ASA.Apim.Library

Document version 3.14

NuGet version 2.1.0.1

Document history:

Date	Version	Who	Description
29-03-2018	1.00	DHI	First version of the document and library released for testing.
06-04-2018	1.01	DHI	Updated section 3 with required .Net Framework
19-04-2018	1.10	DHI	Updated Method overview. Updated The Library description, expanding section 5.2. Added list of scenarios using methods in section 7. Added code samples in section 8.
02-05-2018	1.11	DHI	Small changes to the introduction
14-06-2018	1.20	DHI	Adding a new section 6.4 with catalogue related methods Added new methods for course reservations to section 6.2
03-08-2018	1.21	DHI	Adjustments to Scenarios in section 7
24-09-2018	2.0	DHI	Added new section for Course Reservations and Enrollments.
25-4-2019	3.0	DHI	Updated section 4 with preconditions Specified how to change environment – section 4.4 Added section with Nuget history
16-5-2019	3.1	DHI	New release 1.0.1.8 *Includes a breaking Change*
11-6-2019	3.2	DHI	Added new methods with unique ids across categories and subcategories. See section 6.1 and 6.3.
23-8-2019	3.3	DHI	Added new NuGet version. See section 10.10
26-9-2019	3.4	DHI	Added new NuGet version. See section 10.11 *Includes a breaking Change* *Only works in apimanager-asa-test.*
26-9-2019	3.5	DHI	Added new NuGet version. See section 10.12
12-5-2020	3.6	DHI	Added new NuGet version. See section 10.13
22-5-2020	3.7	DHI	Added new NuGet version. See section 10.14
3-6-2020	3.8	DHI	Added new NuGet version. See section 10.15
22-6-2020	3.9	DHI	Added new NuGet version. See section 10.16
31-5-2021	3.10	DHI	Added new NuGet version. See section 10.17
9-6-2021	3.11	DHI	Added new NuGet version. See section 10.18
15-6-2021	3.12	DHI	Added new NuGet version. See section 10.19
2-7-2021	3.13	DHI	Added new NuGet version. See section 10.20 – *gælder kun for Test*
29-10-2021	3.14	DH	Added new NuGet Version See section 10.21

1 CONTENT

2 INTRODUCTION

ASA.Apim.Library is a .Net class library that can be used in applications based on the .Net Framework version. The library is released as a NuGet NuGet package on a private NuGet server.

This document describes how to start up using the ASA.Apim.Library where

- Section 3 describes how to install the library into your application
- Section 4 how to sign-up to the API Manager which is a precondition to use the Library
- Section 5 describes the structure of the Library and how to use the implemented methods
- Section 6 gives an overview and short description of the methods
- Section 7 list some scenarios and how to use the methods in the Library
- Section 8 contains some code samples.

3 INSTALLING THE LIBRARY

The ASA.Apim.Library for .Net Framework is distributed as a NuGet package on a private NuGet server - https://nugetserveriqi20180327105515.azurewebsites.net/nuget. The library has to be installed in the .Net application and only applications based on .**Net Framework 4.6.1**.

Further description of how to use install the NuGet package can be found in separate document (ASA.ApimLibrary NuGet pakke.pdf only available in Danish).

4 PRECONDITION & CREDENTIALS

Before you can start using the API you need to obtain a subscriptionkey and know which account(s) to access. In the following sections we will describe how to subscribe to get access to the APIs on the API Manager and how to get access to accounts where you want data from.

Description of your first time at ASA Test API Developer and how to explore and try out the services directly in the API Developer can be found in a separate document (API Manager som Developer - Kom-i-gang.pdf – only available in Danish).

4.1 SUBSCRIPTION

To be able to use the library you need to have a subscription to the API Manager where the web services are published. You cannot use the same subscription for different environment and therefor needs to sent request for each environment.

The subscription can be obtained by sending a mail to support@iqidigital.com with

- first name
- last name
- email
- the account(s)* you need access to.
- Environment (Test or Live)

iQiDigital will then verify the request, set up the user and sent an invite from the requested API Manager.

It is not possible to use sign-up in API Manager Developer portal and get immediately access to API because of the verification process and granting rights to requested account(s).

4.2 ACCOUNT

Account is equivalent with school e.g. TESTAOF-TEST, TESTLOF-TEST, TESTDOF-TEST or TESTFORA-TEST. The name of the account is the same as created and used in the Navision Client.

It is possible to have access to more than one account if needed and have to be specified when sending a request for subscription to APIM.

4.3 ENVIRONMENT

ASA.Apim.Library supports API Managers for both Test and Live APIs, and other API Managers if needed by providing the URL to the API manager.

The Library provides a static class Configuration.cs under the folder Helpers, which contains some helper methods. Use SetApimUrl(string apimUrl) to set the environment Url (= Apim Url). The parameter apimUrl can be either ”test” or “live” which are aliases for the URL to the environments.

4.3.1 APIM-Test

To use ASA.Apim.Library with API Manager in Test you call

ASA.Apim.Library.Helpers.Configuration.SetApimUrl(“test”);

and if nothing is set with SetApimUrl, “test” is also default.

4.3.2 APIM-Live

To use ASA.Apim.Library with API Manager in Live you call

ASA.Apim.Library.Helpers.Configuration.SetApimUrl(“live”);

4.3.3 Sample of using SetApimUrl

The following is an example of how to use SetApimUrl in .NET Framework 4.6.1 and above:

To the Web.config appSettings of your project add:

- Test Env: <add key="ApimUrl" value="test" />
- Live Env: <add key="ApimUrl" value="live" />

In Global.asax, at the end of Application_Start() add

```
//Configure APIM
ASA.Apim.Library.Helpers.Configuration.SetApimUrl(ConfigurationManager.AppSettings["ApimUrl"]);
```

5 THE LIBRARY

The ASA.Apim.Library have two layers to access data from the ASA solution. On the top there is a list of implemented methods divided into 5 domains CourseService, Locationservice, TeacherService, EnrollmentService and UtilityService all explained in section 6. And in the bottom, there is a list of service references (web references) where each service and their operations can be reached directly.

