



matchIT SQL

Information Pack



1 Contents

1	Contents.....	2
2	Introduction.....	6
3	Processing Unicode Data.....	7
4	Getting Started Tutorial	8
4.1	Introduction	8
4.2	Quick start with example data	8
4.3	Getting set up with your own database schema.....	8
4.3.1	Match Keys.....	8
4.3.2	General Settings	9
4.3.3	Output Settings.....	10
4.3.4	Data Sources	10
4.3.5	matchIT API Settings	12
4.3.6	Overview of XML Configuration File	13
4.3.7	Running the application configured to your data	13
4.3.8	Triggers	14
4.3.9	Running in Passive Mode	15
5	Overview of XML UI.....	16
6	matchIT SQL Component Overview	19
6.1	Stored Procedures	19
6.2	SSIS Components	19
6.3	Worker Processes	19
6.4	matchIT SQL Service	19
6.5	Error Logging	19
6.6	matchIT SQL Monitor	19
7	Stored Procedures.....	20
7.1	Address Correction	20
7.1.1	msp_GenerateCorrectedAddresses	20
7.1.2	msp_GenerateNCOAAddresses.....	40
7.2	Key Generation.....	54
7.2.1	msp_CreateKeysTable	54
7.2.2	msp_AddKeyFieldsToTable	55
7.2.3	msp_GenerateKeys	56
7.2.4	msp_BulkGenerateKeys	62
7.2.5	msp_CreateIndexes	62
7.3	Exact Deduplication	64
7.3.1	msp_FindExactMatches	64
7.3.2	msp_FindExactOverlap	66

7.3.3	msp_GroupExactMatches.....	67
7.3.4	msp_GroupExactOverlap	68
7.4	Fuzzy Deduplication	69
7.4.1	msp_FindMatches	69
7.4.2	msp_FindOverlap	76
7.4.3	msp_GroupMatches.....	78
7.4.4	msp_GroupOverlap	81
7.5	Output	83
7.5.1	msp_OutputMatchingPairs	83
7.5.2	msp_OutputMatchingGroups.....	84
7.5.3	msp_OutputDuplicates	85
7.5.4	msp_OutputDedupedTable	86
7.5.5	msp_TagMatchingResultsWithGroupLevel	87
7.5.6	msp_OutputOverlapMatchingPairs.....	88
7.5.7	msp_OutputOverlapMatchingGroups	89
7.5.8	msp_OutputOverlapDuplicates	90
7.5.9	msp_OutputOverlapDedupedTable	91
7.5.10	msp_TagOverlapMatchingResultsWithGroupLevel.....	92
7.6	Triggers	93
7.6.1	msp_CreateTableTriggers.....	93
7.6.2	msp_DeleteTableTriggers	93
7.6.3	msp_GenerateSingleKeys	93
7.7	Miscellaneous	94
7.7.1	msp_CreateCustomMatchesTable.....	94
7.7.2	msp_CreateCustomGroupedMatchesTable	94
7.7.3	msp_CreateUniqueRefField	94
7.7.4	msp_SingleRecordMatch.....	95
7.7.5	mfn_SingleRecordMatch	96
7.7.6	mfn_SingleRecordMatch2	96
7.7.7	mfn_SingleRecordMatchEx.....	97
7.7.8	mfn_SingleRecordMatchEx2.....	97
7.7.9	mfn_SingleGenerateKeys	98
7.8	General Progress Logging Table	99
7.8.1	log	99
8	Summary Reporting.....	101
8.1	GenerateCorrectedAddresses Summary Report	102
8.2	FindMatches Summary Report	103
8.3	FindOverlap Summary Report.....	104
8.4	GroupMatches Summary Report	105
8.5	GroupOverlap Summary Report	105

8.6	FindExactMatches Summary Report	106
8.7	FindExactOverlap Summary Report.....	106
8.8	GroupExactMatches Summary Report	106
8.9	GroupExactOverlap Summary Report	106
9	Address Correction	107
9.1	Installation.....	107
9.2	Setup.....	107
9.3	Usage	107
10	Mail Sortation	108
10.1	Installation.....	108
10.2	Usage.....	108
10.3	Advanced Automation.....	110
11	Suppression Processing	111
11.1	Installation.....	111
11.2	Utilising the SSIS package.	111
11.2.1	Suppression Priorities	113
11.2.2	Using the options config.	113
11.2.3	Suppression Loader.....	114
11.2.4	Select the data to be suppressed	115
11.2.5	Suppression Output.....	115
11.2.6	Re-generating Output without Re-running suppressions	115
11.2.7	Cancelling a job from the log	116
11.2.8	Hardware Recommendations	116
12	SSIS Tasks	117
12.1	Installation.....	117
12.2	Setup	117
12.3	Usage.....	117
13	Troubleshooting.....	119
13.1	A .NET Framework error occurred during execution of user-defined routine or aggregate "msp_GenerateKeys":	119
13.1.1	Possible Cause:.....	119
13.2	A .NET Framework error occurred during execution of user-defined routine or aggregate msp_GenerateCorrectedAddresses.....	119
13.2.1	Possible Cause:.....	119
13.3	Column '<column>' has invalid type: <type>	120
13.3.1	Cause:	120
Appendix A – SSIS Screenshots		121
Generate Corrected Addresses Task.....		123
Source Tables		123
Input Mappings.....		124

Output Mappings.....	125
API Settings.....	126
Generate Keys Task.....	127
Source Tables.....	127
Input Mappings.....	128
Output Settings.....	129
Filtering and Sampling Settings.....	130
Advanced Settings.....	131
Appendix B - Key Generation Flag field.....	132

2 Introduction

Welcome to the matchIT SQL Product information pack. This product has been developed to allow users to run batched data cleansing by utilizing the matchIT API.

matchIT SQL makes use of *SQL CLR*, a feature of SQL Server 2005 that allows stored procedures and functions for SQL Server to be written using .NET languages, as well as using ADO.NET rather than classic ADO for increased performance.

matchIT SQL is highly stable and robust. The unmanaged matchIT API COM component is loaded into an external application – the 'worker' process – so that the stored procedures assembly can be loaded into SQL Server using External Access privileges.

The main core of the matchIT SQL functionality is configurable through an XML file. This is the heart of matchIT SQL, which is used to modify settings of the matchIT API and to map your data to the relevant fields of the matchIT Record object. The XML Configuration file is used in all CLR procedures that involve working with your data, and is described in more detail below.

This guide should be read in conjunction with the documentation for the matchIT API and has been split into the following chapters.

- Getting Started Tutorial – tutorial detailing how you would go about making changes to the default xml configuration file to work with your own database and matching keys;
- Technical Reference – technical reference describing each of stored procedures offered by matchIT SQL;
- Troubleshooting – this section will help you with the most common error messages you are likely to see when using the product.

3 Processing Unicode Data

matchIT SQL is fully able to handle Unicode data. It does this by *transliterating* Unicode characters into characters from the Latin1 code page (Windows-1252) so that they can be processed by the matchIT API core. The Latin1 code page is generally used by Western European languages, including English, Spanish, French, and German.

Transliteration is not the same as *translation*, in which words are converted from one language to another; when transliterating, it's the characters themselves that are converted from one alphabet to another. For example, the Chinese character 昌 means "prosperous" and is pronounced "chang", and the Chinese character 李 means "plum" and is pronounced "li". Transliteration converts the Chinese name 昌李 into "chang li" (translation would convert this to "prosperous plum").

Characters from alphabets such as Cyrillic (languages include Russian) and scripts such as CJK (Chinese, Japanese, Korean) can be input into matchIT SQL. Transliteration will be performed in two places:

1. when match keys are generated and output via the BulkGenerateKeys stored procedure or GenerateKeys SSIS task (this produces output using Latin1 characters);
2. when records are compared via the FindMatches and FindOverlap stored procedures and SSIS tasks.

4 Getting Started Tutorial

4.1 Introduction

matchIT SQL comes with some pre-written SQL scripts to create tables and stored procedures, and also run various de-duplication methods on the example data included. Importing this example data and running the scripts is explained in the following section, 'Quick start with example data'.

A guide on modifying the XML Configuration file and how to set up the product to communicate with your own data is set out in the section 'Getting set up with your own database schema'. The guide will focus on setting up matchIT SQL to perform a single data source de-duplication on your own data.

4.2 Quick start with example data

The easiest way to run matchIT SQL with the example data is to run the four .sql scripts, 'Import Example Data', 'Generate Keys', 'Find Matches', and 'Find Overlap', which can be accessed through the Windows start menu (Start > Programs > matchIT SQL > Demo) once the application has been installed. Run them in order: the first will install the example database with the relevant tables; the second script will generate the search keys; the third will find matches in the first table; and the last will find the overlap between the two tables. Opening any script from the Start Menu will automatically open it in SQL Server management studio (provided that is how your .sql file association is set up) from where it can be executed. 'Find Matches' will run a single de-duplication on the first set of data, example1, producing a results table called 'matches'. 'Find Overlap' will perform a two-data source overlap between the example1 and example2 data.

The other way to import the example data and run the de-duplication processes is to simply browse to the scripts in SQL Server Management Studio manually. The demo scripts are located in 'C:\matchIT SQL\demo', and further scripts can be found in 'C:\matchIT SQL\scripts'. When you browse to the latter location you will see that there are a wide variety of scripts available to run. To install the data, you need to run 'ImportExampleData.sql' (found in the demo folder) to actually install the database. After that you can then run the script of your choice to perform the de-duplication you want. In the 'scripts' folder, 'FindMatches.sql' will run a single data source de-duplication on the example1 data, 'FindOverlap.sql' will run a two-data source de-duplication on the example1 data and example2 data, and 'DedupeExample.sql' (in the demo 'folder') will run both. The same applies to running FindMatches.sql and FindOverlap.sql in that you will need to amend Config.xml as mentioned above before running them. The scripts have been split into separate folders distinguish between ones that are for demonstration purposes (in the demo folder) and ones that are for more practical use (in the scripts folder).

4.3 Getting set up with your own database schema

Now let's take a closer look at the XML Configuration of matchIT SQL, namely the file Config.xml located in 'C:\matchIT SQL\demo'. It is easiest to break down the XML file into sections as they appear in the file itself.

4.3.1 Match Keys

The first section that appears in the XML configuration file is the <matchKeys> node, within which there are definitions for three types of keys: <fuzzyKeys>, <exactKeys> and <duplicatePreventionKeys>.

In short, a match key is used to group potential matches within data. The keys you will want to use will depend on the kind of data you are working with. An example of a match key that is commonly used is the 'name key', which is the phonetic surname combined with the initial of the first name. The name key for 'John Smith' would be 'smyTJ', which would group names such as Jonathan Smith and Jon Smythe. These potential matches would then be compared with each other and scored to rate the validity of the match.

By default, three 'Composite keys' are defined for the <fuzzyKeys> node - A composite key is a collection of single keys that are used together in a query to search for data. You can add or remove composite keys (note that one search is done per composite key, so more composite keys means more processing time) by adding or removing <key> nodes within the <fuzzyKeys> node, as well as modifying the single keys of the <key> nodes themselves. The <fuzzyKeys> are used in the fuzzy matching processes such as findMatches and findOverlap.

Also by default, a 'Composite key' containing 8 keys is defined for the <exactKeys> node – They can be modified in exactly the same way that the <fuzzyKeys> are, and are used in the exact matching processes, such as findExactMatches and findExactOverlap. Records with identical exactKeys are deemed to be exact matches without any further comparisons or processing, hence there is not scoring. It is therefore important to use valid keys for this node. For example, just using 'name key' like in the example above may cause 2 John Smiths at completely different addresses, who are therefore most likely completely different people, to be classed as exact matches.

