

# Student Data Aggregation Calculator

by



## Developer Guide

**Version:** 1.0

**Date:** February 21, 2013

## Table of Contents

<b>Overview.....</b>	<b>3</b>
<b>System Requirements.....</b>	<b>3</b>
Client Requirements.....	3
Server Requirements.....	3
<b>Internet Information Services (IIS).....</b>	<b>4</b>
Create a New site in IIS 7.....	4
<b>API Reference.....</b>	<b>8</b>
Overview of API.....	8
Objective.....	8
Integration Approach.....	8
How to Use this Reference.....	8
<b>inBloom REST API.....</b>	<b>9</b>
<b>inBloom REST API Features.....</b>	<b>9</b>
Example response for ApplicationAuthorization .....	10
Example response for applicationAuthorization/{uuid} .....	10
Example response for customRoles.....	10
<b>inBloom REST API Example Requests &amp; Responses.....</b>	<b>11</b>
Example response for applicationAuthorization .....	13
Example response for applicationAuthorization/{uuid}.....	13
Example response for customRoles.....	13
<b>inBloom REST API Example Requests &amp; Responses.....</b>	<b>14</b>
<b>Source Code Directory Structure .....</b>	<b>18</b>
SDAC.Core.....	18
SDAC.Data.....	18
SDAC.DomainModel.....	18
SDAC.UI.Web.....	20
UI Files (aspx pages).....	20
Error Page.....	22
<b>Custom Entity.....</b>	<b>23</b>
How to Retrieve Custom Entity Data.....	23

## Overview

Student Data Aggregation Calculator (SDAC) is a flexible and intuitive app that allows teachers to mark, create and store flags with specific conditions as per the preferences of the user, and run these particular conditions across the student database to identify further subsets of the students who meet the set criteria.

These flags can even be saved for future use, thus creating an easy-to-use tool that can quickly identify students who, for example, have turned in specific homework assignments, achieved academic honors or got a specific grade on an exam or assignment.

The inBloom through its shared technology infrastructure has initialized open doors for district school staff, who have typically only interacted with software systems as a means for data entry, creating and viewing static reports and generating spreadsheets, that are often outdated as well as incomplete, not to be mentioned the complicity of the structure and searching process of the data. Given the vastness and depth of the data the inBloom Data Store encompasses, access to real- time, accurate and relevant student data is now within touching distance, and success of the project will ensure a faster and more efficient student assessment system.

With a simple, user friendly, yet highly effective user interface in place along with the features and functions that have been provided with an extensive focus on the usability and the technology stack can be termed as highly reliable and scalable. SDAC is sure to make an instant impact on the way that data-driven decisions have been affecting the student achievements through the years, and it can be stated with absolute and utmost confidence that there are multiple stakeholders who will be benefiting from the introduction of SDAC on field.

## System Requirements

### Client Requirements

- Supported client browsers include: IE 8+, Firefox 4+, Safari 4+, Chrome 12+
- Also supports iPad

### Server Requirements

- Windows Server 2008 R2 64bit
- Microsoft .NET Framework 4.0 SP2 or higher

- Microsoft Internet Information Services (IIS) 7.0+ or Visual Studio 2010/2012 built-in web server.
- Microsoft SQL Server 2008 R2, 2012 (supports free SQL Server Express Edition)

## Internet Information Services (IIS)

### Create a New site in IIS 7

The following section explains how to create a new website on your Windows server in IIS 7. There are two types of websites that can be configured, IP based sites and Name based sites.

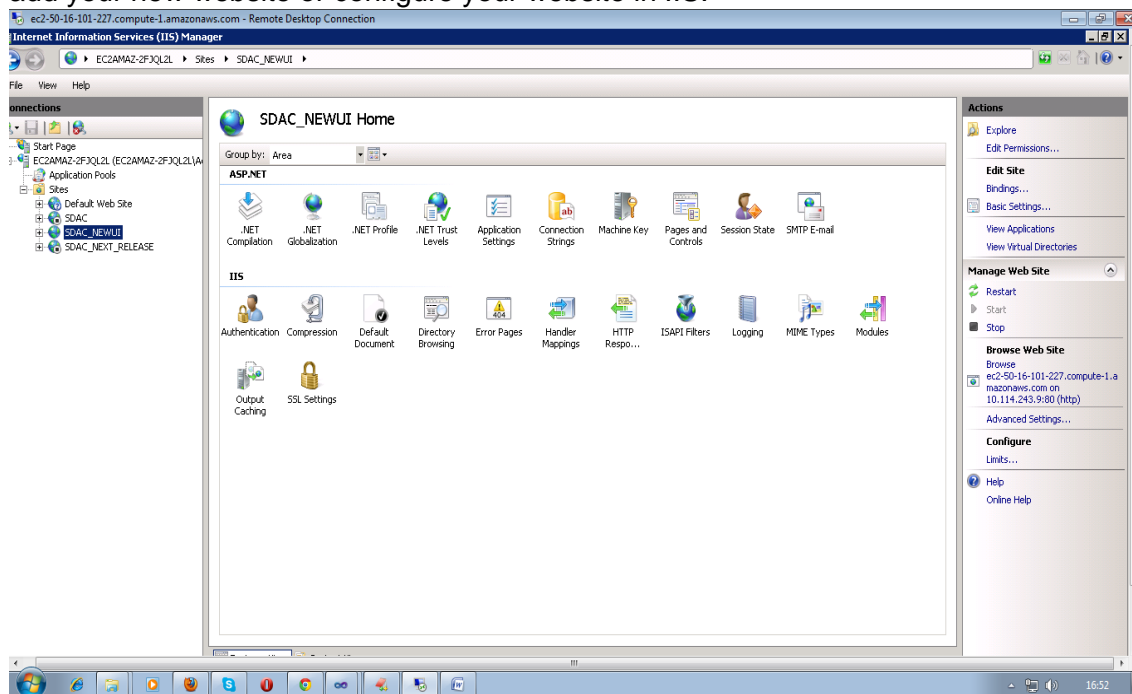
**Step 1:** Log into your server through Remote Desktop Connection

