# Data Match Enterprise API Installation Guide

## Before the starting

To keep some backward compatibility with previous version of DME API folders structure left without changes.

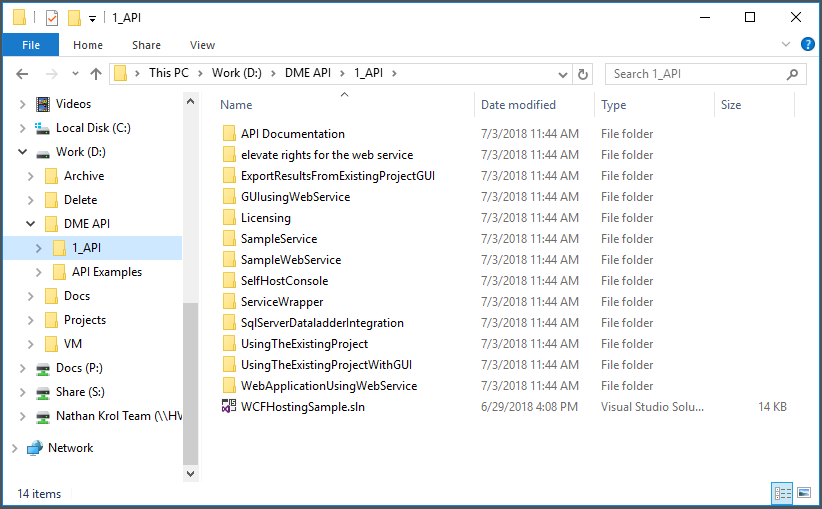


Fig. 1. Directories structure of Data Match API.

The folder API Examples contains only subfolder “dlls” with the next files (Files related to API are marked with green color):

Table 1. Data Match Enterprise API libraries.

|  |  |
| --- | --- |
| **File Name** | **Description** |
| \_3GramsHashes.dll | Contains special subroutines for calculating hashes. |
| AccuAddress.dll | Library that is related to Address Verification module. |
| DataMatch.Api.dll | Provides high level interface for developers and includes such entities as project information and data source info. |
| DataMatch.Collections.dll | Module that provides the functionality for working with huge collections of standard .NET types in multi-threaded way. Also sorting functionality is exposed. |
| DataMatch.Core.dll | The main part of API. Includes common shared entities that are used in other libraries. Licensing system, Paths settings, widespread interfaces and delegates are placed in this .dll file. |
| DataMatch.Data.dll | All the functionality related to import and export of data is located there. Different connectors and data source configurations you can find in this library. |
| DataMatch.DataStorage.dll | This module includes API for working with Data Match internal storages. OnDriveTable and InMemoryTable are located here. |
| DataMatch.Matching.dll | A part of functionality responsible for match configuration, match definitions, matching process is located in this library. |
| DataMatch.Project.dll | Base classes responsible for storing projects, data sources, Data Match application configuration are placed here. |
| DataMatch.Transformation.dll | Base classes required for data cleansing and transformation are located in this dll. |
| IBM.Data.DB2.dll | Required for IBM DB2 import/export. |
| ICSharpCode.SharpZipLib.dll | SharpZipLib is a compression library that supports Zip files. |
| iGeoCd.dll | A part of Address Verification module. |
| Interop.MSDASC.dll | Necessary for OLE Universal connector |
| log4net.dll | Logging system |
| MySql.Data.dll | Required for import/export data from/to MySql. |
| NPOI.dll | Library for reading Excel files. |
| Oracle.DataAccess.dll | Required for Oracle connector. |
| SimMetrics.dll | Contains routines for calculating string similarity. |
| Trinet.Core.IO.Ntfs.dll | Utilities for working with alternate data streams on NTFS file systems. |

## Getting started

Data Match Enterprise API also contains Visual Studio solution that includes several projects with samples of API usage and best practices.

### 2.1. Pre-requirements for using examples:

* Microsoft Visual Studio 2015 or later;
* Microsoft .NET Framework 4.5.2;
* Microsoft Windows Vista SP2, Windows 7 SP1, Windows 8, 10 x86, x64; Windows Server 2008 and higher (x86, x64);
* Processor: Minimum: 2 GHz Dual Core, Recommended: 3.0 GHz Quad Core;
* RAM: Minimum: 4 GB; Recommended 16+ Gb
* HDD: Using of SSD will increase the performance significantly;

To get started unpack DME\_API archive into your working directory and run the solution WCFHostingSample.sln.

## Samples explanation

### Export Results From Existing Project GUI

This example shows direct using of API. Build this project and make sure that all DLLs from table 1 are present in output directory. (bin\Debug or bin\Release).

Also output directory should contain configuration.ini file with the next content:

**[AppSettings]**

pathForRegistrationFile= C:\ProgramData\DataMatch Enterprise\Registration\

projectsPath= c:\Users\[USER\_NAME]\Documents\DataMatch Enterprise\projects

dataPath=D:\enterprise\API persistent

tempDataPath=D:\enterprise\API temp

**[PkFieldName]**

After running application and opening some DME project you’ll see something like on the fig. 2:

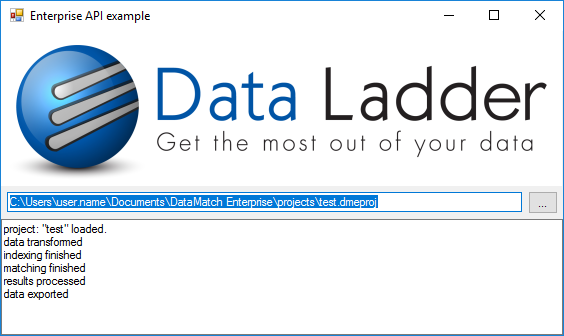


Fig. 2. Appearance of the first GUI sample project.