The <duplicatePreventionKeys> node can similarly contain any number of composite keys (one is defined by default), and these can be modified in exactly the same way as the others, by adding or removing 'keyX' attributes (where X is an integer). The duplicate prevention keys are used in the msp_SingleRecordMatch procedure and mfn_SingleRecordMatch table-valued function (TVF), which will typically be used in a passive manor, described further on.

It is also worth noting here that it is possible to use matchIT record fields as search keys as well as key fields themselves (i.e. fields with the 'mk' prefix). Note however that if there is no field mapping defined for a particular matchIT record field (discussed in the <datasource> section below), the key will not be able to be used as a match key, and an exception will occur if it is.

Note – It is possible to declare SQL string functions with the key definitions, that is, LEFT, RIGHT and SUBSTRING. If you wanted to use the fist 2 letters of post out for example as a match key, you would set the 'keyN' attribute to be LEFT(mkPostOut, 2). You would do the same with the RIGHT and SUBSTRING functions (with SUBSTRING taking a third counter argument). The following other functions can be used: LTRIM, RTRIM, TRIM, PUNTRIM, UPPER, and LOWER.

4.3.2 General Settings

The <generalSettings> section of the configuration contains nodes that define settings for various methods of matchIT SQL. Explanations for each are listed below –

Node	Description
progressInterval	Sets the length of time (in milliseconds) between progress reports sent to the sql pipe for various methods in the product. Increasing the progress interval will marginally improve performance at the cost of losing information sent to the user.
tempFileDirectory	Sets the directory that files such as the bulk generation file and log files are created in. If left blank, the windows Temp folder is used as a default.
maxClusterSize	Sets the maximum size that a cluster can be before being reported as a large cluster. Clusters are the groupings of records created by the initial matching done on the match keys.
minimumIndividualScore	Sets the minimum score for a match to pass at

	individual level.
minimumFamilyScore	Sets the minimum score for a match to pass at family level.
minimumHouseholdScore	Sets the minimum score for a match to pass at household level.
minimumBusinessScore	Sets the minimum score for a match to pass at business level.
minimumCustomScore	Sets the minimum score for a match to pass at custom level.
comms/port	Sets the port to be used when a stored procedure connects to the matchIT SQL Service.

4.3.3 Output Settings

The <outputSettings> node contains custom definitions of two standard tables that are produced by matchIT SQL (namely the matches and matches_grouped) that can be created from the standard tables by running the relevant procedure. (In the case of producing a custom matches table, you would run msp_CreateCustomMatchesTable, and for the matches_grouped table you would run msp_CreatedCustomGroupedMatchesTable). For both the <matchesOutputTable> and <groupedMatchesOutputTable> nodes, you must define a custom table title with the 'customTitle' attribute, and then for each column subnode in either definition it is possible to define a custom column name. If you want to leave one of the columns from the standard tables out of your custom definition, simply leave the 'customName' attribute blank.

The <outputSettings> node also contains tableSuffixes which allows you to specify suffixes for tables which are automatically generated, such as the keys table and the output table, but we recommend leaving these set to the defaults.

4.3.4 Data Sources

The <dataSources> node is where the mapping of your data takes place. The <dataSources> node can contain any number of <dataSource> sub nodes, within which you define your sources that you want to operate on. Each <datasource> node must have an 'id' attribute specified as it is used when calling procedures to specify which data source you want the procedure to work on.

The first sub node within the <dataSource> node is the <connectionString> node, which contains a connection string to the database. By default, as mentioned in 'Quick Start with Example Data', it has blank credentials and points to a database called 'matchIT_SQL_demo', which is the sample database created by matchIT SQL for use with the sample SQL scripts. You need to amend the connection string as necessary to point to your data.

Following the < dataSources > node is the <tables> node. This node contains all the definitions of the tables within your database that contain the data to be worked with. A single table is defined in a <table> node. The <table> node can contain any of the following attributes:

Attribute	Description
Name	The name of the table in the database (mandatory for all tables).
uniqueRef	The name of the unique reference column in the table (mandatory for all tables).
join	The name of the table that the table in question joins to (the main table of data will not have this attribute).
joinType	The type of join to use between the table in question and the table it is joining to – either INNER, LEFT or RIGHT. Uses LEFT by default.
joinColumn	The name of the column in the join table that the unique ref column of the table in question joins with (mandatory for any table definition with a join attribute).
isKeysTable	Indicates that the table in question is the keys table definition (can only have one keys table definition per data source, and it must join to the main table of data)
isOutputTable	Indicates that the table in question is the output table which will contain cleaned and normalised data after key generation (can only have one output table defined per data source)
tableHints	Here you can specify a comma delimited list of Table Hints that will be used in SELECT queries for the table in question, for example 'NOEXPAND' if the table you have defined is an indexed view and you want the query to reference the view rather than its base tables.

Within a <table> node, it is possible to have multiple <conditionalColumn> sub nodes that define columns in the table in question that must meet a condition for a record to be included (for example, a 'deleted' flag column). A <conditionalColumn> node must have the following attributes defined:

Attribute	Description
name	The name of the column in the table.
isEqualTo	Specifies whether the value should or should not be equal to the value.
Value	The value of the flag column (linked to the isEqualTo attribute as mentioned above).
isIntegerType	Specifies whether the column is an integer type or not.

One thing to note about the <tables> section is that SQL server supports views, so it is possible to use the name of a view for the 'name' attribute of a table node.

4.3.4.1 Using a Sample from your datasource

The next part of the datasource definition is Sampling. This allows you to specify a specific sample of data to use during your processing rather than the complete datasource. This can be useful if you are testing your scripts and refining matching settings.

There are several sampling options available:

- Percentage – uses a percentage from your datasource. For example, if you select 10%, then 1 in 10 records from your datasource would be used.
- NinM - uses a sample based on N in every M records of the source data.
- Range - Uses a sample over the specified range of the specified field.
- RandomN - Uses a sample of N randomly selected records from the source data.
- MaxFromTop - uses a sample of the top N records from the source data. If this option is applied with any of the previous options, then the limit setting is used as a restriction on the maximum number of records that will be used in the sample.

When sampling is enabled, only a sample of your data will be used by the matchIT SQL stored procedures; remember to de-activate this when you are ready to process your complete datasource!

4.3.4.2 Field Mappings

The last node within the <dataSource> node is the <fieldMappings> node - It contains <fieldMapping> sub nodes that define how fields in your data map to the matchIT record object. The 'matchITField' attributes of these nodes are pre-set and must not be changed – The attribute that you will need to modify is the 'databaseField'. Simply put the values of the 'databaseField' nodes to the names of the fields in your data that best match the corresponding 'matchITField'. Any matchITFields that do not have an equivalent in your database should have their databaseField attributes left blank.

Note that it is also possible to specify custom fields that do not have a specific matchIT record equivalent, that you wish to match on, to any of the 9 generic custom fields that are available in the matchIT record object. By default, there is a node included in the xml on installation with its matchITField attribute set to 'CustomField1' – You can in fact have 9 nodes in total with their matchITField attribute values ranging from CustomField1 to CustomField9. A database field such as 'National Insurance Number' could be mapped to one of these fields.

If you open a configuration file that has been edited and saved through the Web UI (described below) you will also notice that there is an <addressing> node within each data source that contains the settings used when running the addressing stored procedure. The output contained within this node varies depending on what addressing API is being used, and is best edited through the UI. For the sake of getting the demo scripts to work with your own data however, you do not need to be concerned with this node and whether or not it exists, as the addressing procedure will not be used.

4.3.5 matchIT API Settings

The <matchITAPISettings> node defines all the settings related to the matchIT API. Perhaps the one of the more important nodes in this section is the first node - <nationality>. This node needs to be set to whatever the nationality of the data is that you are working on.

The structure of the XML within the <matchITAPISettings> node follows the COM hierarchy that is described in the matchIT API guide. Please refer to the matchIT API guide for more detail on the settings and properties of the matchIT API and what they mean.

4.3.6 Overview of XML Configuration File

Below is a simple list summarising the sections of the XML Configuration that have been described above. If you are unsure of a particular section, refer back though the explanation above to be sure as it may cause the application to not work properly in your case if not configured correctly.

- Match Keys – Define the keys that you wish to use for fuzzy matching, exact matching and duplicate prevention matching.
- General Settings – Define the various general settings you wish to use during processing.
- Output Settings – Define the schemas you wish to use to produce custom 'matches' and 'grouped matches' tables from the standard ones produced by matchIT SQL.
- Data Sources – Defined the tables and mappings for your database(s) here.
- matchIT API Settings – Configure the matchIT API settings you wish to use in this section.

Lastly, there are a couple of amendments that need to be made to the sql file FindMatches.sql. If you open the file in SQL Server Management Studio, the lines that need modifying are the following:

```
EXEC msp_CreateKeysTable @config='C:\matchIT SQL\demo\Config.xml',
@dataSourceID='1'
EXEC msp_BulkGenerateKeys @config='C:\matchIT SQL\demo\Config.xml',
@dataSourceID='1'
EXEC msp_FindMatches @config='C:\matchIT SQL\demo\Config.xml',
@dataSourceID='1'
EXEC msp_GroupMatches @config='C:\matchIT SQL\demo\Config.xml',
@dataSourceID='1', @level='individual'
```

Simply substitute '@dataSourceID='1'' for whichever data source you wish to perform the procedures on and '@level='individual'' for whichever level you wish to group the matches at (with the levels being individual, family, household, business and custom).

Note here how all the procedures listed above accept 2 common parameters – one is the file path to the configuration file with your desired set up for the process, and the other is the id of the data source in the configuration for the procedure to run on.

4.3.7 Running the application configured to your data

Once you are happy that all of the necessary changes have been carried out, you are ready to execute the sql script FindMatches.sql in SQL Server Management studio.

As you will be able to see from the script, and as is shown in the code snippet above, this script runs 4 stored procedures. First the msp_CreateKeysTable procedure will run, creating the keys table called by whatever name it was given in the specified data source in the XML. Secondly, the msp_BulkGenerateKeys procedure will generate the keys for the data and populate the keys table. Thirdly, the msp_FindMatches procedure will run and produce 2 tables – 'large_clusters' and 'matches' in the specified data source. The former table stores logs of any instances where a cluster of matches has exceeded the default maximum cluster size (set in the <generalSettings> section of the XML). The latter table stores records of matches that occur, their associated scores at each matching level as well as a column containing an integer indicating which levels the matches passed at (with regards to the minimum scores set in the <generalSettings> section). The value in the last column is made up from the sum of values which correspond to the levels at which the match passed at, which are as follows

- Individual = 1
- Family = 2

- Household = 4
- Business = 8
- Custom = 16

So for example, if a match contained a value of 9 in this column, that would mean that it passed at both individual and business level (1 + 8).

Any errors that occur with procedures will appear in the output pane in SQL Server Management Studio. From the error message you will be able to track down what the error applies to, and modify the configuration / sql script accordingly.

Once you have got this working and have a good idea about the configuration, you can have a look at running the other .sql scripts that are available in the demo folder, namely DedupeExample.sql and FindOverlap.sql. You will see in each script from the lines that start with EXEC what stored procedures are used. Note that in the case of overlapping in these two scripts, both data sources in the configuration are used, and you have the freedom to specify which data source to use as the main file and which one to use as the overlap.

