId,  folder0,  filter: string.Concat(CmisProperties.ObjectId, CmisProperties.Path),  orderBy: null,  includeRelationships: null,  includeAllowableActions: null,  renditionFilter: null,  includePathSegment: null,  maxItems: null,  skipCount: null).Result;  var objectResponse = edkObjectService.GetObjectByPath(  repositoryId,  repositoryPath + "/Folder1/Data.xml",  filter: string.Concat(CmisProperties.ObjectId, CmisProperties.Path),  includeAllowableActions: null,  includeRelationships: null,  renditionFilter: null,  includePolicyIds: null,  includeAcl: null).Result;  Console.WriteLine();  Console.WriteLine("File1 ObjectInfo: {0}:{1}", CmisProperties.ObjectId, objectResponse.Id);  var parent = edkNavigationService.GetFolderParent(repositoryId, folder2, filter: CmisProperties.ObjectId).Result;  Console.WriteLine();  Console.WriteLine("Folder2 parent: {0}", parent.Id);  var queryResponse = edkDiscoveryService.Query(  repositoryId: repositoryId,  statement: string.Format(  "SELECT \* FROM cmis:document WHERE cmis:name = '{0}' And IN\_TREE ('{1}')",  "Data.xml", folder2),  searchAllVersions: null,  includeAllowableActions: null,  includeRelationships: null,  renditionFilter: null,  maxItems: null,  skipCount: null).Result;  Console.WriteLine();  Console.WriteLine("File2 ObjectInfo: {0}:{1}", CmisProperties.ObjectId, queryResponse.First().Id);  Console.ReadKey();  }  finally {  CleanUp();  } } |

#### Objektu pārvaldības servisu izmantošana (ObjectServices)

Piemērā attēlota EDK objektu pārvaldības servisu darbības demonstrācija.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string repoFolder1 = repositoryPath + "/Folder1";  string repoFolder2 = repositoryPath + "/Folder2";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  string localFile2Path = Path.Combine(appRootFolder, "Data2.xml");  var edkObjectService = new EdkWcfBinding(new EdkConfiguration()).GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  void CleanUp() {  var deletedIds = edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Result;  }  CleanUp();  try {  string rootFolder = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  string folder1 = edkObjectService.GetOrCreateFolderPath(repositoryId, repoFolder1);  var file = edkObjectService.CreateFile(repositoryId, repoFolder1, localFilePath);  string fileId = file.Id;  Console.WriteLine("Create a file: Data.xml({0}), in: {1}{2}", fileId, repoFolder1, Environment.NewLine);  var getPropResponse = edkObjectService.GetProperties(repositoryId, file.Id, CmisProperties.ChangeToken).Result;  getPropResponse.TryGetValue(CmisProperties.ChangeToken, out IProperty fileChangeTokenProperty);  string fileToken = fileChangeTokenProperty.FirstValueAsString;  string folder2 = edkObjectService.GetOrCreateFolderPath(repositoryId, repoFolder2);  var properties = new PropertyDictionary {  [CmisProperties.Name] = new PropertyString(new PropertyStringDefinition { Id = CmisProperties.Name }, new[] { "DataCopy.xml" }),  [CmisProperties.Description] = new PropertyString(new PropertyStringDefinition { Id = CmisProperties.Description }, new[] { "Copy description" }),  };  var fileCopy = edkObjectService.CreateDocumentFromSource(repositoryId, sourceId: file.Id, folderId: folder2, properties: properties, policies: null, addAces: null, removeAces: null, versioningState: null).Result;  Console.WriteLine("Copy 'Data.xml' to: {1}, rename to: 'DataCopy.xml' ({0}) (NB! file content and filename were not changed):", fileCopy.Id, repoFolder2);  var contentStreamResponse = edkObjectService.GetContentStream(repositoryId, fileCopy.Id, streamId: null, offset: null, length: null).Result;  Console.WriteLine("--> verify that current content file name of the 'DataCopy.xml' is same as source document's ({0}): {1}{2}", fileCopy.Id, contentStreamResponse.FileName, Environment.NewLine);  var stream = new ContentStream(File.ReadAllBytes(localFile2Path), "NewData.xml", length: null, mimeType: null, extension: null);  Console.WriteLine("Set a new file content and change filename to 'NewData.xml'.", fileCopy.Id, repoFolder2, Environment.NewLine);  var setContentStreamResponse = edkObjectService.SetContentStream(repositoryId, fileCopy.Id, true, fileToken, stream).Result;  getPropResponse = edkObjectService.GetProperties(repositoryId, fileCopy.Id, CmisProperties.ChangeToken).Result;  getPropResponse.TryGetValue(CmisProperties.ChangeToken, out fileChangeTokenProperty);  fileToken = fileChangeTokenProperty.FirstValueAsString;  var appendContentStreamResponse = edkObjectService.AppendContentStream(repositoryId, fileCopy.Id, true, fileToken, stream).Result;  contentStreamResponse = edkObjectService.GetContentStream(repositoryId, fileCopy.Id, streamId: null, offset: null, length: null).Result;  Console.WriteLine("Current content file name of the 'DataCopy.xml'({0}): {1}{2}", fileCopy.Id, contentStreamResponse.FileName, Environment.NewLine);  getPropResponse = edkObjectService.GetProperties(repositoryId, fileCopy.Id, CmisProperties.ChangeToken).Result;  getPropResponse.TryGetValue(CmisProperties.ChangeToken, out fileChangeTokenProperty);  string fileCopyChangeToken = fileChangeTokenProperty.FirstValueAsString;  var deleteContentStreamResponse = edkObjectService.DeleteContentStream(repositoryId, fileCopy.Id, fileCopyChangeToken).Result;  edkObjectService.DeleteTree(repositoryId, rootFolder, true, continueOnFailure: true, unfileObjects: null).Wait();  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Objektu īpašību izmaiņas (FileProperties)

Piemērā attēlota EDK objektu īpašību izmaiņas servisu darbības demonstrācija.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  string localFile2Path = Path.Combine(appRootFolder, "Data2.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  void CleanUp() {  edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Wait();  }  CleanUp();  try {  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  #region File1  var file1 = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  var getPropResponse = edkObjectService.GetProperties(repositoryId, file1.Id, string.Concat(CmisProperties.ChangeToken,",", CmisProperties.Name)).Result;  getPropResponse.TryGetValue(CmisProperties.ChangeToken, out IProperty fileChangeTokenProperty);  string file1Token = fileChangeTokenProperty.FirstValueAsString;  getPropResponse.TryGetValue(CmisProperties.Name, out IProperty nameProperty);  Console.WriteLine("File1 name: {0}", nameProperty.FirstValueAsString);    string file1Id = file1.Id;  #endregion  #region File2  var file2 = edkObjectService.CreateFile(repositoryId, repositoryPath, localFile2Path);  getPropResponse = edkObjectService.GetProperties(repositoryId, file2.Id, string.Concat(CmisProperties.ChangeToken, ",", CmisProperties.Name)).Result;  getPropResponse.TryGetValue(CmisProperties.ChangeToken, out fileChangeTokenProperty);  string file2Token = fileChangeTokenProperty.FirstValueAsString;  getPropResponse.TryGetValue(CmisProperties.Name, out nameProperty);  Console.WriteLine("File2 name: {0}", nameProperty.FirstValueAsString);  string file2Id = file2.Id;  #endregion  #region UpdateProperties  Console.WriteLine("Single doc. update");  var properties = new PropertyDictionary {  [CmisProperties.Name] = new PropertyString(new PropertyStringDefinition { Id = CmisProperties.Name }, new[] { "NameChangedWithSingleUpdate" }),  };  var gg = edkObjectService.UpdateProperties(repositoryId, file1Id, file1Token, properties).Result;  #endregion  #region GetProperties  var propResponse = edkObjectService.GetProperties(repositoryId, file1Id, filter: CmisProperties.Name).Result;  propResponse.TryGetValue(CmisProperties.Name, out nameProperty);  Console.WriteLine("File1 name: {0}", nameProperty.FirstValueAsString);  propResponse = edkObjectService.GetProperties(repositoryId, file2Id, filter: CmisProperties.Name).Result;  propResponse.TryGetValue(CmisProperties.Name, out nameProperty);  Console.WriteLine("File2 name: {0}", nameProperty.FirstValueAsString);  #endregion  #region BulkUpdateProperties  properties = new PropertyDictionary {  [CmisProperties.Name] = new PropertyString(new PropertyStringDefinition { Id = CmisProperties.Name }, new[] { "NameChangedWithBulkUpdate" }),  };  var tokenIdList = new List<IObjectIdChangeToken> {  new ObjectIdChangeToken((ObjectId)file1Id, file1Token, (ObjectId)file1Id),  new ObjectIdChangeToken((ObjectId)file2.Id, file2Token, (ObjectId)file2.Id),  };  Console.WriteLine("Bulk update");  var response = edkObjectService.BulkUpdateProperties(repositoryId, bulkUpdateData: new BulkUpdate() { Properties = properties, ObjectIdAndChangeToken = tokenIdList }).Result;  var objIdAndToken = response.FirstOrDefault(t => t.Id.Id == file2.Id.Id);  file1Id = objIdAndToken.NewId?.Id ?? objIdAndToken.Id.Id;  file1Token = objIdAndToken.ChangeToken;  objIdAndToken = response.FirstOrDefault(t => t.Id.Id == file2.Id.Id);  file2Id = objIdAndToken.NewId?.Id ?? objIdAndToken.Id.Id;  file2Token = objIdAndToken.ChangeToken;  #endregion  #region Get updated properties  propResponse = edkObjectService.GetProperties(repositoryId, file1Id, filter: CmisProperties.Name).Result;  propResponse.TryGetValue(CmisProperties.Name, out nameProperty);  Console.WriteLine("File1 name: {0}", nameProperty.FirstValueAsString);  propResponse = edkObjectService.GetProperties(repositoryId, file2Id, filter: CmisProperties.Name).Result;  propResponse.TryGetValue(CmisProperties.Name, out nameProperty);  Console.WriteLine("File2 name: {0}", nameProperty.FirstValueAsString);  #endregion  Console.ReadKey();  }  finally {  CleanUp();  } |

#### Objektu sasaistes servisu izmantošana (RelationshipServices)

Piemērā attēlota EDK objektu sasaistes servisu darbības demonstrācija.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  string localFile2Path = Path.Combine(appRootFolder, "Data2.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var relationshipService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.IRelationshipService>();  void CleanUp() {  var deletedIds = edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Result;  }  CleanUp();  try {  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  var file1 = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  var file2 = edkObjectService.CreateFile(repositoryId, repositoryPath, localFile2Path);  Console.WriteLine("Create relationship:{0}Source: {1}, Target: {2}", Environment.NewLine,  file1.Id, file2.Id);  string relationshipId = edkObjectService.CreateRelationship(repositoryId, file1.Id, file2.Id, "TestRelationship");  Console.WriteLine("{0}Load relationship:", Environment.NewLine);  var relationships = relationshipService.GetObjectRelationships(repositoryId, file1.Id, filter: CmisProperties.TargetId, relationshipDirection: null, typeId: null, maxItems: null, skipCount: null, includeAllowableActions: null, includeSubRelationshipTypes: null).Result;  foreach (var relationship in relationships) {  relationship.Properties.TryGetValue(CmisProperties.TargetId, out IProperty targetId);  if (targetId != null) {  Console.WriteLine("Relationship TargetId: {0}", targetId.FirstValueAsString);  }  }  Console.ReadKey();  }  finally {  CleanUp();  } |