**Step 2:** From Visual Studio, publish your Web application.

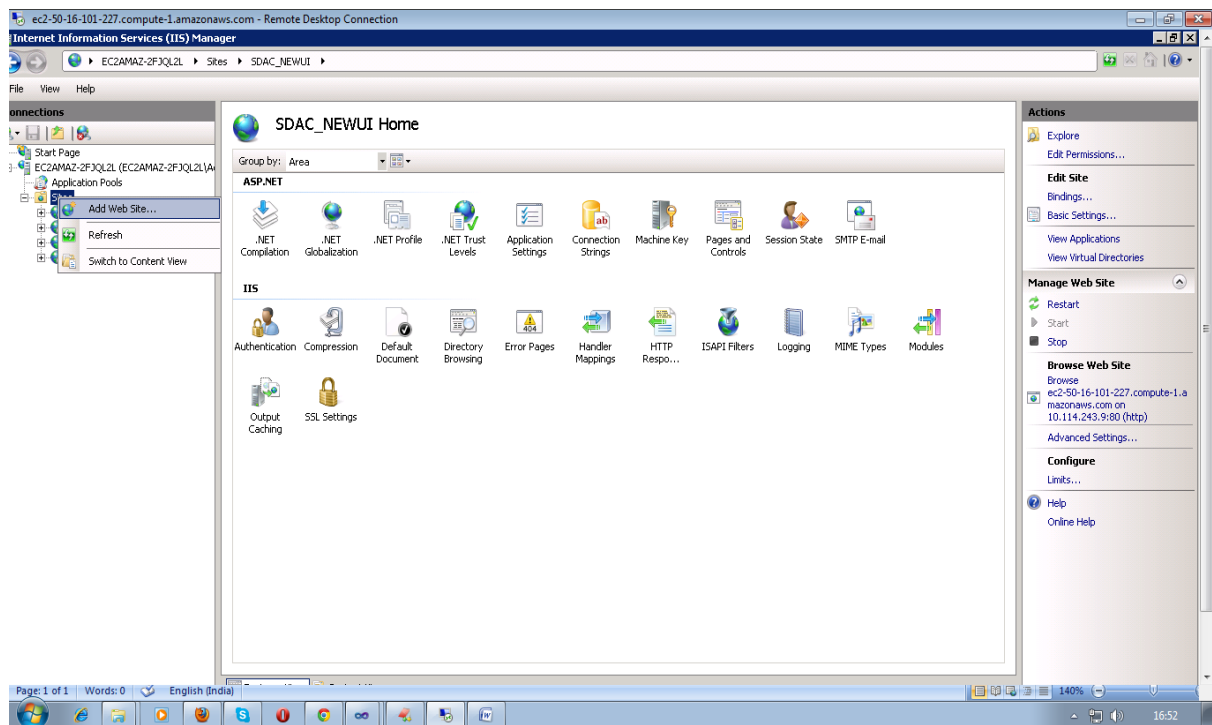
**Step 3:** Copy the published application folder to "C:\inetpub\wwwroot" [default] folder.

**Step 4:** From Run - > inetmgr -> OK or Use Start-> Internet Information Services (Manager)

The following screen will display and is the main page for any application, where you can add your new website or configure your website in IIS.



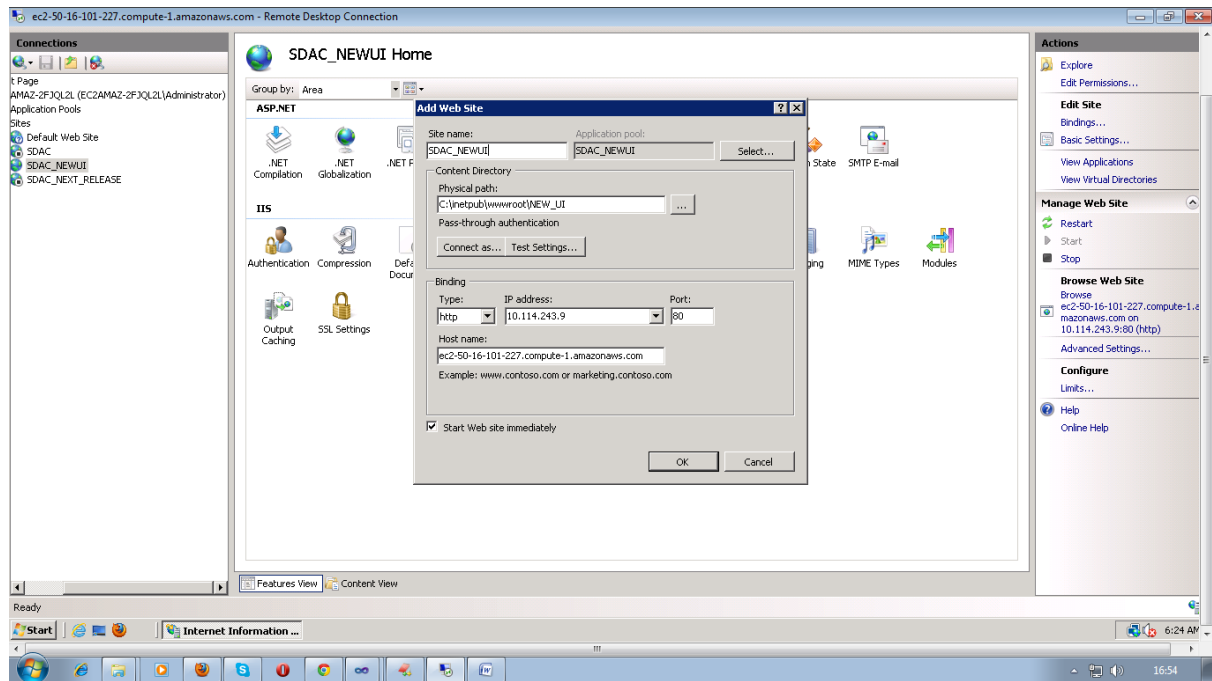
**Step 5:** To add new website Right click on Sites Choose Add Website option



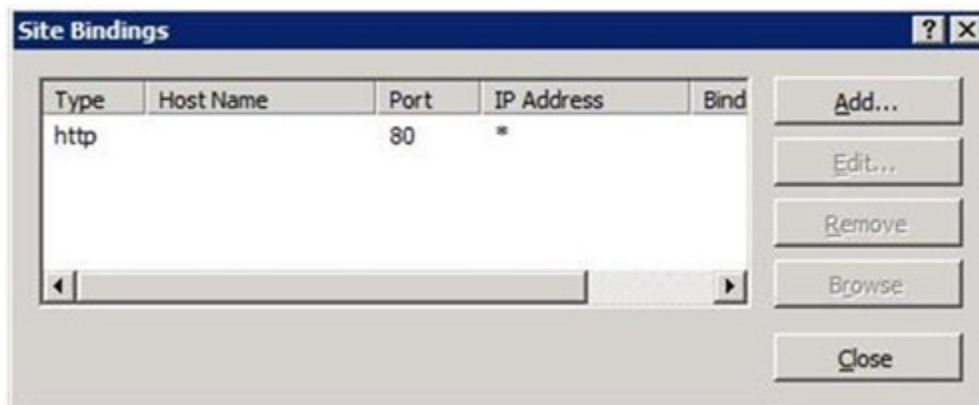
**Step 6:** Type in the Site Name. A description or the domain name is best to put into this field. This will automatically create an Application Pool for the site.