Depending on your situation, you may wish to amend the sql scripts provided or write your own sql script from scratch to process your data. An overview of all stored procedures available is available in the Technical Reference section.

4.3.8 Triggers

You may wish to have a system in place that keeps the keys table in a <dataSource> produced by matchIT SQL kept in sync and up to date with any changes that occur within the tables of data specified in the data source. matchIT SQL offers a system to take care of this in the form of triggers. In the 'scripts' folder, you will see two scripts called 'CreateTriggers.sql' and 'DropTriggers.sql', the names of which are self-explanatory. If you open up the first script, 'CreateTriggers.sql', the lines which you will need to amend are the following.

```
USE matchIT_SQL_demo
```

```
ALTER DATABASE matchIT_SQL_demo
```

Simply amend these lines to point to the database that contains the tables you are creating the triggers on.

```
'C:\Program Files\matchIT SQL\bin\StoredProcedures.dll'
```

Amend this file path if necessary to point to the assembly you wish to create the stored procedures from.

```
'C:\matchIT SQL\demo\Config.xml'
```

Amend this file path to point to the configuration file that contains the data source defining the tables you wish to create the triggers on.

Note – Once created, the triggers rely on the Configuration file being unchanged and staying in the same location as when created. Triggers are only really recommended for long term set ups that are not going to change. The triggers created on the tables in data source specified when calling msp_CreateTableTriggers also rely on the fact that the database in question has the msp_GenerateSingleKeys stored procedure on it too (which gets created along with the assembly in CreateTriggers.sql). If this procedure is not available on the database, the triggers will simply do nothing. Triggers should be deployed with care and testing to make sure no errors occur that could prevent database updates, inserts and deletions occurring in the tables that the triggers exist on.

As an overview then, and as can be seen in the following line

```
EXEC msp_CreateTableTriggers @config='C:\matchIT SQL\demo\Config.xml', @dataSourceID='1'
```

Triggers are created on a particular database using the specified configuration file on the tables defined in the specified data source. Once created, the triggers, when fired, rely on the Configuration File they were created with being in the location that was specified on creation, and for accurate results, for it to be unchanged.

To remove triggers, and all associated procedures and the assembly, make the same style modifications as per the create script above – The configuration file and data source used to remove the triggers will need to be the same as when they were created.

4.3.9 Running in Passive Mode

It may be necessary, depending on the scenario, to run some match processing in the background that is not required in real time. An example of this might be an e-commerce web site that has members joining every week – where it may be necessary to run a weekly process to gather all the new joiners and then search for matches in the database. In this situation, the best thing to do would be to gather a collection of records for which you wish to check against the database for duplicates, and run the `msp_SingleRecordMatch` stored procedure (described in the technical reference) for each one and log the matches for each record as suits the situation. See the description of `msp_SingleRecordMatch` for information on what it returns and the parameters it accepts.

5 Overview of XML UI

matchIT SQL ships with a web user interface for making amendments to XML configuration files. During installation you would have been asked what web server you wish to use for this UI - IIS, Cassini or none. If you chose the latter then this section does not really apply as the UI would not have been installed. If you did however choose to install the UI by selecting one of the other options, then you will be able to access it through the start menu – Start > Programs > matchIT SQL > matchIT SQL UI. If you are using IIS, you will need to make a change to a default value in the Web.config file (found in 'C:\Program Files\matchIT SQL\bin\UI'), namely the following –

```
<identity impersonate="false" userName="Domain\User"  
password="Password" />
```

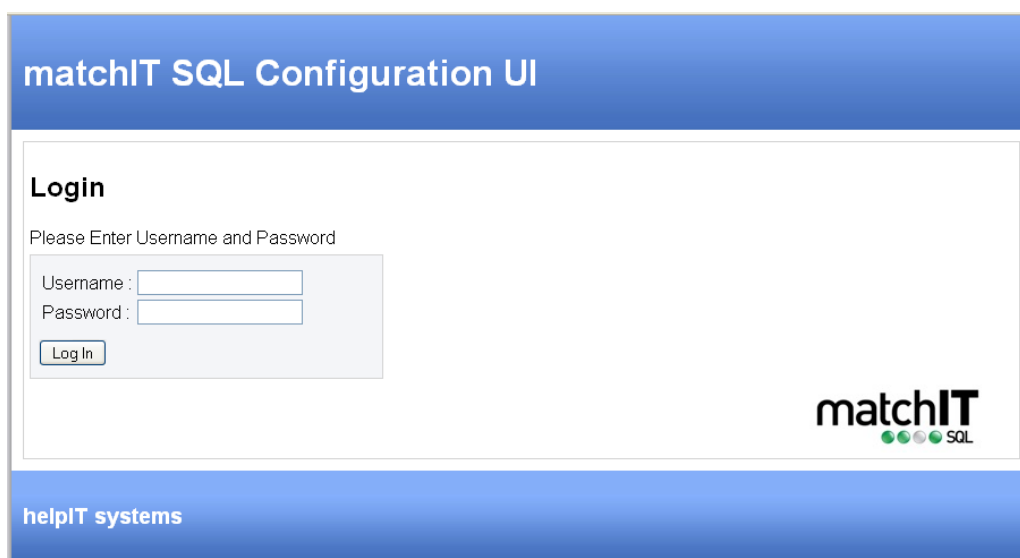
The value for `impersonate` will need to be set to `true` and the values of `userName` and `password` will have to be changed to some valid credentials for the machine that the UI is hosted on – The UI will run under the context of this user. The reason for this is so that when an XML file is saved, the application has the permissions to write to the file (which, in the case of IIS, running under an anonymous user, it would not be able to do so). It is best to set up a user with the minimum privileges necessary to perform this action to preserve security.

In the authentication section of the Web.config file, you will notice the following node –

```
<user name="admin" password="21232f297a57a5a743894a0e4a801fc3" />
```

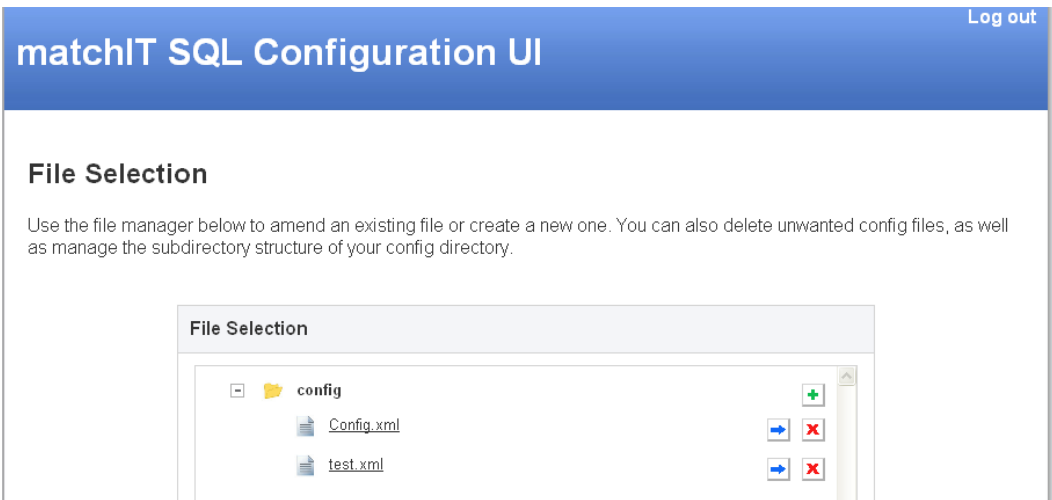
These are the credentials (with the password stored as an MD5 hash) that you will need to enter when logging into the application. The default values for the username and password for a fresh install are 'admin' and 'admin' respectively. We recommend that you generate an MD5 hash for a stronger password and use this instead.

After browsing to the UI through the start menu you should see the following.

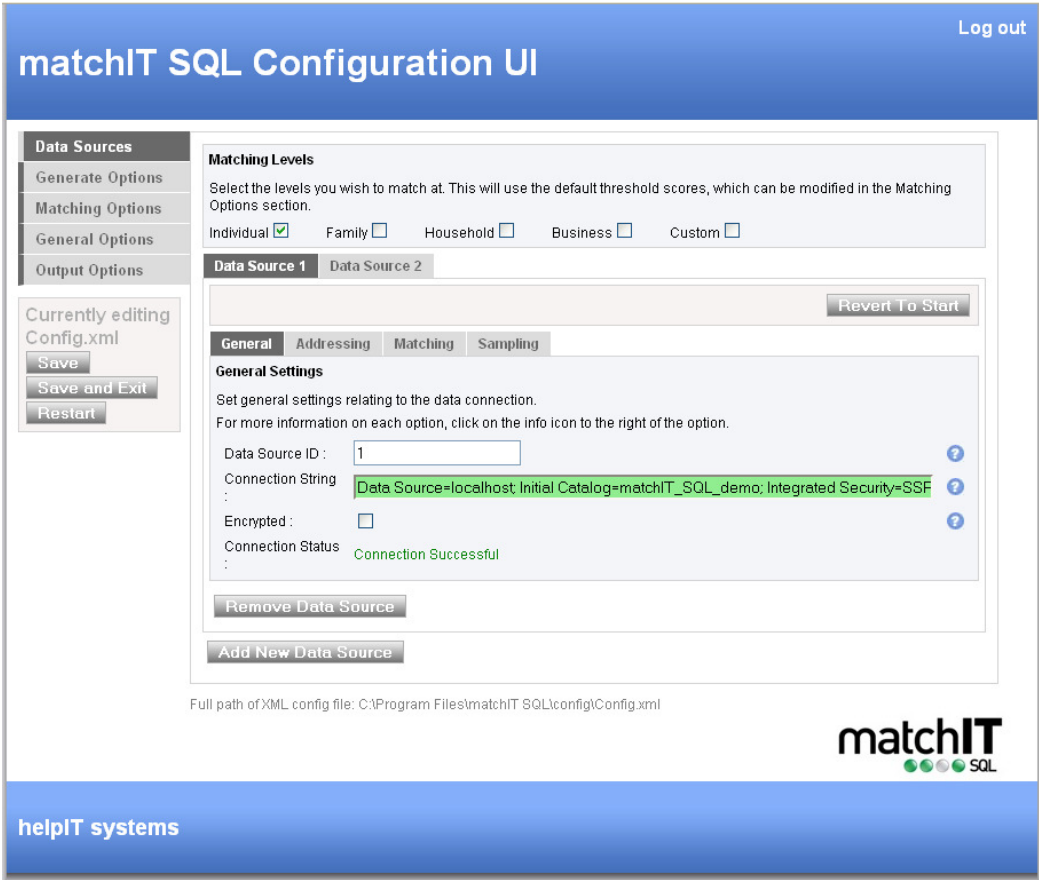


The screenshot shows the matchIT SQL Configuration UI login page. The page has a blue header with the text "matchIT SQL Configuration UI". Below the header is a white box containing a "Login" section. The "Login" section has the text "Please Enter Username and Password" and two input fields: "Username:" and "Password:". Below the input fields is a "Log In" button. In the bottom right corner of the white box is the matchIT logo, which consists of the text "matchIT" and "SQL" with three colored dots (green, yellow, red) between them. At the bottom of the page is a blue footer with the text "helpIT systems".

Simply enter the default username and password (admin/admin or otherwise if you have changed it) to log into the UI. After doing so, you will be presented with the following screen, where you can either choose to edit and existing configuration file or create a new one.