After running the application matched data will be exported to database defined by connection string that is defined in source code:

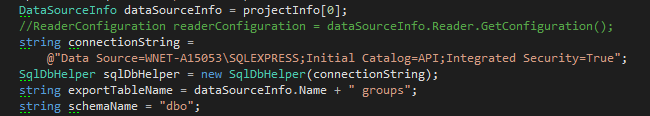


Fig. 3. Connection string definition in the source code.

### GUI Using Web Service

Pre-requirements: before running this example SelfHostConsole application should be executed. Second project is a host for WCF Service that interacts with API.

SelfHostingConsole project contains App.config file where you can define custom bindings. By default, base address is: http://localhost/SelfHostedServiceConsole . The same address should be declared as URL of ServiceClient in the file ServiceClient.cs of the first project (GUIusingWebService).

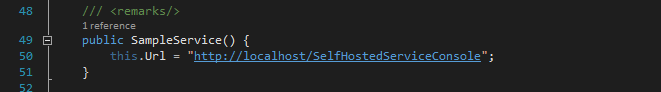


Fig. 4. Defining of WCF Service address for ServiceClient.cs in GUIusingWebService.

SelfHostingConsole project also requires special configuration. First of all, ensure that output directory contains all required DLLs (see table 1). Also a file webservice.ini should be present in this folder with the next configuration (values of variables should be changed according to your system configuration):

**[AppSettings]** (except connection string all other values are mandatory)

connectionstring=Server=[SERVER];Database=[DATABASE];Trusted\_Connection=True;

pathForRegistrationFile=C:\ProgramData\DataMatch Enterprise\Registration\

projectsPath=c:\Users\user.name\Documents\DataMatch Enterprise\projects

dataPath=D:\enterprise\API persistent

tempDataPath=D:\enterprise\API temp

**[PkFieldName]** (mandatory if insert/update/delete is performed on the database table)

Example1=ID

Companies1M=ID

s2tech=SSN

ttt3=fullname

test=fullname

**[RAM MB per project]** (mandatory for all DME projects used by service, integer value is in kB)

If system’s free memory is less than value service uses, system returns a message about this. It continues working after the memory is available again)

business\_names=1024

person\_names=2048

DCSFMatch\_API=4096

ttt3=2048

test=2048

**[RAM MB]** (both mandatory)

AllInMemory=true - for now should always be true

minFree=4096 – won’t start if this amount in kB is not available.

Also it might be necessary to grant network privileges for WCF Service. For this purposes you can use windows command file that is located in the next subdirectory: elevate rights for the web service. You can edit run.bat file according to your system configuration.

After running the application if everything goes correctly you’ll see the next screen:

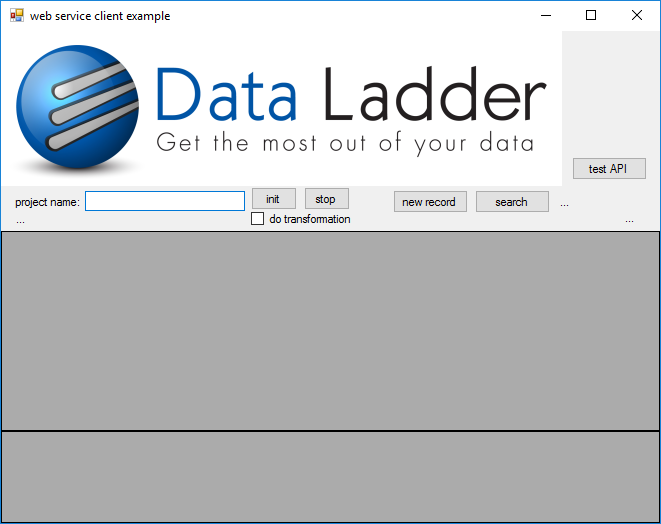


Fig. 5. GUI sample application that uses web service.

In the “project name” textbox you should define project name from your DME directory with projects that is defined in webservice.ini file in projectsPath section. In the section **[PkFieldName]** you should define a field for a project by which you want to search in opened project.

Click “new record” button to add a field where you can define text to search. Then enter the text to search and click “search” button. You’ll see search results in the tabular format:

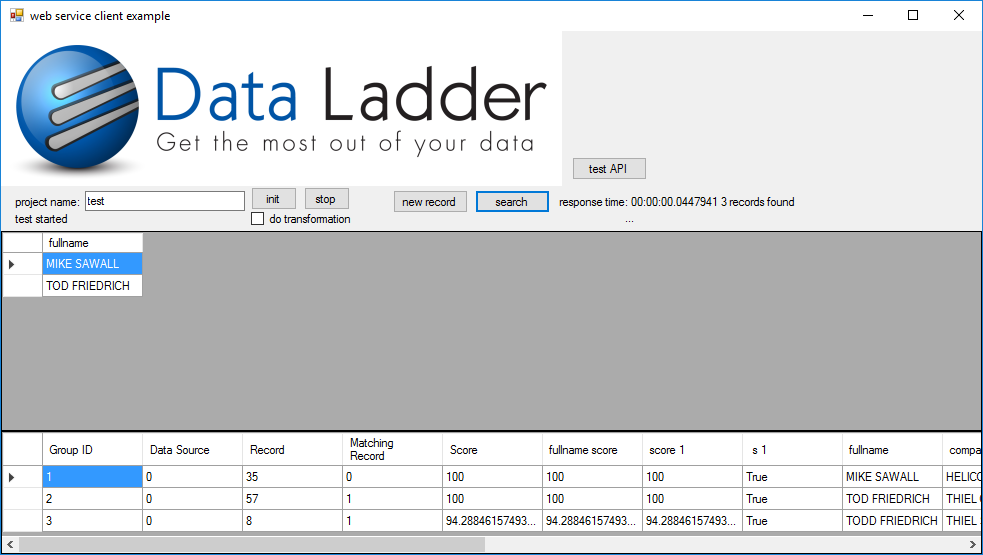


Fig. 6. Search results.