#### ACL servisu izmantošana (ACLServices)

Piemērā attēlota EDK objektu piekļuves kontroles servisu darbības demonstrācija.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| string repositoryId = ConfigurationManager.AppSettings["RepositoryURN"];  string principalId = "TestPrincipalId";  string canApplyAclPermission = "edk:applyACL";  string repositoryPath = "/Samples";  string appRootFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);  string localFilePath = Path.Combine(appRootFolder, "Data.xml");  var edkWcfBinding = new EdkWcfBinding(new EdkConfiguration());  var edkObjectService = edkWcfBinding.GetService<Abc.Edk.Abstractions.Bindings.Services.IEdkObjectService>();  var edkAclService = edkWcfBinding.GetService<Abc.Cmis.Abstractions.Bindings.Services.IAclService>();  void CleanUp() {  var deletedIds = edkObjectService.DeleteTree(repositoryId, edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath), true, continueOnFailure: true, unfileObjects: null).Result;  }  CleanUp();  try {  // Create folder  string folderId = edkObjectService.GetOrCreateFolderPath(repositoryId, repositoryPath);  // Get AllowableActions on folder  var allowableActionsResp = edkObjectService.GetAllowableActions(repositoryId, folderId).Result;  Console.WriteLine("Can ApplyACL = {0}, Folder = {1}", allowableActionsResp.Contains(CmisAllowableActions.CanApplyACL), folderId);  // Create document  var file = edkObjectService.CreateFile(repositoryId, repositoryPath, localFilePath);  // Get AllowableActions on file  allowableActionsResp = edkObjectService.GetAllowableActions(repositoryId, folderId).Result;  Console.WriteLine();  Console.WriteLine("Can ApplyACL = {0}, File = {1}", allowableActionsResp.Contains(CmisAllowableActions.CanApplyACL), file.Name);  // Get ACL on file  var aclResponse = edkAclService.GetAcl(repositoryId, file.Id, false).Result;  Console.WriteLine();  Console.WriteLine("{0} ACL:", file.Name);  foreach (var ace in aclResponse.Aces) {  Console.WriteLine("Principal = {0}, Permissions = {1}", ace.Principal.PrincipalId, string.Join(", ", ace.Permissions));  }  // Apply ACL  // Use GetRepositoryInfo to obtain supported permissions  HashSet<string> permissions = new HashSet<string> { canApplyAclPermission };  var aclEntry = new Ace(principalId: principalId, permission: permissions);  var aclEntries = new List<IAce> { aclEntry };  var acl = new Acl(aclEntries);  Console.WriteLine("{0}Apply ACL to folder:{0}", Environment.NewLine);  edkAclService.ApplyAcl(repositoryId: repositoryId, objectId: folderId, addAces: acl, removeAces: null, aclPropagation: AclPropagation.Propagate);  // Get ACL on file  aclResponse = edkAclService.GetAcl(repositoryId, file.Id, false).Result;  Console.WriteLine("{0} ACL:", file.Name);  foreach (var ace in aclResponse.Aces) {  Console.WriteLine("Principal = {0}, Permissions = {1}", ace.Principal.PrincipalId, string.Join(", ", ace.Permissions));  }  // Remove ACL on file  permissions = new HashSet<string> { canApplyAclPermission };  aclEntries = new List<IAce> {  new Ace(principalId: principalId, permission: permissions)  };  var removeAcl = new Acl(aclEntries);  Console.WriteLine("{0}Remove ACL on folder:{0}", Environment.NewLine);  edkAclService.ApplyAcl(repositoryId, folderId, null, removeAcl, AclPropagation.Propagate).Wait();  // Get ACL on file  aclResponse = edkAclService.GetAcl(repositoryId, file.Id, false).Result;  Console.WriteLine("{0} ACL:", file.Name);  foreach (var ace in aclResponse.Aces) {  Console.WriteLine("Principal = {0}, Permissions = {1}", ace.Principal.PrincipalId, string.Join(", ", ace.Permissions));  }  Console.ReadKey();  }  finally {  CleanUp();  } |

## Integrācijas apraksts Java platformai

Zemāk ir sniegts demonstrācijas apraksts projektam “edk-sample”, kura ietvaros tiek implementēts:

STS autentifikācijas uzturēšana, griežoties pie EDK servera;

Klases (moduļi) darbam ar EDK serveri;

Automātiskā Java klašu ģenerācija, balstoties uz WSDL.

Demonstrācijas projekta pamatā tiek izmantota atvērtā platforma Apache CXF, programmatūras projektu vadības rīks Apache Maven, rīki JAXB un JAX-WS kuras kopumā nosaka projekta struktūru. Apache CXF pielietošana projektā ļauj izvairīties no sekojošās funkcionalitātes manuālās programmēšanas:

mijiedarbība ar STS serveri, tokenu iegūšana (*Apache* CXF);

pieprasījumu pilnveidošanas ar tokenu (*Apache* CXF);

JAVA-koda uzģenerēšanas katra veida pieprasījumam un to rezultātu apstrādei (JAXB);

mijiedarbība ar EDK serveri (JAX-WS).

Līdz ar to, gadījumā, ja EDK servera pusē tiek mainīts wsdl apraksts, tad kods, kurš ir atbildīgs par pieprasījumu veidošanas un rezultātu apstrādes var būt noģenerēts jeb atjaunots automātiski.

Projekta realizācija bez minētās platformas ir iespējama, taču šajā gadījumā papildu ir jāimplementē klases, kas izpilda iepriekš minēto funkcionalitāti.

Demonstrācijas mērķim mapē *samples* tiek izvietotas septiņas klases, kuras implementē scenārijus mijiedarbībai ar EDK serveri.

### Savienošana ar sertifikātu

Piemēru darbināšanai nepieciešams VRAA iestādes izsniegts sertifikāts, kas ir reģistrēts VRAA PFAS AUTH risinājumā ar EDK2.EDKAdministrator tiesībam.

Sertifikātu izmantošanai Java projektā ir nepieciešams pielikt sertifikātu ar *.pfx* paplašinājumu projekta katalogā (piemērām mapē “src/main/resources/”). Pēc tam aizpildīt client\_sign.properties failu norādot sertifikāta faila nosaukumu, sertifikāta alias, tipu un paroli.

|  |
| --- |
| 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=123 org.apache.ws.security.crypto.merlin.keystore.alias=le-fb155573-a1b3-4ddd-ab0f-e723f9efd787 org.apache.ws.security.crypto.merlin.file=cert.pfx |

### Wsdl-balstītās pirmkoda automātiska uzģenerēšana un atjaunošana

Projektā izveidojiet mapi, kur glabāt visu servisu *wsdl* datnes. Piemērām:

\src\main\resources\wsdls\

Lejupielādējiet izveidotājā mapē katru *wsdl* datni no attiecīgā servisa adreses:

https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ObjectService.svc?singleWsdl

https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ NavigationService.svc?singleWsdl

https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ RepositoryService.svc?singleWsdl

https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ DiscoveryService.svc?singleWsdl

https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ MultiFilingService.svc?singleWsdl

https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ RelationshipService.svc?singleWsdl

https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ ACLService.svc?singleWsdl

Katra *wsdl* datne ir jāmodificē, pievienojot sekojošās politikas fragmentus (visiem failiem fragmenti ir vienādi):

1. wsp policy:

*Wsdl* koda piemērs ir parādīts zemāk:

|  |
| --- |
| <wsdl:definitions  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  xmlns:cmism="http://docs.oasis-open.org/ns/cmis/messaging/200908/"  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"  xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"  xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"  xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"  xmlns:xsd="http://www.w3.org/2001/XMLSchema"  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"  xmlns:tns="http://ivis.eps.gov.lv/ISS/EDKService/v1-0/"  xmlns:wsa10="http://www.w3.org/2005/08/addressing"  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"  xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/"  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"  name="ObjectService"  targetNamespace="http://ivis.eps.gov.lv/ISS/EDKService/v1-0/">   <wsp:Policy wsu:Id="Mtom.xml">  <wsoma:OptimizedMimeSerialization xmlns:wsoma="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization"/>  </wsp:Policy> |

1. wsp: Policy

<wsp:Policy wsu:Id="ws2007FederationNoSctMtom\_policy">  
 <wsp:ExactlyOne>  
 <wsp:All>  
 <wsoma:OptimizedMimeSerialization xmlns:wsoma="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization"/>  
 <sp:TransportBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
 <wsp:Policy>  
 <sp:TransportToken>  
 <wsp:Policy>  
 <sp:HttpsToken>  
 <wsp:Policy/>  
 </sp:HttpsToken>  
 </wsp:Policy>  
 </sp:TransportToken>  
 <sp:AlgorithmSuite>  
 <wsp:Policy>  
 <sp:Basic256/>  
 </wsp:Policy>  
 </sp:AlgorithmSuite>  
 <sp:Layout>  
 <wsp:Policy>  
 <sp:Strict/>  
 </wsp:Policy>  
 </sp:Layout>  
 <sp:IncludeTimestamp/>  
 </wsp:Policy>  
 </sp:TransportBinding>  
 <sp:EndorsingSupportingTokens xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
 <wsp:Policy>  
 <sp:IssuedToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
 <Issuer xmlns="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
 <Address xmlns="http://www.w3.org/2005/08/addressing">  
 https://epak2.abcsoftware.lv/PFAS/Pfas.STS/v1-3/STS/Issue.svc/trust/13/certificatemixed  
 </Address>  
 <Metadata xmlns="http://www.w3.org/2005/08/addressing">  
 <Metadata xmlns="http://schemas.xmlsoap.org/ws/2004/09/mex" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
 <wsx:MetadataSection xmlns="">  
 <wsx:MetadataReference>  
 <Address xmlns="http://www.w3.org/2005/08/addressing">  
 https://epak2.abcsoftware.lv/PFAS/Pfas.STS/v1-3/STS/Issue.svc/mex  
 </Address>  
 </wsx:MetadataReference>  
 </wsx:MetadataSection>  
 </Metadata>  
 </Metadata>  
 </Issuer>  
 <sp:RequestSecurityTokenTemplate>  
 <trust:KeyType xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</trust:KeyType>  
 <trust:Claims xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512" Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://www.oasis-open.org/RSA2004/attributes/AUTHORITY"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://docs.oasis-open.org/wsfed/authorization/200706/claims/action"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="urn:ivis:100001:name.id-viss"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://schemas.microsoft.com/ws/2008/06/identity/claims/role"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>  
 <wsid:ClaimType xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sid"/>  
 </trust:Claims>  
 <trust:KeyWrapAlgorithm xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p</trust:KeyWrapAlgorithm>  
 <trust:EncryptWith xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://www.w3.org/2001/04/xmlenc#aes256-cbc</trust:EncryptWith>  
 <trust:SignWith xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://www.w3.org/2000/09/xmldsig#hmac-sha1</trust:SignWith>  
 <trust:CanonicalizationAlgorithm xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://www.w3.org/2001/10/xml-exc-c14n#</trust:CanonicalizationAlgorithm>  
 <trust:EncryptionAlgorithm xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://www.w3.org/2001/04/xmlenc#aes256-cbc</trust:EncryptionAlgorithm>  
 </sp:RequestSecurityTokenTemplate>  
 <wsp:Policy>  
 <sp:RequireInternalReference/>  
 </wsp:Policy>  
 </sp:IssuedToken>  
 </wsp:Policy>  
 </sp:EndorsingSupportingTokens>  
 <sp:Wss11 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
 <wsp:Policy/>  
 </sp:Wss11>  
 <sp:Trust13 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
 <wsp:Policy>  
 <sp:MustSupportIssuedTokens/>  
 <sp:RequireClientEntropy/>  
 <sp:RequireServerEntropy/>  
 </wsp:Policy>  
 </sp:Trust13>  
 <wsaw:UsingAddressing/>  
 </wsp:All>  
 </wsp:ExactlyOne>  
</wsp:Policy>

<wsp:UsingPolicy wsdl:Required="true"/>

1. jāievieto uzreiz pēc taga <wsp:Policy … </wsp:Policy>. Ievietošanas rezultāta paraugu var apskatīt demonstrācijas projekta jebkura servisa wsdl datnē.
2. Aizvietot wsdl:types

<wsdl:types>  
 <xsd:schema targetNamespace="http://docs.oasis-open.org/ns/cmis/ws/200908/" xmlns:wsp="http://schemas.xmlsoap.org/wsdl/" xmlns:wsu="http://docs.oasis-open.org/ns/cmis/messaging/200908/" xmlns:wsdl="http://ivis.eps.gov.lv/ISS/EDKService/v1-0" xmlns:cmism2="http://www.w3.org/2006/05/addressing/wsdl" xmlns:tns="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/" xmlns:cmism="http://docs.oasis-open.org/ns/cmis/messaging/200908/" xmlns:i0="http://ivis.eps.gov.lv/ISS/EDKService/v1-0" xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap122="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns2="http://ivis.eps.gov.lv/ISS/EDKService/v1-0" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsa10="http://www.w3.org/2005/08/addressing" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:wsdl2="http://schemas.xmlsoap.org/wsdl/" xmlns:wsu2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
 <xsd:import namespace="http://docs.oasis-open.org/ns/cmis/messaging/200908/" schemaLocation="http://ivis.eps.gov.lv/RC/xsd/100001/CMISMessaging/1-1/CMISMessaging.xsd"/>  
 </xsd:schema>  
</wsdl:types>

1. Pievienot Wsp atsauci

<wsp:Policy>  
 <wsp:PolicyReference URI="#Mtom.xml"/>  
</wsp:Policy>

<wsp:PolicyReference URI="#ws2007FederationNoSctMtom\_policy"/>

1. jāievieto taga <wsdl:binding> … </wsdl:binding> ietvaros un pirms tagam <soap12:binding>.

*Wsdl* koda piemērs ir parādīts zemāk:

|  |
| --- |
| <wsdl:binding name="ws2007FederationNoSctMtom" type="tns:ObjectService">  <wsp:Policy>  <wsp:PolicyReference URI="#Mtom.xml"/>  </wsp:Policy>   <wsp:PolicyReference URI="#ws2007FederationNoSctMtom\_policy"/>  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>  …  </wsdl:binding> |

Lai *maven* zinātu, kur un, kāds pirmkods jāuzģenerē, pievienojamajam servisu aprakstam ir jānorāda pom.xml datne, spraudņa “org.apache.cxf” konfigurēšanas ietvaros.

<wsdlOption>

<wsdl>${basedir}/src/main/resources/wsdls/**ACLService**.wsdl</wsdl>

<wsdlLocation>classpath:wsdls/**ACLService**.wsdl</wsdlLocation>

<extraargs>

<extraarg>-p</extraarg>

<extraarg>vid.ldk.**aclservice**</extraarg>

</extraargs>

</wsdlOption>

Paraugu var apskatīt demonstrācijas projekta *pom.xml* datnē.

