Data Ladder
Get the most out of your data

Guide

DataMatch Enterprise SDK Developers Guide

68 Bridge, St. Suite 307
Suffield, CT 06708

+1 888-779-6578

Sales@DataLadder.com

www.DataLadder.com

DataMatch Enterprise™

SDK Developers Guide

1. Introduction

The Data Match Enterprise API is a component written by Data Ladder for state-of-the-art fuzzy matching, data formatting, and data cleansing. Common uses are duplicate prevention, inquiry, real-time deduplication, and merge/purge.

The Data Match Enterprise API splits and cases names and addresses, generates match keys for phonetic matching, generates 3-grams for more accurate fuzzy match, and grades matching records. The component provides a compact and efficient solution for data quality issues on any Windows-based system. This is the help file for the .NET Framework Data Match Enterprise API.

The API is written in C# programming language. This document assumes that you have familiarity with at least one .NET Framework programming language. Experience with the utilization of .NET components from within programs would be an advantage, but not essential. If you have any questions, please contact us and we'd be happy to help.

The Data Match Enterprise (hereafter referred to as DME) SDK consists of several parts:

- Data Match Enterprise Application – functionality is determined by DME registration key;
- Address Verification module – installed optionally;
- DME API – a subset of DME .dll libraries and files that can be used for calling DME functionality programmatically;
- DME API Samples – Microsoft Visual Studio projects that are intended for demonstration of the main DME functionality that can be used programmatically.
- Developers Guide (this document) and User Guide – instruction set for installing, configuring and running API and samples.

2. DME API Parts

DME API is a part of DME application binary files, with the exception of business and presentation layers. A full list of required libraries with their descriptions presented in table 1:

Table 1. Data Match Enterprise API libraries.

File Name	Description
DataMatch.AddressVerification.dll	A library that is related to the Address Verification module.
DataMatch.Api.dll	Provides a high-level interface for developers and includes such entities as project information and data source info.
DataMatch.Collections.dll	A module that provides the functionality for working with huge collections of standard .NET types in a multi-threaded way. Sorting functionality is also exposed.
DataMatch.Connectors.dll	This library contains the functionality responsible for the creation of different data connectors: – Access – ApacheHBase – DynamicsCRM – Email – Excel – Facebook – JSON.dll – MongoDB – MySQL – Salesforce – SugarCRM – Twitter – XML
DataMatch.Core.dll	The main part of API. Includes commonly shared entities that are used in other libraries. Licensing system, Path 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.Data.Core.dll	Basic functionality related to data connectors. Interfaces, enumerations, shared entities, and helpers are declared here.
DataMatch.DataStorage.dll	This module includes API for working with DME internal storages. OnDriveTable and InMemoryTable are located here.
DataMatch.Matching.dll	A part of functionality responsible for match configuration, match definitions, and the matching process is located in this library.
DataMatch.Project.dll	Base classes responsible for storing and loading projects are placed here. Descriptor classes for every DME entity that can be stored in .dmeproj file are defined in this module.
DataMatch.Transformation.dll	Base classes required for data cleansing and transformation are located in this dll.
TrigramHashes.dll	Contains special subroutines for calculating hashes.
IBM.Data.DB2.dll	Required for IBM DB2 import/export.
ICSharpCode.SharpZipLib.dll	SharpZipLib is a compression library that supports Zip files.
Interop.MSDASC.dll	Necessary for OLE Universal connector
log4net.dll	Logging system
MySql.Data.dll	Required to import/export data from/to MySQL.
Newtonsoft.Json.dll	Required for JSON serialization.
Ninject.dll	This module is needed for SalesForce connector.
NPOI.dll	Library for reading Excel files.
Oracle.ManagedDataAccess.dll	Required for Oracle connector.
Teradata.Client.Provider.dll, Teradata.Net.Security.Tdgss.dll	Teradata connector libraries.
Trinet.Core.IO.Ntfs.dll	Utilities for working with alternate data streams on NTFS file systems.

(Files related to API are marked in green)

3. Getting started

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

3.1. Pre-requisites to be able to use examples:

- Microsoft Visual Studio 2017 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: Deployment of SSD will increase the performance significantly;
- Microsoft SQL Server Express 2016 /2017 (required for a Web Sample).