### Licensing

This example demonstrates how to deal with DME API licensing system. You can get your PC Signature using this program:

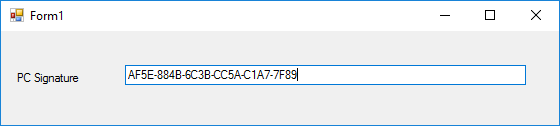


Fig. 7. Licensing demo

There is no special configuration for this project.

### Sample Service

This is a sample of WCF service that consumes DME API functionality and can be used in your WEB and Desktop applications. Configuring of this service is usually assigned to host and explained in the section 3.2. in the part related to SelfHostConsole configuration.

SampleService project includes App.config file that contains just one important part, – address binding:

…

<baseAddresses>

<add baseAddress="http://localhost:8732/Design\_Time\_Addresses/SampleService/Service1"/>

</baseAddresses>

…

This address is not used if you configure host application or service for this WCF.

### Self Host Console

This project is a console application that serves for hosting WCF service. You can use IIS or Windows service instead.

Configuring of this application was detailed described in the section 3.2.

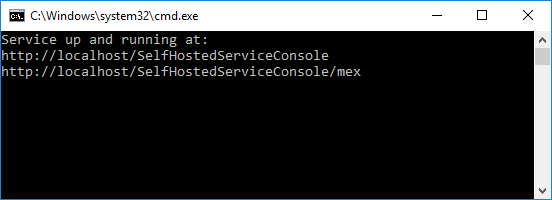


Fig. 8. Appearance of host console application for WCF service

This is a standard approach that is used for WCF Services development. You can check if the server executed successfully just opening the link from console in your web browser (see fig. 9).

Running of this program is a pre-requirement for most of demo samples in this VS project.

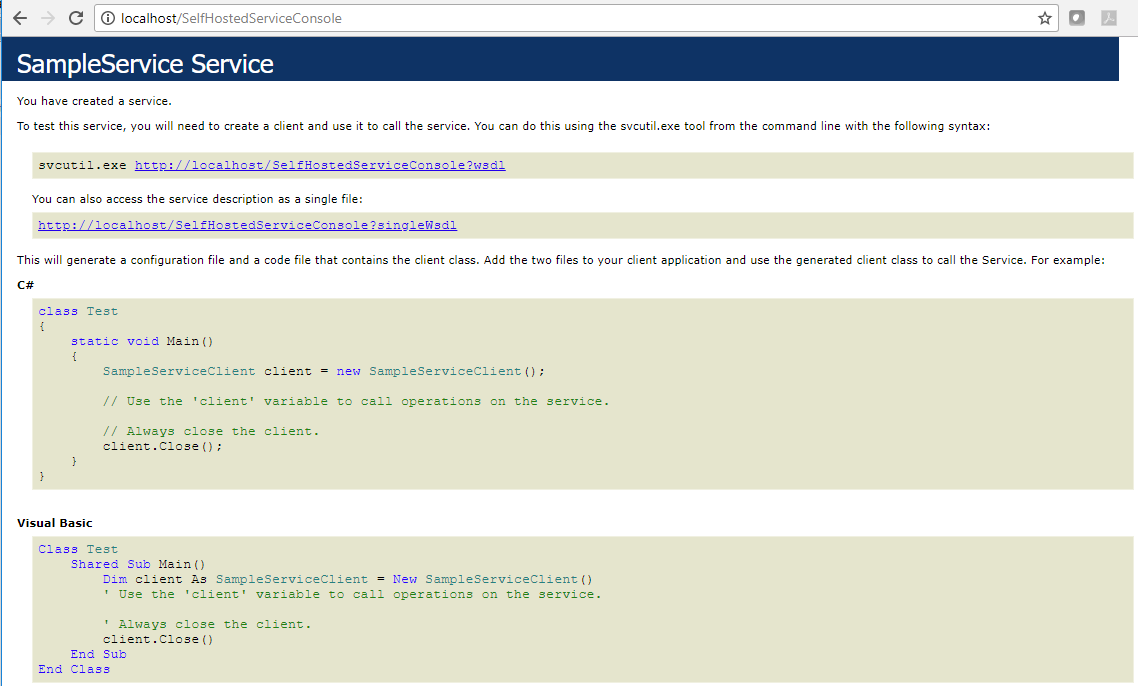


Fig. 9. WCF Service in a web browser

### Using the Existing Project

This example is the simplest console application demonstrating API functionality. It loads some DME project from hardcoded path, performs transformation, matching and exports results to table in DB defined by hardcoded connection string:

…

projectInfo.Load(@"C:\Users\user.name\Documents\DataMatch Enterprise\projects\test.dmeproj");

…

string connectionString = @"Data Source=WNET-A15053\SQLEXPRESS;Initial Catalog=API;Integrated Security=True";

…

The results you can see on the next screenshot:

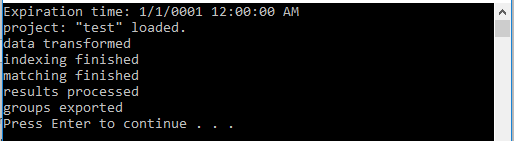


Fig. 10. Simplest console application that demonstrates the main API functionality

### Live Search Demo and Real Time Duplicate Check

### (Using the Existing Project With GUI)

This is WinForms demo application that uses DME API directly, without using of service. The program is intended for searching text in Data Match Enterprise projects. Please note that **only the projects with single data source can be used.**

Application uses both configuration.ini and webservice.ini files.

The first file is similar to the one described in the section 3.1.

