1. Introduction

These code examples are made to help you build integrations using

- Fælleskommunalt Sags- og Dokumentindeks
- Fælleskommunalt Organisationssystem
- Fælleskommunalt Klassifikationssystem

The code examples demonstrate how to:

- generate code classes from WSDL/XSD files (see chapter 6)
- get a token from Secure Token Service
- call webservices via Serviceplatformen
- call webservices directly (i.e. not using Serviceplatformen)

The code examples demonstrate how to use the following services and operations:

Integration	Service	Operations
Sags- og Dokumentindeks (SF1470)	SagDokumentIndeksService v6.0	Importer, Fremsoeg, Fjern
Organisation (SF1500)	VirksomhedService v6.0	Soeg
Organisation (SF1500)	OrganisationService v6.0	Soeg, Laes
Klassifikation (SF1510)	KlasseService v7.0	Soeg, List

Sags- og Dokumentindeks v6.0 use Liberty Basic Soap Binding and are called via Serviceplatformen. Organisation v6.0 and Klassifikation v7.0 uses the IDWS Binding (OIO IDWS SOAP profile) and is called directly, i.e. not via Serviceplatformen. The signing of the SOAP Envelope content has been simplified with OIOIDWS. The examples demonstrate how both security profiles can be used in the same context. Read more at Digitaliseringskataloget

Chapter 4 describes the code examples in more detail.

2. Prerequisites

- .NET Framework 6.x
- Visual Studio or similar
- Certificates (see 2.1)
- Service agreements (see 2.2)

2.1 Certificates

In order for the code examples to work, the following certificates are needed:

- A client certificate (funktionscertifikat) representing your it-system.

    - Read more about how to create and install your client certificate here.
- Serviceplatformen Certificate (and root/intermediate trust certificates)
    - Included in this package - can also be downloaded from Digitaliseringskataloget.
- Secure Token Service Certificate (and root/intermediate trust certificates)
    - Included in this package - can also be downloaded from Digitaliseringskataloget.
- Organsation Certificate - Needed to verify responeses from Organsation
    - Included in this package - can also be downloaded from Digitaliseringskataloget
- Klassifikation Certificate - Needed to verify responeses from Klassifikation
    - Included in this package - can also be downloaded from Digitaliseringskataloget

To install to certificates for Serviceplatformen and Secure Token Service:

- Run `/Certificates/install.bat` in a command promt

2.2 Service agreement

In order for the code examples to work, you will need service agreements which must include the following services:

- Sags- og dokumentindeks v. 6.0 (Required roles: Rediger and Udstil)
- Organisation v. 6.0 (Required roles: Udstil)
- Klassifikation v. 7.0 (Required roles: Udstil)

Read more about how to create the service agreements

3. Build & execute

Verify that all required certificates are installed in Windows (see 2.1) and that you have a valid service agreement (see 2.2)

- Open the solution by opening the `Kombit.InfrastructureSamples.sln` solution file.
- Open `ConfigVariables.cs` and modify the variables in the first section (Variables which MUST BE MODIFIED before running the code examples). You will need to set:
    - Thumbprint for you client certificate
    - UUID and name of your it-system in Fælleskommunalt Administrationssystem
    - CVR and name of the municipality (myndighed) that will be used to test
    - A randomly generated UUID which will be used for your test case
- Go to Build Menu → Build Solution. This will create a `.exe` file in `\Kombit.InfrastructureSamples\Kombit.InfrastructureSamples\bin\Debug\`
- Run the `Kombit.InfrastructureSamples.exe` file from the terminal and follow the instructions written in the console.

4. Code examples

The following story is used for all code examples:

> Ulla Jakobsen works at Korsbaek Kommune (CVR 11111111).
>
> On May 12th 2020 she receives an inquiry from Godtfred Lund, an elderly citizen in the municipality, who would like to learn about how the municipality can help him in his everyday life which is getting increasingly difficult. Ulla agrees with Godtfred, that an employee from the municipality will visit him for a so-called preventive conversation.
>
> Ulla creates a case in the municipality's case management system. The case is assigned case number 2020-123456789 by the system and Ulla names the case "Agreement on preventive home visit". She assigns the case KLE number 27.35.04 (preventive home visit) and KLE handlingsfacet G01 (general cases). As the primary party on the case she assigns Godtfred Lund, who has CPR number 012345-6789. Ulla assigns herself as the primary case worker and her team, the Preventive Team, as the responsible unit. Ulla sends a confirmation of the agreement to Godtfred and adds it as a document on the case.
>
> On June 2nd 2020 Ulla closes the case.

Example 1 - Import Case

Demonstrates how to import a case to Sags- og Dokumentindeks with the minimum required attributes and relations (i.e. all mandatory attributes and relations)

- The corresponding Organisation UUID for the supplied CVR number is found (see example 5).
- The corresponding Klasse UUID for the supplied KLE subject is found (see example 6).
- The corresponding Klasse UUID for the supplied KLE facet is found (see example 6).
- The `Import` operation in `SagDokumentIndeksService` is used to import the case

State (tilstand)

As the case has been created and later closed, both states must be included in the import:

- Opstaaet (emerged)
  - VirkningTil is set to the date when the case was closed *Afsluttet (completed)
  - VirkningTil is set to +infinite.