5.1 THE METHODS

The methods are implemented under each domains in the namespace ASA.Apim.Library.Services, so the methods for e.g. Location services can be found under ASA.Apim.Library.Services.LocationService. For most of the services there are 3 implemented methods:

1. **get{*servicename*}(**Subscriptionkey,Accountkey, size), where
   Subscriptionkey is from the user sign-up (see section 4)
   Accountkey is the account in Capto (see section 4)
   Size is optional and default set to 0 meaning everything is returned.

   Sample: getClassRoom("1234567890123456654","AOF") which returns all classrooms for the test account "AOF"

2. **get{*servicename*}(**Subscriptionkey,Accountkey, filter, size)
   Subscriptionkey, Accountkey and **Size** same as 1.
   Filter is a list of search criteria.

   Sample: getClassRoom("1234567890123456654","AOF", {Course_Location_Name,"Birkerød Skole"}) will return all classrooms on "Birkerød Skole"

3. **get{*servicename*}ById**(Subscriptionkey,Accountkey, id, size)
   Subscriptionkey, Accountkey and **Size** same as 1.
   Id is a string containing the of search criteria.

   Sample: getClassRoomById("1234567890123456654","AOF", "1008") will return classroom number 1008.

In the samples above, it all returns data from the ClassRoom service.

For each method there is mouse-over descriptions in the code, so it should be self-explanatory.

A full list of all available methods are listed in section 6 with a short description and the Danish term.

See section 9.1 for code sample on how to use a method.

5.2 CUSTOM METHODS (WEB REFERENCES)

If the list of methods is not sufficient and there is a need for using other features and or functionalities of the webservices on the API Manager it is possible to create customized methods. All web services published in ASA Api manager can be accessed in the library though their web reference, e.g. CourseHeader service can be accessed using CourseHeader_Reference etc.

The main part of the webservices published is Basic Page Operation which has a set of default operations where the current methods almost only uses the operation ReadMultible except for one that uses Read. Further description of the Basic Page Operations can be found here: https://docs.microsoft.com/da-dk/dynamics-nav/basic-page-operations and here: https://docs.microsoft.com/en-us/previous-versions/dynamicsnav-2016/dd301179(v%3dnav.90).

See section 9.2 for code samples on how to use the web reference for implementing a method.

5.3 FILTERING

For the webservices that have the operation ReadMultible there is a possibility to create and use a filter when calling the service. A filter contains of a field name and a criteria. The field names are dependent on the definition of the webservices and typically consist of the fields in the response. Dynamic NAV webservices has a specific syntax for setting the criteria in filters which is documented here: https://docs.microsoft.com/en-us/previous-versions/dynamicsnav-2016/hh879066(v=nav.90).

See section 0 of a code sample for how to setup a filter I ReadMultible.

6 METHODS OVERVIEW

As described in previous section the methods are divided into 5 domains:

- CourseService
- LocationService
- TeacherService
- EnrollmentService
- UtilityService

- CatalogService

CourseService contains methods that are directly related to a course like getting course detail, course prices etc. CourseService has relations to LocationService and TeacherService which can be seen in the diagram in section 6.1. See the full list of methods in section 6.1 Basic services.

LocationService contains methods that are related to geographical places and organisations like school and department. LocationService has relations to CourseService which can be seen in the diagram in section 6.1. See the full list of methods in section 6.1 Basic services.

TeacherService contains methods that are related to Teachers. TeeacherService has relations to CourseService which can be seen in the diagram in section 6.1. See the full list of methods in section 6.1 Basic services.

EnrollmentService contains methods related to course reservations and enrollments. See the full list of methods in section 6.2 Course Enrollments.

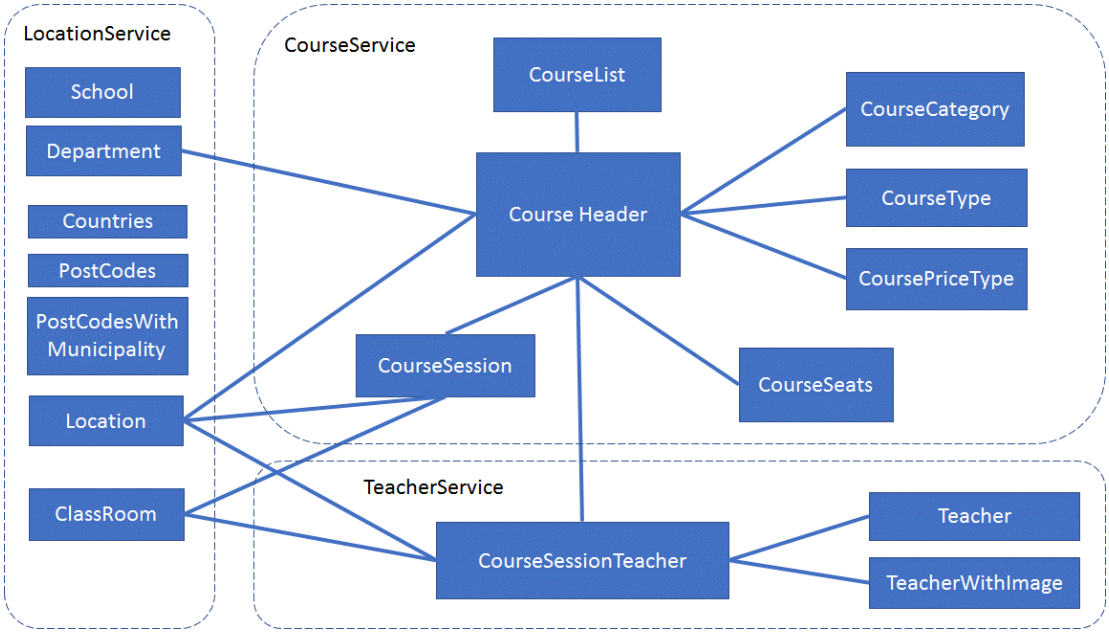
UtilityService contains methods that are not directly related but supports the solution. See the full list of methods in section 6.3 Utility Services.