**Step 7:** Enter or browse to the path where the website will have its root directory. This is the directory where the home page should go.

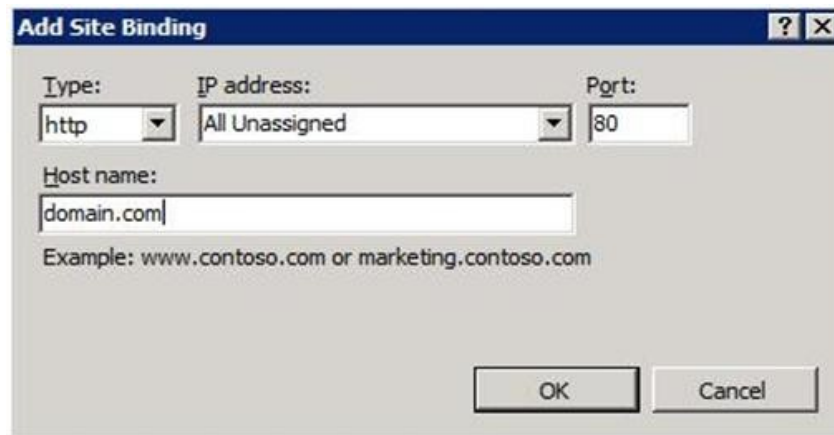
**Step 8:** For the Binding, choose HTTP, choose an IP address that is free, leave the port as port 80, and enter in your domain name (i.e. www.domain.com) as the Host Name. Leave the checkbox for Start Web site immediately checked and click OK.



**Step 9:** Go back to the IIS7 main screen and select the site under the Sites folder. Click Bindings... on the right sidebar so there is a host header for domain.com as well.



**Step 10:** Click the Add button, choose HTTP, use the same IP address as setup in step 7, leave the port as port 80, and enter in your domain name (i.e.domain.com) as the Host Name. Click OK



**Add Site Binding**

Type:  IP address:  Port:

Host name:

Example: www.contoso.com or marketing.contoso.com

OK Cancel

An IP-based website has a unique IP address therefore no other website can use this IP address. If you need multiple IP-based websites, you will need to contact us to add the IP addresses to your server.

A name-based website shares a single IP address with each website on the server. You are able to make as many name-based sites as you wish with your single IP address.

# API Reference

## Overview of API

The Application Programming Interface contains the building blocks that are necessary to create inBloom applications. Application access to the inBloom Data Store is strictly governed by this API.

## Objective

The overall objective of the API is to provide a stable, well-defined interface for software developers to build applications that use the inBloom Data Store and adapt existing applications to leverage the inBloom Data Store. The API is a real-time transactional interface intended for interactive applications. It is not intended to provide bulk data load or file extract and download services.

## Integration Approach

Application integration approach depends on the scope of the application integration effort being proposed.

## How to Use this Reference

The inBloom REST API is the interface that enables you to interact with the inBloom Data Store and services. This reference lists the API resources that developers need when forming the URIs in their REST API calls.

Each resource listing includes the following information:

- A description of the resource and its context
- The URI path
- Request body and response code information for each method
- Links to relevant schema documentation (for the data store URIs)
- Resource-specific fields and parameters
- Links to examples

See the inBloom developer documentation at [inbloom.org/for-developers](http://inbloom.org/for-developers) for more details about the REST API itself and how to develop applications that use API calls, including:

- Example usage scenarios
- URI patterns with examples
- Media types
- Global parameters
- Query operators
- Authentication and authorization



## inBloom REST API

inBloom provides this Restful, client-server Web service that includes these features:

- Predictable, resource-oriented URIs
- HTTP methods to exchange representations of resources
- HTTP response codes to indicate API errors
- Built-in HTTP features such as HTTP basic access authentication
- Support of JSON and XML data-interchange formats
- Hypertext As The Engine Of Application State (HATEOAS) constraint to provide standardized link relations

### inBloom REST API Features

The following table provides an overview of the inBloom REST API for quick reference. The chapters and sections that follow cover the details you need for your application development tasks.

REST API Feature	inBloom Implementation	Explanation
Architecture	RESTful	REST-style client/server architecture with HATEOAS constraint.
Protocol	HTTPS	APIs are exclusively exposed through HTTPS.
Base URL	<code>\$BASE_URL/api/rest</code>	Use this URL to locate REST API resources.
HTTP methods	GET, PUT, POST, DELETE	Standard REST implementation.
Media Types	JSON, XML	To explicitly request a content type in the Accept header, specify one of these: <ul style="list-style-type: none"><li>• <code>application/vnd.slc+json</code></li><li>• <code>application/vnd.slc+xml</code></li></ul>
Response Codes	HTTP	inBloom uses conventional HTTP response codes to indicate success or failure of an API request.
Application Authentication	OAuth 2.0, SAML 2.0	inBloom uses a combination of OAuth 2.0 and SAML 2.0 protocols to allow web-based authentication and authorization, including single sign-on (SSO).
Encryption	HTTPS	Network traffic between an application and the inBloom requires SSL/TLS. Connecting over HTTPS is handled by the inBloom API libraries over standard ports to ensure portability, accessibility, and mobility.
REST API Version	vNURI path element	The version of the API is requested by a vNURI path element, for example:

### Example response for ApplicationAuthorization

The following is the JSON syntax of a response when making a GET request to this URL as an LEA administrator:

```
[{
  "authId": "b2c6e292-37b0-4148-bf75-c98a2fcc905f",
  "authType": "EDUCATION_ORGANIZATION",
  "applds": ["78f71c9a-8e37-0f86-8560-7783379d96f7"],
  "link":
"http://localhost:8080/api/rest/applicationAuthorization/4726e42f-b265-372a-3c17-dc8d5d5fb263",
  "id": "4726e42f-b265-372a-3c17-dc8d5d5fb263"
}]
```

### Example response for applicationAuthorization/{uuid}

The following is the JSON syntax of a response when making a GET request to this URL:

```
{
  "id": "4726e42f-b265-372a-3c17-dc8d5d5fb263",
  "authId": "b2c6e292-37b0-4148-bf75-c98a2fcc905f",
  "authType": "EDUCATION_ORGANIZATION",
  "entityType": "applicationAuthorization",
  "applds": ["78f71c9a-8e37-0f86-8560-7783379d96f7"],
  "metaData": {
    "tenantId": "Midgar",
    "updated": 1352993607658,
    "created": 1332785105123
  }
}
```