Lai ieviestu servisu projektā, attiecīgais apraksts ir jānorāda *cxf.xml* datnē. Piemērām, servisam *ACLService* piekļūšanas apraksts izskatās šādi:

|  |
| --- |
| <jaxws:client id="as"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.ldk.aclservice.ACLService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ACLService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties> </jaxws:client> |

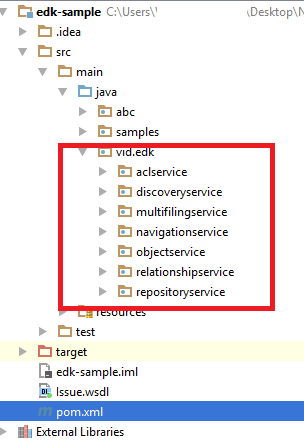
Pilnu paraugu ar visiem iekļautiem servisiem var apskatīt demonstrācijas projekta *cxf.xml* datnē. Pēc vienas vai visas *wsdl* datnes modificēšanas un *xml* datni atjaunošanās var automātiski noģenerēt Java klases ar *maven* komandu:

clean generate-sources

Rezultātā projekta mapē jāparādās jaunai mapei (skat. 7. attēlā, sarkanā rāmī), kas satur klases, interfeisus un fabrikas attiecīgā servisa piekļūšanai. Sakompilēt projektu var, izmantojot *maven* komandu:

compile

Gadījumā, ja EDK serverī kāda servisa apraksts tika mainīts, nepieciešams atjaunot attiecīgā servisa lokālo i datni, pilnveidot ar politikas fragmentiem un izpildīt koda ģenerēšanas *maven* komandu. Atsevišķos gadījumos pirms koda ģenerēšanas var dzēst mapi ar atjaunojama servisa kodu.



7.attēls. Mapes, kas satur klases, interfeisus un fabrikas servisa piekļūšanai

### EDK servera mijiedarbības demonstrācijas scenāriji

Lai demonstrētu darbu ar EDK serveri, projekts “edk-sample” implementē sekojošus scenārijus:

1. scenārijs. Informācija par repozitoriju un tā objektiem.

|  |  |  |  |
| --- | --- | --- | --- |
| N.p.k. | Operācija | Izpildes precizējumi | Gaidāmais rezultāts |
| 1 | getRepositories | – | Atgriezts repozitoriju saraksts |
| 2 | getRepositoryInfo | Ar EDK repozitorija identifikatoru | Atgriezta informācija par repozitoriju |
| 3 | getTypeChildren | Ar typeId=”cmis:document” | Atgriezta informācija par cmis:document |
| 4 | getTypeDefinition | Ar typeId=”edk:d:customDocument” | Atgriezta informācija par edk:d:customDocument |

2. scenārijs. Navigācijas metodes.

|  |  |  |  |
| --- | --- | --- | --- |
| N.p.k. | Operācija | Izpildes precizējumi | Gaidāmais rezultāts |
| 1 | getObjectByPath | Ar parametru *createMissingFolders*, izveidot trīs mapju hierarhijas līmeņus | Ir izveidoti hierarhijas elementi |
| 2 | createDocument | Otrajā mapju hierarhijas līmenī | Ir izveidots dokuments |
| 3 | createDocument | Trešajā mapju hierarhijas līmenī | Ir izveidots dokuments |
| 4 | getFolderTree | Mapju hierarhijas saknes līmenī | Atgriezts mapju saraksts |
| 5 | GetDescendants | Mapju hierarhijas saknes līmenī | Atgriezts mapju saraksts |
| 6 | getObjectByPath | Dokumentam otrajā mapju hierarhijas līmenī | Atgriezts dokuments |
| 7 | getFolderParent | Mapei otrajā hierarhijas līmenī | Atgriezta mape |
| 8 | query | Visiem objektiem ar tipu “ldk:document” | Atgriezta informācija par dokumentiem |

3. scenārijs. Kataloģizācija (multi-filing).

|  |  |  |  |
| --- | --- | --- | --- |
| N.p.k. | Operācija | Izpildes precizējumi | Gaidāmais rezultāts |
| 1 | createFolder | – | Izveidota pirmā mape |
| 2 | createDocument | Izveidot dokumentu pirmajā mapē | Izveidots dokuments |
| 3 | createFolder | Ārpus pirmās mapes | Izveidota otrā mape |
| 4 | addObjectToFolder | No pirmās mapes otrajā | Otrajā mapē tiek izveidota dokumenta instance |
| 5 | getObjectParents | Izveidotām dokumentam | Divas mapes |
| 6 | removeObjectFromFolder | Izņemt no otrās mapes | Tiek dzēsta dokumenta instance otrajā mapē |
| 7 | getObjectParents | Izveidotām dokumentam | Pirmajā mape |
| 8 | moveObject | No pirmās mapes otrajā | Dokuments ir pārvietots |
| 9 | getObjectParents | Izveidotām dokumentam | Otrajā mape |

4. scenārijs. Darbs ar objektiem.

|  |  |  |  |
| --- | --- | --- | --- |
| N.p.k. | Operācija | Izpildes precizējumi | Gaidāmais rezultāts |
| 1 | createFolder | Mapju hierarhijas saknes līmenī | Ir izveidota pirmā mape |
| 2 | createFolder | Pirmajā mapē | Ir izveidota otrā mape |
| 3 | createDocument | Otrajā mapē | Ir izveidots dokuments otrā mapē |
| 4 | createFolder | Pirmajā mapē | Ir izveidota trešā mape |
| 5 | createDocumentFromSource | Trešajā mapē izveidot otro dokumentu, izmantojot pirmo dokumentu kā avotu | Trešajā mapē ir izveidots otrais dokuments uz pirmā dokumenta bāzes |
| 6 | setContentStream | Otrajam dokumentam | Otrā dokumenta saturs aizvietots ar jaunu plūsmu |
| 7 | appendContenStream | Otrajam dokumentam | Otrā dokumenta saturs papildināts ar jaunu plūsmu |
| 8 | getContentStream | No otrā dokumenta | Jaunas plūsmas saturs |
| 9 | deleteContentStream | Otrajam dokumentam | Notika bināro datu dzēšana, changeToken ir ar jaunu vērtību |
| 10 | getContentStream | No otrā dokumenta | Atgriezta kļūda, jo nav bināro datu |
| 11 | deleteObject | Pirmajam dokumentam | Pirmā dokumenta dzēšana |
| 12 | deleteTree | Mapju hierarhijas saknē | Visi pakārtotie hierarhijas elementi tiek dzēsti |

5. scenārijs. Dokumentu atribūtu maiņa.

|  |  |  |  |
| --- | --- | --- | --- |
| N.p.k. | Operācija | Izpildes precizējumi | Gaidāmais rezultāts |
| 1 | createFolder | Mapju hierarhijas saknes līmenī | Ir izveidota pirmā mape |
| 2 | createFolder | Mapju hierarhijas saknes līmenī | Ir izveidota otrā mape |
| 3 | createDocument | Pirmajā mapē | Ir izveidots pirmais dokuments |
| 4 | createDocument | Otrajā mapē | Ir izveidots otrais dokuments |
| 5 | updateProperties | Pirmajam dokumentam izmanīt nosaukumu | Pirmajam dokumentam nosaukums ir izmainīts |
| 6 | getProperties | Pirmajam dokumentam | Atgriezts jauns nosaukums pirmajam dokumentam |
| 7 | bulkUpdateProperties | Abiem dokumentiem izmainīt nosaukumu | Abi dokumentu nosaukumi ir izmainīti |
| 8 | getProperties | Pirmajam dokumentam | Atgriezts jauns nosaukums pirmajam dokumentam |
| 9 | getProperties | Otrajam dokumentam | Otrajam dokumentam atgriezts tāds pats nosaukums kā pirmajam dokumentam |

6. scenārijs. Objektu sasaistes.

|  |  |  |  |
| --- | --- | --- | --- |
| N.p.k. | Operācija | Izpildes precizējumi | Gaidāmais rezultāts |
| 1 | createFolder | Mapju hierarhijas saknes līmenī | Ir izveidota pirmajā mape |
| 2 | createDocument | Iepriekš izveidotajā mapē | Ir izveidots pirmais dokuments |
| 3 | createDocument | Iepriekš izveidotajā mapē | Ir izveidots otrais dokuments |
| 4 | createRelationship | Starp izveidotājiem dokumentiem | Ir izveidota saite |
| 5 | getObjectRelationships | Pirmajai datnei | Ir atgriezts saites identifikators |

7. scenārijs. Darbs ar objektu tiesībām.

|  |  |  |  |
| --- | --- | --- | --- |
| N.p.k. | Operācija | Izpildes precizējumi | Gaidāmais rezultāts |
|  | Operācijas no 1. līdz 7. izpildīt ar “EDK.Administrator” lietotāju | | |
| 1 | createFolder | Mapju hierarhijas saknes līmenī | Ir izveidota mape |
| 2 | applyACL | Pielietot mapei, ar parametru addACEs un propagate.  Gruppa = “VIDISS\_EDK\_SENDERS”, tiesības = "edk:createRelationship" | Mapei ir pievienotas jaunas tiesības |
| 3 | createDocument | Izveidotajā mapē | Ir izveidots dokuments |
| 4 | getACL | Izveidotajam dokumentam | Ir atgriezts ACL saraksts |
| 5 | getAllowableActions | Izveidotajam dokumentam | Ir atgriezts atļauto darbību uzskaitījums |
| 6 | applyACL | Pielietot mapei, ar parametru removeACEs un propagate. Gruppa = “VIDISS\_EDK\_SENDERS”, tiesības: "edk:createRelationship" un "edk:createFolder" | Palika tikai tiesības lasīšanai |
| 7 | getAllowableActions | Izveidotajam dokumentam | Ir atgriezts atjaunots atļauto darbību uzskaitījums |
| 8 | Ar lietotāju “EDK.Sender” Izveidot jaunu savienojumu, izmantot savienojumu nākamajā operācijā. | | |
| 9 | createFolder | Pamēģināt izveidot jauno mapi (ņemot vērā, ka šādas tiesības tika izņemtas) | Kļūda (tiesību ierobežojumu dēļ). |

### Demonstrācijas projekta moduļu apraksts

Projekta struktūra ir attēlota 7.attēlā. Paketē *abc.edk.clien*t tiek izvietotas klases, kas paredzētas pieprasījuma struktūras sagatavošanai un servisa metodes izsaukšanai. Pakete *samples* satur demonstrācijas klases, kas uzbūvētas uz scenāriju pamata (skat. 2.2.3.sadaļu). Pakete *vid.edk* satur ar *Apache* CXF noģenerētās klases, kas realizē servisu izsaukšanu. Šīs paketes klases nav paredzētas manuālai modificēšanai. Pakete *resources/payload* satur failus, kurus projekts pārsuta uz EDK serveri. Pakete *resources/wsdls* satur *wsdl* datnes, kas bija lejupielādētās no EDK servera un pilnveidotas ar drošības politikas fragmentiem (skat. 2.2.2.sadaļu). Līdz ar to pakete *resources* satur sertifikātu un *Apache* CXF konfigurācijas failu *cxf.xml.* Zemāk tiek paskaidrotas konfigurācijas moduli.

#### Cxf.xml

Datne *cxf.xml* ir paredzēta *cxf* klienta parametru konfigurācijai. Zemāk tiek parādīts, kā konfigurēt ziņojuma izvadi, SSL sertifikātu krātuvi un Web servisu pievienošanu *Spring* kontekstam.

Koda piemēra fragments ir parādīts zemāk:

|  |
| --- |
| <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:cxf="http://cxf.apache.org/core"  xmlns:jaxws="http://cxf.apache.org/jaxws"  xmlns:soap="http://cxf.apache.org/bindings/soap"  xmlns:http="http://cxf.apache.org/transports/http/configuration"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:sec="http://cxf.apache.org/configuration/security"  xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"  xsi:schemaLocation=" http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd http://cxf.apache.org/transports/http/configuration http://cxf.apache.org/schemas/configuration/http-conf.xsd http://cxf.apache.org/configuration/security http://cxf.apache.org/schemas/configuration/security.xsd http://cxf.apache.org/bindings/soap http://cxf.apache.org/schemas/configuration/soap.xsd">    <!-- CXF klienta konfigurācija -->  <cxf:bus>  <cxf:features>  <cxf:logging />  </cxf:features>  </cxf:bus>   <bean id="default.sts-client" class="org.apache.cxf.ws.security.trust.STSClient">  <constructor-arg ref="cxf"/>  <property name="wsdlLocation" value="https://epak2.abcsoftware.lv/PFAS/Pfas.STS/v1-3/STS/Issue.svc?singleWsdl"/>  <property name="serviceName"  value="{http://schemas.microsoft.com/ws/2008/06/identity/securitytokenservice}SecurityTokenService"/>  <property name="endpointName"  value="{http://schemas.microsoft.com/ws/2008/06/identity/securitytokenservice}CertificateWSTrustBinding\_IWSTrust13Sync"/>  <property name="properties">  <map>  <entry key="security.signature.properties" value="client\_sign.properties"/>  <entry key="security.callback-handler">  <ref bean="keystorePasswordCallback"/>  </entry>  <entry key="security.sts.applies-to" value="https://epak2.abcsoftware.lv/EDK2"></entry>  </map>  </property>  </bean>   <jaxws:client id="os"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.edk.objectservice.ObjectService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ObjectService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties>  </jaxws:client>   <jaxws:client id="ns"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.edk.navigationservice.NavigationService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/NavigationService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties>  </jaxws:client>   <jaxws:client id="rs"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.edk.repositoryservice.RepositoryService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/RepositoryService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties>  </jaxws:client>   <jaxws:client id="ds"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.edk.discoveryservice.DiscoveryService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/DiscoveryService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties>  </jaxws:client>   <jaxws:client id="ms"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.edk.multifilingservice.MultiFilingService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/DiscoveryService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties>  </jaxws:client>   <jaxws:client id="rss"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.edk.relationshipservice.RelationshipService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/RelationshipService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties>  </jaxws:client>    <jaxws:client id="as"  name="{http://ivis.eps.gov.lv/ISS/EDKService/v1-0}ws2007FederationNoSctMtom"  serviceClass="vid.edk.aclservice.ACLService"  address="https://epak2.abcsoftware.lv/EDK/v2-0/Logic/ACLService/ws2007FederationNoSctMtom"  createdFromAPI="true">  <jaxws:properties>  <entry key="ws-security.sts.client" value-ref="default.sts-client"/>  </jaxws:properties>  </jaxws:client>   <bean id="keystorePasswordCallback"  class="abc.edk.client.KeystorePasswordCallback">  <property name="keyPassword" value="123"/>  </bean>  </beans> |

### Piemēri

Projekta paketē *sample* tika izvietoti visi paraugi darbam ar EDK serveri. Zemāk parādīti daži piemēri. Katrā piemērā pirmkodu veido divas daļas: augstākā līmeņa izsaukšanas piemērs (pirmkods tiek ņemts no paketes *sample* klasēm) un pieprasījumu datu struktūras sagatavošanas piemērs (tika ņemts no paketes abc.edk.client).