CatalogService contains methods that are directly related to a course, but combined with a Catalog, meaning that you can call CatalogCourses with a catalog id and get all courses for that specific catalog. The same applies to prices, locations, teachers etc. See the full list of methods in section 6.4 Catalog services.

In the following sections the overview diagrams contain blue and grey boxes representing services. The blue boxes can be accessed using the methods and the grey boxes can only be accessed using the web references in this version of the Library.

In the list of services and methods there is a reference in [] in the first column under the method/service name. That is a reference to Capto's description of the *"Medlems og Kursusportalen NAV 2017"* where further descriptions of the services can be found. The document can be requested from Capto.

6.1 BASIC SERVICES



Figur 1Basic domain

Method	Description	Danish term
CourseService		
getCourseHeader *[CourseHeader]*	Returns details for one course	Hold
getCourseHeaders *[CourseHeader]*	Returns list of courses with details, where status of the course is either booked or launched.	Liste af Hold
getcourseTypes *[CourseType]*	Returns list of course types	Hold type
getCourseLists *[CourseList]*	Returns overview of courses	Hold oversight
getCourseCategory *[CH Attribute Values WS]*	Returns categories for course, see section describing categories.	Hold kategorier
getCoursePriceType *[CoursePriceTypeList]*	Returns courses with list of price types and prices.	Pristyper
getCourseSeats	Returns courses with details on seats e.g. available seats, number of participants and number on the waiting list.	Hold pladser
getCourseSessions *[courseBookingLineList]*	Returns courses with list of booked sessions including date, time, location, cancel status etc for each course.	Mødegange

getCourseHeadersWithDepartments *[Courseheader, SchoolDepartmentList]*	Returns list of full course details with department details. There is only one department for a given course and the list only contains courses with a department.	Hold med afdeling
imageLibraryList	Library for images.	Billede bibliotek
getCourseCategoriesUnique *[Catalog]*	Returns the same list as *getCourseCategories*, but with unique ids across categories and aubcategories.	
getCourseCategoriesByIdUnique *[Catalog]*	Returns the same list as *getCourseCategoriesById*, but with unique ids across categories and aubcategories.	
TeacherService		
getTeachers *[TeacherList]*	Returns list of teachers with details e.g. name, phone and email.	Lærer
getCourseSessionTeachers *[teacherCourseSessionLineList]*	Returns courses with list of booked sessions including date, time and teachers for each course.	Mødegange med lærere
getTeacherWithImage *[TeacherWithImage]*	Returns list of teachers with details and an image.	Lærer med Billede
LocationService		
getSchool *[SchooleDepartmentList]*	Return school with details e.g. name, address, phone, email, homepage, school manager, bank and giro information. There is one school for each account.	Skole
getDepartment *[SchooleDepartmentList]*	Return departments with details e.g. address, email, phone, homepage etc. Referenced in CourseHeader.CourseDepartment.	Afdeling
getLocation *[CourseLocationList]*	Return locations with details e.g. address, municipality, county and GPS coordinates. Referenced in CourseSession.Location_No.	Lokation
getClassRoom *[ClassRoomList]*	Return Classrooms with details, e.g. name, location and GPS coordinates. Referenced in CourseSession.ClassRoom_No.	Lokale
getCountries *[Countries]*	Returns a list of country code and name.	Landekoder
getPostCodes *[PostCodes]*	Returns a list of post code, city and country code.	Postnr
getpostCodesWithMunicipality *[PostCodesWithMunicipality]*	Returns a list of post code, city and country code and municipality.	Postnr m/kommune

6.2 COURSE RESERVATIONS AND ENROLLMENTS

EnrollmentService		
createCourseEnrollments *[CourseEnrollments]*	Create a course enrolment for one or more participants including payment information with "PortalOrderId" as unique identifier.	Tilmelding
getCourseParticipant *[CourseParticipantList]*	Returns for each course a list of participants with e.g. participant name, number and type.	Hold deltagere
CourseWebStatus [CourseWebStatus]	Check status of the course enrolment already created using "PortalOrderId".	Tilmeldingsstatus
webshopBasketManagement [webshopBasketManagement]	Handles reservation of a course before enrolment: - Create reservation - Remove reservation - Reservation exists	Reservation

6.3 UTILITY SERVICES

UtilityService		
getCategories *[Catalog]*	Returns list of categories, where each category has a main category (Attribute) and a subcategory (Attribute value).	Kategori
getCatalogs *[Catalog]*	Returns a list of catalogs (Attribute) with description and from_date.	Katalog
getSubCategories *[SubjectAreas]*	Returns a list of subcategories. Subcategories can be either related to Categories or Catalogs.	Underkatalog
getCatagoryMatrix	Returns a matrix of categories and subcategories with the same structure as *getCourseCategories*.	
getCatalogMatrix	Returns a matrix of catalogs and subcategories with the same structure as *getCourseCategories*.	
getSubCategoriesUnique	Returns the same as *getCategories* but with unique ids across categories and subcategories.	

getCategoryMatrixUnique *[Catalog]*	Returns the same as *getCategoriesMatrix* but with unique ids across categories and subcategories.	

6.4 CATALOG SERVICES

Common for following methods are that Catalog is the overall selection criteria. Meaning that just referring to a catalog id it is possible to get the related courses, sessions, prices, teachers etc.