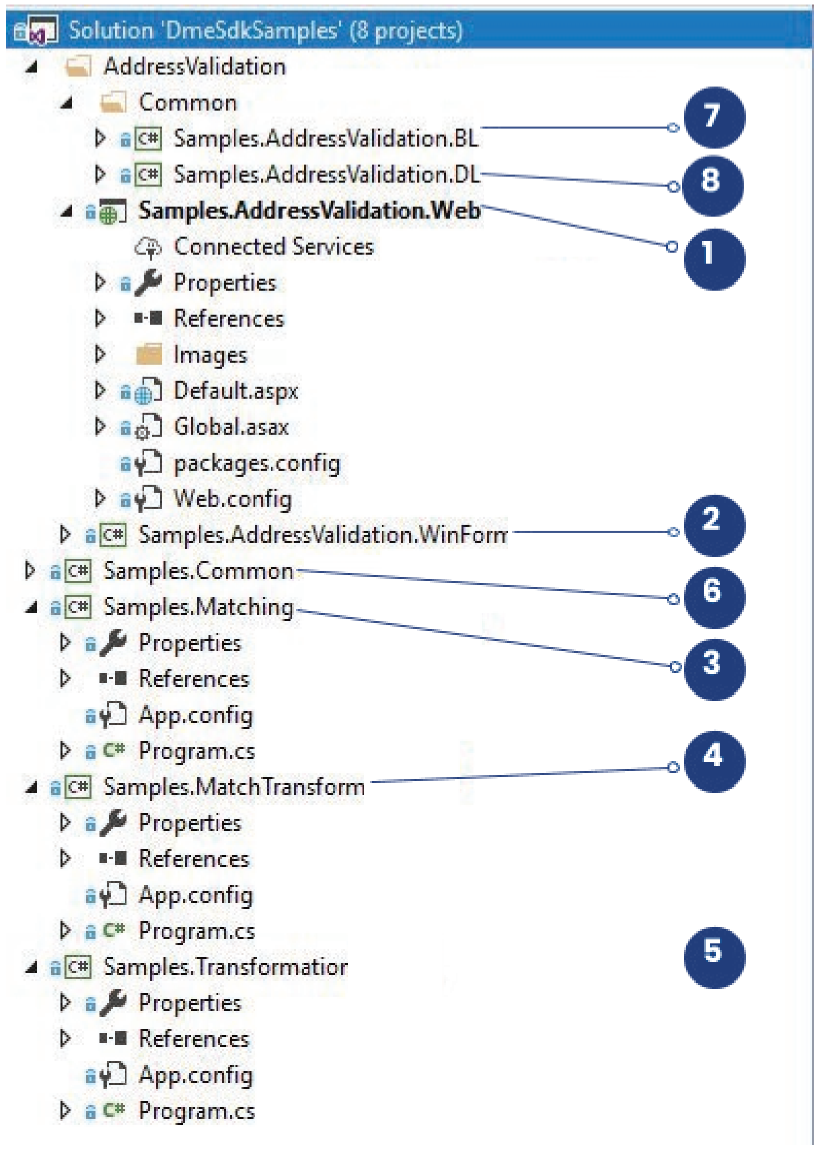
4. API Explanation

4.1. Overview

Current SDK includes a list of examples and incorporates "best practices."

Samples explanation

Current SDK includes a Visual Studio solution with the projects shown below:



Pic. 1. Visual Studio Solution with SDK samples

1. *Web sample,* which demonstrates Address Suggestions module usages, Address parsing and duplicate searching using Match Engine from DME API. MS SQL Server DB, is used as a data source for this demo. The current sample project consists of several logical layers:
   - Business Layer (7), contains the main functionality related just to demo application;
   - Data Layer (8), contains the main operations related to communication with sample DB.
   - Common functionality project (6) also included. It is shared between all the demo projects and contains useful functionality.
2. The next sample is similar to *Web Sample*, but uses WinForms approach:

   a. This console application shows how to initialize the Match Engine, define Match Definitions, create Match Criteria, and set up Field Mapping programmatically. In this sample, a small demo data source is created programmatically to show OnDriveTable class usages. Similar records are searched in this data source and match results are represented in console output.

   b. In this sample, 2 data tables are created. One of them is loaded from a demo excel file that contains a contact list and includes fields such as First Name, Last Name, Address, Phone, Email, etc. The second table is created programmatically and contains just a single record that utilizes the Address Verification Transformation Blocks which are applied to the smaller table. The larger table and transformed smaller table is added to MatchEngine input.

Several match criteria are added to only one match definition that is used for matching. The match is performed in the current sample and match results are represented in tabular format.