Ņemot vērā, ka visu servisu klienta konstruktora kods ir līdzīgs, un atšķiras tikai ar servisa nosaukumu, turpmāk konstruktora kods netiks izklāstīts. Zemāk tiek parādīts konstruktora piemērs objektu servisam.

|  |
| --- |
| package abc.edk.client;  import org.apache.cxf.Bus; import org.apache.cxf.message.Message; import org.apache.cxf.ws.security.trust.STSClient; import vid.edk.objectservice.\*; import java.util.Map;  public class ObjectServiceClient {  private ObjectService os;  public ObjectServiceClient() {   ObjectService\_Service service = new ObjectService\_Service();  os = service.getWs2007FederationNoSctMtom(); }  …   //...  } |

#### Mapes izveidošanas vai pārbaudes piemērs (CreateFolder)

Izmantojot metodes palīdzību, var izveidot mapes. Ja ceļā (parametra), kuru padod metode, mapēs neeksistē EDK, tad metode pati izveidos šo mapi un atgriezīs izveidoto vai eksistējošu gala mapes EDK identifikatoru.

Priekšnosacījums – lai izveidotu operāciju, vajag izveidot klases *ObjectServiceClient* objektu.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.objectservice.CmisContentStreamType;  ...  SpringBusFactory bf = new SpringBusFactory(); Bus bus = bf.createBus("cxf.xml"); BusFactory.setDefaultBus(bus);  ObjectServiceClient osc = new ObjectServiceClient();  String folderId = osc.createFolder("URN:IVIS:100266:EDK-0000001  ","URN:IVIS:100266:FLD-0000962","MyFolder");  System.out.println("New folder id is " + folderId); |

Klases *ObjectServiceClient* konstruktoru skaties 2.2.5.sadaļā.

Zemāk parādīts datu struktūras sagatavošanas un servisa izsaukšanas kods (klases *ObjectServiceClient.java* fragmenti).

|  |
| --- |
| public String createFolder(String repositoryId, String parentFolderId, String newFolderName) {   // typeId property  CmisPropertyId propTypeId = new CmisPropertyId();  propTypeId.setPropertyDefinitionId(EnumPropertiesBase.CMIS\_OBJECT\_TYPE\_ID.value());  propTypeId.setQueryName(EnumPropertiesBase.CMIS\_OBJECT\_TYPE\_ID.value());  propTypeId.getValue().add(EnumBaseObjectTypeIds.CMIS\_FOLDER.value());   // name property  CmisPropertyString propName = new CmisPropertyString();  propName.setPropertyDefinitionId(EnumPropertiesBase.CMIS\_NAME.value());  propName.setQueryName(EnumPropertiesBase.CMIS\_NAME.value());  propName.getValue().add(newFolderName);   // building properties list  CmisPropertiesType properties = new CmisPropertiesType();  properties.getProperty().add(propTypeId);  properties.getProperty().add(propName);   Holder<String> objectId = new Holder<>("");   try {  os.createFolder(repositoryId, properties, parentFolderId, null, null, null, null, objectId);  } catch (ObjectServiceCreateFolderCmisFaultFaultFaultMessage objectServiceCreateFolderCmisFaultFaultFaultMessage) {  objectServiceCreateFolderCmisFaultFaultFaultMessage.printStackTrace();  }   return objectId.value; }  public void moveObject(String repositoryId, Holder<String> objectId, String targetFolderId, String sourceFolderId) {   try {  os.moveObject(repositoryId, objectId, targetFolderId, sourceFolderId, null);  } catch (ObjectServiceMoveObjectCmisFaultFaultFaultMessage objectServiceMoveObjectCmisFaultFaultFaultMessage) {  objectServiceMoveObjectCmisFaultFaultFaultMessage.printStackTrace();  } }  public CmisAllowableActionsType getAllowableActions(String repositoryId, String objectId){   CmisAllowableActionsType cmisAllowableActions = null;   try {  cmisAllowableActions = os.getAllowableActions(repositoryId, objectId, null);  } catch (ObjectServiceGetAllowableActionsCmisFaultFaultFaultMessage objectServiceGetAllowableActionsCmisFaultFaultFaultMessage) {  objectServiceGetAllowableActionsCmisFaultFaultFaultMessage.printStackTrace();  }   return cmisAllowableActions; } |

#### Datnes izveidošana (createDocument)

Izmantojot metodi *createDocument,* tiek padots baitu masīvs, mapes identifikators un datnes nosaukums.

Priekšnosacījums – lai veidotu operāciju, vajag izveidot klases *ObjectServiceClient* objektu.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.objectservice.CmisContentStreamType;  …  SpringBusFactory bf = new SpringBusFactory(); Bus bus = bf.createBus("cxf.xml"); BusFactory.setDefaultBus(bus);  ObjectServiceClient osc = new ObjectServiceClient();  String sourceFileId = osc.createDocument(" URN:IVIS:100266:EDK-0000001  ", "URN:IVIS:100266:FLD-0000962",  "src/main/resources/payload/importantfile.gif",  "image/gif"); System.out.println("Object Id: " + sourceFileId); |

Klases *ObjectServiceClient* konstruktoru skaties 2.2.5.sadaļā.

Zemāk parādīts datu struktūras sagatavošanas un servisa izsaukšanas kods (klases *ObjectServiceClient.java* fragmenti).

|  |
| --- |
| public String createDocument(String repositoryId, String folderId, String localFilename, String username, String mimeType) {   // typeId property  CmisPropertyId propTypeId = new CmisPropertyId();  propTypeId.setPropertyDefinitionId(EnumPropertiesBase.CMIS\_OBJECT\_TYPE\_ID.value());  propTypeId.setQueryName(EnumPropertiesBase.CMIS\_OBJECT\_TYPE\_ID.value());  propTypeId.getValue().add("edk:d:customDocument");   // name property  String filenamewhitoutpath = Paths.get(localFilename).getFileName().toString();  CmisPropertyString propName = new CmisPropertyString();  propName.setPropertyDefinitionId(EnumPropertiesBase.CMIS\_NAME.value());  propName.setQueryName(EnumPropertiesBase.CMIS\_NAME.value());  propName.getValue().add(filenamewhitoutpath);  // // owner property  CmisPropertyString propOwner = new CmisPropertyString();  propOwner.setPropertyDefinitionId("edk:owner");  propOwner.setQueryName("edk:owner");  propOwner.getValue().add(username);   CmisPropertyString propretentionGroup = new CmisPropertyString();  propretentionGroup.setPropertyDefinitionId("edk:retentionGroup");  propretentionGroup.setQueryName("edk:retentionGroup");  propretentionGroup.getValue().add("nenoteikta");   // building properties list  CmisPropertiesType properties = new CmisPropertiesType();  properties.getProperty().add(propTypeId);  properties.getProperty().add(propName);  properties.getProperty().add(propretentionGroup);  properties.getProperty().add(propOwner);   // streaming file content  CmisContentStreamType contentStream = new CmisContentStreamType();  contentStream.setFilename(filenamewhitoutpath);  contentStream.setMimeType(mimeType);  FileDataSource fileDS = new FileDataSource(new File(localFilename));  DataHandler dataHandler = new DataHandler(fileDS);  contentStream.setStream(dataHandler);   Holder<String> objectId = new Holder<>("");   try {  os.createDocument(repositoryId, properties, folderId, contentStream, EnumVersioningState.MAJOR, null, null, null, null, objectId);  } catch (ObjectServiceCreateDocumentCmisFaultFaultFaultMessage objectServiceCreateDocumentCmisFaultFaultFaultMessage) {  objectServiceCreateDocumentCmisFaultFaultFaultMessage.printStackTrace();  }   return objectId.value;  }  public String createRelationship(String repositoryId, String sourceId, String targetId, String relationshipName, CmisAccessControlListType addACEs, CmisAccessControlListType removeACEs) {   // typeId property  CmisPropertyId propTypeId = new CmisPropertyId();  propTypeId.setPropertyDefinitionId(EnumPropertiesBase.CMIS\_OBJECT\_TYPE\_ID.value());  propTypeId.setQueryName(EnumPropertiesBase.CMIS\_OBJECT\_TYPE\_ID.value());  propTypeId.getValue().add(EnumBaseObjectTypeIds.CMIS\_RELATIONSHIP.value());   // sourceId property  CmisPropertyId propSourceId = new CmisPropertyId();  propSourceId.setPropertyDefinitionId(EnumPropertiesRelationship.CMIS\_SOURCE\_ID.value());  propSourceId.setQueryName(EnumPropertiesRelationship.CMIS\_SOURCE\_ID.value());  propSourceId.getValue().add(sourceId);   // targetId property  CmisPropertyId propTargetId = new CmisPropertyId();  propTargetId.setPropertyDefinitionId(EnumPropertiesRelationship.CMIS\_TARGET\_ID.value());  propTargetId.setQueryName(EnumPropertiesRelationship.CMIS\_TARGET\_ID.value());  propTargetId.getValue().add(targetId);   // name property  CmisPropertyString propName = new CmisPropertyString();  propName.setPropertyDefinitionId(EnumPropertiesBase.CMIS\_NAME.value());  propName.setQueryName(EnumPropertiesBase.CMIS\_NAME.value());  propName.getValue().add(relationshipName);   // building properties list  CmisPropertiesType properties = new CmisPropertiesType();  properties.getProperty().add(propTypeId);  properties.getProperty().add(propSourceId);  properties.getProperty().add(propTargetId);  properties.getProperty().add(propName);   Holder<String> objectIdHolder = new Holder<>(null);   try {  os.createRelationship(repositoryId, properties, null, addACEs, removeACEs, null, objectIdHolder);  } catch (ObjectServiceCreateRelationshipCmisFaultFaultFaultMessage objectServiceCreateRelationshipCmisFaultFaultFaultMessage) {  objectServiceCreateRelationshipCmisFaultFaultFaultMessage.printStackTrace();  }   return objectIdHolder.value; } |

#### Objekta dzēšana (deleteObject)

Izmantojot metodi *deleteObject,* tiks dzēsts ar identifikatoru norādīts objects.

Priekšnosacījums – lai veidotu operāciju, vajag izveidot klases *ObjectServiceClient* objektu.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; ...  ...  SpringBusFactory bf = new SpringBusFactory(); Bus bus = bf.createBus("cxf.xml"); BusFactory.setDefaultBus(bus);  ObjectServiceClient osc = new ObjectServiceClient();  osc.deleteObject("URN:IVIS:100266:EDK-0000001", "URN:IVIS:100266:DOC-0001283-V1.0"); |

Klases *ObjectServiceClient* konstruktoru skaties 2.2.5.sadaļā.

Zemāk parādīts datu struktūras sagatavošanas un servisa izsaukšanas kods (klases *ObjectServiceClient.java* fragments).

|  |
| --- |
| private final static boolean ALL\_VERSIONS\_\_TRUE = true;  private final static javax.xml.ws.Holder<CmisExtensionType> EXTENSION\_\_NULL = null;  ...  public void deleteObject(String repositoryId, String objectId) {  try {  os.deleteObject(repositoryId, objectId, ALL\_VERSIONS\_\_TRUE, EXTENSION\_\_NULL);  } catch (ObjectServiceDeleteObjectCmisFaultFaultFaultMessage objectServiceDeleteObjectCmisFaultFaultFaultMessage) {  objectServiceDeleteObjectCmisFaultFaultFaultMessage.printStackTrace();  } } |

#### Mapes dzēšana (deleteTree)

Izmantojot metodi *deleteTree,* tiks dzēstas visas datnes no mapes, un pēc tam dzēsta arī pati mape.

Priekšnosacījums – lai veidotu operāciju, vajag izveidot klases *ObjectServiceClient* objektu.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.objectservice.CmisContentStreamType;  ...  SpringBusFactory bf = new SpringBusFactory(); Bus bus = bf.createBus("cxf.xml"); BusFactory.setDefaultBus(bus);  ObjectServiceClient osc = new ObjectServiceClient();  vid.ldk.objectservice.DeleteTreeResponse.FailedToDelete deleteTreeResponse = osc.deleteTree("URN:IVIS:100266:EDK-0000001", "URN:IVIS:100266:FLD-0001127"); |

Klases *ObjectServiceClient* konstruktoru skaties 2.2.5.sadaļā.