Note that the folder tree on the left shows the list of XML files that are in the 'Config' directory of the matchIT SQL installation – The UI is set to point to this in the Web.config file in the appSettings section, as well as specifying what file to use as a base file for creating new configuration files and reverting to default settings (Template.xml by default, which is omitted from the edit list on the left – this file is read-only). You will also notice a 'Log out' link at the top right of the screen that you can use to get back to the 'Log In' screen. Once you select to edit or create a file, the screen will look as follows.



You are now in a position to edit / create the file you have selected. The navigation of the form follows the logical structure of the XML and should be easy to use. On the left hand side you will see three buttons - 'Save', 'Save and Exit' and 'Restart' – which allow you to save changes and return to the config file selection page. At the top of most sections you will see the 'Revert To Start' and 'Default Settings' buttons which revert the changes you have made to the section you are on back to the settings you started with or to the default settings respectively. Note that the Data Sources section doesn't have a 'Default Settings' button for each data source, as there are no real 'Default Settings' for a data source.

If you are unsure of any section you are looking at, you can use the 'help' prompts in each section to point you in the right direction. These are the small blue circles containing a white question mark located on the right. Clicking on them will expand information on the section they are related to.

6 matchIT SQL Component Overview

This chapter guides you through at a high level the key components that comprise matchIT SQL.

6.1 *Stored Procedures*

These provide the high-level functionality; they're responsible for such things as accessing the database, finding records to compare, and importing results.

6.2 *SSIS Components*

These provide the same level of high-level functionality as the stored procedures but are delivered through Microsoft's SSIS Software. SSIS packages provide a rapid way of implementing your matching process and can be scheduled from SQL Management Studio's SQL Server Agent.

6.3 *Worker Processes*

These provide a safe interface to the matchIT API, an unmanaged COM component for generating search keys and comparing records. Should a problem be encountered within the unmanaged code, the service can automatically spawn a new worker and thus provide a highly stable and robust batch processing environment.

From a user perspective you will not need to worry about the workers, but we mention them here for the purposes of completeness.

6.4 *matchIT SQL Service*

This provides a link between the stored procedures and worker processes. When a stored procedure runs, it connects to the service which, in turn, creates a new worker. The stored procedure then connects to the worker, which will then provide the stored procedure with all matchIT API-related functionality.

From a user perspective you will not need to worry about the service, but we mention it here for the purposes of completeness.

6.5 *Error Logging*

When an error is generated through the matchIT SQL stored procedures or SSIS tasks, an error log will normally be created in the specified temporary folder (note that where there are actual problems with the XML configuration file itself, i.e. invalid XML then this error log cannot be produced). Normally if an error occurs, you should be able to review this error log and quickly see what the problem is. In the rare scenario that you are unable to determine the cause of the error yourself, then feel free to contact our support team and send through the error log file to support@helpit.com who will then be able to actively investigate the problem.

6.6 *matchIT SQL Monitor*

This isn't an essential component of the matchIT SQL system. It's a simple program that's useful for monitoring the status of the service and any active worker processes. It can be used to detect deadlocked and crashed workers and provides a convenient mechanism for terminating them so that the stored procedure can resume processing.

7 Stored Procedures

matchIT SQL provides stored procedures, organised into the following categories:

- Address Correction
- Key Generation
- Fuzzy Deduplication
- Exact Deduplication
- Output
- Triggers
- Miscellaneous.

7.1 Address Correction

The stored procedures within this section relate to validation of addresses against post office address files.

7.1.1 msp_GenerateCorrectedAddresses

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.

Generates corrected addresses for the source addresses specified in the supplied configuration. The corrected addresses will be written to the output table specified in the configuration, and can be subsequently used in matching processes.

The source and output mappings can be configured through the Web UI, or directly in the XML in the section called Addressing within the Datasource settings. The output table generated from the addressing step can then be used as the source table for your addresses within the matching step.

Setting	Description
dataSources	Specifies the database connection, table and column mappings used to define the dataset being processed. To use the resulting corrected address table produced on completion of this process, you should add this table into the datasource.
addressing->databaseSettings->SourceSettings->sourceTables	Table name containing the source address data that is to be used during processing.
addressing->databaseSettings->SourceSettings->sourceColumns	This allows the mapping of the source table's columns to specific address data types.
addressing->databaseSettings->OutputSettings->outputTable	Name of the table containing the corrected addresses that will be produced following completion of this process.
addressing->databaseSettings->OutputSettings->outputColumns	This allows the configuration of the output columns. The CorrectedMapping attribute is used to specify the address element that should go into a specific field should the source address be validated.

	The SourceMapping attribute specifies the source table's data field that will populate this field, should the source address fail validation.
addressing->databaseSettings->OutputSettings->outputColumns->TIGERData	This section allows configuration of TIGER Data appending (US Only). The enabled attribute specifies whether or not appending is turned on, and the onlyCodeValidAddresses attribute specifies whether or not to append data for non-verified addresses also. Each TIGER field can be specified as a sub node of this node, with the name of the field as the name of the node, and an enabled attribute to specify whether or not to append the given field. This defaults to false. See the section below for an explanation of each TIGER Data field. (Only applicable for US addressing.)
addressing->apiSettings->threadCount	Number of threads that may be used during the processing. Recommendation is to use 1 thread per machine core.
addressing->apiSettings->defaultAPI	When licenced for the UK and US addressing APIs, but not the international addressing API, this setting specifies which addressing API (UK or US) to use for international addresses.
addressing->progressLoggingSettings->filePath	Location of a progress log file which is output during Address processing.
addressing->progressLoggingSettings->interval	How frequently updates will be written to the progress log file (in milliseconds).

7.1.1.1 US Addressing Settings

Setting	Description
Addressing->apiSettings->enabledUS	When set to "false" the US addressing API is not used even when licenced. Default "true".
addressing->apiSettings->processing	Only applicable for US addresses that contain both a box number and a street number. Valid values are "PreferBox" and "PreferStreet"; on output the former will place box numbers on the first address line, while the latter will instead place street numbers on the first address line.
addressing->apiSettings->useMixed	If this is set to "true", data is returned in mixed-case format. If this is set to "false", result data is returned in all upper case characters.
addressing->apiSettings->useAlias	If this is set to "true", alias street names are returned when they are used in the input. If this is set to "false", 'official' street names are returned even when alias street names are used in the input. (An alias street name is an alternative name for a street that is acceptable to the USPS. It may be a name by which a street was formerly known, a commonly-used nickname for the street, or one the community prefers to use.)

7.1.1.2 UK Addressing Settings

Setting	Description
Addressing->apiSettings->enabledUK	When set to "false" the UK addressing API is not used even when licenced. Default "true".
addressing->apiSettings->processType	Value can be AddressCorrection or Postcode correction. Postcode correction is an option for UK addressing only and indicates that only postcodes are to be updated and not address lines. DPSOnly is an option for UK addressing that appends the DPS code without changing the address or postcode.
addressing->apiSettings->scoreThresholds	Any address level match falling below the "address" score threshold and any postcode level match falling below the "postcode" score threshold is downgraded to "tentative". The default values of 50 for both address and postcode are already quite strict. If your input address data contains a lot of extraneous text you might want to consider lowering these thresholds. You can safely lower these thresholds to 0 – in which case you will be relying solely on the addressing engine's internal checks.

addressing->apiSettings->capitalisePostTown	When set to True, postal towns will be capitalised in the output table.
addressing->apiSettings->keepNonPafData	When set to True, matchIT SQL will attempt to keep non PAF elements (i.e. extra data that may not be in the postal address file) when updating addresses. Sub options control which types of non-PAF data to keep: "above" – elements above the top of address (e.g. flat names) "within" – elements within the address (e.g. misspelled localities) "PNR" – elements recognised as PNR (Postally Not Required) localities.
addressing->apiSettings->preventCompanyUpdates	Stops organizations from being updated.
addressing->apiSettings->advancedFuzzy	When Advanced Fuzzy is disabled, the fuzzy matching algorithm employed only allows for one or two spelling errors in an entire name. Advanced Fuzzy acts at a word level, and allows names to be matched even if some words are missing (depending on how highly occurring the missing words are). Default "true" (Advanced Fuzzy enabled).
addressing->apiSettings->resubmitFailures	When enabled, failures are resubmitted with the first two populated address lines swapped. This is useful when subpremise details are on the line following the street name, instead of the line preceding the street name, e.g. "15 Grand Avenue, Flat 1". Default "true".
addressing->apiSettings->removePostcodeFromInputAddress	When a partial match is found only the postcode field is updated - any output address fields will be populated from the source address fields. With this option enabled a postcode or outward code found in the source address fields will be removed.
addressing->apiSettings->referenceDatabase	The name of a database configuration to use. This must be the name of a database configuration (or pool) defined in mconfig.ini (see below). Normally this will be "PAF".
addressing->apiSettings->outputDefaultDPS	With this option enabled a default "9Z" DPS code will be output when the address does not match to premise level but has a valid postcode.

The UK addressing engine uses a service called "Capscan Pool Manager" – this manages a pool of server processes that show up in task manager as "mcserver". The configuration of these server processes is via a file in the UK address data installation folder, called mconfig.ini. The matchIT SQL setup process installs and configures this automatically, so you shouldn't normally need to do any configuration manually, but you might want to increase the number of mcserver processes if running on a machine with multiple cores – to do this increase the values of InitialServers and MaxServers in parallel. Any changes to mconfig.ini will take effect when the "Capscan Pool Manager" service is restarted. The following table lists the other settings in the mconfig.ini file:

Section	Setting	Description
Link	Host	Must be localhost.
Link	DefConnectionMode	Must be 0.
Link	LogFile	Location of error log file.
Link	LicPath	Location of licence files.
Pool1	Name	Must be "PAF".
Pool1	InitialServers	The initial number of mcserver processes to launch. If the machine has multiple cores, set this to about half the number of cores.
Pool1	NewServers	The number of new mcserver processes to launch if none are available. This is only used in the unlikely event of an mcserver process crashing and needing to be replaced. Set to 1.
Pool1	MaxServers	The maximum number of mcserver processes to launch. Set this to the same value as InitialServers.
Pool1	Module	Path and name of the addressing engine dll: cpsvrnc5.dll
Pool1	ServerPath	Path and name of the addressing server: mcserver.exe
Pool1	PAF	The address database(s) to search. Normally just Capscan.paf.
Pool1	RCDB	The folder containing overlay data files. E.g. NSPDO.RCD (National Statistics

		Postcode Directory).
Pool1	MCDParam	Path and name of a file containing advanced configuration settings for addressing engine.
Pool1	AddrFrmt	Path and name of a file containing address formatting configuration.