### Example response for customRoles

The following is the JSON syntax of a response when making a GET request to this URL:

```
[{
  "id": "4f6718f4-18bc-4a35-83d5-1817e6d72961",
  "roles": [{
    "groupTitle": "Educator",
    "isAdminRole": false,
    "names": ["Educator"],
    "rights": ["READ_GENERAL", "AGGREGATE_READ", "READ_PUBLIC"]
  }, {
    "groupTitle": "IT Administrator",
    "isAdminRole": true,
    "names": ["IT Administrator"],
```

```

    "rights": ["WRITE_RESTRICTED", "READ_GENERAL", "AGGREGATE_READ",
"READ_PUBLIC", "READ_RESTRICTED", "WRITE_GENERAL"]
  }, {
    "groupTitle": "Leader",
    "isAdminRole": false,
    "names": ["Leader"],
    "rights": ["READ_GENERAL", "AGGREGATE_READ", "READ_PUBLIC",
"READ_RESTRICTED"]
  }, {
    "groupTitle": "Aggregate Viewer",
    "isAdminRole": false,
    "names": ["Aggregate Viewer"],
    "rights": ["AGGREGATE_READ", "READ_PUBLIC"]
  }],
  "customRights": [],
  "entityType": "customRole",
  "metaData": {
    "tenantId": "Midgar"
  },
  "realId": "e5c12cb0-1bad-4606-a936-097b30bd47fe"
}]

```

## inBloom REST API Example Requests & Responses

**Request:** Requests the specified resource or collection of resources.

There is no request body for GET. When forming your GET request URI, use REST API's global parameters and URI patterns.

GET <https://localhost/api/rest/v1/sections/{id}>

**Response:** The *TotalCount* header contains the number of items that were returned. The response body contains the requested resource representations, including HATEOAS links to reachable URIs.

**Response:**

```

{
  "id": "{id}",
  "sessionId": "{id}",
  "courseOfferingId": "{id}",
  "populationServed": "Regular Students",
  "sequenceOfCourse": 3,
  "mediumOfInstruction": "Independent study",
  "uniqueSectionCode": "World History II - Sec 6s10",
  "links": [
    {

```

```

    "rel": "self",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}"
  },
  {
    "rel": "custom",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}/custom"
  },
  {
    "rel": "getSchool",
    "href": "https://local.slidev.org/api/rest/v1/schools/{id}"
  },
  {
    "rel": "getEducationOrganization",
    "href": "https://local.slidev.org/api/rest/v1/educationOrganizations/{id}"
  },
  {
    "rel": "getSession",
    "href": "https://local.slidev.org/api/rest/v1/sessions/{id}"
  },
  {
    "rel": "getCourseOffering",
    "href": "https://local.slidev.org/api/rest/v1/courseOfferings/{id}"
  },
  {
    "rel": "getGradebookEntries",
    "href": "https://local.slidev.org/api/rest/v1/gradebookEntries?sectionId={id}"
  },
  {
    "rel": "getStudentSectionAssociations",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}/studentSectionAssociations"
  },
  {
    "rel": "getStudents",
    "href": "https://local.slidev.org/api/rest/v1/sections/
{id}/studentSectionAssociations/students"
  },
  {
    "rel": "getTeacherSectionAssociations",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}/teacherSectionAssociations"
  },
  {
    "rel": "getTeachers",
    "href": "https://local.slidev.org/api/rest/v1/sections/
{id}/teacherSectionAssociations/teachers"
  },
  {

```

```

    "rel": "getStudentGradebookEntries",
    "href": "https://local.slidev.org/api/rest/v1/studentGradebookEntries?sectionId={id}"
  }
],
"programReference": [

],
"schooId": "{id}",
"entityType": "section"
}

```

### Example response for applicationAuthorization

The following is the JSON syntax of a response when making a GET request to this URL as an LEA administrator:

```

[
  {
    "authId": "b2c6e292-37b0-4148-bf75-c98a2fcc905f",
    "authType": "EDUCATION_ORGANIZATION",
    "appls": ["78f71c9a-8e37-0f86-8560-7783379d96f7"],
    "link":
    "http://localhost:8080/api/rest/applicationAuthorization/4726e42f-b265-372a-3c17-dc8d5d5fb263",
    "id": "4726e42f-b265-372a-3c17-dc8d5d5fb263"
  }
]

```

### Example response for applicationAuthorization/{uuid}

The following is the JSON syntax of a response when making a GET request to this URL:

```

{
  "id": "4726e42f-b265-372a-3c17-dc8d5d5fb263",
  "authId": "b2c6e292-37b0-4148-bf75-c98a2fcc905f",
  "authType": "EDUCATION_ORGANIZATION",
  "entityType": "applicationAuthorization",
  "appls": ["78f71c9a-8e37-0f86-8560-7783379d96f7"],
  "metaData": {
    "tenantId": "Midgar",
    "updated": 1352993607658,
    "created": 1332785105123
  }
}

```

### Example response for customRoles

The following is the JSON syntax of a response when making a GET request to this URL:

```

[
  {
    "id": "4f6718f4-18bc-4a35-83d5-1817e6d72961",
    "roles": [

```

```

    "groupTitle": "Educator",
    "isAdminRole": false,
    "names": ["Educator"],
    "rights": ["READ_GENERAL", "AGGREGATE_READ", "READ_PUBLIC"]
  }, {
    "groupTitle": "IT Administrator",
    "isAdminRole": true,
    "names": ["IT Administrator"],
    "rights": ["WRITE_RESTRICTED", "READ_GENERAL", "AGGREGATE_READ",
"READ_PUBLIC", "READ_RESTRICTED", "WRITE_GENERAL"]
  }, {
    "groupTitle": "Leader",
    "isAdminRole": false,
    "names": ["Leader"],
    "rights": ["READ_GENERAL", "AGGREGATE_READ", "READ_PUBLIC",
"READ_RESTRICTED"]
  }, {
    "groupTitle": "Aggregate Viewer",
    "isAdminRole": false,
    "names": ["Aggregate Viewer"],
    "rights": ["AGGREGATE_READ", "READ_PUBLIC"]
  }],
  "customRights": [],
  "entityType": "customRole",
  "metaData": {
    "tenantId": "Midgar"
  },
  "realmId": "e5c12cb0-1bad-4606-a936-097b30bd47fe"
}]

```

## inBloom REST API Example Requests & Responses

**Request:** Requests the specified resource or collection of resources.

GET : <https://localhost/api/rest/v1/sections/{id}>

Response: The *TotalCount* header contains the number of items that were returned. The response body contains the requested resource representations, including HATEOAS links to reachable URIs.