Zemāk parādīts datu struktūras sagatavošanas un servisa izsaukšanas kods.

|  |
| --- |
| public vid.edk.objectservice.DeleteTreeResponse.FailedToDelete deleteTree(String repositoryId, String folderId) {   vid.edk.objectservice.DeleteTreeResponse.FailedToDelete response = null;   try {  response = os.deleteTree(repositoryId, folderId, true, EnumUnfileObject.DELETE, true, null);  } catch (ObjectServiceDeleteTreeCmisFaultFaultFaultMessage objectServiceDeleteTreeCmisFaultFaultFaultMessage) {  objectServiceDeleteTreeCmisFaultFaultFaultMessage.printStackTrace();  }   return response; } |

#### Izgūt mapes struktūru (getFolderTree)

Izmantojot metodi *GetFolderTree,* tiks izgūtas mapes struktūra, kura tika norādīta parametrā.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| package abc.edk.client;  import vid.edk.navigationservice.\*;  import java.math.BigInteger; import java.util.List;   public class NavigationServiceClient {   private static String FILTER\_NOT\_SET = "Not set";  private NavigationService ns;   public NavigationServiceClient() {   NavigationService\_Service navigationService\_service = new NavigationService\_Service();  ns = navigationService\_service.getWs2007FederationNoSctMtom();  }   public String getFolderParent(String repositoryId, String folderId){   String someRes = null;   try {  vid.edk.navigationservice.CmisObjectType resultObject = ns.getFolderParent(repositoryId, folderId, null, null);  vid.edk.navigationservice.CmisPropertiesType resProperties = resultObject.getProperties();   for (vid.edk.navigationservice.CmisProperty cmisResProperty : resProperties.getProperty()) {  if (cmisResProperty instanceof vid.edk.navigationservice.CmisPropertyId) {  if (cmisResProperty.getPropertyDefinitionId().equals("cmis:parentId")) {  List<String> valueList = ((CmisPropertyId) cmisResProperty).getValue();  if (valueList.size() > 0) {  someRes = valueList.get(0);  break;  }  }  //vid.edk.navigationservice.CmisPropertyId propertyId = (vid.edk.navigationservice.CmisPropertyId) resProperties.getProperty().get(0);  //someRes = propertyId.getValue().get(0);  }  }   } catch (NavigationServiceGetFolderParentCmisFaultFaultFaultMessage e) {  e.printStackTrace();  }   return someRes;  }    public List<CmisObjectParentsType> getObjectParents(String repositoryId, String objectId) {  List<CmisObjectParentsType> objectParents = null;  try {  objectParents = ns.getObjectParents(repositoryId, objectId, null, false, EnumIncludeRelationships.NONE, null, false, null);  } catch (NavigationServiceGetObjectParentsCmisFaultFaultFaultMessage navigationServiceGetObjectParentsCmisFaultFaultFaultMessage) {  navigationServiceGetObjectParentsCmisFaultFaultFaultMessage.printStackTrace();  }   return objectParents;  }    public List<CmisObjectInFolderContainerType> getFolderTree(String repositoryId, String folderId, BigInteger depth) {   List<CmisObjectInFolderContainerType> tree = null;   try {  tree = ns.getFolderTree(repositoryId, folderId, depth, FILTER\_NOT\_SET, Boolean.FALSE, EnumIncludeRelationships.NONE, "", Boolean.FALSE, null);  } catch (NavigationServiceGetFolderTreeCmisFaultFaultFaultMessage navigationServiceGetFolderTreeCmisFaultFaultFaultMessage) {  navigationServiceGetFolderTreeCmisFaultFaultFaultMessage.printStackTrace();  }   return tree;   }   public List<CmisObjectInFolderContainerType> getDescendants(String repositoryId, String folderId, BigInteger depth) {   List<CmisObjectInFolderContainerType> descendants = null;   try {  descendants = ns.getDescendants(repositoryId, folderId, depth, FILTER\_NOT\_SET, Boolean.FALSE, EnumIncludeRelationships.NONE, "", Boolean.FALSE, null);  } catch (NavigationServiceGetDescendantsCmisFaultFaultFaultMessage navigationServiceGetDescendantsCmisFaultFaultFaultMessage) {  navigationServiceGetDescendantsCmisFaultFaultFaultMessage.printStackTrace();  }   return descendants;  }  } |

#### Izgūt datnes saturu (getContentStream)

Izmantojot metodi *getContentStream,* tiks izgūts datnes saturs.

Priekšnosacījums – lai veidotu operāciju, vajag izveidot klases *ObjectServiceClient* objektu.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.objectservice.CmisContentStreamType;  ...  ...  SpringBusFactory bf = new SpringBusFactory(); Bus bus = bf.createBus("cxf.xml"); BusFactory.setDefaultBus(bus);  ObjectServiceClient osc = new ObjectServiceClient();  CmisContentStreamType contentStream = osc.getContentStream("URN:IVIS:100266:EDK-0000001",  " URN:IVIS:100266:DOC-0001283-V1.0", BigInteger.valueOf(1), BigInteger.valueOf(1000));  System.out.println("contentStream.getFilename(): " + contentStream.getFilename()); System.out.println("contentStream.getLength() : " + contentStream.getLength()); |

Klases ObjectServiceClient konstruktoru skaties 2.2.5. sadaļā.

Zemāk parādīts datu struktūras sagatavošanas un servisa izsaukšanas kods.

|  |
| --- |
| public CmisContentStreamType getContentStream(String repositoryId, String objectId, BigInteger offset, BigInteger length) {   try {  return os.getContentStream(repositoryId, objectId, null, offset, length, null);  } catch (ObjectServiceGetContentStreamCmisFaultFaultFaultMessage objectServiceGetContentStreamCmisFaultFaultFaultMessage) {  objectServiceGetContentStreamCmisFaultFaultFaultMessage.printStackTrace();  }   return new CmisContentStreamType(); } |

#### Izgūt informāciju par repozitoriju (getRepositoryInfo)

Piemērā attēlota repozitorija informācijas izgūšana.

Priekšnosacījums – lai veidotu operāciju, vajag izveidot klases *RepositoryServiceClient* objektu.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| package samples;  import abc.edk.client.RepositoryServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.repositoryservice.CmisRepositoryEntryType; import vid.edk.repositoryservice.CmisRepositoryInfoType; import vid.edk.repositoryservice.CmisTypeDefinitionType;  import java.util.List;  public class RepositoryServiceSample {   private static final String OBJECT\_TYPEID\_CMIS\_DOCUMENT = "cmis:document";  private static final String OBJECT\_TYPEID\_LDK\_DOCUMENT = "edk:d:customDocument";   public static void main(String[] args) {   System.out.println("Discovering information about a repository, including information about an object-types deﬁned for a repository\n");   SpringBusFactory bf = new SpringBusFactory();  Bus bus = bf.createBus("cxf.xml");  BusFactory.setDefaultBus(bus);   RepositoryServiceClient rsc = new RepositoryServiceClient();   // (1)  System.out.println("Calling method getRepositories");   List<CmisRepositoryEntryType> repositoriesList = rsc.getRepositories();   System.out.println("Result: ");  for (CmisRepositoryEntryType elem : repositoriesList) {  System.out.println("repositoryId = " + elem.getRepositoryId());  System.out.println("repositoryName = " + elem.getRepositoryName());  }  System.out.println("==================================================================================\n\n");    // (2)  System.out.println("Calling method getRepositoryInfo\n");  for (CmisRepositoryEntryType elem : repositoriesList) {  String repositoryId = elem.getRepositoryId();  System.out.println("For repositoryId = " + repositoryId + " ...");  System.out.println("Result: ");  CmisRepositoryInfoType repositoryInfo = rsc.getRepositoryInfo(repositoryId);  System.out.println("RepositoryName = " + repositoryInfo.getRepositoryName());  System.out.println("RepositoryDescription = " + repositoryInfo.getRepositoryDescription());  System.out.println("vendorName = " + repositoryInfo.getVendorName());  System.out.println("productName = " + repositoryInfo.getProductName());  System.out.println("productVersion = " + repositoryInfo.getProductVersion());  System.out.println("rootFolderId = " + repositoryInfo.getRootFolderId());  }  System.out.println("==================================================================================\n\n");    // (3)  System.out.println("Calling method getTypeChildren\n");  for (CmisRepositoryEntryType elem : repositoriesList) {  String repositoryId = elem.getRepositoryId();  System.out.println("For repositoryId = " + repositoryId + " AND object-type = " + OBJECT\_TYPEID\_CMIS\_DOCUMENT );  System.out.println("Result: ");  vid.edk.repositoryservice.CmisTypeDefinitionListType typechildren = rsc.getTypeChildren(repositoryId, OBJECT\_TYPEID\_CMIS\_DOCUMENT, Boolean.TRUE);  System.out.println("total obtained types: " + typechildren.getTypes().size());  for (CmisTypeDefinitionType dt:typechildren.getTypes()) {  System.out.println("Type id = " + dt.getId());  System.out.println(" DisplayName = " + dt.getDisplayName());  System.out.println(" Description = " + dt.getDescription());  }  }  System.out.println("==================================================================================\n\n");     // (4)  System.out.println("Calling method getTypeDefinition\n");  for (CmisRepositoryEntryType elem : repositoriesList) {  String repositoryId = elem.getRepositoryId();  System.out.println("when repositoryId = " + repositoryId + " AND object-type = " + OBJECT\_TYPEID\_LDK\_DOCUMENT);  System.out.println("Result: ");  vid.edk.repositoryservice.CmisTypeDefinitionType typeDefinitionType = rsc.getTypeDefinition(repositoryId, OBJECT\_TYPEID\_LDK\_DOCUMENT);  System.out.println("id = " + typeDefinitionType.getId());  System.out.println("LocalName = " + typeDefinitionType.getLocalName());  System.out.println("DisplayName = " + typeDefinitionType.getDisplayName());  System.out.println("Description = " + typeDefinitionType.getDescription());  System.out.println("baseId = " + typeDefinitionType.getBaseId().value());  }  System.out.println("==================================================================================\n\n");   } } |

Klases *RepositoryServiceClient* konstruktoru skaties zemāk.

Zemāk arī parādīts datu struktūras sagatavošanas un servisa izsaukšanas kods.

|  |
| --- |
| package abc.edk.client;  import org.apache.cxf.Bus; import vid.edk.repositoryservice.\*;  import java.math.BigInteger; import java.util.List;  public class RepositoryServiceClient {   private RepositoryService rs;    public RepositoryServiceClient() {   RepositoryService\_Service repositoryService\_service = new RepositoryService\_Service();  rs = repositoryService\_service.getWs2007FederationNoSctMtom();  }   public List<CmisRepositoryEntryType> getRepositories(){   List<CmisRepositoryEntryType> resultList = null;   try {  resultList = rs.getRepositories(null);  } catch (RepositoryServiceGetRepositoriesCmisFaultFaultFaultMessage repositoryServiceGetRepositoriesCmisFaultFaultFaultMessage) {  repositoryServiceGetRepositoriesCmisFaultFaultFaultMessage.printStackTrace();  }   return resultList;  }   public CmisRepositoryInfoType getRepositoryInfo(String repositoryId) {   CmisRepositoryInfoType repositoryInfo = null;   try {  repositoryInfo = rs.getRepositoryInfo(repositoryId, null);  } catch (RepositoryServiceGetRepositoryInfoCmisFaultFaultFaultMessage repositoryServiceGetRepositoryInfoCmisFaultFaultFaultMessage) {  repositoryServiceGetRepositoryInfoCmisFaultFaultFaultMessage.printStackTrace();  }   return repositoryInfo;  }   public vid.edk.repositoryservice.CmisTypeDefinitionListType getTypeChildren(String repositoryId, String typeId, java.lang.Boolean includePropertyDefinitions) {   vid.edk.repositoryservice.CmisTypeDefinitionListType cmisTypeDefinitionListType = null;   try {  cmisTypeDefinitionListType = rs.getTypeChildren(repositoryId, typeId, includePropertyDefinitions, BigInteger.valueOf(0), BigInteger.valueOf(0), null);  } catch (RepositoryServiceGetTypeChildrenCmisFaultFaultFaultMessage repositoryServiceGetTypeChildrenCmisFaultFaultFaultMessage) {  repositoryServiceGetTypeChildrenCmisFaultFaultFaultMessage.printStackTrace();  }   return cmisTypeDefinitionListType;  }   public vid.edk.repositoryservice.CmisTypeDefinitionType getTypeDefinition(String repositoryId, String typeId){   vid.edk.repositoryservice.CmisTypeDefinitionType cmisTypeDefinitionType = null;   try {  cmisTypeDefinitionType = rs.getTypeDefinition(repositoryId, typeId, null);  } catch (RepositoryServiceGetTypeDefinitionCmisFaultFaultFaultMessage repositoryServiceGetTypeDefinitionCmisFaultFaultFaultMessage) {  repositoryServiceGetTypeDefinitionCmisFaultFaultFaultMessage.printStackTrace();  }   return cmisTypeDefinitionType;  }  } |

#### Vaicājumu izmantošana informācijas meklēšanai (DiscoveryServices)

Piemērā attēlota vaicājumu izveide un izmantošana dokumentu saraksta saņemšanai.