7.1.1.3 International Addressing Settings

Setting	Description
Addressing->apiSettings->enabledInt	When set to "false" the international addressing API is not used even when licenced. Default "true".
addressing->apiSettings->outputCasing	Used to specify the letter case to use for output fields. Valid options are 'Upper' and 'Title' (default is 'Title').
addressing->apiSettings->confidenceThreshold	Used to specify the threshold at which to stop returning parse interpretations, as a percentage of the confidence of the top result (default is 90).
addressing->apiSettings->minimumMatchscore	Used to specify the minimum matchscore a record must reach in order to avoid reversion (default is 0, valid values are 0-100).
addressing->apiSettings->minimumPostcode	Used to specify the minimum postcode status a record must reach in order to avoid reversion (default is 0, valid values are 0-8).
addressing->apiSettings->minimumVerificationLevel	Used to specify the minimum verification level a record must reach in order to avoid reversion (default is 4, valid values are 0-5).
addressing->apiSettings->minimumVerificationLevelPerCountry	This can be used to override the minimum verification level (see above) for individual countries. This setting can contain multiple items separated by spaces, where each item is of the format "Country=Level" and Country is the three-letter country code as defined by ISO 3166-1. For example, "CAN=3 IND=3" provides levels for Canada and India only whilst all other countries will use the minimum verification level specified above.
addressing->apiSettings->intelligentScoring	If enabled, both the verification status (verified, partial, no match) and the verification level (delivery point, premise, thoroughfare, town, or region) are considered; if disabled, only the verification status is used.
addressing->apiSettings->defaultCountry	3-character country code (ISO 3166-1 alpha-3) of the country to assume when no country name or country code is explicitly specified and matchIT is unable to determine the country from other address details.

7.1.1.4 International Addressing Batch Cloud Settings

Setting	Description
addressing->apiSettings->cloudKey	Authenticates calls to the international addressing cloud web service; if not specified, then locally installed data will be used.
addressing->apiSettings->cloudUsername	Web service username of account to use.
addressing->apiSettings->cloudPassword	Password for web service account.
addressing->apiSettings->cloudJobname	Arbitrary name for job – useful for tracking itemised billing.

7.1.1.5 Input Fields

Some combination of the following fields must be input to the Generate Corrected Addresses process.

7.1.1.6 US Addressing Input Fields

Name	Description
Organization <i>or</i> Company	Specifies the company name.
Suite	Specifies the suite or unit.
Address1, Address2 <i>or</i> Street, Street2	Specifies address lines.

Urbanization	Specifies the urbanization.
City <i>or</i> Town	Specifies the city.
State <i>or</i> County <i>or</i> Region	Specifies administrative area.
Zip or Zipplus4 or Postcode	Specifies the 5-digit or 9-digit ZIP code.
Plus4	Specifies the ZIP+4 (if these are in separate columns).

7.1.1.7 UK Addressing Input Fields

Name	Description
Organization <i>or</i> Company	Specifies the company name.
Address1 – Address8	Up to 8 address lines can be specified.
Town <i>or</i> City	Specifies the posttown.
County <i>or</i> Region <i>or</i> State	Specifies the county.
Postcode <i>or</i> Zip	Specifies the postcode.

7.1.1.8 International Addressing Input Fields

Name	Description
Organization <i>or</i> Company	Specifies the company name.
Suite	Specifies the suite or unit.
Street, Street2, Address1 – Address12	Up to 12 address lines can be specified.
Urbanization	Specifies the urbanization.
City <i>or</i> Town	Specifies the city.
State <i>or</i> County <i>or</i> Region	Specifies administrative area.
Zip or Zipplus4 or Postcode	Specifies the 5-digit or 9-digit ZIP code.
Plus4	Specifies the ZIP+4 (if these are in separate columns).
Country	Specifies the country

7.1.1.9 Output Fields

The following result code fields are always output:

Name	Description
AddrEngine	This indicates the addressing API (UK, US, INTERNATIONAL) that each record is verified with (this is only output if licensed for multiple APIs).
PafFlag	A numeric representation of the PafDesc column.
PafDesc	Indicates the level at which an address was validated ("Verified", "Good", "Partial", "Tentative", and "NoMatch"). See engine specific sections below for details.
PafDescExtra	
ErrorCode	See engine specific description below.
AddrScore	See engine specific description below.
UpdateFlag	See engine specific description below.

User specified output fields have a source mapping and one or more corrected mappings.

Source Mapping - The default mapping for data to populate in the selected column when no address match can be found for the given input. Available values will be the Address Types assigned to the Source Columns. Select 'None' for no default data.

Corrected Mappings - The corrected elements to populate in the selected column when a match is found. See the available engine specific output fields in the following sections. Multiple elements are permitted for a single output column, and will be separated with a space when combined.

7.1.1.10 UK Addressing Output Fields

Corrected Mappings for UK Addressing may include:

Name	Description
Address1-8	A combination of the address elements listed below depending on their availability in the PAF file in the following order, SubBuildingName & BuildingName, BuildingNumber & Thoroughfare, DependantLocality. The Town and County details are output to specific Town and County fields.
Organisation	The organisation name listed on the PAF file.
BuildingName	The building name of the house or commercial premises.
BuildingNumber	The number of the building on a thoroughfare.
SubBuilding	When a building is split into a number of flats or business units, a sub building name will be returned.
Thoroughfare	The street that contains the delivery point.
DependentThoroughfare	If a thoroughfare exists more than once within a town, the PAF file may contain additional information to uniquely identify each one.
DoubleDependentLocality	Further subdivision within a DependentLocality.
DependentLocality	Locality (e.g. village or borough name within a town) used to differentiate between streets with the same name.
Town	The postal town for the address
County	The country the address relates to.
Postcode	The postcode the address relates to.
POBox	The POBox listed on the PAF for the address.
AddressKey	A combination of the 8 digit address key and 8 digit organisation key that uniquely identifies each delivery point padded with a leading 0 (where there is no organisation, 0's will be used).
DeliveryPointSuffix	The delivery point associated with the address to uniquely identify the postcode.
Barcode	CBC Barcode
Easting	5 digit code relating to the location of the postcode to the National Grid.
Northing	5 digit code relating to the location of the postcode to the National Grid.
CountryCode	ONS Country Code: L93000001 - Channel Islands E92000001 - England M83000003 - Isle of Man N92000002 - Northern Ireland S92000003 - Scotland W92000004 - Wales
Latitude	WGS84 (World Geodetic Standard 1984) datum Latitude in degrees decimal. E.g. 51.5267183130
Longitude	WGS84 (World Geodetic Standard 1984) datum Longitude in degrees decimal. E.g. -0.1888023154
LEA	Local Education Authority
DHA	Health Area Code
PCG	Primary Care Trust Code
OSCTY	Ward County Code

OSLAUA	Ward District Code
OSWard	Ward Code
GridQuality	Grid Quality
GOR	Government Office Region Code
CONS	Constituency
EER	European Electoral Region

7.1.1.11 UK Addressing Return Codes

This section describes the following return codes: PafFlag, PafDesc, ErrorCode, AddrScore, UpdateFlag.

The PafFlag and PafDesc fields indicate whether a record has been verified or not during the addressing process:

PafFlag	PafDesc	Description
1	Verified	The record achieved a premise level match from the addressing engine and reached the address level score threshold. This is a confident match. The address and postcode fields will be updated.
2	Good	The record achieve a postcode level match and all input address elements were matched so the result was upgraded to an address level match. The address and postcode fields will be updated.
3	Partial	The record achieved a postcode level match and reached the postcode level score threshold. The postcode field will be updated.
4	Tentative	No Premise information (Org, Sub-building, Building Number, Building Name) was returned, or the score threshold was not met. The address and postcode fields will not be updated.
5	NoMatch	No match was found.
6	Foreign	The record was identified as foreign.
9	ProcessingFailure	An error has occurred processing the record.

The ErrorCode field is a 17 character field that indicates the verification status. For example, "PREMP===CVVVARXXF". This consists of nine parts in the following format:

ErrorCode part	Width	Description
Match level	4	Match level can have the following values: "PREM" – Match to a single premise; "PCOD" – Match to a complete postcode; "PART" – Match to a partial postcode; "NOMA" – No satisfactory match could be found; "FORN" – Foreign address detected.
Passing level	1	Passing level can have the following values: "A" – Address level; "P" – Postcode level; "N" – No match.
Postcode update	4	Postcode update can have the following values: " " (four spaces) – No postcode or no match; "----" – Input postcode, no output postcode; "++++" – Output postcode, no input postcode; "====" – No change to postcode; "===C" – Change at postcode level;

		"=CC" – Change at sector level; "=CCC" – Change at outward code level; "CCCC" – Change at area code level.
Premise flag	1	Premise flag can have the following values: "X" – Not matched; "V" – Verified; "R" – Retained; "C" – Corrected; "A" – Added.
Street flag	1	See premise flag.
Locality flag	1	See premise flag.
Postcode flag	1	See premise flag.
Vanity flags	3	Leading flag can have the following values: "XXX" – Not matched; "SSS" – Some vanity elements retained; "RRR" – Vanity elements retained; "DDD" – Vanity elements removed.
Reformat flag	1	Reformat flag can have the following values: "X" – Not matched; "O" – Original address kept as-is; "F" – Address fully reformatted.

Values for AddrScore are between 0 and 110 – this is essentially a percentage score boosted by 10 if there are no unmatched words in the input.

The UpdateFlag field indicates whether the record was updated:

UpdateFlag	Description
Address	Record address and postcode were updated.
Postcode	Record postcode was updated.
None	Record not updated.

7.1.1.12 UK Addressing Optional Output Fields

The following sets of fields are made available via separately licensed optional datasets.

The Royal Mail UDPRN database provides a Unique Delivery Point Reference Number for each address that does not change during the lifetime of the premises. This contains the following fields:

Name	Description
UDPRN	Unique Delivery Point Reference Number

The Royal Mail Multiple Residence database provides details of addresses that share a communal front door – typically flats. In these cases, the standard PAF database contains only one address representing the communal front door, the "owning address". The Multiple Residence database has the standard address fields for each of the flats behind the communal front door plus these additional fields:

Name	Description
UMRRN	Unique Multiple Residence Reference Number

OwningUDPRN	UDPRN of owning DP
OwningAKOK	Address and Organisation Key of owning DP

The Royal Mail Not Yet Built database contains addresses where planning permission has been granted but the premise is not yet built. N.B. these are not plot number addresses. At the time that planning permission is granted, the local authority assigns an official address and Royal Mail assigns a postcode. As soon as the premise is able to receive mail the address is moved from the Not Yet Built database to standard PAF.