Method	Description	Danish term
CatalogService		
getCatalogCourses	Returns list of courses with details for specified catalog.	Liste af Hold for et katalog
getCatalogCourseLists	Returns overview of courses for specified catalog.	Hold oversigt for et katalog
getCatalogCoursePrice	Returns list of course prices for specified catalog..	Pristyper for et katalog
getCatalogCourseSeats	Returns courses with details on seats e.g. available seats, number of participants and number on the waiting list for specified catalog.	Hold pladser for et katalog
getCatalogCourseSessions	Returns courses with list of booked sessions including date, time, location, cancel status etc for each course.	Mødegange for et katalog

7 How to use the Library Methods

This section contains list of scenarios with samples of how and which method or combination of methods in the Library to use.

7.1 Courses and course details

7.1.1 Scenario: I want to get the course details for one or list of courses.

getCourseHeaders is the central service to provide all details of one or more courses depending on the filter. And **getCourseHeadersById** is just an easier way to get the list of course details using the Course number as parameter where the parameter will be the criteria where the criteria (see section 5.3 for further info on filters).

getCourseHeader only gets one course which takes a course number as an input parameter, which is almost the same as **getCourseHeadersById** except the latter can take a filter criteria as input parameter and therefor return a list of courses.

GetCourseheaders and GetCourseHeadersById will only return Courses that have generalCourseStatus set to either Booked or launched. - Updated

7.1.2 Scenario: I want to know if social security number or Birthdate is required. Updated

The course type will determine if social security number is required.,To get the information you need to use **getCourseTypeById** with the coursetype from the course details which will return the information if social security number is required or Birthdate is required. Payers_name_only is set if a lecture where it is the quantity of seats that are requested and no participant information is required.

7.2 Catalog and Courses

7.2.1 Scenario: I want to show possible catalogs so I can select which one I am interested in.

getCatalogs will get a list of all catalogs with an Id, Description and start date.

7.2.2 Scenario: I want to get the courses for a specific catalog.

A course can belong to multiple catalogs and to get courses belonging to a specific catalog use **GetCourseCategoryById** and use the id found using **getCatalogs**.

7.2.3 Scenario: I want to get course details or sessions for a specific catalog.

If you want to use the list of courses for a given catalog when selecting Course details like in section 7.1 or course sessions in section 7.3 you need to create a criteria that can be used as a filter. The criteria are a concatenation of all course numbers separated by a '|' e.g. "85100|67000|65123". Use the criteria like in **getCourseheardersById**(…,"85100|67000|65123".,…) to get your course details for a specific catalog.

7.3 Course sessions and teachers

7.3.1 Scenario: I want to get time and location for all the sessions of a course

getCourseSessionById with course number as input parameter returns list of booked sessions including date, time, location, cancel status etc for each course.

7.3.2 Scenario: I want to get time and teacher for all the sessions of a course

getCourseSessionTeachersById with course number as input parameter returns list of booked sessions and the teachers per session.

7.3.3 Scenario: I want to get list of teachers that are active on one or more courses

getTeachersByCourseNo with course number(s) as input parameter will return a distinct list of teachers that are active on at least one of the courses.

7.4 COURSE SEATS

7.4.1 Scenario: I want to show status of a course. - Updated

getCourseSeatsById will return the maximum number of seats on a course, number of participants and number of seats still available. You need to use the field availableSeats even if it is not always equal to the difference between Maximum and number of seats taken, because ASA also adjust with reservations. You will also get the deadline for enrolment, so you are able to show if it is still possible to enrol to a course.

7.5 PRICING

7.5.1 Scenario: I want to present a list of prices for a course. - Updated

The Prices for a course are defined by a combination of price_type_name and municipality, meaning price_type_name can have different prices depending on municipality . Using **getCoursePriceTypesById** given course number will give you a list CoursePriceTypes that includes price_type, price_type_name and municipality, both code and name.

If you need to select prices for a specific municipality or a specific price_type_name we suggest that you either create your own filter criteria to **getCourcePriceType** or do the filtering using e.g. LINQ on the list returned from **getCoursePriceTypesById.**

8 RESERVATION AND ENROLLMENT – OUT-OF-DATE

In the Library there are a list of methods to support reservation of courses as well as checkout Enrollment. The validations of Enrollments are handled by the receiving administration system in Navision, when the enrollments are created.

The flow of a typical shopping basket has the following steps as shown in the diagram.

The green boxes are directly supported by the Enrollment services and methods, while the orange boxes represent the checkout where filling out the additional information for the selected courses and lectures before payment are decided and made. Methods to support UI validation are described in various scenarios in section 7.

The overall flow is that during selection of courses and or lectures reservations are made to secure the seats for the next 20 minutes. When finish selecting courses and or lectures the checkout starts adding information about participants, price type etc. Hereafter the payer and payment information are added before the checkout is confirmed and enrollments are created in Navision.

In section 8.1 each step is descripted in more detail including which methods in the Library that can be used.

The data that is collected during step 2, 3 and 4 are contained in one instance of the Enrollment Model and described in more details in section 8.2.

8.1 THE METHODS IN ENROLLMENTSERVICE

[TBD]

8.1.1 Create Reservation

EnrollmentService.ReserveCourseEnrollment

8.1.2 Remove reservation

EnrollmentService.RemoveReservation

8.1.3 Create Enrollment

EnrollmentService.CreateEnrollment

8.1.4 CourseWebStatus

EnrollmentService.Status

8.2 ENROLLMENT MODEL

Because the original model of the enrolment services is difficult to use it was decided to implement a new Enrollment model on top of the original model and create a mapping between the two models. However this only works when using the Library, so when using the SOAP service directly from the API Manager it is necessary to know the original model which is covered by the documentation from Capto. Whenever Enrollment Model is mentioned in the remaining of this section it is therefore a reference to the new Enrollment Model.