Priekšnosacījums – lai veidotu operāciju, vajag izveidot klases *DiscoveryServiceClient* objektu.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.discoveryservice.QueryResponse;  ...  SpringBusFactory bf = new SpringBusFactory(); Bus bus = bf.createBus("cxf.xml"); BusFactory.setDefaultBus(bus);  abc.ldk.client.DiscoveryServiceClient dsc =  new abc.ldk.client.DiscoveryServiceClient();  // (8) System.out.println("Calling method query."); String queryStatement = "SELECT \* FROM edk:d:customDocument"; QueryResponse queryResponse = dsc.query(REPOSITORY\_ID,  queryStatement); System.out.println("query result items count: " +  queryResponse.getObjects().getNumItems().toString()); |

Moduļa *DiscoveryServiceClient.java* koda piemērs ir parādīts zemāk:

|  |
| --- |
| package abc.edk.client;  import org.apache.cxf.Bus; import vid.edk.discoveryservice.\*;  public class DiscoveryServiceClient {   private DiscoveryService ds;   public DiscoveryServiceClient() {   DiscoveryService\_Service discoveryService\_service = new DiscoveryService\_Service();  ds = discoveryService\_service.getWs2007FederationNoSctMtom();  }   public QueryResponse query(String repositoryId, String queryStatement){   QueryResponse queryResponse = null;   Query query = new Query();  query.setRepositoryId(repositoryId);  query.setStatement(queryStatement);   try {  queryResponse = ds.query(query);  } catch (DiscoveryServiceQueryCmisFaultFaultFaultMessage discoveryServiceQueryCmisFaultFaultFaultMessage) {  discoveryServiceQueryCmisFaultFaultFaultMessage.printStackTrace();  }   return queryResponse;  } } |

#### Navigācijas servisu izmantošana (NavigationServices)

Piemērā tiek attēlota EDK navigācijas servisu operācijas izsaukšana un datu struktūru sagatavošana.

Koda piemērs ir parādīts zemāk:

|  |
| --- |
| package abc.edk.client;  import vid.edk.navigationservice.\*;  import java.math.BigInteger; import java.util.List;  public class NavigationServiceClient {   private static String FILTER\_NOT\_SET = "Not set";  private NavigationService ns;   public NavigationServiceClient() {   NavigationService\_Service navigationService\_service = new NavigationService\_Service();  ns = navigationService\_service.getWs2007FederationNoSctMtom();  }   public String getFolderParent(String repositoryId, String folderId){   String someRes = null;   try {  vid.edk.navigationservice.CmisObjectType resultObject = ns.getFolderParent(repositoryId, folderId, null, null);  vid.edk.navigationservice.CmisPropertiesType resProperties = resultObject.getProperties();   for (vid.edk.navigationservice.CmisProperty cmisResProperty : resProperties.getProperty()) {  if (cmisResProperty instanceof vid.edk.navigationservice.CmisPropertyId) {  if (cmisResProperty.getPropertyDefinitionId().equals("cmis:parentId")) {  List<String> valueList = ((CmisPropertyId) cmisResProperty).getValue();  if (valueList.size() > 0) {  someRes = valueList.get(0);  break;  }  }  }  }   } catch (NavigationServiceGetFolderParentCmisFaultFaultFaultMessage e) {  e.printStackTrace();  }   return someRes;  }    public List<CmisObjectParentsType> getObjectParents(String repositoryId, String objectId) {  List<CmisObjectParentsType> objectParents = null;  try {  objectParents = ns.getObjectParents(repositoryId, objectId, null, false, EnumIncludeRelationships.NONE, null, false, null);  } catch (NavigationServiceGetObjectParentsCmisFaultFaultFaultMessage navigationServiceGetObjectParentsCmisFaultFaultFaultMessage) {  navigationServiceGetObjectParentsCmisFaultFaultFaultMessage.printStackTrace();  }   return objectParents;  }    public List<CmisObjectInFolderContainerType> getFolderTree(String repositoryId, String folderId, BigInteger depth) {   List<CmisObjectInFolderContainerType> tree = null;   try {  tree = ns.getFolderTree(repositoryId, folderId, depth, FILTER\_NOT\_SET, Boolean.FALSE, EnumIncludeRelationships.NONE, "", Boolean.FALSE, null);  } catch (NavigationServiceGetFolderTreeCmisFaultFaultFaultMessage navigationServiceGetFolderTreeCmisFaultFaultFaultMessage) {  navigationServiceGetFolderTreeCmisFaultFaultFaultMessage.printStackTrace();  }   return tree;   }   public List<CmisObjectInFolderContainerType> getDescendants(String repositoryId, String folderId, BigInteger depth) {   List<CmisObjectInFolderContainerType> descendants = null;   try {  descendants = ns.getDescendants(repositoryId, folderId, depth, FILTER\_NOT\_SET, Boolean.FALSE, EnumIncludeRelationships.NONE, "", Boolean.FALSE, null);  } catch (NavigationServiceGetDescendantsCmisFaultFaultFaultMessage navigationServiceGetDescendantsCmisFaultFaultFaultMessage) {  navigationServiceGetDescendantsCmisFaultFaultFaultMessage.printStackTrace();  }   return descendants;  }  } |

#### Kataloģizācijas izmantošana (Multi-filing)

Piemērā tiek attēlota EDK kataloģizācijas paņēmienu demonstrācija, kas izmanto *MultiﬁlingService*, *NavigationService* un *ObjectService* servisu operācijas:

* *createFolder* (metodes implementāciju skat. 2.2.5.1.paragrāfā);
* *createDocument* (metodes implementāciju skat. 2.2.5.2. paragrāfā);
* *addObjectToFolder* (metodes implementāciju skat. šī paragrāfa beigās);
* *getObjectParents* (metodes implementāciju skat. 2.2.5.8. paragrāfā);
* *removeObjectFromFolder* (metodes implementāciju skat. šī paragrāfa beigās);
* *moveObject* (metodes implementāciju skat. 2.2.5.1. paragrāfā).

Piemērs atbilst 3. scenārija darbību secībai (skat.2.2.3. sadaļā). Koda piemērs ir parādīts zemāk:

|  |
| --- |
| package samples;  import abc.edk.client.MultiFilingServiceClient; import abc.edk.client.NavigationServiceClient; import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.navigationservice.CmisObjectParentsType; import vid.edk.navigationservice.CmisPropertyId;  import javax.xml.ws.Holder; import java.util.List;  public class MultiFilingServiceSample {   private static final String REPOSITORY\_ID = "URN:IVIS:100266:EDK-0000001";  private static final String REPO\_PATH = "/Samples";  private static final String ROOT\_FOLDER\_NAME = "/Samples/";  private static final String LOCAL\_RES\_FOLDER = "src/main/resources/payload/";  private static final String SOURCE\_FOLDER\_NAME = "SourceFolder";  private static final String TARGET\_FOLDER\_NAME = "TargetFolder";  private static final String FILE\_NAME = "SampleFragment3";  private static String ROOT\_FOLDER\_ID = "";   public static void main(String[] args) {   System.out.println("File/un-ﬁle objects into/from folders (if the repository supports the multiﬁling or unﬁling optional capabilities)");   SpringBusFactory bf = new SpringBusFactory();  Bus bus = bf.createBus("cxf.xml");  BusFactory.setDefaultBus(bus);   MultiFilingServiceClient msс = new MultiFilingServiceClient();  ObjectServiceClient osc = new ObjectServiceClient();  NavigationServiceClient nsc = new NavigationServiceClient();   osc.deleteTree(REPOSITORY\_ID, osc.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true));  ROOT\_FOLDER\_ID = osc.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true);   // (1)  System.out.println("Calling method createFolder. Params: sourceFolderName=" + SOURCE\_FOLDER\_NAME + ", parentFolderId = " + ROOT\_FOLDER\_ID);  String sourceFolderId = osc.createFolder(REPOSITORY\_ID, ROOT\_FOLDER\_ID, SOURCE\_FOLDER\_NAME);  System.out.println("Object Id for path " + ROOT\_FOLDER\_NAME + SOURCE\_FOLDER\_NAME + " : " + sourceFolderId);  System.out.println("==================================================================================\n\n");    // (2)  System.out.println("Calling method createDocument in folder with id = " + sourceFolderId);  String fileId = osc.createDocument(REPOSITORY\_ID, sourceFolderId, LOCAL\_RES\_FOLDER + FILE\_NAME, "scienceNewsUser", "text/plain");  System.out.println("Object Id for path " + ROOT\_FOLDER\_NAME + SOURCE\_FOLDER\_NAME + "/" + FILE\_NAME + " : " + fileId);  System.out.println("==================================================================================\n\n");    // (3)  System.out.println("Calling method createFolder. Params: sourceFolderName=" + TARGET\_FOLDER\_NAME + ", parentFolderId = " + ROOT\_FOLDER\_ID);  String targetFolderId = osc.createFolder(REPOSITORY\_ID, ROOT\_FOLDER\_ID, TARGET\_FOLDER\_NAME);  System.out.println("Object Id for path " + ROOT\_FOLDER\_NAME + TARGET\_FOLDER\_NAME + " : " + targetFolderId);  System.out.println("==================================================================================\n\n");    // (4)  System.out.println("Calling method addObjectToFolder. Params: objectId=" + fileId + ", targetFolderId = " + targetFolderId);  msс.addObjectToFolder(REPOSITORY\_ID, fileId, targetFolderId);  System.out.println("==================================================================================\n\n");    // (5)  System.out.println("Calling method getObjectParents. Params: objectId=" + fileId);  List<CmisObjectParentsType> objectParents = nsc.getObjectParents(REPOSITORY\_ID, fileId);  printOutParentDetails(objectParents);  System.out.println("==================================================================================\n\n");    // (6)  System.out.println("Calling method removeObjectFromFolder. Params: objectId=" + fileId + ", folderId = " + targetFolderId);  msс.removeObjectFromFolder(REPOSITORY\_ID, fileId, targetFolderId);  System.out.println("==================================================================================\n\n");    // (7)  System.out.println("Calling method getObjectParents. Params: objectId=" + fileId);  List<CmisObjectParentsType> objectParents2 = nsc.getObjectParents(REPOSITORY\_ID, fileId);  printOutParentDetails(objectParents2);  System.out.println("==================================================================================\n\n");    // (8)  System.out.println("Calling method moveObject. Params: objectId=" + fileId + ", targetFolderId="+targetFolderId+", sourceFolderId="+ sourceFolderId);  Holder<String> fileIdHolder = new Holder<>(fileId);  osc.moveObject(REPOSITORY\_ID, fileIdHolder, targetFolderId, sourceFolderId);  System.out.println("initial fileId: " + fileId);  System.out.println("returned fileIdHolder: " + fileIdHolder.value);  System.out.println("==================================================================================\n\n");    // (9)  System.out.println("Calling method getObjectParents. Params: objectId=" + fileId);  List<CmisObjectParentsType> objectParents3 = nsc.getObjectParents(REPOSITORY\_ID, fileId);  printOutParentDetails(objectParents3);  System.out.println("==================================================================================\n\n");   }   private static void printOutParentDetails(List<CmisObjectParentsType> parents) {  System.out.println("Parents amount: " + parents .size());  System.out.println("Parents: ");  for (CmisObjectParentsType parent : parents) {  CmisPropertyId propertyId = (CmisPropertyId) parent.getObject().getProperties().getProperty().get(0);  List<String> propValueList = propertyId.getValue();  if (propValueList.size() > 0) {  System.out.println(" " + propValueList.get(0));  }  }  }  }  } |

Moduļa *MultiFilingServiceClient.java* koda piemērs ir parādīts zemāk:

|  |
| --- |
| package abc.edk.client;  import org.apache.cxf.Bus; import vid.edk.multifilingservice.MultiFilingService; import vid.edk.multifilingservice.MultiFilingServiceAddObjectToFolderCmisFaultFaultFaultMessage; import vid.edk.multifilingservice.MultiFilingServiceRemoveObjectFromFolderCmisFaultFaultFaultMessage; import vid.edk.multifilingservice.MultiFilingService\_Service;  public class MultiFilingServiceClient {   private MultiFilingService ms;   public MultiFilingServiceClient() {   MultiFilingService\_Service multiFilingService\_service = new MultiFilingService\_Service();  ms = multiFilingService\_service.getWs2007FederationNoSctMtom();  }   public void addObjectToFolder(String repositoryId, String objectId, String destinationFolderId) {   try {  ms.addObjectToFolder(repositoryId, objectId, destinationFolderId, Boolean.TRUE, null);  } catch (MultiFilingServiceAddObjectToFolderCmisFaultFaultFaultMessage multiFilingServiceAddObjectToFolderCmisFaultFaultFaultMessage) {  multiFilingServiceAddObjectToFolderCmisFaultFaultFaultMessage.printStackTrace();  }   }   public void removeObjectFromFolder(String repositoryId, String objectId, String folderId) {   try {  ms.removeObjectFromFolder(repositoryId, objectId, folderId, null);  } catch (MultiFilingServiceRemoveObjectFromFolderCmisFaultFaultFaultMessage multiFilingServiceRemoveObjectFromFolderCmisFaultFaultFaultMessage) {  multiFilingServiceRemoveObjectFromFolderCmisFaultFaultFaultMessage.printStackTrace();  }   } } |

#### Objektu īpašību izmaiņas (FileProperties)

Piemērā tiek attēlota EDK objektu īpašību izmaiņas servisu darbības demonstrācija.