The Not Yet Built database has the standard address fields plus these additional fields:

Name	Description
NOTYETBUILT	Not Yet Built flag – 'Y' for all Not Yet Built addresses
NYB_UDPRN	Unique Delivery Point Reference Number

The Dun & Bradstreet Business Database consists of over 3.4 million actively trading organizations, ranging from small businesses and shops through to blue-chip corporations. In addition to the standard address fields, the D&B Business data has the following fields:

Name	Description
DUNS	D-U-N-S Number
TEL	Telephone number
FAX	Fax number
SIC03	UK 2003 SIC code
SIC72	US '72 SIC code
NUMEMP	Number of employees at site
NUMEMPCO	Number of employees at company
HOF	Head office flag
TO	Turnover
YEARSTAR	Year started

The Dun & Bradstreet Business Contact Details Database provides the following contact details:

Names	Description
ExecutiveCount	Count of named executives
Ex1_Function ... Ex8_Function	Executive function
Ex1_FirstName ... Ex8_FirstName	Executive first name
Ex1_Surname ... Ex8_Surname	Executive surname
Ex1_Salutation ... Ex8_Salutation	Executive salutation
Ex1_Sex ... Ex8_Sex	Executive gender
FinFunction, FinFirstName, FinSurname, FinSalutation, FinSex	Contact details for finance responsibility
HRFunction, HRFirstName, HRSurname, HRSalutation, HRSex	Contact details for HR responsibility
ITFunction, ITFirstName, ITSurname, ITSalutation, ITSex	Contact details for IT responsibility
MktingFunction, MktingFirstName, MktingSurname, MktingSalutation, MktingSex	Contact details for marketing responsibility
NworkFunction, NworkFirstName, NworkSurname, NworkSalutation, NworkSex	Contact details for networks responsibility
PurchFunction, PurchFirstName, PurchSurname, PurchSalutation, PurchSex	Contact details for purchasing responsibility

SalesFunction, SalesFirstName, SalesSurname, SalesSalutation, SalesSex	Contact details for sales responsibility
DecsFunction, DecsFirstName, DecsSurname, DecsSalutation, DecsSex	Contact details for decisions responsibility
CommsFunction, CommsFirstName, CommsSurname, CommsSalutation, CommsSex	Contact details for tech comms responsibility

7.1.1.13 US Addressing Output Fields

Corrected Mappings for US Addressing may include:

Name	Description
Company	Mapping a company will provide the capability of appending the secondary (suite) information to a business address providing that the input address is determined to be a highrise default record.
StreetOnly	This is the first half of a parsed version of Street, when inputting an address of "123 Main St Ste 9" in Street, the StreetOnly element would contain only the "123 Main St" when outputting the corrected mappings. StreetOnly would also contain the PO Box and #, if the address were a PO Box.
Alternates	In situations where there are 2 valid addresses provided (a PO Box and a street address) – The user can specify a preference to choose when both are valid. When inputting dual addresses with a PO Box preference set, then the street address would be moved to alternates, as the zip+4 is assigned based on the primary address. In cases where there are dual addresses but one is not valid, then the invalid address elements will be put into the Street2 column.
Street	This is a full first address line – it will include the PO Box, or the street and suite information in a single line.
Street2	Used when there is more than one street input line, if the user has multiple street lines to input, or if street and suite elements are already parsed then map the columns in as Street and Street2 – Processing on that address may happen twice in case data comes in transposed to help compensate for bad elements being put into the street line – when dealing with dual address elements. When an address is verified – data returned into street2 is not a valid part of the verified address and may include attention lines, or other address elements that were not put into Alternates or Leftovers.
Suite	This element contains both the Suite descriptor (Ste, Apt, Fl) and the secondary number. A valid input of "123 Main St Ste 9" in Street, would result in "Ste 9" being populated into the output column.
Urbanization	In Puerto Rico, repeated street names and address number ranges can be found within the same ZIP Code (e.g., CALLE 1, CALLE 2, etc.). These streets can have the same house number ranges (e.g., 1-99). In these cases, the urbanization name is the only element that correctly identifies the location of a particular address.
City	The City part of the address.
State	State Abbreviations - From AL (Alabama) to WY (Wyoming).
Zip	The first five digits of the zip code.
Plus4	The last four digits of the zip code.
ZipPlus4	The Zip and Plus4 concatenated into a single column with the dash between the Zip and Plus4. Ex: 90078-0907
DPV	This indicates whether an address is a valid delivery point – See Section 6.1.1.2 for additional information.
DPVnotes	USPS Standardized Footnotes – See Section 6.1.1.2 for additional information.
Corrections	Same as Errorcode which is output by default– CASS Correction Codes – See Section 6.1.1.2 for more detailed information.
Score	Same as AddrScore which is output by default – CASS Certification Return Codes - See Section 6.1.1.2 for more detailed information.
Housenum	This is the first numeric number in the street line. Examples: 422 W 5th St – Housenum would be 422

	PO Box 37 – Housenum would be 37
PreDir	The predirectional part of the street line Example: 422 W 5th St – PreDir would be W
StreetName	The name of the street in the street line, or when the address is a PO Box then it will indicate the PO Box. Examples: 422 W 5th St – Streetname would be 5th PO Box 37 – Streename would be PO Box
PostDir	Post Directional part of the street line after the street’s suffix Example: 422 Anderson St NW – Post Direction would be the NW
StreetSuffix	The street line’s suffix information – For example, AVE (Avenue), BYU (Bayou), BLVD (Boulevard) or other type of RD (Road). Example: 422 W 5th St – Streetsuffix would be St
Sud	Secondary Unit Designators - Indicates the type of residential or commercial unit mail is sent to, such as APT (apartment), STE (suite), or TRLR (trailer). Example: 422 W 5th St Ste 240 – Sud would be Ste
UnitName	The unit number of the address Example: 422 W 5th St Ste 240 – UnitNum would be 240
PMB	This is the number of a Private Mail Box.
Leftovers	Preserved information at the end of a street address line that are not part of a valid address, but makes the address valid when removed.
Dpb	A combination of the Delivery point code and the check digit. When printing barcodes you would concatenate the Zip, plus 4, then this Dpb column.
Delivery Point Code	The code by which a mail piece can be sorted by its address. This code is formatted from the ZIP + 4.
Check Digit	The one-digit number that is part of each bar code. It is required for all customer-generated special services forms to delete errors resulting from manual data entry or errors from transmitted data.
Barcode	DPBC - The delivery point barcode, or DPBC, is a line of short and tall bars that helps identify a mailing's exact destination. 11-digit code representing a nine-digit ZIP code plus two additional digits, used to address mail plus the check digit – There are 12 digits in total. This field contains a numeric representation of the barcode.
Crrt	Carrier Route - The addresses which a carrier delivers mail. A Carrier Route includes city routes, rural routes, highway contract routes, Post Office box sections and general delivery units.
LotCode	eLOT™ sequence number eLOT Travel Line Product was developed to give mailers the ability to sort their mailings in approximate carrier-casing sequence. To aid in mail sortation, eLOT contains an eLOT sequence number, this number indicates the first occurrence of delivery made to the add-on range within the carrier route.
LotDir	eLOT Ascending/Descending Indicator The ascending/descending code indicates the approximate delivery order within the sequence number.
CountyNum	The FIPS county code is the Federal Information Processing Standard (FIPS) county code.
CountyName	County - The largest administrative division of most states in the United States.
CongDist	Congressional District.

LACS	LAC Indicator The LAC Address Conversion Service indicator describes records that have been converted by LAC Service from rural route to city-style addresses so that emergency vehicles (e.g., ambulances, police cars, etc.) can more easily find these locations.
DpvAnswer	DPV match indicator – First part of the DPV field, see Section 6.1.1.2 for additional information.
DpvCMRA	Commercial Mail Receiving Agency Indicator – Second part of the DPV field, see Section 6.1.1.2 for additional information.
DpvFalsePositive	3rd part of DPV field, indicates whether it was found or not in the false positive table when DPV processing was performed, see Section 6.1.1.2 for additional information.
Rdi	RDI or Residential Delivery Indicator identifies whether an address is classified as residential or business.

7.1.1.14 US Addressing Return Codes

This section describes the following return codes: PafFlag, PafDesc, ErrorCode, AddrScore, UpdateFlag, DPV, DPVNotes.

The PafFlag and PafDesc fields indicate whether a record has been verified or not during the addressing process:

PafFlag	PafDesc	Description
1	Verified	The record received an AddrScore of 0.
2	Good	The record achieved an AddrScore of 1.
3	Partial	The record achieved an AddrScore of 5, 8 or 9.
4	NoMatch	The record did not receive an AddrScore of 0,1,5,8 or 9.

An ErrorCode field will be created containing correction code information as follows:

ErrorCode	Description
A	Normal street match.
B	PO Box match
C	Route type match
D	Unique Zip match
E	Small town match
F	Alias match.
G	Highrise alternate match.
H	Firm match.
I	Highrise match.
J	Highrise default match.
K	Route default match.
L	Street name corrected.
M	Street Suffix corrected.
N	Pre-directional corrected.
O	Post-directional corrected.
P	City corrected.
Q	State corrected.
R	Zip corrected.
S	Urbanization corrected.
T	Zip+4 corrected.

U	House number corrected.
V	Unit number corrected.
W	Secondary unit designator corrected.
X	Firm corrected.
Y	Street swapped with firm.
Z	Street swapped with alternate.
0	Dual address changed to PO Box.
1	Dual address street match.
2	Input city is not preferred but is acceptable.
3	Street standardised.
4	Unit not verified.
5	Leftovers found.
6	Zip move match.
7	LACSLINK match.
8	SuiteLink Match.

The AddrScore field can have the following values:

AddrScore	Description
-1	An Error occurred when processing the address.
0	Address was successfully verified.
1	Address is coded but undeliverable (i.e. on side of street known to contain no houses).
2	The zip code was not found and the city and state cannot be used to determine a geographical area to search.
3	Coding would result in changing both the zip and city.
4	The best match would result in too many suspicious changes.
5	The street was identified as an alias, but was out of the range restricted for that alias.
6	No street address was given.
7	There are no street name matches in the given zip code or in any geographically-related zip code.
8	The street may contain superfluous components which cannot be discarded with confidence.
9	The house number could not be matched.
10	The best match was made to a zip move record but was not an exact match.
11	A zip move match was made, but no exact match could be found in the new zip.
12	Insufficient address information. It's not even possible to guess as to what might be correct.
13	There are multiple matches with the same degree of confidence.
14	Incorrect suffix, directional, street name or unit resulted in multiple matches with the same degree of confidence.
15	Incorrect zip, city or urbanization resulted in multiple matches with the same degree of confidence.
16	A Corrected field was too long to fit into the supplied field.
17	Media Error. The database could not be read because of a hardware or system problem.

The UpdateFlag field indicates whether the record was updated:

UpdateFlag	Description
Address	Record address and zipcode were updated.
None	Record not updated.

The DPV field is 5 characters wide; it enhances data by adding the Delivery Point Validation information generated by the addressIT module. The placement of the character code indicates which process was performed. The meanings of the five character positions are summarized in the following table.

DPV character	Position	Description
DPV Confirmation Indicator	1	The DPV Confirmation Indicator is the primary method used by the USPS to determine whether an address was considered deliverable or undeliverable. Blank - Address was not assigned a Zip+4 and therefore no DPV processing was performed. Y - Address was DPV confirmed for both primary and (if present) secondary numbers. D - Address was DPV confirmed for primary number only, and Secondary number information was missing. S - Address was DPV confirmed for primary number only and secondary number information was missing. N - Both Primary and (if present) Secondary number information failed to DPV Confirm.
DPV CMRA Indicator	2	CMRA (Commercial Mail Receiving Agency) Indicates a private business that acts as a mail-receiving agent for specific clients. Blank - Address was not assigned a Zip+4 and therefore no DPV processing was performed. Y - Address was found in CMRA table. N - Address was not found in CMRA table
DPV False Positive Indicator	3	The False Positive table flags the False Positive addresses. This is a flag to determine whether a mailing list is being generated or created during validation. Creating a mailing list through DPV certification is not allowed by the USPS. Blank - Address was not assigned a Zip+4 and therefore no DPV processing was performed. Y - Address was found in False Positive table. N - Address was not found in False Positive table.
Vacant Indicator	4	A delivery point was active in the past, but is currently vacant (in most cases, unoccupied over 90 days) and not receiving delivery. Blank - Address was not assigned a Zip+4 and therefore no DPV processing was performed. Y - Address was found in the VACANT table. N - Address was not found in the VACANT table
DSF2 No Stats Indicator	5	Indicates the address is not receiving delivery, and the address is not counted as a possible delivery. These addresses are not receiving delivery because A) delivery has not been established; B) customer receives mail as a part of a drop; or C) the address is no longer a possible delivery because the carrier destroys or returns all of the mail. Blank - Address was not assigned a Zip+4 and therefore no DPV processing was performed. Y - Address was found in NOSTATS table. N - Address was not found in NOSTATS table

The DPVNotes field can contain any combination of the following codes.