**[AppSettings]**

pathForRegistrationFile=C:\enterprise API\registration

projectsPath=c:\Users\user.name\Documents\DataMatch Enterprise\projects

dataPath=c:\enterprise\API persistent

tempDataPath=c:\enterprise\API temp

**[PkFieldName]**

The second file should be configured in the next way (your project names will definitely be different):

**[AppSettings]**

connectionstring=Server=WNET-A15053\SQLEXPRESS;Database=API;Trusted\_Connection=True;

pathForRegistrationFile=C:\enterprise API\registration

projectsPath=c:\Users\[USER\_NAME]\Documents\DataMatch Enterprise\projects

dataPath=C:\enterprise API\data

tempDataPath=C:\enterprise API\tmp

**[PkFieldName]**

Example1=ID

Companies1M=ID

### Live Search Demo

If everything configured properly you can run the example:

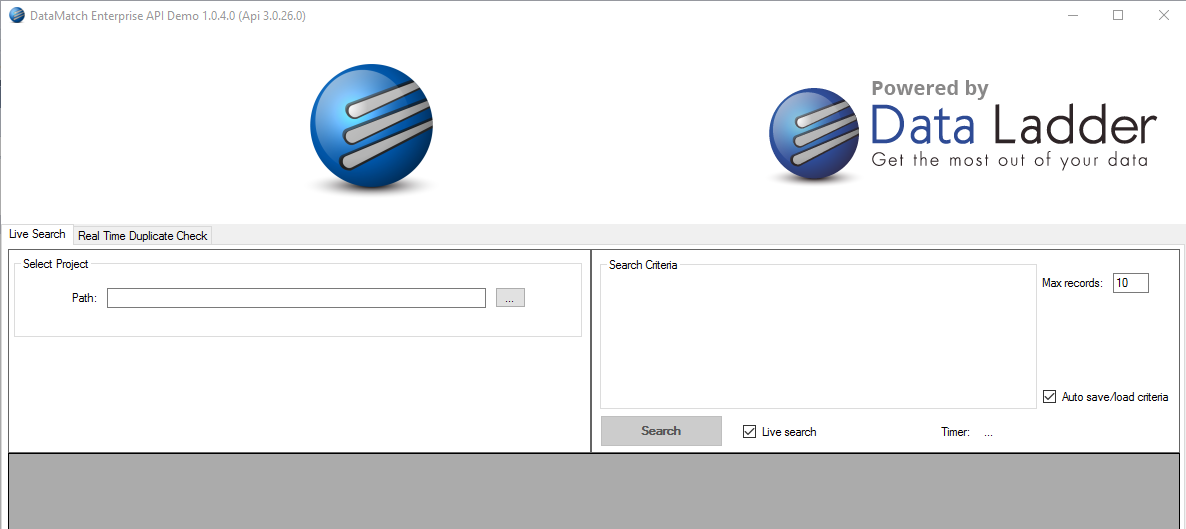


Fig. 11. Main application form appearance

You can prompt project file using standard open dialog clicking on the button …

Then it’s possible to search in matched results using live search or simple search by demand:

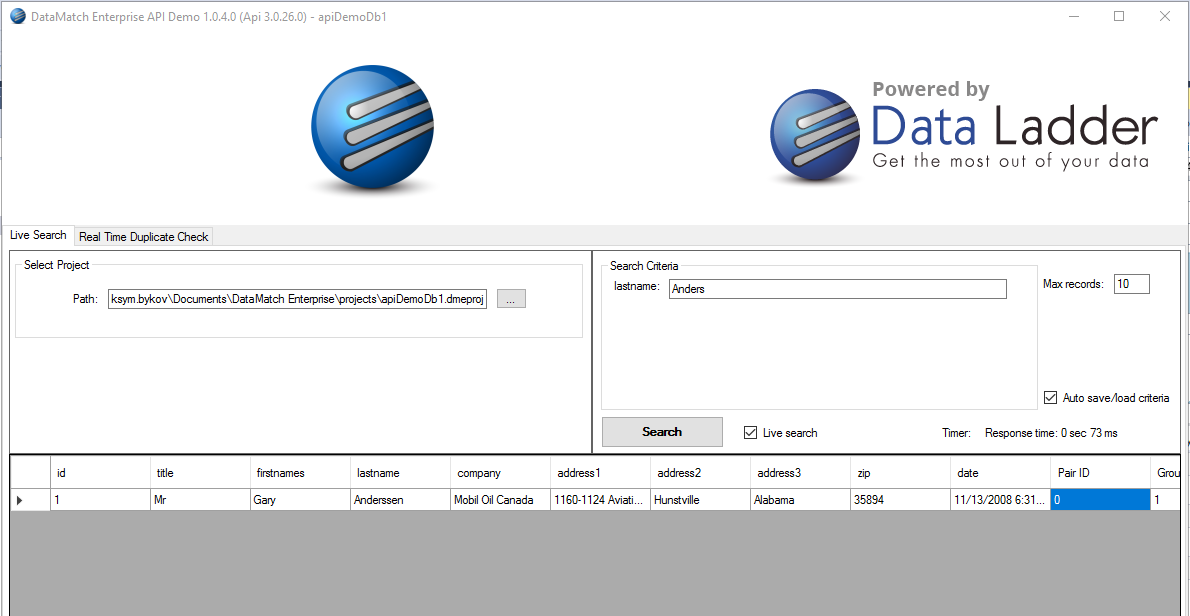


Fig. 12 Live Search Demo application in action

Please note that you should define at least 1 match criteria in the project you want to search in. Textboxes in the top-right corner are corresponded to fields used in match criteria.

Search results are displayed in the tabular result. You can define max record count to show to increase the speed of the search.

### Real Time Duplicate Check

This is second example that this project contains. This is second tab of the application. It implements checking records before they will be inserted in the database. If the record is not similar any existing record it will be inserted in the database. If it is similar it will be rejected.

On Fig.13 the attempt inserting similar record is shown.

Content of database is shown in the bottom data table. In the middle of page duplicates or inserted data are shown.