3. The next sample contains a small sample table and is created programmatically. It contains Address and Full Name information. The Address Verification module is used as a transformation block. Some of the transformation output fields are represented in a tabular format at the end of the sample.

There are two fundamental parts to the Data Match Enterprise API:

- record indexing
- record matching

These parts may be utilized in different scenarios:

- single data source matching
- cross data source matching
- data capture and duplicate prevention

Match Definitions

Match Definition is a set of rules we apply on the fields on which the matching process needs to be performed.

Match Definition for one field consists of the following:

Matching type which can be:

- Fuzzy
- Exact

Before conducting a match process, we can transform the input into its phonetic equivalent:

- Phonetic

Example: phonetic transformation of words "Dayton" and "Deighton" is equal.

If Match Definition is Fuzzy, then we need to apply a value for the:

- Level
    - Level defines the threshold for the comparator. If the results of the comparison are equal to or higher than Level, then the match is considered successful.

Matching Scores

Matching Score is the average value of all matching scores per individual fields. If any field has a matching level below the level, the final score will be 0.

4.2. Classes, Properties, and Methods

The Data Match Enterprise API consists of many classes. These classes are listed and described here along with their properties and/or methods.

Listed below you may find the most commonly utilized classed:

Table 2. The core classes of DME API

Class	Description
MatchEngine	Provides the core interface to use and configure the Data Match Enterprise API.
MatchDefinitionBuilder	Contains all settings used by the MatchEngine class.
MatchDefinitionSingle	Contains all settings for set of mapped fields
MatchDefinitionsList	A list of MatchDefinitionSingle.
MultipleMatchDefinitionsManager	More than one MatchDefinitionsList can be used in the matching process and this class contains them.
OnDriveTable	Permanent table used for storage of imported data sets, indexes, temporary and final results of the matching process.
IReaderHelper	Interface used to import/export data from various data sources (SQL Server, mySql, Excel, CSV, etc.).
ReaderConfiguration	Used to configure the reader.
ReaderToVariableTableConvertor	Converts data from any reader to OnDriveTable for later use in the API.

MatchEngine Class

- Before the instance of this class is created, the MultipleMatchDefinitionsManager instance must be initiated.

MatchEngine Methods

- Public void Add(string mapperName, ITable2CoordsMapper mapper);
    - mapperName is the name for the imported data set which is used in the match definitions, also
    - mapper is the imported data set (can be any class which implements the ITable2CoordsMapper interface)

- Public void AddPairToMatchList(intdatasourceIndexA, int datasourceIndexB);

Add a pair of indexes of datasets for matching (e.g. 1, 1 means to match within data set 1. And 2, 3 means to match between data sets 2 and 3)

- public bool DoIndex()
    - Stores the indexed data into files for later use with matching process
- public bool DoMatch()
    - Initiates the matching process. All settings must be completed before this method is called
- public void ProcessFinalResults();
    - Creates finals score pairs and groups

MatchDefinitionSingle Class

MatchDefinitionSingle Properties

- Exact
- Fuzzy
- Phonetic
- Level

MatchDefinitionSingle Methods

- public void MapField(string dataSourceName, string fieldName);
    - For all data sets that need to be matched, this method must called once with the data source name and field name for mapping.

MultipleMatchDefinitionsManager Class

- More than one MatchDefinitionsList can be used in the matching process and this class contains each unique MatchDefinitionsList.

MultipleMatchDefinitionsManager Methods

- public void Add(MatchDefinitionsList matchDefinitionsList);
    - Adds a MatchDefinitionsList to the manager.

We hope this guide helped explain how to navigate the DataMatch Enteprise SDK.

If you have any questions, please feel free to reach out to our technical support department at **Support@DataLadder.com**.

Data Ladder
Get the most out of your data
Our Customers
GE
hp
Department of Transportation United States of America
The Department of the Treasury 1789
edp consulting inc.
amec foster wheeler
LAMB Insurance Services
ZURICH
SLDS
Deloitte.
GRAINGER. FOR THE ONES WHO GET IT DONE
DHL
Cleveland Brothers CAT
GOODYEAR
ON IMPACT STRATEGIES, LLC
ARLINGTON Power Equipment Inc.
Buckle
QRP
Turn Key Auto Events
Cabarrus County Schools
ECS ENTERPRISE CONTENT SOLUTIONS
USDA
sja
Duke UNIVERSITY
yp
CardinalHealth