USPS Standardized Footnotes Reporting CASS Zip+4 Certification

- AA – Input address matched to the ZIP+4 file.
- A1 – Input address not matched to the ZIP+4 file.

Footnotes Reporting DPV Validation Observations

- BB – Matched to DPV (all components).
- CC – Primary number matched to DPV, but secondary number not matched (present but invalid).
- F1– Input Address Matched to a Military Address.
- G1– Input Address Matched to a General Delivery Address.
- N1 – Primary number matched to DPV, but high-rise address missing secondary number.
- M1 – Primary number missing.
- M3 – Non-postal Primary number invalid.
- P1 – Input Address RR or HC Box number Missing.
- P3 – Input Address PO, RR, or HC Box number Invalid.
- U1– Input Address Matched to a Unique ZIP Code.

Footnotes Reporting CMRA Observation

A commercial mail-receiving agency (CMRA) is a private business that acts as the mail receiving agent for specific clients by providing a delivery address and other services. If the address matches to a CMRA location one of the following footnotes will appear.

- RR – Matched to CMRA.
- R1 – Matched to CMRA but Secondary Number not Present.

If you require further details of these error codes, please contact the support team at support@helpit.com.

7.1.1.15 US Addressing Optional Output Fields

TIGER (Topologically Integrated Geographic Encoding and Referencing system) Data is a USPS data set that allows ZIP codes to be matched to geographical location. A user will submit a ZIP+4 code and the additional data files will work out the location of the centroid for this location, and then present longitude and latitude data back in the XML return value. TIGER Data is enabled for matchIT SQL only if the relevant TIGER Data files exist in the Addressing Data Directory. These are called tiger.dat and tiger.idx.

The following are the fields available from the TIGER Data processing, and their descriptions.

Name	Description
InputZip	The formatted zip code that is passed to the TIGER Data engine. Zips are formatted to be 9 characters long by the GenerateCorrectedAddresses process, which involves padding 5 digit zips with 4 zeros to make a Zip+4.
Latitude	The angular distance north or south from the equator of a point on the earth's surface, measured on the meridian of the point.
Longitude	The angular distance east or west on the earth's surface, measured by the angle contained between the meridian of a particular place and some prime meridian, as that of Greenwich, England, and expressed either in degrees or by some corresponding difference in time.
Block	Blocks are numbered uniquely within each census tract with a 3-character number that identifies the collection block used in the census and a character block suffix. This character block suffix is often blank.
CMSA	A 4-digit code assigned to areas that consist of primary metropolitan statistical areas.

LatFrom	The north/south measurement indicating the beginning point of the address.
LatTo	A north/south measurement indicating the ending point of the address.
LongFrom	The east/west measurement indicating the beginning point of the address.
LongTo	The east/west measurement indicating the ending point of the address.
Plus4	Describes the last four positions of a ZIP+4 Code. Most delivery addresses are assigned a single ZIP+4 Code. However, large companies may be given a range of ZIP+4 Codes that can be used to route mail to a specific department.
PMSA	A 4-digit code assigned to areas that comprise one or more counties, including a major population nucleus and nearby communities that have a high degree of interaction.
Side	Sides indicate what side of the line segment the location occurs at, this is why Lat/Long From and To are returned, to allow identification of the location based on the range of the location. The side is loosely related to the side of a street or highway the ZIP+4 is located, but translating this to an actual side of a road is unreliable.
Tract	Small, locally delineated statistical areas within selected counties, generally having stable boundaries and, when first established by local communities, designed to have relatively homogeneous demographic characteristics.
MatchLevel	The level at which TIGER data was retrieved for the given zip. The levels are zip5, zip7 and zip9.
ProcessingMessage	Describes the status of the TIGER data result

7.1.1.16 International Addressing Output Fields

Corrected Mappings for International Addressing may include:

Name	Description
Latitude	This field holds the WGS 84 latitude in decimal degrees format.
Longitude	This field holds the WGS 84 longitude in decimal degrees format.
GeoAccuracy	This field holds the GeoAccuracy (GAC) code.
GeoDistance	This field holds the radius of accuracy in meters, giving an indication of the likely maximum distance between the given geocode and the physical location. For Point-level results (GAC = "PX"), this value may be empty or not returned.
Address	This field holds the full address.
Address1-12	These fields can be used to specify input address line data, and on output will contain the correctly formatted address for mailing in the relevant country, split into individual address lines.
DeliveryAddress	This field holds the full address minus the Organization, Locality hierarchy, AdministrativeArea hierarchy and PostalCode hierarchy fields, correctly formatted for mailing in the relevant country, including line breaks specified using the AddressLineSeparator option.
DeliveryAddress1-DeliveryAddress12	These fields contain the individual lines contained within the DeliveryAddress field.
CountryName	This field holds the ISO 3166 official country name.
ISO3166-2	This field holds the ISO 3166 2-character country code.
ISO3166-3	This field holds the ISO 3166 3-character country code.
ISO3166-N	This field holds the ISO 3166 3-digit numeric country code.
SuperAdministrativeArea	This field holds the largest geographic data element within a country.
Region	This field holds the most common geographic data element within a country. For instance, USA State, and Canadian Province.
SubAdministrativeArea	This field holds the smallest geographic data element within a country. For instance, USA County.
Town	This field holds the most common population center data element within a country. For instance, USA City, Canadian Municipality.
Thoroughfare	This field holds the most common street or block data element within a country. For instance, USA Street.

DependentThoroughfare	This field holds the dependent street or block data element within a country. For instance, UK Dependent Street.
DoubleDependentLocality	This field holds the smallest population center data element, dependent on both the contents of the Locality and DependentLocality fields. For instance, UK Village.
DependentLocality	This field holds a smaller population center data element, dependent on the contents of the Locality field. For instance, Turkish Neighborhood.
Building	This field contains the descriptive name identifying an individual location, should one exist.
Premise	This field contains the alphanumeric code identifying an individual location, should one exist.
SubBuilding	This field contains the secondary identifiers for a particular delivery point. For instance, "FLAT 1" or "SUITE 212".
Postcode	This field contains the complete postal code for a particular delivery point, should such detail be able to be determined.
PostcodePrimary	This field contains the primary postal code used for a particular country. For instance, USA Zip, Canadian Postcode, Indian PINcode.
PostcodeSecondary	This field contains secondary postal code information, if used in a particular country and if such detail is able to be determined and reference data is available. For instance, USA Zip Plus 4.
Organization	This field contains the business name associated with a particular delivery point, should one exist.
PostBox	This field contains the post box for a particular delivery point, should one exist.
LeftOvers	
Contact	
Function	
Department	
ThoroughfarePreDirection	This field holds the prefix directional contained within the Thoroughfare field, should one exist. For instance, if Thoroughfare contains "N MAIN ST" ThoroughfarePreDirection contains "N" if a sufficient level of parsing detail exists for the particular country.
ThoroughfareLeadingType	This field holds the leading thoroughfare type indicator within the Thoroughfare field, should one exist. For instance, if Thoroughfare contains "RUE DE LA GARE" ThoroughfareLeadingType contains "RUE" if a sufficient level of parsing detail exists for the particular country.
ThoroughfareName	This field holds the name indicator within the Thoroughfare field, should one exist. For instance, if Thoroughfare contains "N MAIN ST" ThoroughfareName contains "MAIN" if a sufficient level of parsing detail exists for the particular country.
ThoroughfareTrailingType	This field holds the trailing thoroughfare type indicator within the Thoroughfare field, should one exist. For instance, if Thoroughfare contains "N MAIN ST" ThoroughfareTrailingType contains "ST" if a sufficient level of parsing detail exists for the particular country.
ThoroughfarePostDirection	This field holds the postfix directional contained within the Thoroughfare field, should one exist. For instance, if Thoroughfare contains "MAIN ST N" ThoroughfarePostDirection contains "N" if a sufficient level of parsing detail exists for the particular country.
DependentThoroughfarePreDirection	See description for ThoroughfarePreDirection
DependentThoroughfareLeadingType	See description for ThoroughfareLeadingType
DependentThoroughfareName	See description for ThoroughfareName
DependentThoroughfareTrailingType	See description for ThoroughfareTrailingType
DependentThoroughfarePostDirection	See description for ThoroughfarePostDirection
BuildingLeadingType	This field holds the leading building type indicator within the Building field, should one exist. For instance, if Building contains "BLOC C" BuildingLeadingType contains "BLOC" if a sufficient level of parsing detail exists for the particular country.

BuildingName	This field holds the name indicator within the Building field, should one exist. For instance, if Building contains "WESTMINSTER HOUSE" BuildingName contains "WESTMINSTER" if a sufficient level of parsing detail exists for the particular country.
BuidlingTrailingType	This field holds the trailing building type indicator within the Building field, should one exist. For instance, if Building contains "WESTMINSTER HOUSE" BuildingTrailingType contains "HOUSE" if a sufficient level of parsing detail exists within a particular country.
PremiseType	This field contains the leading premise type indicator within the Premise field, should one exist. For instance, if Premise contains "Plot 7/7A" PremiseType contains "Plot" if a sufficient level of parsing detail exists within a particular country.
PremiseNumber	This field contains the alphanumeric indicator within the Premise field, should one exist. For instance, if Premise contains "Plot 7/7A" PremiseNumber contains "7/7A" if a sufficient level of parsing detail exists within a particular country.
SubBuildingType	This field contains the sub-building type indicator within the SubBuilding field, should one exist. For instance, if SubBuilding contains "FLAT 1" SubBuildingType contains "FLAT" if a sufficient level of parsing detail exists within a particular country.
SubBuildingNumber	This field contains the alphanumeric indicator within the SubBuilding field, should one exist. For instance, if SubBuilding contains "FLAT 1" SubBuildingNumber contains "1" if a sufficient level of parsing detail exists within a particular country.
SubBuildingName	This field contains the descriptive name within the SubBuilding field. For instance, if SubBuilding contains "BASEMENT FLAT" SubBuildingName contains "BASEMENT FLAT".
OrganizationName	This field contains the name indicator within the Organization field, should one exist. For instance, if Organization contains "Loqate Inc" OrganizationName contains "Loqate" if a sufficient level of parsing detail exists for the particular country.
OrganizationType	This field contains the trailing type indicator contained within the Organization field, should one exist. For instance, if Organization contains "Loqate Inc" OrganizationType contains "Inc" if a sufficient level of parsing detail exists for the particular country.
PostBoxType	This field contains the type indicator contained within the PostBox field, should one exist. For instance, if PostBox contains "PO BOX 1234" PostBoxType contains "PO BOX" if a sufficient level of parsing detail exists for the particular country.

7.1.1.17 International Addressing Return Codes

This section describes the following return codes: PafFlag, PafDesc, ErrorCode, AddrScore, UpdateFlag.

The PafFlag and PafDesc fields indicate whether a record has been verified or not during the addressing process:

PafFlag	PafDesc	Description
1	Verified	A complete match was made between the input data and a single record from the available reference data.
3	Partial	A partial match was made.
5	NoMatch	No match was found.
9	ProcessingFailure	An error has occurred processing the record.