Also there are two thresholds ’Auto Match’ and ‘Manual Review ‘. If score similarity of new records is more than it is set by ‘Auto Match’ the record will be rejected. If score less than ‘Manual Review’ it will be inserted. If the score is between thresholds, then manual accepting is needed for inserting the record into the database

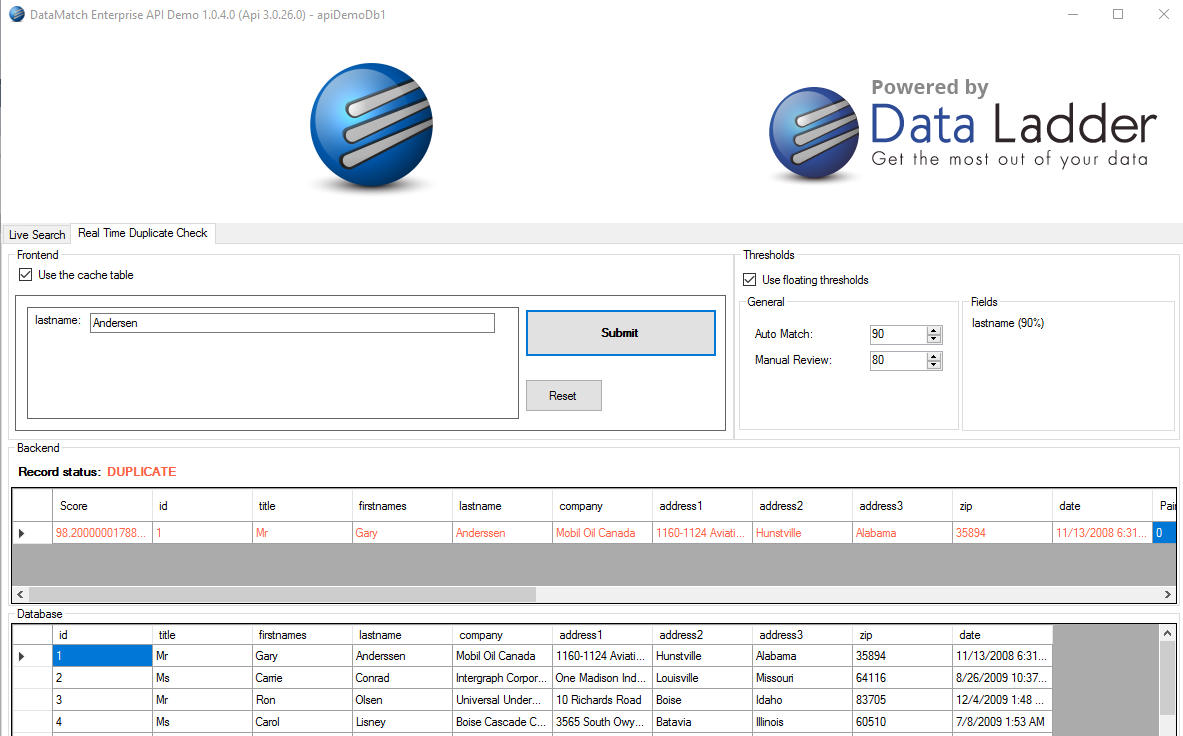


Fig. 13. Real Time Duplication Check application in action (suppressing a duplicate record)

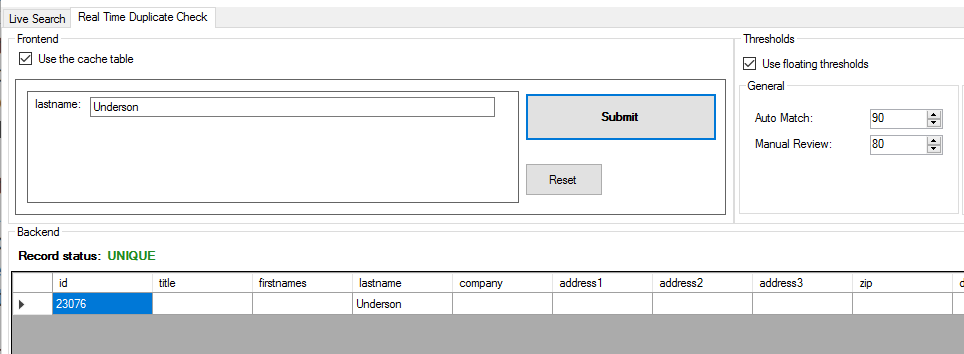


Fig. 14. Real Time Duplication Check application in action (inserting the unique record)

### Real time examples with Web Application Using Web Service

This example shows how DME API can be used in Web applications. This case is most justified for using WCF Service.

First of all, address binding should be defined in Web.config file:

<system.serviceModel>

<bindings>

<basicHttpBinding>

<binding name="BasicHttpBinding\_ISampleService"/>

</basicHttpBinding>

</bindings>

<client>

<endpoint address="http://localhost/SelfHostedServiceConsole"

binding="basicHttpBinding"

bindingConfiguration="BasicHttpBinding\_ISampleService"

contract="SelfHostedService.ISampleService"

name="BasicHttpBinding\_ISampleService"/>

</client>

</system.serviceModel>

In this section address should correspond to the address defined in the app.config of SelfHostedConsole project (see chapter 3.2).

SelfHostedConsole application should be executed before opening web site in some browser.