Relations

The import request creates the following mandatory relations:

- Sagsaktør.Ejer
- Sagsaktør.Ansvarlig
- Sagsaktør.Primær behandler
- Sagspart.Primær part
- Sagsklasse.Primær klasse
- Sagsklasse.Handlingsklasse
- Sagsarkiv.Behandlingsarkiv
- IT-system.Master
- IT-system.Afsender

For all relations the "Virking" element is filled out as follows:

Field	Content
VirkningFra	Current time
VirkingTil	Infinite
AktoerRef	UUID of the user responsible for the modification in the master system (in this case Ulla)
AktoerTypeKode	Bruger (user)

Example 2 - Search Case

Demonstrates how to search for a case in Sags- og Dokumentindeks.

- The `FREMSOEG` operation in `SagDokumentIndeksService` is used to find the case based on the case UUID.
- The number of the case (sagsnummer) is found in the response and displayed in the console.

Example 3 - Remove Case

Demonstrates how to remove a case from Sags- og Dokumentindeks.

- The `FJERN` operation in `SagDokumentIndeksService` is used to remove the case

Example 4 - Import, search and remove Case

Executes examples 1-3 in one sequence.

Example 5 - Search Organization

Demonstrates how to find the Organisation object related to a Virksomhed object with a given CVR-number.

- The `SOEG` operation in `VirksomhedService` is used to find the Virksomhed object which holds the CVR-number.
- The `SOEG` operation in `OrganisationService` is used to find the Organisation object related to the Virksomhed object.
- The `LAES` operation in `OrganisationService` is used to read the full Organisation object.

Example 6 - Search Class

Demonstrates how to find a Klasse object based on a Brugervendt Nøgle (key).

- The `SOEG` operation in `KlasseService` is used to find the Klasse UUID based on a brugervendt nøgle (key).
- The `LIST` operation in `KlasseService` is used to get the full Klasse object based on the Klasse UUID found by searching.

- Details about the Klasse object are found in the response and displayed in the console.

5. Solution structure

Folder	Contents
Certificates	Required Certificates for Serviceplatformen and Secure Token Service (see 2.1)
Digst.OioIdws.*	Packages for handling tokens and security.
Kombit.InfrastructureSamples	Program.cs (main class of the project) ConfigVariables.cs (all variables needed to configure the solution)
Kombit.InfrastructureSamples\Klassifikation	Code related to Fælleskommunalt Klassifikationssystem
Kombit.InfrastructureSamples\Organisation	Code related to Fælleskommunalt Organisationssystem
Kombit.InfrastructureSamples\SagDokumentIndeks	Code related to Fælleskommunalt Sags- og Dokumentindeks
Kombit.InfrastructureSamples\Token	Helper classes for fetching and caching tokens
wsdl	Contains wsdl and xsd files for the webservices used in the examples. These files can alse be found in the documentation package for each service (see links i section 1).

6. Code generation from WSDL and XSD files

The project contains code which is autogenerated from the service WSDL and XSD files.

The code is autogenerated using dotnet-svcutil.exe (additional steps for SagDokumentIndeks - see below). The project contains a bacth file for each service which will generate the code:

Batch file	Autogenerated code in Kombit.InfrastructureSamples
wsdl\klasse\run.bat	Klassifikation\KlasseService.cs
wsdl\organisation\run.bat	Organisation\OrganisationService.cs
wsdl\virksomhed\run.bat	Organisation\VirksomhedService.cs

Batch file	Autogenerated code in Kombit.InfrastructureSamples
wsdl\sagdokumentindeks\run.bat	SagDokumentIndeks\SagDokumentIndeksService.cs SagDokumentIndeks\SagDokumentIndeksService.LokalUdvidelse.c

The file `\wsdl\runall.bat` will execute all of the above batch files.

Follow this procedure if you wish to regenerate the code:

- Open a shell from VS by going to Tools → Command Line → Developer PowerShell.
- Navigate to wsdl directory and execute ./runall.bat.

Generating code for Sags- og Dokumentindeks extensions (lokaludvidelser)

The run.bat file for SagDokumentIndeksService contains additional steps in order to autogenerate code for the service.

This is due to the fact that the OIO-standard, on which Sags- og Dokumenindeks is based, allows for the use of extensions (Lokaludvidelser) in order to handle custom elements/fields. Sags- og Dokumentindeks makes use of such extensions (Lokaludvidelser) in various places.

dotnet-svcutil.exe does not generate code for the extensions (Lokaludvidelser) as they are not explicitly referenced in the XSD schema. I.e. the XSD schema does not state *when and where* to use a specific extension - only that an extension is allowed.

Therefore xsd.exe is used to generate code for the extensions (Lokaludvidelser). The autogenerated code is placed in `SagDokumentIndeksService.LokalUdvidelse.cs`.

When generating code for the extensions (Lokaludvidelser) using xsd.exe, a number of classes/enums already generated by dotnet-svcutil.exe will be generated once again. To avoid naming conflicts, a Powershell command is executed to modify the duplicate classes/enums in `SagDokumentIndeksService.LokalUdvidelse.cs`. Some of the classes/enums used in the extensions will therefore appear with an "X" appended to the name. This renaming will not affect the naming in the resulting XML structure but allows the project to compile and XML to be properly serialized.

Finally, in order to handle XML serialization of the extensions (Lokaludvidelser), attributes are added to the autogenerated class `LokalUdvidelseListeType` in `SagDokumentIndeksService.LokalUdvidelseListeType.cs`.