Response:

```

{
  "id": "{id}",
  "sessionId": "{id}",
  "courseOfferingId": "{id}",

```

```

"populationServed": "Regular Students",
"sequenceOfCourse": 3,
"mediumOfInstruction": "Independent study",
"uniqueSectionCode": "World History II - Sec 6s10",
"links": [
  {
    "rel": "self",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}"
  },
  {
    "rel": "custom",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}/custom"
  },
  {
    "rel": "getSchool",
    "href": "https://local.slidev.org/api/rest/v1/schools/{id}"
  },
  {
    "rel": "getEducationOrganization",
    "href": "https://local.slidev.org/api/rest/v1/educationOrganizations/{id}"
  },
  {
    "rel": "getSession",
    "href": "https://local.slidev.org/api/rest/v1/sessions/{id}"
  },
  {
    "rel": "getCourseOffering",
    "href": "https://local.slidev.org/api/rest/v1/courseOfferings/{id}"
  },
  {
    "rel": "getGradebookEntries",
    "href": "https://local.slidev.org/api/rest/v1/gradebookEntries?sectionId={id}"
  },
  {
    "rel": "getStudentSectionAssociations",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}/studentSectionAssociations"
  },
  {
    "rel": "getStudents",
    "href": "https://local.slidev.org/api/rest/v1/sections/
{id}/studentSectionAssociations/students"
  },
  {
    "rel": "getTeacherSectionAssociations",
    "href": "https://local.slidev.org/api/rest/v1/sections/{id}/teacherSectionAssociations"
  },
]

```

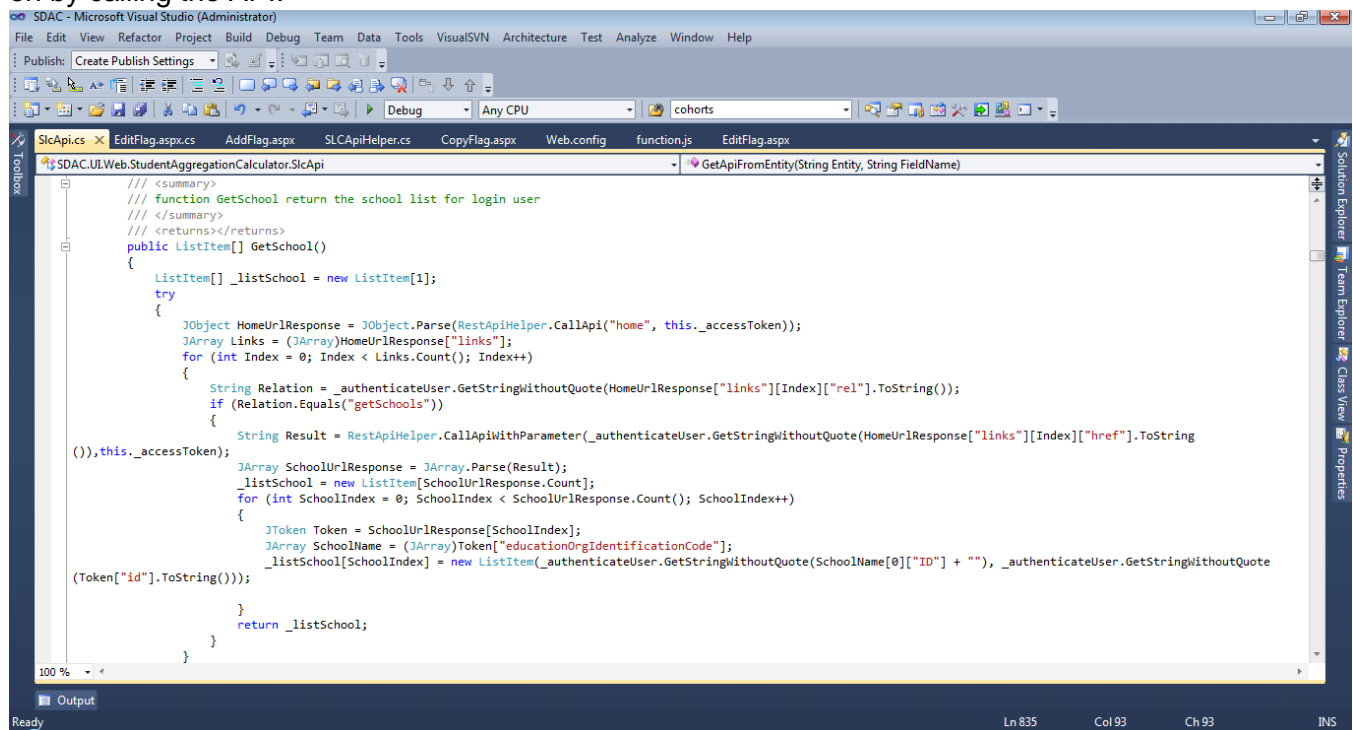
```

{
    "rel": "getTeachers",
    "href": "https://local.slidev.org/api/rest/v1/sections/
{id}/teacherSectionAssociations/teachers"
},
{
    "rel": "getStudentGradebookEntries",
    "href": "https://local.slidev.org/api/rest/v1/studentGradebookEntries?sectionId={id}"
}
],
"programReference": [

],
"schoolId": "{id}",
"entityType": "section"
}

```

The following example shows how the school list is returned for the user currently logged on by calling the API:



```

SDAC - Microsoft Visual Studio (Administrator)
File Edit View Refactor Project Build Debug Team Data Tools VisualSVN Architecture Test Analyze Window Help
Publish: Create Publish Settings
Debug Any CPU cohorts
SLCApi.cs EditFlag.aspx AddFlag.aspx SLCApiHelper.cs CopyFlag.aspx Web.config function.js EditFlag.aspx
SDAC.ULWeb.StudentAggregationCalculator.SLCApi GetApiFromEntity(String Entity, String FieldName)
/// <summary>
/// function GetSchool return the school list for login user
/// </summary>
/// <returns></returns>
public ListItem[] GetSchool()
{
    ListItem[] _listSchool = new ListItem[1];
    try
    {
        JObject HomeUrlResponse = JObject.Parse(RestApiHelper.CallApi("home", this._accessToken));
        JArray Links = (JArray)HomeUrlResponse["links"];
        for (int Index = 0; Index < Links.Count(); Index++)
        {
            String Relation = _authenticateUser.GetStringWithoutQuote(HomeUrlResponse["links"][Index]["rel"].ToString());
            if (Relation.Equals("getSchools"))
            {
                String Result = RestApiHelper.CallApiWithParameter(_authenticateUser.GetStringWithoutQuote(HomeUrlResponse["links"][Index]["href"].ToString()), this._accessToken);
                JArray SchoolUrlResponse = JArray.Parse(Result);
                _listSchool = new ListItem[SchoolUrlResponse.Count];
                for (int SchoolIndex = 0; SchoolIndex < SchoolUrlResponse.Count(); SchoolIndex++)
                {
                    JToken Token = SchoolUrlResponse[SchoolIndex];
                    JArray SchoolName = (JArray)Token["educationOrgIdentificationCode"];
                    _listSchool[SchoolIndex] = new ListItem(_authenticateUser.GetStringWithoutQuote(SchoolName[0]["ID"] + ""), _authenticateUser.GetStringWithoutQuote(Token["id"].ToString()));
                }
                return _listSchool;
            }
        }
    }
}

```

To get the API call from the entity we can call APIs for example to get the Student Academic Record we have the API "studentAcademicRecords?studentId={StudentId}".