The Enrollment Model covers all information to create Enrollments for a customer's shopping basket and consist of:

- PortalOrderId
- List of CourseDetails
- List of LectureDetails
- Payer
- Payment

When a customer has selected a list of courses and or lectures and move on to Checkout an instance of the EnrollmentModel is created with the unique identifier PortalOrderId (see 8.2.1 for further information on PortalOrderId). PortalOrderId is the same id used whenever creating course and lecture reservations for the customer.

In the following sections all elements of the EnrollmentModel are described I further details.

8.2.1 PortalOrderId

```
public string PortalOrderId { get; set; }
```

When instantiating the EnrollmentModel a unique identifier is assigned, also called a PortalOrderId. PortalOrderId is a unique id for a given account and can only be used once when creating enrollments in ASA, therefor it is suggested to generate a GUID. If Enrollments are sent to ASA with a PortalOrderId that has already been sent before the enrollments will be rejected by ASA.

8.2.2 CourseDetails

CourseDetails is for course numbers where CourseType has Payers_Name_only set to false - see section 7.1.2 for further details.

```
public string CourseNo { get; set; }
public string SocialSecurityNo { get; set; }
public string Birthdate { get; set; }
public string PriceType { get; set; }
public decimal Price { get; set; }
public Participant Participant { get; set; }
```

CourseDetails consist of a list of all selected courses enriched with price, participant information and if required social security number or birthdate. See section 7.1.2 for further details on how to decide if social security number or birthdate is required.

Price is given by a Price_Type and the actual price for the actual participant – see section 7.5 for further details on how to determine the price for the participant.

The personal information about the participant is given by

```
public int Id { get; set; }
public string Name { get; set; }
public string Address { get; set; }
public string PostCode { get; set; }
public string Email { get; set; }
public string PhoneNo { get; set; }
public string MobilePhoneNo { get; set; }
```

Where Name, Address and PostCode is required.

Name can be full name or be provided as

```
public string FirstName { get; set; }
public string SurName { get; set; }
```

And Address can be full address or be provided as

```
public string StreetName { get; set; }
public int StreetNo { get; set; }
public string StreetNoExtension { get; set; }
public string FloorNo { get; set; }
public string FloorNoExtension { get; set; }
public string City { get; set; }
```

Id is an internal id in one instance of EnrollmentModel, meaning that it will never occur in any form for identification in ASA after creating enrollments. The id is used internally in the EnrollmentModel to identify if a given participant occur on more than one course. And if that is the case, the participant information including the id should be identical for all courseDetails with the same participant. The Id is an integer and should start with 1 for the first participant in the first CourseDetail.

See section 8.3.2 for a JSON sample of CourseDetails.

8.2.3 LectureDetails

LectureDetails is for course numbers where CourseType has Payers_Name_only set to true - see section 7.1.2 for further details.

```
public string CourseNo { get; set; }
public string PriceType { get; set; }
public decimal Price { get; set; }
public int ParticipantQuantity { get; set; } = 1;
```

Price is given by a Price_Type and the actual price for one participant – see section 7.5 for further details on how to determine the price for the participant.

ParticipantQuantity is the number of seats on a given lecture.

See section 8.3.3 for a JSON sample of LectureDetails.

8.2.4 Payer

Payer is of type Participant and detailed description can be found in section 8.2.2 for Personal information. Many times, payer is one of the participants in CourseDetails, and if that is the case then please just copy information from actual participant in courseDetail, including the id, to payer.

```
public Participant Payer { get; set; }
```

If Payer is a company following fields will more likely be used

```
public int Id { get; set; }
public string Name { get; set; }
public string Address { get; set; }
public string PostCode { get; set; }
public string Email { get; set; }
public string PhoneNo { get; set; }
public string MobilePhoneNo { get; set; }
public string VATno { get; set; }
public string EANno { get; set; }
public string MobilePhoneNo { get; set; }
```

See section 8.2.2 for a more extended version of Address id needed.

See section 8.3.4 for a JSON sample of Payer.

8.2.5 PaymentInfo

```
public string WayOfPayment { get; set; }
public string  PaymentReference { get; set; }
public decimal TotalOrderAmount { get; set; }
```

WayOfPayment is how the payment is completed and ASA supports following three ways: creditcard, or invoice or payment link. And the value in WayOfPayment are 1=creditcard, 2=invoice and 3=payment link.

If WayOfPayment is creditcard, paymentreference need to contain the reference to the actual in e.g in ePay or similar services.

TotalOrderAmount has to be the total price of the Enrollments.

See section 8.3.5 for a JSON sample of Payment information

8.3 SAMPLES

8.3.1 PortalOrderId

```
{
 "PortalOrderId": "f08f7e68-be0f-4b7a-bfd8-dba65e7f64b3",
}
```

8.3.2 List of CourseDetails

One course and one participant

```
"CourseDetails": [
{
"CourseNo": ”XX3333”
"PriceType": ”2”,
"Price": 489,
"Participant" : {
                "Id": 1,
                "Name": "Jens Olsen",
                "Address": "Viberødvej 2",
                "PostCode": ”2450”,
                "City": "København SV",
                "Email": "jo@hotmail.com",
                "PhoneNo": ”41799701”,
                "MobilePhoneNo": ”41799701”
                }
}
]
```

Extended name and address

```
"CourseDetails": [
{
"CourseNo": ”864008”,
"PriceType": ”4”,
"Price": 1200,
"Participant" : {
                "Id": 1,
                "FirstName": "Jens",
                "SurName": "Olsen",
                "StreetName": "Viberødvej",
                "StreetNo": "2",
                "PostCode": ”2450”,
                "City": "København SV",
                "Email": "jo@hotmail.com",
                "PhoneNo": ”41799701”,
                "MobilePhoneNo": ”41799701”
                }
}
]
```