Izmantotās metodes jeb operācijas:

* *createFolder* (metodes implementāciju skat. 2.2.5.1. paragrāfā);
* *createDocument* (metodes implementāciju skat. 2.2.5.2. paragrāfā);
* *updateProperties* (metodes implementāciju skat. šīs sadaļas beigās);
* *getProperties* (metodes implementāciju skat. šīs sadaļas beigās);
* *bulkUpdateProperties* (metodes implementāciju skat. šīs sadaļas beigās).

Piemērs atbilst 5. scenārija darbību secībai (skat.2.2.3. sadaļā EDK servera mijiedarbības demonstrācijas scenāriji). Koda piemērs ir parādīts zemāk:

|  |
| --- |
| package samples;  import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.objectservice.CmisBulkUpdateType; import vid.edk.objectservice.CmisObjectIdAndChangeTokenType; import vid.edk.objectservice.CmisPropertiesType; import vid.edk.objectservice.CmisProperty; import vid.edk.objectservice.CmisPropertyString; import vid.edk.objectservice.EnumPropertiesBase;  import javax.xml.ws.Holder; import java.util.List;  public class PropertiesUpdatingSample {   private static final String REPOSITORY\_ID = "URN:IVIS:100266:EDK-0000001";  private static final String REPO\_PATH = "/Samples";  private static final String CMIS\_ROOT\_FOLDER\_NAME = "/Samples/";  private static final String LOCAL\_RES\_FOLDER = "src/main/resources/payload/";  private static final String FILTER\_CMIS\_NAME = EnumPropertiesBase.CMIS\_NAME.value();  private static final String REMOTE\_FOLDER\_1\_NAME = "TestFolderPropertiesUpdating1";  private static final String REMOTE\_FOLDER\_2\_NAME = "TestFolderPropertiesUpdating2";  private static final String FILE\_1\_NAME = "SampleFragment1";  private static final String FILE\_2\_NAME = "SampleFragment2";  private static final String NEW\_NAME = "NewFileName.txt";  private static final String MIME\_TYPE = "text/plain";  private static String CMIS\_ROOT\_FOLDER\_ID = "";   public static void main(String[] args) {   System.out.println("Demonstration of properties updating");   SpringBusFactory bf = new SpringBusFactory();  Bus bus = bf.createBus("cxf.xml");  BusFactory.setDefaultBus(bus);   ObjectServiceClient osc = new ObjectServiceClient();   osc.deleteTree(REPOSITORY\_ID, osc.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true));  CMIS\_ROOT\_FOLDER\_ID = osc.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true);   // (1)  System.out.println("Calling method createFolder. Params: folderName=" + REMOTE\_FOLDER\_1\_NAME + ", parentFolderId = " + CMIS\_ROOT\_FOLDER\_ID);  String folderId = osc.createFolder(REPOSITORY\_ID, CMIS\_ROOT\_FOLDER\_ID, REMOTE\_FOLDER\_1\_NAME);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + REMOTE\_FOLDER\_1\_NAME + " : " + folderId);  System.out.println("==================================================================================\n\n");    // (2)  System.out.println("Calling method createFolder. Params: folderName=" + REMOTE\_FOLDER\_2\_NAME + ", parentFolderId = " + CMIS\_ROOT\_FOLDER\_ID);  String folder2Id = osc.createFolder(REPOSITORY\_ID, CMIS\_ROOT\_FOLDER\_ID, REMOTE\_FOLDER\_2\_NAME);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + REMOTE\_FOLDER\_2\_NAME + " : " + folder2Id);  System.out.println("==================================================================================\n\n");    // (3)  System.out.println("Calling method createDocument in folder" + REMOTE\_FOLDER\_1\_NAME + " (" + folderId+")");  String file1Id = osc.createDocument(REPOSITORY\_ID, folderId, LOCAL\_RES\_FOLDER + FILE\_1\_NAME, "Lem", MIME\_TYPE);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + REMOTE\_FOLDER\_1\_NAME + "/" + FILE\_1\_NAME + " : " + file1Id);  System.out.println("==================================================================================\n\n");    // (4)  System.out.println("Calling method createDocument in folder" + REMOTE\_FOLDER\_2\_NAME + " (" + folder2Id+")");  String file2Id = osc.createDocument(REPOSITORY\_ID, folder2Id, LOCAL\_RES\_FOLDER+ FILE\_2\_NAME, "Lem", MIME\_TYPE);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + REMOTE\_FOLDER\_2\_NAME + "/" + FILE\_2\_NAME + " : " + file2Id);  System.out.println("==================================================================================\n\n");    // (5)  System.out.println("Calling method updateProperties. Params: in folder" + REMOTE\_FOLDER\_1\_NAME + " (" + folderId+")");   CmisPropertyString propName = new CmisPropertyString();  propName.setPropertyDefinitionId(FILTER\_CMIS\_NAME); // lets change name property  propName.setQueryName(FILTER\_CMIS\_NAME);  propName.getValue().add("HowTheWorldWasSaved");   CmisPropertiesType properties = new CmisPropertiesType();  properties.getProperty().add(propName);   String file1ChangeToken = "1";   ObjectServiceClient.UpdatedChangeToken updatedChangeToken = osc.updateProperties(REPOSITORY\_ID, file1Id, file1ChangeToken, properties);  file1ChangeToken = updatedChangeToken.getNewChangeToken();  System.out.println("objectId = " + updatedChangeToken.getObjectId());  System.out.println("new changeToken = " + file1ChangeToken);  System.out.println("==================================================================================\n\n");    // (6)  System.out.println("Calling method getProperties. Params: objectId=" + file1Id + ", filter="+FILTER\_CMIS\_NAME);  CmisPropertiesType resultProperties = osc.getProperties(REPOSITORY\_ID, file1Id, FILTER\_CMIS\_NAME);  CmisProperty prop = resultProperties.getProperty().get(0);  System.out.println("resultProperties: " + ((CmisPropertyString) prop).getValue().get(0));  System.out.println("==================================================================================\n\n");    // (7)  String changingProperty = FILTER\_CMIS\_NAME; // lets change value for property cmis:name   System.out.println("Calling method bulkUpdateProperties for two elements: setting for property [" + changingProperty+ "] value ["+NEW\_NAME+"]");   String file2ChangeToken = "1";   // objects list  vid.edk.objectservice.CmisBulkUpdateType bulkUpdateData = new CmisBulkUpdateType();  CmisObjectIdAndChangeTokenType elem1 = new CmisObjectIdAndChangeTokenType();  elem1.setId(file1Id);  elem1.setChangeToken(file1ChangeToken);  CmisObjectIdAndChangeTokenType elem2 = new CmisObjectIdAndChangeTokenType();  elem2.setId(file2Id);  elem2.setChangeToken(file2ChangeToken);  bulkUpdateData.getObjectIdAndChangeToken().add(elem1);  bulkUpdateData.getObjectIdAndChangeToken().add(elem2);   // properties list  CmisPropertyString descriptionProperty = new CmisPropertyString();  descriptionProperty.setPropertyDefinitionId(changingProperty);  descriptionProperty.setQueryName(changingProperty);  descriptionProperty.getValue().add(NEW\_NAME);   CmisPropertiesType cmisPropertiesType = new CmisPropertiesType();  cmisPropertiesType.getProperty().add(descriptionProperty);  bulkUpdateData.setProperties(cmisPropertiesType);   Holder<List<CmisObjectIdAndChangeTokenType>> cmisObjectIdAndChangeTokenTypeHolder = osc.bulkUpdateProperties(REPOSITORY\_ID, bulkUpdateData);  System.out.println("Updated objects: ");  for (CmisObjectIdAndChangeTokenType elem : cmisObjectIdAndChangeTokenTypeHolder.value )  System.out.println(" id = " + elem.getId());  System.out.println("==================================================================================\n\n");    // (8)  System.out.println("Calling method getProperties. Params: objectId=" + file1Id + ", filter="+changingProperty);  CmisPropertiesType resultBulketProperties1 = osc.getProperties(REPOSITORY\_ID, file1Id, changingProperty);  CmisPropertyString cmisPropertyString1 = (CmisPropertyString) resultBulketProperties1.getProperty().get(0);  System.out.println("Property [" + changingProperty + "] new value: " + cmisPropertyString1.getValue().get(0));  System.out.println("==================================================================================\n\n");    // (9)  System.out.println("Calling method getProperties. Params: objectId=" + file2Id + ", filter="+changingProperty);  CmisPropertiesType resultBulketProperties2 = osc.getProperties(REPOSITORY\_ID, file2Id, changingProperty);  CmisPropertyString cmisPropertyString2 = (CmisPropertyString) resultBulketProperties2.getProperty().get(0);  System.out.println("Property [" + changingProperty + "] new value: " + cmisPropertyString2.getValue().get(0));  System.out.println("==================================================================================\n\n");   } } |

Zemāk parādīts datu struktūras sagatavošanas un servisa izsaukšanas kods (klases *ObjectServiceClient.java* fragmenti).

|  |
| --- |
| public CmisPropertiesType getProperties(String repositoryId, String objectId, String filter) {  CmisPropertiesType properties = null;  try {  properties = os.getProperties(repositoryId, objectId, filter, null);  } catch (ObjectServiceGetPropertiesCmisFaultFaultFaultMessage objectServiceGetPropertiesCmisFaultFaultFaultMessage) {  objectServiceGetPropertiesCmisFaultFaultFaultMessage.printStackTrace();  }  return properties; }  public Holder<List<CmisObjectIdAndChangeTokenType>> bulkUpdateProperties(String repositoryId, CmisBulkUpdateType bulkUpdateData) {   Holder<List<CmisObjectIdAndChangeTokenType>> objectIdAndChangeToken = new Holder<>();   try {  os.bulkUpdateProperties(repositoryId, bulkUpdateData, null, objectIdAndChangeToken);  } catch (ObjectServiceBulkUpdatePropertiesCmisFaultFaultFaultMessage objectServiceBulkUpdatePropertiesCmisFaultFaultFaultMessage) {  objectServiceBulkUpdatePropertiesCmisFaultFaultFaultMessage.printStackTrace();  }  return objectIdAndChangeToken; } |

#### Objektu sasaistes servisu izmantošana (RelationshipServices)

Piemērā tiek attēlota EDK objektu sasaistes servisu darbības demonstrācija.

Izmantotās metodes jeb operācijas:

* *createFolder* (metodes implementāciju skat. 2.2.5.1. paragrāfā);
* *createDocument* (metodes implementāciju skat.2.2.5.2. paragrāfā);
* *createRelationship* (metodes implementāciju skat. 2.2.5.2. paragrāfā);
* *getObjectRelationships* (metodes implementāciju skat. šīs sadaļas beigās).

Piemērs atbilst 6. scenārija darbību secībai (skat.2.2.3. sadaļā). Koda piemērs ir parādīts zemāk:

|  |
| --- |
| package samples;  import abc.edk.client.ObjectServiceClient; import abc.edk.client.RelationshipServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.relationshipservice.CmisObjectListType; import vid.edk.relationshipservice.CmisObjectType; import vid.edk.relationshipservice.CmisPropertyId;  public class RelationshipsSample {   private static final String REPOSITORY\_ID = "URN:IVIS:100266:EDK-0000001";  private static final String REPO\_PATH = "/Samples";  private static final String CMIS\_ROOT\_FOLDER\_NAME = "/Samples/";  private static final String LOCAL\_RES\_FOLDER = "src/main/resources/payload/";  private static final String FOLDER\_NAME = "TestFolderObjectRelationships2";  private static final String MIME\_TYPE = "text/plain";  private static final String FILE\_1\_NAME = "SampleFragment1";  private static final String FILE\_2\_NAME = "SampleFragment2";  private static String CMIS\_ROOT\_FOLDER\_ID = "";   public static void main(String[] args) {   System.out.println("Demonstration of object relationships");   SpringBusFactory bf = new SpringBusFactory();  Bus bus = bf.createBus("cxf.xml");  BusFactory.setDefaultBus(bus);   ObjectServiceClient osc = new ObjectServiceClient();  RelationshipServiceClient rsc = new RelationshipServiceClient();   osc.deleteTree(REPOSITORY\_ID, osc.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true));  CMIS\_ROOT\_FOLDER\_ID = osc.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true);   // (1)  System.out.println("Calling method createFolder. Params: folderName=" + FOLDER\_NAME + ", parentFolderId = " + CMIS\_ROOT\_FOLDER\_ID);  String folderId = osc.createFolder(REPOSITORY\_ID, CMIS\_ROOT\_FOLDER\_ID, FOLDER\_NAME);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + FOLDER\_NAME + " : " + folderId);  System.out.println("==================================================================================\n\n");   // (2)  System.out.println("Calling method createDocument in folder" + FOLDER\_NAME + " (" + folderId + ")");  String file1Id = osc.createDocument(REPOSITORY\_ID, folderId, LOCAL\_RES\_FOLDER + FILE\_1\_NAME, "Lem", MIME\_TYPE);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + FOLDER\_NAME + "/" + FILE\_1\_NAME + " : " + file1Id);  System.out.println("==================================================================================\n\n");   // (3)  System.out.println("Calling method createDocument in folder" + FOLDER\_NAME + " (" + folderId + ")");  String file2Id = osc.createDocument(REPOSITORY\_ID, folderId, LOCAL\_RES\_FOLDER + FILE\_2\_NAME, "Lem", MIME\_TYPE);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + FOLDER\_NAME + "/" + FILE\_2\_NAME + " : " + file2Id);  System.out.println("==================================================================================\n\n");    // (4)  System.out.println("Calling method createRelationship. Params: sourceId: " + file1Id + ", targetId: " + file2Id);  String resultObjectId = osc.createRelationship(REPOSITORY\_ID, file1Id, file2Id, "fragment2followsAfterFragment1", null, null);  System.out.println("resulting objectid: " + resultObjectId);  System.out.println("==================================================================================\n\n");    // (5)  System.out.println("Calling method getObjectRelationships. Params: objectId: " + file1Id);  CmisObjectListType cmisObjectListType = rsc.getObjectRelationships(REPOSITORY\_ID, file1Id, null, "");  System.out.println("ObjectList.getNumItems(): " + cmisObjectListType.getNumItems().toString());  for (CmisObjectType object : cmisObjectListType.getObjects()) {  CmisPropertyId value = (CmisPropertyId) object.getProperties().getProperty().get(0);  System.out.println(" " + value.getValue());  }  System.out.println("==================================================================================\n\n");  } } |