How to call an API according to the entity is shown as follows:

```

SDAC - Microsoft Visual Studio (Administrator)
File Edit View Refactor Project Build Debug Team Data Tools VisualSVN Architecture Test Analyze Window Help
Publish: Create Publish Settings
Debug Any CPU cohorts
EnumHelper.cs ConfigurationHelper.cs SessionEnum.cs QueryStringTokenEnum.cs ConfigurationSettingEnums.cs SLCApi.cs EditFlag.aspx.cs AddFlag.aspx SLCApiHelper.cs CopyFlag.aspx
SDAC.UI.Web.StudentAggregationCalculator.SLCApi
GetApiFromEntity(String Entity, String FieldName)
/// <summary>
/// Get the api call according to entity
/// </summary>
/// <param name="Entity"></param>
/// <param name="FieldName"></param>
/// <returns></returns>
public String GetApiFromEntity(String Entity, String FieldName)
{
    String Api="";
    switch (Entity)
    {
        case "Student": Api = ""; break;
        case "StudentAcademicRecord": Api = "studentAcademicRecords?studentId={StudentId}"; break;
        case "StudentSchoolAssociation": Api = "students/{StudentId}/studentSchoolAssociations"; break;
        case "StudentAssessment" : Api = "students/{StudentId}/studentAssessments"; break;
        case "StudentGradebookEntry": Api = "students/{StudentId}/studentGradebookEntries"; break;
        case "StudentParentAssociation": Api = "students/{StudentId}/studentParentAssociations"; break;
        case "Parent": Api = "students/{StudentId}/studentParentAssociations/parents"; break;
        case "Assessment": Api = "students/{StudentId}/studentAssessments/assessments"; break;
        case "StudentProgramAssociation": Api = "students/{StudentId}/studentProgramAssociations"; break;
    }
}

```

To get the API call with the parameter the method which is used contains the API call, student Id, school Id, course Id, and Section Id by which the data is fetched:

```

SDAC - Microsoft Visual Studio (Administrator)
File Edit View Refactor Project Build Debug Team Data Tools VisualSVN Architecture Test Analyze Window Help
Publish: Create Publish Settings
Debug Any CPU cohorts
Section.cs School.cs EnumHelper.cs ConfigurationHelper.cs SessionEnum.cs QueryStringTokenEnum.cs ConfigurationSettingEnums.cs SLCApi.cs EditFlag.aspx.cs AddFlag.aspx
SDAC.UI.Web.StudentAggregationCalculator.SLCApi
RunFlag(String FieldName, String DataType, bool ResponseType, String UserId, String SchoolId, String CourseId, String SectionId)
/// <param name="CourseId"></param>
/// <param name="SectionId"></param>
/// <returns></returns>
public String GetApiWithParameter(String ApiCall,String StudentId, String SchoolId, String CourseId, String SectionId)
{
    if (ApiCall.Contains("{SchoolId}"))
    {
        ApiCall = ApiCall.Replace("{SchoolId}", SchoolId);
    }
    if (ApiCall.Contains("{CourseId}"))
    {
        ApiCall = ApiCall.Replace("{CourseId}", CourseId);
    }
    if (ApiCall.Contains("{SectionId}"))
    {
        ApiCall = ApiCall.Replace("{SectionId}", SectionId);
    }
    if (ApiCall.Contains("{StudentId}"))
    {
        ApiCall = ApiCall.Replace("{StudentId}", StudentId);
    }
    return ApiCall;
}
/// <summary>

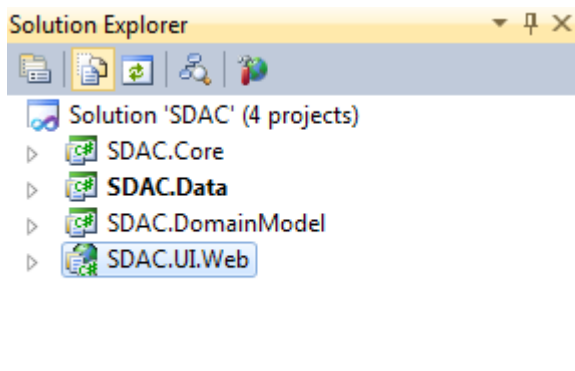
```

# Source Code Directory Structure

This directory holds the all the main system source code and has the following files and directories.

It contains two projects:

- SDAC.Core
- SDAC.Data
- SDAC.DomainModel
- SDAC.UI.Web



## SDAC.Core

This includes core logic of the application and following are the files and folders.

- Properties
- Reference

This contains the required references for the entity model.

- Classes

These are classes which will contain business logic of the application.

## SDAC.Data

This includes .NET wrapper integration logic of the application and following are the files and folders.

- Properties
- Reference

This contains the required references for the entity model.

- Classes

These are classes which will interact with RESTful APIs from the .NET wrapper of the application.

See the inBloom and Upeo .NET wrapper detailed documentation at

<https://github.com/upeo>.

## SDAC.DomainModel

This includes the entity model and contains the following files and folder.

- Properties

- Reference

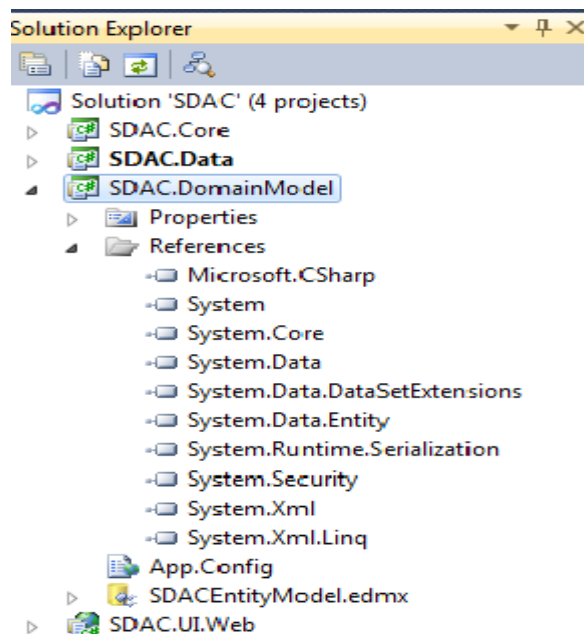
This contains the required references for the entity model.

- App.config file

This contains the connection string of the entity model.

