MatchUp® Object 3

MatchUp Object

Quick Start Guide

Melissa Data Corporation

Copyright

Trademarks

Melissa Data Corporation

22382 Avenida Empresa
Rancho Santa Margarita, CA 92688-2112

Phone: 1-800-MELISSA (1-800-635-4772)
Fax: 949-589-5211

E-mail: info@MelissaData.com
Internet: www.MelissaData.com

For the most recent version of this document, visit
http://www.melissadata.com/tech/matchup-object.htm

Document Code: DQTAPIMUOQSG
Revision Number: 14022013.14

Table of Contents

MatchUp Object API

Features

- Fast processing, about 10-50 million records per hour
- Extremely flexible and customizable
- Now with 23 powerful matching algorithms
- Split name, address, and city/state/ZIP fields on the fly
- Easy to learn and use
- Sample Code provided in C#, VB.NET, C++, FoxPro, Java, SQL Server
- Free tech support

MatchUp Object Interfaces

This API provides the developer with three different interfaces, or deduping methods, allowing for maximum flexibility in selecting the best method for the application.

The Read/Write Deduper

The Read Write Deduper checks an entire list of records against itself, flagging unique records and duplicates. This deduper is best for checking existing databases and purging duplicate data.

The Incremental Deduper

The incremental deduper checks a single record against a persistent historical key file. This enables an application to field incoming records, such as from a website or a customer service system, easily detect duplicate records, pointing the application to the original unique record.

The Hybrid Deduper

The Hybrid Deduper allows the developer more flexibility in constructing the deduping logic. Since the Hybrid deduper does not have the same level of match key management, it also

requires a greater degree of programming complexity. It is a more advanced interface for the few developers whose needs are not met by the other two dedupers.

System Requirements

MatchUp Object is shipped on a CD. It is recommended that you copy the data files to your local or network hard drive in order to access the data faster. Because these objects are in essence programmer's tools, they should be installed on a system that has a development environment in order to utilize the power of the objects.

The following are additional hardware and software requirements:

- 50MB hard disk space (for data files).
- Microsoft® Windows® users — 64-bit Windows Vista, Windows 7, Server 2003 or Server 2008. Most Windows-based programming languages. .NET Framework 3.5 or better required to register the COM object.
- Linux users — Red Hat 64-bit (x64) distribution. GNU C++ 3.4 or later; glibc 3.2 or later.
- Solaris users — Solaris 8,9,10, SPARC platform, 32 or 64-bit. Sun Workshop or Sun ONE Studio compiler. G++ 3.3 and later can also be used.
- AIX users — Version 5.2 or 5.3; POWER, rs/6000, PPC, 64-bit. gcc 3.4.6 or Visual Age.
- HP-UX users — Version 11.11, 11.23; PA-RISC or Itanium, 64-bit. gcc 3.4.6 or aCC A.03.70.

The actual deployment system does not require use of the development tools.

Installing MatchUp Object

Windows

1. Close all applications that are open, including any anti-virus and e-mail software.
2. Place the MatchUp Object CD in your CD-ROM drive.
3. Click the Start button, and then select Run from the pop-up menu. The Run dialog box opens.
4. Type your CD drive letter followed by :\SETUP.EXE in the Command Line box. (For example, D:\SETUP.EXE). After typing the path, click OK.
5. Read the license agreement, and then click Yes if you agree to the terms. (To proceed with the installation, you must accept this agreement.)
6. Select the options that you want to install by placing a check mark next to its name and clicking Next.
7. The installer will display a screen showing the install location and the selected options. Click Next again.
8. Click Install.

Linux/Solaris/HP-UX/AIX

You do not need any special privileges when installing MatchUp Object, nor do you have to log in as root.

During installation, nothing will be modified outside of the target directory.

To install:

1. Place the MatchUp Object CD in your CD-ROM drive.
2. From the shell prompt ($), mount the CD and run the applicable setup.sh script to install the object to the desired directory.

All of the Unix-based OS sample programs assume that they can locate the required data files and object libraries in the current directory. It is not necessary to modify your PATH or LD_LIBRARY_PATH. If you prefer, you can run the sample program straight from the CD-ROM — there is no need to install anything.

The final deployment install has to be done manually or by using your system administration utilities. Since the deployment standards vary widely, Melissa Data does not provide any specific instructions. Remember the following:

- MatchUp Object does not require any special privileges.
- All files can be made read-only.
- There is no need for a setuid or setgid, neither as file permissions nor anywhere in your application code.

File Locations

Windows

Most of the files are placed in subdirectories of "C:\Program Files\Melissa Data\DQT\."

To provide maximum compatibility with Windows, three files are installed in your 'Common App Data' directory. For Windows Vista and Windows 7 the default location is "C:\ProgramData\MelissaDATA\MatchUp." For Windows XP the default location is "C:\Documents and Settings \All Users\Application Data\Melissa DATA\MatchUp." The location of this directory can be changed by users so please note this, as it can often be the source of issues when running the samples/demos.

During the installation process your Microsoft C runtime may be updated, if needed.

Uninstalling MatchUp Object

To remove MatchUp Object from your computer, do the following:

Windows:

1. Click the Start button.
2. Select Settings from the pop-up menu.
3. Click Control Panel.
4. Double-click Add/Remove Programs.

Linux/Solaris/HP-UX/AIX:

1. Verify that all files in the target directory can be safely deleted.
2. Type rm -rf target-directory.

If the target directory contains files that cannot be deleted without impacting other software, you will need to manually erase only files from MatchUp Object.