Moduļa *RelationshipServiceClient.java* koda piemērs ir parādīts zemāk:

|  |
| --- |
| package abc.edk.client;  import org.apache.cxf.Bus; import vid.edk.relationshipservice.EnumRelationshipDirection; import vid.edk.relationshipservice.RelationshipService; import vid.edk.relationshipservice.RelationshipServiceGetObjectRelationshipsCmisFaultFaultFaultMessage; import vid.edk.relationshipservice.RelationshipService\_Service;   import java.math.BigInteger;  public class RelationshipServiceClient {   private RelationshipService rss;   public RelationshipServiceClient() {   RelationshipService\_Service service = new RelationshipService\_Service();  rss = service.getWs2007FederationNoSctMtom();  }     public vid.edk.relationshipservice.CmisObjectListType getObjectRelationships(String repositoryId, String objectId, String typeId, String filter){   vid.edk.relationshipservice.CmisObjectListType ObjectList = null;   try {  ObjectList = rss.getObjectRelationships(repositoryId, objectId, false, EnumRelationshipDirection.SOURCE, typeId, filter, false, BigInteger.valueOf(10) , BigInteger.valueOf(0), null);  } catch (RelationshipServiceGetObjectRelationshipsCmisFaultFaultFaultMessage relationshipServiceGetObjectRelationshipsCmisFaultFaultFaultMessage) {  relationshipServiceGetObjectRelationshipsCmisFaultFaultFaultMessage.printStackTrace();  }   return ObjectList;  } } |

#### ACL servisu izmantošana (ACL Services)

Piemērā attēlota EDK objektu piekļuves kontroles servisu darbības demonstrācija.

Izmantotās metodes jeb operācijas:

* *createFolder* (metodes implementāciju skat. 2.2.5.1. paragrāfā);
* *applyACL* (metodes implementāciju skat. šīs sadaļas beigās);
* *createDocument* (metodes implementāciju skat. 2.2.5.2. paragrāfā);
* *getACL* (metodes implementāciju skat. šīs sadaļas beigās);
* *getAllowableActions* (metodes implementāciju skat. 2.2.5.1. paragrāfā).

Piemērs atbilst 7. scenārija darbību secībai (skat. 2.2.3. sadaļā). Koda piemērs ir parādīts zemāk:

|  |
| --- |
| package samples;  import abc.edk.client.ACLServiceClient; import abc.edk.client.ObjectServiceClient; import org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.bus.spring.SpringBusFactory; import vid.edk.aclservice.\*; import vid.edk.objectservice.CmisAllowableActionsType;  import java.util.HashMap; import java.util.Map;  public class ACLServiceSample {   private static final String REPOSITORY\_ID = "URN:IVIS:100266:EDK-0000001";  private static final String REPO\_PATH = "/Samples";  private static final String CMIS\_ROOT\_FOLDER\_NAME = "/Samples/";  private static final String LOCAL\_RES\_FOLDER = "src/main/resources/payload/";  private static final String LOCAL\_RESOURCE\_FILENAME = "SphereHorse.bmp";  private static final String CMIS\_PRINCIPAL\_ID = "VIDISS\_EDK\_SENDERS";  private static final String CMIS\_PERMISSION\_CREATE\_REL = "edk:createRelationship";  private static final String CMIS\_PERMISSION\_CREATE\_FLD = "edk:createFolder";  private static final String ACL\_TEST\_FOLDER\_NAME = "ACLTestFolder";  private static final String FAILING\_FOLDER\_NAME = "FolderWhichShouldNeverBeCreated";  private static String CMIS\_ROOT\_FOLDER\_ID = "";    public static void main(String[] args) {   System.out.println("Demonstration of Access Control Lists discovering and managing");   SpringBusFactory bf = new SpringBusFactory();  Bus bus = bf.createBus("cxf.xml");  BusFactory.setDefaultBus(bus);   ObjectServiceClient oscAdmin = new ObjectServiceClient();  ACLServiceClient asc = new ACLServiceClient();   oscAdmin.deleteTree(REPOSITORY\_ID, oscAdmin.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true));  CMIS\_ROOT\_FOLDER\_ID = oscAdmin.getObjectByPath(REPOSITORY\_ID, REPO\_PATH, true);   // (1)  System.out.println("Calling method createFolder. Params: folderName=" + ACL\_TEST\_FOLDER\_NAME + ", parentFolderId = " + CMIS\_ROOT\_FOLDER\_ID);   String folderId = oscAdmin.createFolder(REPOSITORY\_ID, CMIS\_ROOT\_FOLDER\_ID, ACL\_TEST\_FOLDER\_NAME);  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + ACL\_TEST\_FOLDER\_NAME + " : " + folderId);  System.out.println("==================================================================================\n\n");    // (2)  System.out.println("Calling method applyACL (addACEs). Params: objectId=" + folderId + ", PrincipalId="+CMIS\_PRINCIPAL\_ID + ", permission="+CMIS\_PERMISSION\_CREATE\_REL);   CmisAccessControlPrincipalType principal = new CmisAccessControlPrincipalType();  principal.setPrincipalId(CMIS\_PRINCIPAL\_ID);   CmisAccessControlEntryType element = new CmisAccessControlEntryType();  element.setPrincipal(principal);  element.getPermission().add(CMIS\_PERMISSION\_CREATE\_REL);  element.setDirect(false);   CmisAccessControlListType addACEs = new CmisAccessControlListType();  addACEs.getPermission().add(element);   ApplyACLResponse applyACLResponse = asc.applyACL(REPOSITORY\_ID, folderId, addACEs, null, EnumACLPropagation.PROPAGATE);   printOutPermissionsForPrincipalId(applyACLResponse, CMIS\_PRINCIPAL\_ID);  System.out.println("==================================================================================\n\n");    // (3)  System.out.println("Calling method createDocument in folder" + ACL\_TEST\_FOLDER\_NAME + " (" + folderId+")");  String file1Id = oscAdmin.createDocument(REPOSITORY\_ID, folderId, LOCAL\_RES\_FOLDER+LOCAL\_RESOURCE\_FILENAME, "Horse", "image/bmp");  System.out.println("Object Id for path " + CMIS\_ROOT\_FOLDER\_NAME + ACL\_TEST\_FOLDER\_NAME + "/"+LOCAL\_RESOURCE\_FILENAME + " : " + file1Id);  System.out.println("==================================================================================\n\n");    // (4)  System.out.println("Calling method getACL. Params: objectId=" + file1Id);  GetACLResponse getACLResponse = asc.getACL(REPOSITORY\_ID, file1Id, Boolean.TRUE);  System.out.println("cmisACLresult permission size = " + getACLResponse.getACL().toString());  System.out.println("==================================================================================\n\n");    // (5)  System.out.println("Calling method getAllowableActions. Params: objectId=" + file1Id);  CmisAllowableActionsType cmisAllowableActions = oscAdmin.getAllowableActions(REPOSITORY\_ID, file1Id);  System.out.println("isCanCreateDocument = " + cmisAllowableActions.isCanCreateDocument());  System.out.println("isCanCreateRelationship = " + cmisAllowableActions.isCanCreateRelationship());  System.out.println("==================================================================================\n\n");   // (6)  System.out.println("Calling method applyACL (removeACEs). Params: objectId=" + folderId + ", PrincipalId="+CMIS\_PRINCIPAL\_ID + ", permission to be removed: "+CMIS\_PERMISSION\_CREATE\_REL + ", " + CMIS\_PERMISSION\_CREATE\_FLD);   CmisAccessControlPrincipalType principalRem = new CmisAccessControlPrincipalType();  principalRem.setPrincipalId(CMIS\_PRINCIPAL\_ID);   CmisAccessControlEntryType elementRem = new CmisAccessControlEntryType();  elementRem.setPrincipal(principalRem);  elementRem.getPermission().add(CMIS\_PERMISSION\_CREATE\_REL);  elementRem.getPermission().add(CMIS\_PERMISSION\_CREATE\_FLD);  elementRem.setDirect(false);   CmisAccessControlListType removeACEs = new CmisAccessControlListType();  removeACEs.getPermission().add(elementRem);   ApplyACLResponse cmisACLrem = asc.applyACL(REPOSITORY\_ID, folderId, null, removeACEs, EnumACLPropagation.PROPAGATE);   printOutPermissionsForPrincipalId(cmisACLrem, CMIS\_PRINCIPAL\_ID);  System.out.println("==================================================================================\n\n");   // (7)  System.out.println("Calling method getAllowableActions. Params: objectId=" + folderId);  CmisAllowableActionsType cmisAllowableActions2 = oscAdmin.getAllowableActions(REPOSITORY\_ID, folderId);  System.out.println("isCanCreateDocument2 = " + cmisAllowableActions2.isCanCreateDocument());  System.out.println("isCanCreateRelationship2 = " + cmisAllowableActions2.isCanCreateRelationship());  System.out.println("==================================================================================\n\n");    // (8)  System.out.println("Creating connection for client 'LDK.Sender'...\n");  ObjectServiceClient oscSender = new ObjectServiceClient();  System.out.println("==================================================================================\n\n");    // (9)  System.out.println("Calling method createFolder in a sender role (awaiting error message permissionDenied). Params: folderName=" + FAILING\_FOLDER\_NAME + ", parentFolder = " + ACL\_TEST\_FOLDER\_NAME + " (" + folderId + ")");  oscSender.createFolder(REPOSITORY\_ID, folderId, FAILING\_FOLDER\_NAME);  System.out.println("==================================================================================\n\n");   }    private static void printOutPermissionsForPrincipalId(ApplyACLResponse cmisACL, String principalId) {  System.out.println("Permissions for [" + principalId + "] :");  for (CmisAccessControlEntryType elem: cmisACL.getACL().getACL().getPermission()) {  if (elem.getPrincipal().getPrincipalId().equals(principalId)) {  for (String permission: elem.getPermission()) {  System.out.println(" " + permission);  }  }  }  }  } |

Moduļa *ACLServiceClient.java* koda piemērs ir parādīts zemāk:

|  |
| --- |
| package abc.edk.client;  import org.apache.cxf.Bus; import vid.edk.aclservice.\*;  import java.util.Map;  public class ACLServiceClient {   private ACLService as;  private ObjectFactory aclObjectFactory;   public ACLServiceClient() {   ACLService\_Service aclService\_service = new ACLService\_Service();  as = aclService\_service.getWs2007FederationNoSctMtom();  aclObjectFactory = new ObjectFactory();   }    public ApplyACLResponse applyACL(String repositoryId, String objectId, CmisAccessControlListType addACEs, CmisAccessControlListType removeACEs, EnumACLPropagation propagation) {   ApplyACLResponse applyACLResponse = null;   ApplyACL applyACL = new ApplyACL();  applyACL.setRepositoryId(repositoryId);  applyACL.setObjectId(objectId);  applyACL.setAddACEs(addACEs);  applyACL.setRemoveACEs(removeACEs);  applyACL.setACLPropagation(aclObjectFactory.createApplyACLACLPropagation(propagation));   try {  applyACLResponse = as.applyACL(applyACL);  } catch (ACLServiceApplyACLCmisFaultFaultFaultMessage aclServiceApplyACLCmisFaultFaultFaultMessage) {  aclServiceApplyACLCmisFaultFaultFaultMessage.printStackTrace();  }   return applyACLResponse;  }    public GetACLResponse getACL(String repositoryId, String objectId, Boolean onlyBasicPermissions){   GetACLResponse cmisACL = null;   GetACL getACL = new GetACL();  getACL.setRepositoryId(repositoryId);  getACL.setObjectId(objectId);  getACL.setOnlyBasicPermissions(aclObjectFactory.createGetACLOnlyBasicPermissions(onlyBasicPermissions));   try {  cmisACL = as.getACL(getACL);  } catch (ACLServiceGetACLCmisFaultFaultFaultMessage aclServiceGetACLCmisFaultFaultFaultMessage) {  aclServiceGetACLCmisFaultFaultFaultMessage.printStackTrace();  }   return cmisACL;  }   } |