Course with required social security number

```
"CourseDetails": [
{
"CourseNo": ”XX1234”,
"PriceType": ”6”,
"Price": 200,
"SocialSecurityNumber": "010145-0253",
"Participant" : {
                "Id": 1,
                "Name": "Jens Olsen",
                "Address": "Viberødvej 2",
                "PostCode": ”2450”,
                "City": "København SV",
                "Email": "jo@hotmail.com",
                "PhoneNo": ”41799701”,
                "MobilePhoneNo": ”41799701”
                }
}
]
```

Course with required Birthdate

```
"CourseDetails": [
{
"CourseNo": ”XX0000”,
"PriceType": ”6”,
"Price": 200,
"Birthdate": "010145",
"Participant" : {
                "Id": 1,
                "Name": "Jens Olsen",
                "Address": "Viberødvej 2",
                "PostCode": ”2450”,
                "City": "København SV",
                "Email": "jo@hotmail.com",
                "PhoneNo": ”41799701”,
                "MobilePhoneNo": ”41799701”
                }
}
]
```

Two courses and same participant

```
"CourseDetails": [
{
"CourseNo": ”111111”,
"PriceType": ”4”,
"Price": 100,
"Participant" : {
                "Id": 1,
                "Name": "Jens Olsen",
                "Address": "Viberødvej 2",
                "PostCode": 2450,
                "City": "København SV",
                "Email": " jensolsen@hotmail.com ",
                "PhoneNo": 41799701,
                "MobilePhoneNo": 41799701
                }
},
{
```

One course two participants

```
"CourseDetails": [
{
"CourseNo": ”111111”,
"PriceType": ”4”,
"Price": 100,
"Participant" : {
                "Id": 1,
                "Name": "Jens Olsen",
                "Address": "Viberødvej 2",
                "PostCode": ”2450”,
                "City": "København SV",
                "Email": "jensolsen@hotmail.com",
                "PhoneNo": ”41799701”,
                "MobilePhoneNo": ”41799701”
                }
},
{
```

```
"CourseNo": ”222222”,
"PriceType": ”4”,
"Price": 200,
"Participant" : {
                 "Id": 1,
                 "Name": "Jens Olsen",
                 "Address": "Viberødvej 2",
                 "PostCode": 2450,
                 "City": "København SV",
                 "Email": "jensolsen@hotmail.com ",
                 "PhoneNo": ”41799701”,
                 "MobilePhoneNo": ”41799701”
                 }
}
]
```

```
"CourseNo": ”111111”,
"PriceType": ”4”,
"Price": 100,
"Participant" : {
                 "Id": 2,
                 "Name": "Lise Olsen",
                 "Address": "Viberødvej 2",
                 "PostCode": 2450,
                 "City": "København SV",
                 "Email": "liseolsen@hotmail.com",
                 "PhoneNo": ”41000022”,
                 "MobilePhoneNo": ”41000022”
                 }
}
]
```


8.3.3 List of LectureDetails

One lecture with 3 participants


```
"LectureDetails": [
{
"CourseNo": ”XX3333”,
"PriceType": ”2”
"Price": 489,
"ParticipantQuantity": 3,
}
]
```


Two lectures with 1 and 5 participants


```
"LectureDetails": [
{
"CourseNo": ”XX3333”,
"PriceType": ”2”,
"Price": 489,
"ParticipantQuantity": 1,
},
{

"CourseNo": ”XX7777”,
"PriceType": ”2”,
"Price": 100,
"ParticipantQuantity": 5,
}
]
```


8.3.4 Payer

Deltager der betaler


```
"Payer":
{
"Id": 1,
"Name": "Jens Olsen",
"Address": "Viberødvej 2",
"PostCode": ”2450”,
"City": "København SV",
"Email": " jensolsen@hotmail.com ",
"PhoneNo": ”41799701”,
"MobilePhoneNo": ”41799701”
}
```


Firma der betaler


```
"Payer":
{
"Id": 1,
"Name": "Fabrikken",
"Address": "Fabrikvej",
"PostCode": ”4400”,
"City": "Søllerup",
"Email": "firma@fabrikken.com ",
"PhoneNo": ”44554455”,
"MobilePhoneNo": ”44554455”,
"VATno": ”12345678”,
"EANno": ”0123456789123”,
}
```


8.3.5 Payment

Payment with creditcard


```
"PaymentInformation": {
 "PaymentReference": ”515835418877731”,
 "WayOfPayment": "1",
 "TotalOrderAmount": 1300
}
```


Payment with invoice


```
"PaymentInformation": {
 "WayOfPayment": "2",
 "TotalOrderAmount": 2500
}
```

9 CODE SAMPLES

This section contains code samples of how to use the library in the implementation. It both contains samples of how to use the methods in the library and also samples of how a method is implemented in the library. The latter can be uses as a sample to how the developer can create their own custom methods.

9.1 METHODS – CODE SAMPLES

This sample shows how to use the getCourseHeaders method in the code. Be aware that the method is in CourseService while the class CourseHeader is in CourseHeader_Reference namespace.

```
using ASA.Apim.Library.CourseHeader_Reference;
using ASA.Apim.Library.Security;
using ASA.Apim.Library.Services;
..
```

```
namespace iQiNugetServerTest
{
   class Program
   {
      static void Main(string[] args)
      {
         var currentCredentials = new ApiManagerCredentials()
         {
            AccountKey = ConfigurationManager.AppSettings["AccountKey"].ToString(),
            SubscriptionKey = ConfigurationManager.AppSettings["SubscriptionKey"].ToString()
         };


         var courseHeaders = new List<CourseHeader>();

         try
         {
            courseHeaders = CourseService.GetCourseHeaders(currentCredentials.SubscriptionKey,
                                                           currentCredentials.AccountKey).ToList();
         }
         catch (Exception ex)
         {

            throw new Exception(ex.Message);
         }

         foreach (var courseHeader in courseHeaders)
         {
            Console.WriteLine($"No: {courseHeader.No}\t Description: {courseHeader.Description} \t Location:
                              {courseHeader.Course_Location_City} ");
         }
         Console.Read();

      }
   }
}
```

9.2 WEB REFERENCES – CODE SAMPLES

Following sections list code samples for how to implement two of the service operations – Read and ReadMultiple.