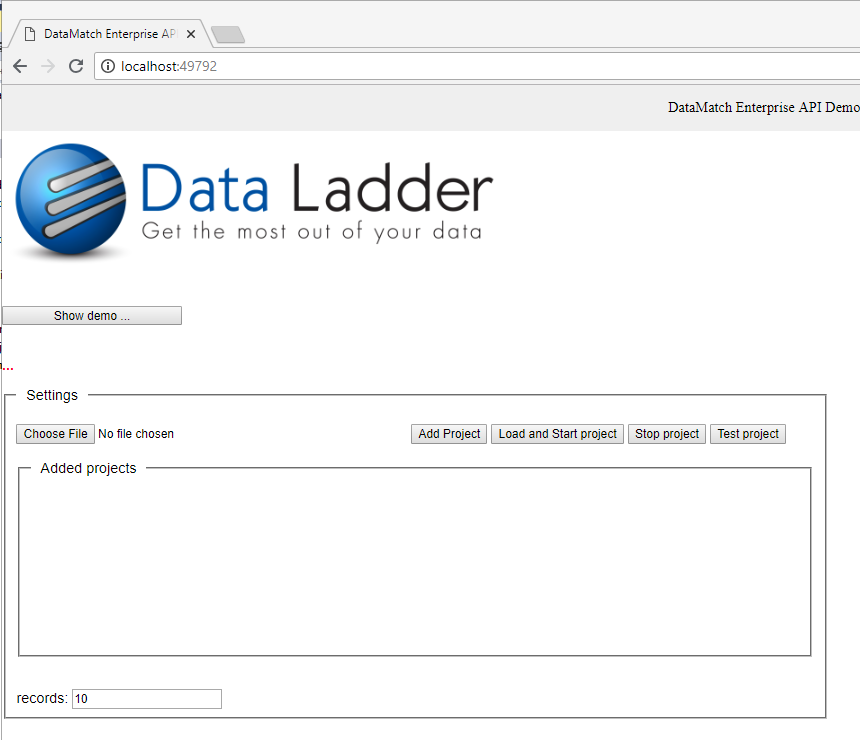


Fig. 15. Sample Web application that uses DME API

This example is similar to GUI Using Web Service, but uses ASP.NET instead of WinForms. Also, a functionality is extended to ‘Live Search’ and ‘Real Time Suppresing’

You can locate your project in the list of group boxes and click “Choose File” button. Then “Add Project”, choose radiobutton and after that “Load and Start project”. Started project is shown on Fig.16

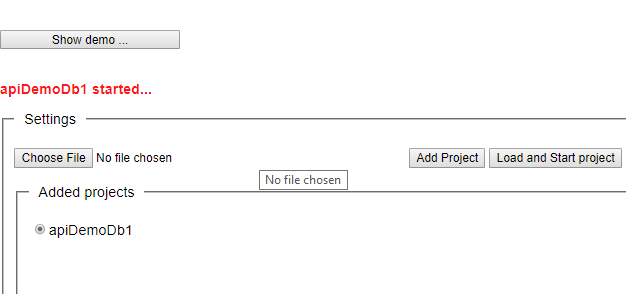


Fig. 16. Web Application Demo started an ‘apiDemoDb1’ project

After the project is loaded you can go to real-time examples: Live Search and Suppressing Management by pressing ‘Show demo...’ button.

The functionality of this demo is similar to the functionality described in p.3.7.1 and p.3.7.2.

On fig.17 results of the live search are shown

For inserting a new record into the database you should type values in text fields in the section Fields and then press button ‘Write’. If a record is unique it will be inserted. How it works is described in p.3.7.2. Also, you can change values of thresholds.

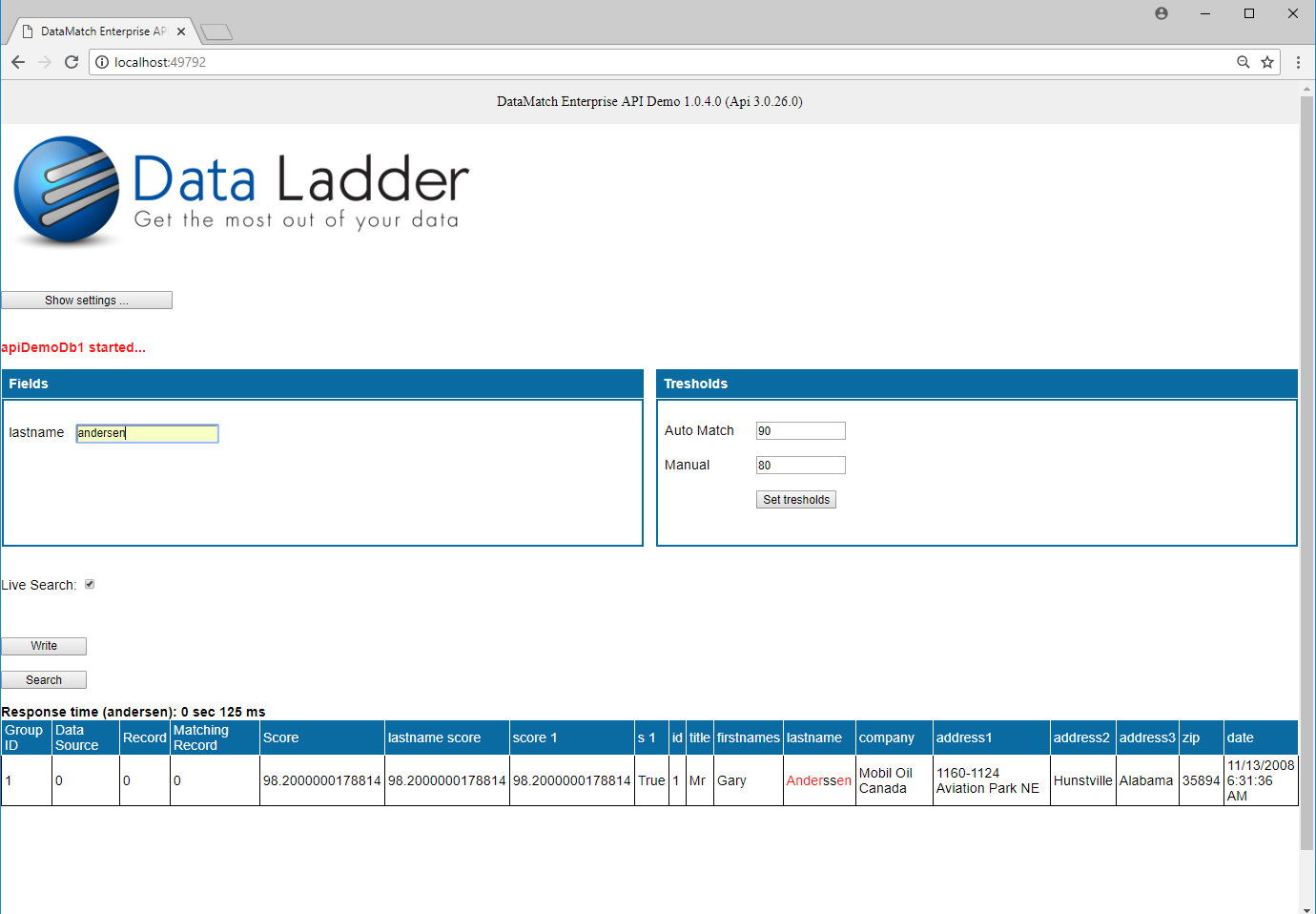


Fig. 17. Web application that uses DME API for searching data in Data Match projects (Results of Live Search)

### Web Suppressing Service for any database

This example consists of three projects.