- SDACEntityModel.edmx File

An .edmx file is an XML file that defines a conceptual model, a storage model, and the mapping between these models. An .edmx file also contains information that is used by the ADO.NET Entity Data Model Designer (Entity Designer) to render a model graphically. The recommended practice for creating an .edmx file is to use the Entity Data Model Wizard. Changes are made to an .edmx file when you use the Entity Designer to modify your model and when you use the Update Model Wizard to update your model based on changes to the underlying database.



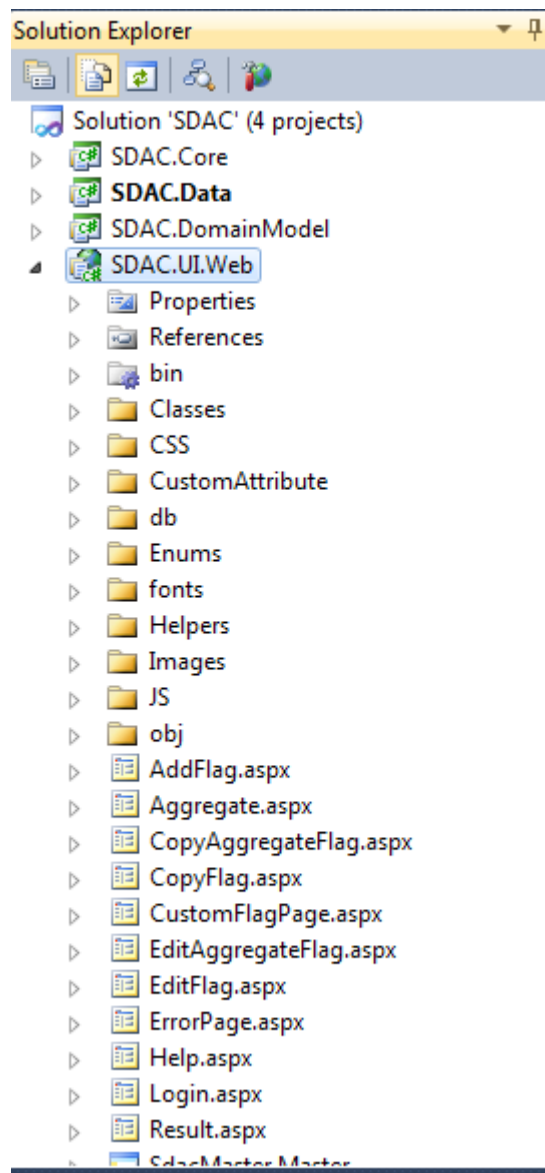
## SDAC.UI.Web

Contains the following folders which include the classes, images, CSS files, JavaScript and UI files.

- Properties
- References
  - This contains the references which are required for the solution.
- Bin
  - Contains the DLL files required for the project.
- Classes
  - This folder contains the API related classes including authenticate user, call for API, get data through API.
- CSS
  - It contains all CSS files.
- Enum
- Helpers
- Images
  - It contains all of the images used in the project.
- JS
  - This contains the JavaScript files.
- Obj


## UI Files (aspx pages)


1. AddFlag
2. AggregateFlag
3. CopyAggregateFlag
4. CopyFlag
5. CustomFlagPage
6. EditAggregateFlag
7. EditFlag
8. ErrorPage
9. Help
10. Login
11. Result
12. Search
13. SDAC Master page



## Error Page

The default error page will display when an unexpected error occurs in the application. The page will display the specific error message as to why the application has stopped. This information may be useful when debugging the application and to determine the cause of the error.

SDAC

Welcome Mrs Linda Kim 

### Student Data Aggregation Calculator

[Help](#)

#### Application Error

The application has to stop due to an error:

<insert error message or stack trace>

Please contact the system administrator for help.

(C) Shared Learning Collaborative, LLC.

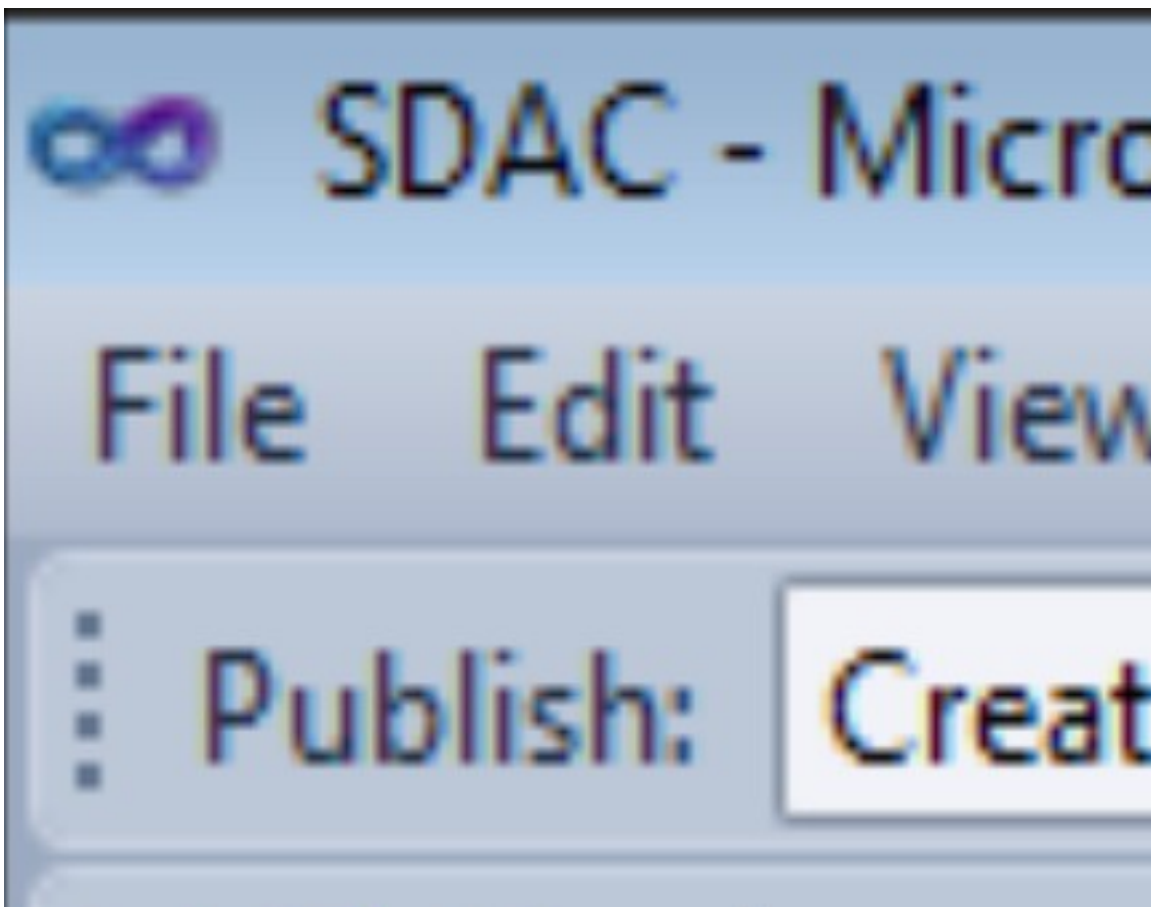
## Custom Entity

Entities stored in the inBloom Data Store include an option for persisting custom data. When custom data is an option for an entity, one can use the `includeCustom` global parameter when forming API calls with the URI for that entity. The data type associated with custom data fields in the inBloom REST API is the JSON binary large object (blob) type.