9.2.1 Service operation: Read

This sample show the implementation of service operation CourseHeader.Read into the method getCourseHeader.

```
Using ASA.Api.Library.CourseHeader_Reference
using ASA.Apim.Library.Security;
using ASA.Apim.Library.Services;
…
```

```
/// <summary>
/// Get one course with details from Navision, using course number
/// </summary>
/// <param name="courseNumber">Course id/number</param>
/// <param name="subscriptionKey">Signup in Api Manager to get a key. [Required]</param>
/// <param name="accountKey">Account key from Navision. [Required]</param>
/// <returns></returns>
public static CourseHeader GetCourseHeader(string courseNumber, string subscriptionKey, string accountKey = null)
{
    var apiKeys = new ApiManagerCredentials()
    {
        AccountKey = accountKey,
        SubscriptionKey = subscriptionKey

    };

    return GetCourseHeader(courseNumber, apiKeys);
}

internal static CourseHeader GetCourseHeader(string courseNumber, ApiManagerCredentials credentials)
{
    try
    {
        var service = new CourseHeader_Service()
        {
            ApiCredentials = credentials
        };

        var courseHeader = service.Read(courseNumber);
        service.Dispose();
        return courseHeader;
    }
    catch (Exception exception)
    {
        throw new Exception($"Error in GetCourseHeader(): {exception.Message}");
    }
}
```

9.2.2 Service operation: ReadMultible

This sample show the implementation of service operation CourseHeader.ReadMultible into the method getCourseHeaders. The request of the operation ReadMultible contains a filter and the method singleCourseHeaderFilter creates a CourseHeader_Filter.

```
Using ASA.Api.Library.CourseHeader_Reference
using ASA.Apim.Library.Security;
using ASA.Apim.Library.Services;

…
```

```
/// <summary>
///
/// Get courses with details from Navision, using your own built filter using CourseHeader_Filter.
/// </summary>
/// <param name="filter">An instance of CourseHeader_Filter.</param>
/// <param name="subscriptionKey">Signup in Api Manager to get a key. [Required]</param>
/// <param name="accountKey">Account key from Navision. [Required]</param>
/// <param name="size">Maximum returned records. 0 returns all records. [Optional]</param>
/// <returns></returns>
public static IEnumerable<CourseHeader> GetCourseHeaders(CourseHeader_Filter[] filter, string subscriptionKey, string
accountKey = null, int size = 0)
{
   var apiKeys = new ApiManagerCredentials()
   {
      AccountKey = accountKey,
      SubscriptionKey = subscriptionKey

   };

   return GetCourseHeaders(filter, apiKeys, size);
}

internal static IEnumerable<CourseHeader> GetCourseHeaders(CourseHeader_Filter[] filter, ApiManagerCredentials credentials,
int size)
{
   try
   {
      var service = new CourseHeader_Service()
      {
        ApiCredentials = credentials
      };

      var list = service.ReadMultiple(filter, "", size).ToList();
      service.Dispose();
      return list;
   }
   catch (Exception exception)
   {
      throw new Exception($"Error in GetCourseHeaders(): {exception.Message}");
   }
}


internal static CourseHeader_Filter[] SingleCourseHeaderFilter(string criteria, CourseHeader_Fields field)
{
```

```
    return new CourseHeader_Filter[]
    {
        new CourseHeader_Filter()
        {
    Field = field,
    Criteria = criteria
        }
    };
}
```

10 ASA.APIM.LIBRARY NUGET VERSION HISTORY

10.1 NUGET VERSION 1.0.0.9

27-9-2018

New and updated Features

- Added new service: DiscountCodes service

- Updated EnrollmentModel with following new fields:

    - Participant.Birthdate : string (mapped to SocialSecurityNo)
    - Participant.FirstName : string (mapped to FirstName)
    - Participant.SurName: string (mapped to SurName)
    - Participant.StreetName: string (mapped to StreetName)
    - Participant.StreetNo: int (mapped to StreetNo)
    - Participant.StreetNoExtension: string (mapped to StreetNoExtension)
    - Participant.FloorNo:string (mapped to FloorNo that is a string)
    - Participant.FloorNoExtension: string mapped to FloorNoExtension)
    - Participant.City: string (mapped to City)

10.2 NUGET VERSION 1.0.1.2

22-11-2018

New and updated Features

- Added new service: Municipality service and added new method
    - GetMenucipalicies(PostCode) that returns a list of City, municipality code and name.

- Updated SubCategories service with following new fields:
    - name="SEO_Headline" type="xsd:string
    - name="SEO_Description" type="xsd:string
    - name="SEO_Imagelink" type="xsd:string
    - name="SEO_Title" type="xsd:string
    - name="SEO_Text_Above" type="xsd:string
    - name="SEO_Text_Belov" type="xsd:string

- Added Helper.Configuration.SetApimUrl which is used to set the target API Manager in runtime.

  SetApimUrl takes one parameter which is the url to the API Manager where ASA.Apim.Libary directs its service calls – the API Manager in test or production. The url can be replaced with an alias for test and production, meaning that if url is set to “test” all calls are directed to the API Manager in test and if url is set to “prod” all service calls are directed to API Manager in production. Test is default so if nothing is set all calls will be directed to API Manager in test.