Configuring MatchUp Object

A current license string is required to use the full functionality of MatchUp Object. Without a license string, the object will work in demo mode.

The license string should be entered as an environment variable named MD_LICENSE. This allows you to update your license string without editing and recompiling your code.

Setting the License String Environment Variable

Windows

Windows users can set environment variables by doing the following:

1. Select Start > Settings, and then click Control Panel.
2. Double-click System, and then click the Advanced tab.
3. Click Environment Variables, and then select either System Variables or Variables for the user X.
4. Click New.
5. Enter the "MD_LICENSE" in the Variable Name box.
6. Enter the license string in the Variable Value box and then click OK.

Please remember that these settings take effect only upon start of the program. You may need to quit and restart the development environment to incorporate the changes.

Linux/Solaris/HP-UX/AIX

Unix-based OS users can simply set their license string via the following:

```
export MD_LICENSE=A1B2C3D4E5
```

If you decide to put this setting in your .profile, remember to restart your shell.

Sample Implementations

This section describes the logic behind simple applications using MatchUp Object. The samples are written in pseudocode so they can be easily adapted into almost any language.

Initialization

Declaring and creating an instance of MatchUp Object is followed by the initialization steps to prepare it for use. These steps only need to be performed once per instance of the object in use.

Input

In this section, you'll pass the input values to the deduping engine, building the match keys and adding them to the key list.

Processing

In the processing stage, you'll switch the deduper engine into processing mode and examine the match results.

Termination

This stage involves destroying the current instance of MatchUp Object when you close the application, freeing up memory to be used by other processes.

MatchUp Object Sample

This sample shows the Read/Write deduper interface. For examples of the other interfaces, see the MatchUp Reference Guide.

Step 1 — Declaring and Creating an Instance

Begin by declaring the Object variable.

```
Create MURW As New Instance of mdMUReadWrite
```

Step 2 — Set Data Paths and Initialize

The next step is to tell the MatchUp Object where it can find its data files.

```
CALL SetPathToMatchUpFiles WITH DataPath
```

Before initialization, the application must specify which matchcode and key file will be used for the current deduping operation. See the MatchUp Object Reference Guide for more information.

```
CALL SetMatchcodeName WITH MatchCodeName
CALL SetKeyFile WITH PathToKeyFile
```

If all of the above have been set correctly, calling the InitializeDataFiles function should return a value of 0. If it does not, call the GetInitializationErrorString function to determine the reason for the failure to initialize.

```
CALL InitializeDataFiles RETURNING Result
CALL GetInitializeErrorString RETURNING
ErrorStr
```

Step 3 — Create Field Mappings

Field mappings define which types of data the Read/Write deduper is expecting. In this case, the selected matchcode looks for a five-digit ZIP Code™, a first name, a last name and a street address.

```
CALL ClearMappings
```

After clearing any mappings from a previous use of the Read/Write deduper, call the AddMapping function once for each field being considered.

```
CALL AddMapping WITH Zip5 RETURNING mapOK
CALL AddMapping WITH First RETURNING mapOK
CALL AddMapping WITH Last RETURNING mapOK
CALL AddMapping WITH Address RETURNING mapOK
```

Step 4 — Loop through database records and build keys

The Read/Write deduper builds a temporary key file out of the data from the database. To do this, the application loops through each record and pulls the data from the fields that match the mappings made above.

```
FOR EACH Record in database
Read Zip5, FirstName, LastName,
StreetAddress, userInfo fields from database
```

After pulling the data from the database, pass it to the Read/Write deduper with the AddField function. The application must do this in the same order that it mapped the data types in the step above.

Even if the fields in a database do not exactly match the components required by the matchcode, MatchUp Object is able to extract only the information it needs. For example, if the database only contained a full name field, that field could be passed twice and MatchUp Object would recognize the first names and last names and only use the parts it needed.

After passing each set of fields, call BuildKey to create the match key according to the mappings and the current matchcode.

```
CALL ClearFields
CALL AddField WITH Zip5
CALL AddField WITH FirstName
CALL AddField WITH LastName
CALL AddField WITH StreetAddress
CALL BuildKey
```

The UserInfo is a unique identifier for each record. The application will need this to later match the deduping information to the original records.

```
CALL SetUserInfo with userInfo
```

The WriteRecord function adds the current key and UserInfo to the key field.

```
CALL WriteRecord
```

Repeat for every record in the current data set.

```
NEXT Record
```

Step 5 — Begin Processing the Records

The Process function switches the Read/Write deduper from writing new keys to comparing the stored keys to each other.

```
CALL Process
```

Step 6 — Examine the Processed Records

At this point, loop through the processed records and get information on each record's unique/duplicate status, and how many duplicates of each record exist in the data set.

```
WHILE mu.ReadRecord does not return 0
```

The status code indicates whether the record is unique, a record with duplicates, or a duplicate of another record.

```
    CASE mu.GetStatusCode OF
        A:PRINT "This record is a duplicate."
        B:PRINT "This record has duplicates."
        C:PRINT "This record is unique."
    ENDCASE
    CALL mu.ClearFields
ENDWHILE
```

Step 7 — Destroying the Instance

Be sure to free allocated memory. We do not believe the MatchUp Object leaks allocated resources, however, you must properly destroy the MatchUp Object to force the release of memory.

ParseFielded is faster and more accurate, but requires that the data is consistently delimited.