The first project is a web application that provides functions of web service. This service can receive commands, data for matching and can write results of matching into the database. Service receives commands by SOAP so any database that can use SOAP can use this service

The second project is created for interacting with the web service. And this project is loaded into MS SQL Server. It allows using of web service from SQL Server.

The third project is SQL CLR assembly that work as a bridge between SQL Server objects (triggers, stored procedures) and the second project

### Sample Web Service

This project provides suppressing management service over HTTP.

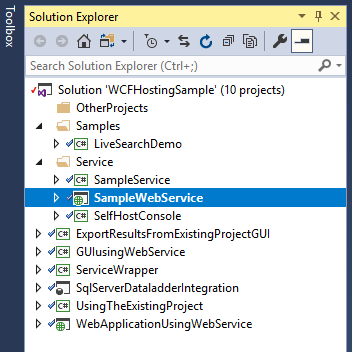


Fig. 18. The web application project inside Visual Studio

In initial stage project loads data from data source. Later, the service receives inserted records and verify are in the data source similar records or not. Results of the check are written into predefined database table.

The table must have definite structure. It must have columns: ‘id’(bigint), ‘StrongDuplicate’(int), ‘Duplicate’(int), how it shown on Fig.19. The table can have other columns but these three columns must exist

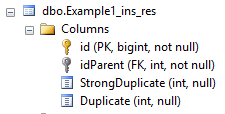


Fig.19 The web application project inside Visual Studio

In initial stage project also loads settings. They defined into ‘Properties.Settings’. Settings define which DME projects will be used, where find the table for results, etc. Settings are shown in Fig.20

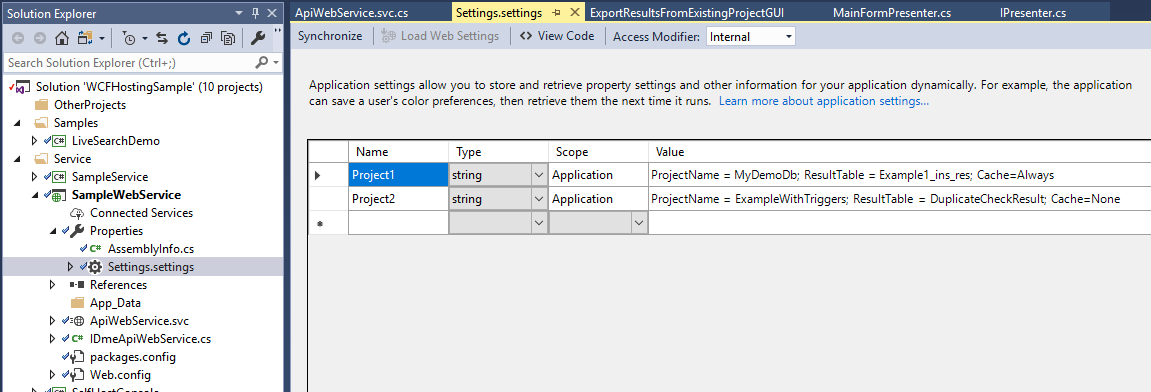


Fig. 20. Settings of Web Application project

The application can work with a few data sources in same time. Every string of settings defines one data source. The string must have such structure:

*“ProjectName = MyDemoDb; ResultTable = Example1\_ins\_res; Cache=Always”*

***ProjectName*** – name of DME project, where the data source, matching fields are defined; ***ResultTable*** – the table for results, it is described above; ***Cache*** – how the application should add checked records into the table with cached data source after checking.

***Cache*** can take one of the next value: *Allways*, *ExceptStrongDuplicates*, *ExceptDuplicates*, *None*.

*Allways* – every record will be added, *ExceptStrongDuplicates* – record that don’t have strong duplicates will be added, *ExceptDuplicates* – record that don’t have strong duplicates and don’t have duplicates will be added, *None* – record will not be added.

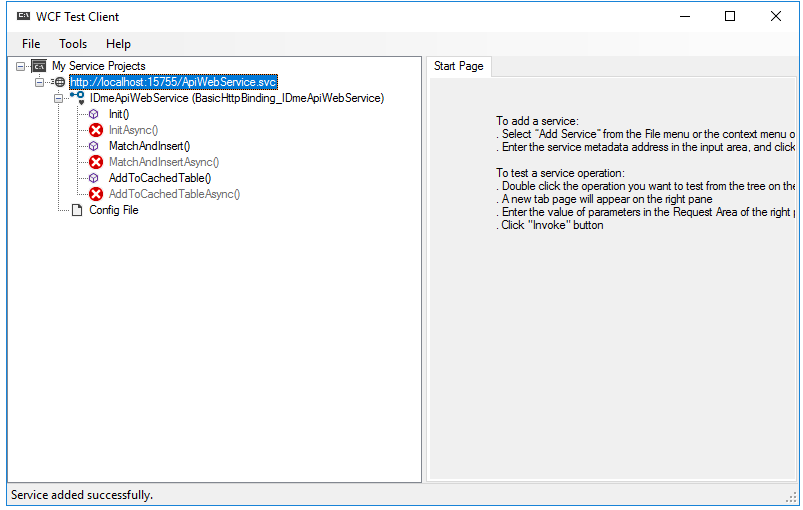
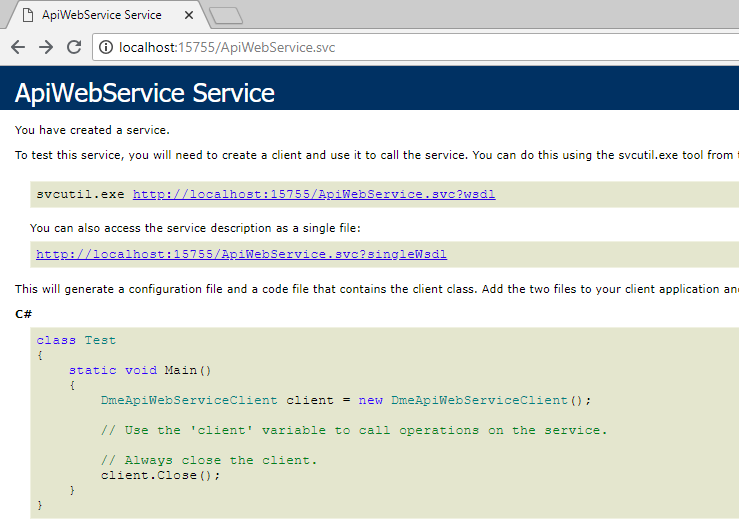