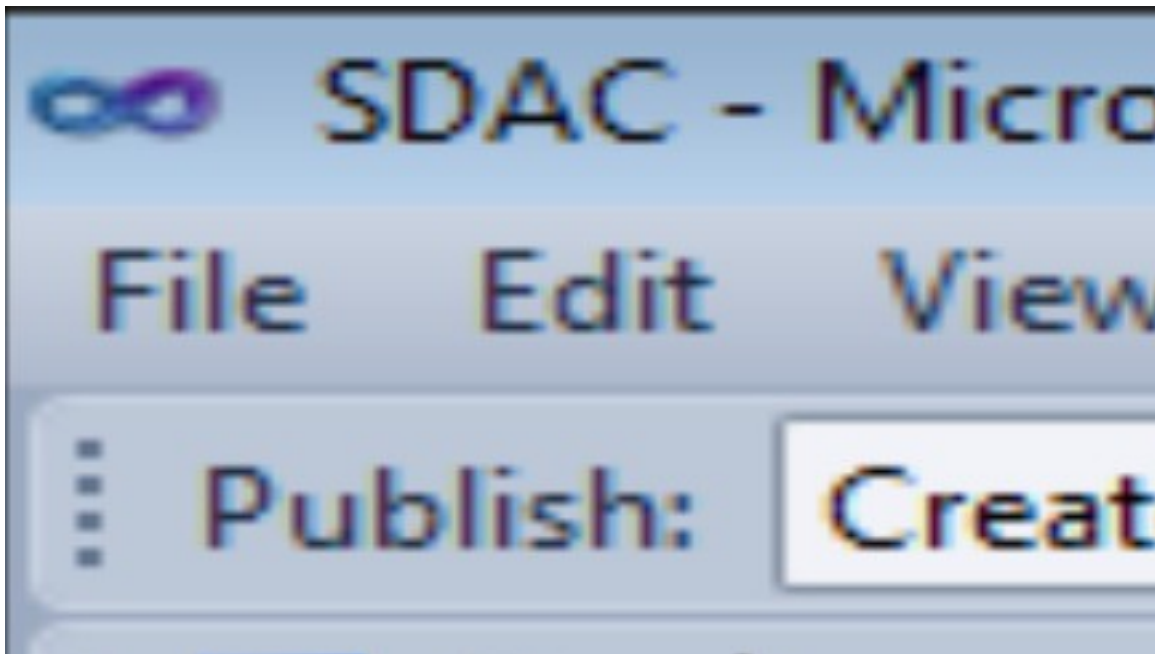
Only a user with admin privileges can access the custom data. To permit non-admin users with this ability, set the `WRITE_GENERAL` permission for each role in the Admin Settings of the inBloom portal.

### How to Retrieve Custom Entity Data

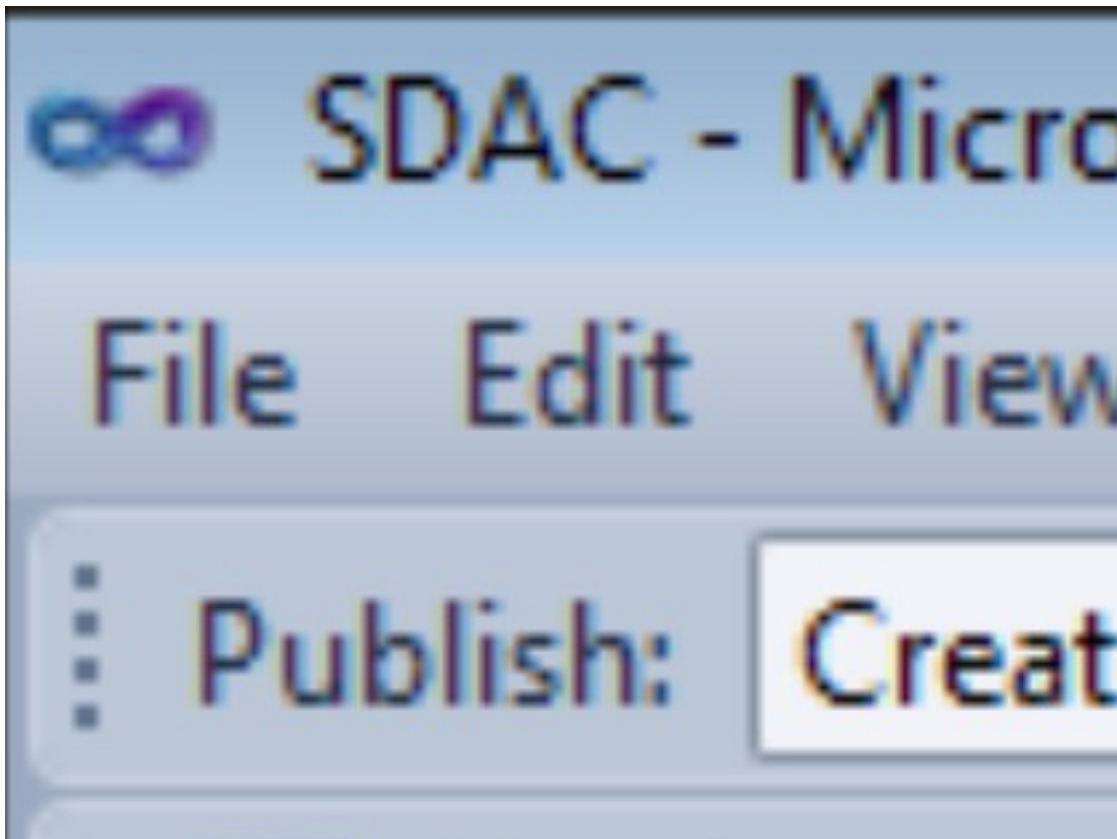
Get custom data associated with the entity



Get the flags custom data using the variables



Once the flag is added successfully we get the custom entity data.





We get the custom entity data shown as follows:



Custom data associated with an entity can be created and queried through the custom end point for the entity. One can use GET & PUT methods to manage the data.