  For a website there could be a call to Helper.Configuration.SetAPImUrl(url) in e.g. Global.asax, and where url has been setup and retrieved from web.config.

10.3 NuGet Version 1.0.1.3

23-11-2018

New and updated Features

- Added new methods to easier extract Categories and sub categories in a matrix. The output includes all categories either there are courses attached or not.
    - GetCategories returns a list of all top categories.
    - GetCategoryMatrix returns a list of top categories and their sub categories. The structure of the output is
        - § Attribute_ID (= top category id)
        - § Attribute (= top category name)
        - § Attribute_Value_ID (= first level category id)
        - § Attribute_Value (= first level category name)
        - § Attribute_Value_ID_2 (= second level category id)
        - § Attribute_Value_2 (= second level category name)
        - § Attribute_Value_ID_3 (= third level category id)
        - § Attribute_Value_3 (= third level category name)

        This is the same structure as in CourseCategory service, but without information about Courses.

- Added similar methods for Catalog
    - GetCatalog (already exists)
    - GetCatalogMatrix is similar as GetCategoryMatrix, but for catalogs only.

10.4 NuGet Version 1.0.1.4

12-02-2019

New and updated Features

- Enrollment:
    - Added new methods for discountcodes to support Enrollment Manager (SDK) version 3.

10.5 NuGet Version 1.0.1.5

05-03-2019

New and updated Features

- Enrollment:
    - Added municipality codes when creating enrollments in ASA.

10.6 NuGet Version 1.0.1.6

07-03-2019

- Issues:

- Updated method Util.CategoryMatrix. Error was found where ids for Categories and Subcategories were identical.

10.7 NuGet Version 1.0.1.7

26-4-2019

New and updated Features

- Enrollment:
  - Added support for enrollments on waitinglist
- Library in Live environment:
  - Added support for using the Library in API Manager Live (see section 4.4)

10.8 NuGet Version 1.0.1.8

26-4-2019

New and updated Features

- EnrollmentModel:
  - Breaking change: Moved the field “DiscountCodes” from EnrollmentModel.CourseDetail.Participant to EnrollmentModel.CourseDetail.

10.9 NuGet Version 1.0.1.9

11-6-2019

New Features

- Added new methods with generated unique id’s across categories and subCategories
  - getCourseCategoriesUnique
  - getCourseCategoriesByIdUnique
  - getSubCategoriesUnique
  - getCategoryMatrixUnique

10.10 NuGet Version 1.0.2.0

23-8-2019

New Features

- Added new field to CourseEnrollment: Comment

Issues

- Fixed issue with missing municipality for Payer when mapping to NAV service.

10.11 NuGet Version 2.0.0.0

26-9-2019

New Features

- CoursePricetype: Added new field Social_Security_No_Required
- CourseType:
  - Added new field Social_Security_No_Not_Required
  - Removed fields Social_Security_No_Required and Birthdate_Required

Issues

- The removed fields in CourseType service will cause a breaking change and therefor the NuGet version will only work with a API Manager that has been updated accordingly.

  Version 2.0.0.0 will therefor only work in apimanager-asa-test and a notification will be made when the Live has been updated.

10.12 NuGet Version 2.0.0.1

26-9-2019

Issue

- DiscountCodes service issue fixed.

10.13 NuGet Version 2.0.0.2

19-5-2020

Issue

- Large number of courses in catalog issue fixed. **No changes to the use of the Library are needed.**

10.14 NuGet Version 2.0.0.3

27-5-2020

Issue

- Large number of courses in catalog issue fixed. **No changes to the use of the Library are needed.**

10.15 NuGet Version 2.0.0.4

3-6-2020

Issue

- Use Bookmark (paging) when retrieving more than 2000 records. **No changes to the use of the Library are needed.**

10.16 NuGet Version 2.0.0.5 – prerelease

22-6-2020

New feature

- Added new Utility service to return list of companies (accounts):
  - GetCompanies(string subscriptionKey, string tenant), where tenant is AOF, LOF, DOF, FOF or FORA.

Issue

- More services are optimized by adding paging.
- Bug fix: When downloading catalog for sydjylland.aof.dk

10.17 NuGet Version 2.0.0.6

31-5-2021

- NuGet version 2.0.0.5 becomes stable.

10.18 NuGet Version 2.0.0.7

9-6-2021

Updated features

- Updated DiscountCodes to always deliver rate (amount or percentage) for any code.

Issues

- Paging in GetCourseHeader has been reduced further to 100 per request. **No changes to the use of the Library are needed.**

10.19 NuGet Version 2.0.0.8

15-6-2021

New Features

- Added new fields to Department: Apply_School_Data_on_Reports

10.20 NuGet Version 2.1.0.0 prerelease

2-7-2021

Release only available on TEST

When updating NuGet package, the version is only available when selecting pre-releases.

New Features

- Added new elements on following services

  - **classroom**
    Description (kursists bemærkning)
    Description_on_Portal (Boolean, skal vises på portal)
    Teacher_Comment (underviser bemærkning)
  - **department**
    DIBS_Merchant_Id
    DIBS_Login
    DIBS_Password
    ePay_Merchant_No
    ePay_Password
    Terms_and_Conditions_URL (url til handelsbetingelser)
  - **location**
    AssistantName (Pedelnavn, LOF)
    email (Pedel email, LOF)
    phoneno (Pedel telefon, LOF)
    TeacherComment (underviser bemærkning)
  - **teacher**
    Occupation (stilling)
    Teacher_Note (underviser bemærkning)
    Active (er underviser aktiv)
  - **teacherwithimage**
    Occupation (stilling)
    Teacher_Note (underviser bemærkning)
    Active (er underviser aktiv)

10.21 NuGet Version 2.1.0.1

29-10-2021

Release of features described under 2.1.0.0 - now also available for LIVE.