Fig. 21. Web Service after starting in Wcf Test Client

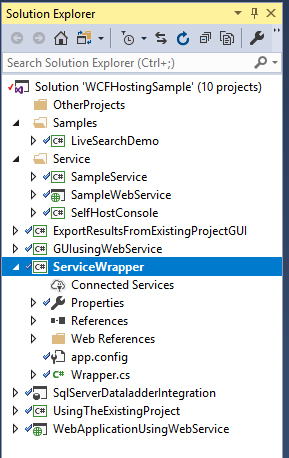
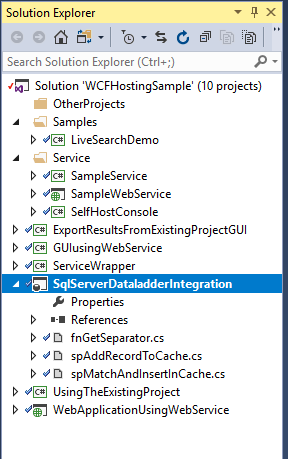
There are two ways to add new record in cache table inside Match Engine. The first automatically after matching. The second using web service method ‘AddToCacheTable’ (see Fig.21), this way can be used for inserting such records straight from the database.

After starting service all existing methods of service shown into ‘WCF Test Client’ (Fig. 21). Also service can be connected with web browser (Fig.22)

 Fig. 22. Web Service in browser

### Library for accessing Web Service from SQL Server

For accessing to web service from SQL Server the application uses the assembly that loaded in the database that can interact with SOAP. How you can find it in Visual Studio Solution Explorer is shown on Fig.23a

1. b)

Fig. 23. Assemblies that interact with SOAP from SQL Server

This project has a Service Reference to the project described in previous.

### SQL CLR library for SQL Server

This assembly connect C# and SQL languages. It is shown on Fig.23b. In this project stored procedures wraps SOAP calls

### Deploying

SQL scripts that allow load assemblies and create stored procedures are shown below

1) Load assemblies described in p.3.9.2 and p.3.9.3 into SQL Server database

Enable unsafe assembly in database

alter database [DatabaseName]

set trustworthy on;

go

Enable CLR assemblies

USE [DatabaseName]

EXEC sp\_configure 'clr enabled', '1';

RECONFIGURE;

Load assemblies into database

IF EXISTS (

SELECT [name]

FROM sys.assemblies

WHERE [name] = N'SqlServerDataladderIntegration')

BEGIN

DROP ASSEMBLY SqlServerDataladderIntegration

ALTER ASSEMBLY SqlServerDataladderIntegration

FROM 'D:\ApiDemo\SqlServerDataladderIntegration.dll'

WITH PERMISSION\_SET = UNSAFE ;

END

ELSE

BEGIN

CREATE ASSEMBLY SqlServerDataladderIntegration

FROM 'D:\ ApiDemo \SqlServerDataladderIntegration.dll'

WITH PERMISSION\_SET = UNSAFE ;

END

2) Create Stored Procedures that allow use loaded assemblies from triggers and other objects of database

Use DatabaseName

Go

IF EXISTS (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[spMatchInsert]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)

DROP PROCEDURE [dbo].[spMatchInsert]

GO

IF EXISTS (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[spAddToCacheTable]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)

DROP PROCEDURE [dbo].[spAddToCacheTable]

GO

IF EXISTS (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[spGetSeparator]') and OBJECTPROPERTY(id, N'IsScalarFunction') = 1)

DROP FUNCTION [dbo].[spGetSeparator]

GO

IF EXISTS (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[spGetVersion]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)

DROP PROCEDURE [dbo].spGetVersion

GO

CREATE PROCEDURE [dbo].spMatchInsert

@parameter1 nvarchar(max),

@parameter2 bigint,

@parameter3 nvarchar(max)

WITH EXECUTE AS CALLER

AS

EXTERNAL NAME SqlServerDataladderIntegration.StoredProcedures.spMatchAndInsertInCache

Go

CREATE PROCEDURE [dbo].spAddToCacheTable

@parameter1 nvarchar(max),

@parameter2 nvarchar(max)

WITH EXECUTE AS CALLER

AS

EXTERNAL NAME SqlServerDataladderIntegration.StoredProcedures.spAddRecordToCache

Go

CREATE FUNCTION [dbo].[spGetSeparator]()

RETURNS

nvarchar(max)

WITH EXECUTE AS CALLER

AS

EXTERNAL NAME SqlServerDataladderIntegration.UserDefinedFunctions.spGetSeparator

Go

CREATE PROCEDURE [dbo].spGetVersion

WITH EXECUTE AS CALLER

AS

EXTERNAL NAME SqlServerDataladderIntegration.StoredProcedures.spGetVersion

Go

3) Execute created stored procedure for checking installation

EXEC dbo.spGetVersion

Expected result (the first number is a version of DME API, the second is a version of API Demo):

3.0.26.0 (1.0.4.0)