The ErrorCode field is a 14 character field that indicates the verification status. For example, "V44-I44-P3-100". This consists of the following single character codes:

ErrorCode part	Width	Description
----------------	-------	-------------

Verification Status	1	<p>Verification status can have the following values</p> <p>V: Verified - A complete match was made between the input data and a single record from the available reference data;</p> <p>P: Partially Verified - A partial match was made between the input data and a single record from the available reference data;</p> <p>U: Unverified - Unable to verify. The output fields will contain the input data;</p> <p>A: Ambiguous - More than one close reference data match;</p> <p>C: Conflict - More than one close reference data match with conflicting values;</p> <p>R: Reverted - Record could not be verified to the specified minimum acceptable level. The output fields will contain the input data.</p>
Post-Processed Verification Match Level	1	<p>The post-processed verification match level gives the level to which the input data matches the available reference data once all changes and additions performed during the verification process have been taken into account.</p> <p>5: Delivery Point (PostBox or SubBuilding);</p> <p>4: Premise (Premise or Building);</p> <p>3: Thoroughfare;</p> <p>2: Locality;</p> <p>1: AdministrativeArea;</p> <p>0: None.</p>
Pre-Processed Verification Match Level	1	<p>The pre-processed verification match level gives the level to which the input data matches the available reference data prior to any changes or additions performed during the verification process.</p> <p>5: Delivery Point (PostBox or SubBuilding);</p> <p>4: Premise (Premise or Building);</p> <p>3: Thoroughfare;</p> <p>2: Locality;</p> <p>1: AdministrativeArea;</p> <p>0: None.</p>
Separator	1	-
Parsing Status	1	<p>I: Identified and Parsed - All input data has been able to be identified and placed into components;</p> <p>U: Unable to parse - Not all input data has been able to be identified and parsed.</p>
Lexicon Identification Match Level	1	<p>The lexicon identification match level gives the level to which the input data has some recognized form, through the use of pattern matching (e.g. a numeric value could be a premise number) and lexicon matching (e.g. 'rd' could be a ThoroughfareType, 'Road'; 'London' could be a Locality)</p> <p>5: Delivery Point (PostBox or SubBuilding);</p> <p>4: Premise (Premise or Building);</p> <p>3: Thoroughfare;</p> <p>2: Locality;</p> <p>1: AdministrativeArea;</p> <p>0: None.</p>
Context Identification Match Level	1	<p>The context identification match level gives the level to which the input data can be recognized based on the context in which it appears. This is the least accurate form of matching and is based on identifying a word as, for instance, a Thoroughfare based on it being preceded by something that could be a Premise, and followed by something that could be a Locality, the latter items being identified through a match against the reference data or the lexicon.</p> <p>5: Delivery Point (PostBox or SubBuilding);</p> <p>4: Premise (Premise or Building);</p> <p>3: Thoroughfare;</p>

		2: Locality; 1: AdministrativeArea; 0: None.
Separator	1	-
Postcode Status	2	P8: PostalCodePrimary and PostalCodeSecondary verified; P7: PostalCodePrimary verified, PostalCodeSecondary added or changed; P6: PostalCodePrimary verified; P5: PostalCodePrimary verified with small change; P4: PostalCodePrimary verified with large change; P3: PostalCodePrimary added; P2: PostalCodePrimary identified by lexicon; P1: PostalCodePrimary identified by context; P0: PostalCodePrimary empty.
Separator	1	-
Matchscore	3	The accuracy matchscore gives the similarity between the input data and closest reference data match as a percentage between 0 and 100. 100% means complete similarity.

The AddrScore field contains the value of Matchscore from the ErrorCode.

The UpdateFlag field indicates whether the record was updated:

UpdateFlag	Description
Address	Record address and postcode were updated.
None	Record not updated.

7.1.2 msp_GenerateNCOAAddresses

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.

GenerateNCOAAddresses can be used to keep your database up-to-date as your customers move or their addresses are corrected. Note that this is available to licensed users only, and can only process US data.

GenerateNCOAAddresses is available as both a stored procedure and an SSIS task. The process involves sending data from the input table to an online service, and writing the received processed data into an output table for subsequent processing and use. (Please see the Security Protocol for further information.)

Stored Procedure

When running as a stored procedure, the data source is used to configure the input table and field mappings that are passed to the NCOA service. If this will follow GenerateCorrectedAddresses, be sure to use the corrected addresses table and its columns as inputs to the NCOA service.

To use the output NCOA tables and columns in following stored procedures (i.e. GenerateKeys), no further configuration is necessary because the stored procedure will be able to determine existence of the NCOA table and automatically make use of it.

SSIS Task

When running as an SSIS task, the user has a choice of how the input table and mappings are obtained: from a preceding GenerateKeys task (in which case no extra configuration is necessary), from a preceding GenerateCorrectedAddresses task (in which case only the required name fields need mapping, while the corrected addresses are automatically used), or via manual configuration of the connection string, tables, and columns.

To use the output NCOA tables and columns in following tasks (i.e. GenerateKeys), it is necessary to map the table and its columns in the task.

Setting	Description
dataSources	Specifies the database connection, table and column mappings used to define the dataset being processed.
ncoa->customerInfo->listProcessor->name	List processor full name.
ncoa->customerInfo->listProcessor->street	List processor street address.
ncoa->customerInfo->listProcessor->lastLine	List processor city, state, ZIP.
ncoa->customerInfo->listProcessor->phone	List processor telephone number.
ncoa->customerInfo->mailer->name	Mailer full name.
ncoa->customerInfo->mailer->street	Mailer street address.
ncoa->customerInfo->mailer->lastLine	Mailer city, state, ZIP.

ncoa->customerInfo->mailer->phone	Mailer telephone number.
ncoa->output->table->name	Can be used to specify the name of the table that will be created. If this is left blank, then the table's name will be automatically generated.
ncoa->output->optionalFields	Any of these fields can be written to the output table. (See Optional Output Fields below.)
ncoa->options->useMixed	If this is set to "true", data is returned in mixed-case format. If this is set to "false", result data is returned in all upper case characters.
ncoa->options->useAlias	If this is set to "true", alias street names are returned when they are used in the input. If this is set to "false", 'official' street names are returned even when alias street names are used in the input. (An alias street name is an alternative name for a street that is acceptable to the USPS. It may be a name by which a street was formerly known, a commonly-used nickname for the street, or one the community prefers to use.)
ncoa->options->validAddressesOnly	If this is set to "true", and GenerateCorrectedAddresses has been run beforehand, then only valid addresses (with a score of 0) are processed by the NCOA service.
ncoa->options->blockSize	Data is uploaded to the NCOA service in 'blocks'. This setting specifies the maximum number of records per block.
ncoa->options->timeout	The maximum time to allow for every 100,000 records processed, in minutes (the default is 15). This should be increased if you have a slow internet connection or are experiencing timeouts. Timeouts can be disabled by specifying 0.
ncoa->options->retries	The maximum number of retry attempts when processing has failed (the default is 1). Specify 0 to disable retry attempts.

7.1.2.1 Initial Setup

Before GenerateNCOAAddresses can be used for the first time, it must be configured on the computer on which matchIT SQL has been installed.

Locate NCOASetup.exe within the matchIT SQL bin folder (for example, the default location is C:\Program Files\matchIT SQL\bin).

Right-click the file and select "Run as administrator". Enter the details of a user with administrative privileges if required.

When the "NCOA Setup" window appears, click PAF Setup then Add New to create a new PAF account. Fill in the required details as necessary and follow the instructions. Select the new PAF in the list, click OK, then close the window. Please email or fax a signed copy of the PAF to helpIT systems. Please allow up to one week for your PAF to be approved by the USPS and your NCOA account to be set up.

Once your NCOA account has been created, re-run NCOASetup. Click Account to input details of your Account, which will have been provided by helpIT systems. Ensure these details are correct by using Test FTP. Click OK. Ensure all details displayed on the NCOA Setup window are correct, including the Account Statistics, and then close the window.

GenerateNCOAAddresses is now ready for use.

7.1.2.2 Input Fields

The following fields must be input to the NCOA process, and will be written to the output table:

Name	Description
FullName <i>or</i>	Fullname field.
FirstNames <i>and</i> LastName	If FullName is not used, then both these fields are required.
Organization	Optional. Can be used to specify the company name.
Address1	Sets the "Street" field of the lookup record. Use this field type to pass all information relating to the primary street address, including the street name, house number, directional's, and street suffix. You can also use this field type to pass all other acceptable forms of primary address information, such as PO Box numbers, rural route numbers, and highway contract numbers. In addition, suite and apartment information can be passed with primary address information through the "Street" field. For example, "123 Main St" and "123 Main St, Apt A" are both acceptable.
Address2	Optional. Can be used to specify any suite and apartment information, if these are in a separate column from the street part of the lookup address (see Address1 above).
Town	Specifies the city.
Region	Specifies the state.
Postcode <i>or</i>	Specifies the 9-digit ZIP code.
PostOut <i>and</i> PostIn	Specifies the 5-digit ZIP code and the ZIP+4, if these are in separate columns.

7.1.2.3 Output Fields

The following fields are written to the output table:

Name	Description
ncoaFullName <i>or</i>	Only if the fullname was specified on input.
ncoaFirstName <i>and</i> ncoaLastName	Only if the fullname was not specified on input.
ncoaCompany	Optional. Retrieves the company name. This is only output if Organization was passed in via the input fields, or if this optional field is enabled (see Optional Output Fields below).
ncoaStreet	Retrieves the street part of the output address.
ncoaSuite	Optional. Retrieves the suite or apartment information of the output address. This is only output if Address2 was passed in via the input fields, or if this optional field is enabled (see Optional Output Fields below).
ncoaCity	Retrieves the city from the output address.
ncoaState	Retrieves the output state abbreviation.
ncoaZip	Retrieves the output ZIP Code.

7.1.2.4 Optional Output Fields

Any of the following fields can be written to the output table:

Name	Description
------	-------------

ncoaCOACode	This field will be populated with the return codes, see Return Code section for the description of codes returned.
ncoaCofound	This field is a True/False flag that tells whether an address change was found.
ncoaMoveEffectiveDate	Date of move. mm/yyyy
ncoaCompany	Retrieves the company name that was entered through the input "ncoaCompany" field, if any. AccuMail Move™ will also verify company moves based off this the company field.
ncoaSuite	Retrieves output secondary street information, such as suite and apartment information, if such information was entered through the input "ncoaSuite" field.
ncoaPlus4	Retrieves the output ZIP+4 Code. The ZIP+4 Code is the 4-digit extension only. You can retrieve the 5-digit ZIP Code using the ncoaZip field type described earlier in this table.
ncoaUrbanization	Retrieves output urbanization information for Puerto Rican addresses.
ncoaDP	This string consists of a 2-byte delivery point code
ncoaCRRT	Retrieves the output carrier route code. This is a 4-digit code assigned to each address on a mail carrier's route.
ncoaLotCode	Retrieves the output Line of Travel identifier consisting of a 4-digit number, plus a 1-character sequence code (either "A" for Ascending or "D" for Descending). The 4-digit number indicates the order in which delivery will be made within a given ZIP+4. The 1-character sequence code indicates whether delivery will be made in ascending or descending order. Once a LOT code is appended to your data file records, you can use it to presort your mailings so that they qualify for Enhanced Carrier Route rates.
ncoaLotDir	The order in which the mail carrier delivers mail within a given carrier route. When you include the Line of Travel information, your mail may be eligible for the USPS Standard Mail Non-Automation Basic Enhanced Carrier Route Presort Rate.
ncoaCountyNumber	Retrieves the output county number. This is the 3-digit USPS code for the county in which the address resides.
ncoaCountyName	Retrieves the output county name.
ncoaCongDistrict	Retrieves the output congressional district code. This is a 2-digit identifier for the United States congressional district to which the input address belongs.
ncoaLACS	A 1-character Locatable Address Conversion Service (LACS) code to identify records that have been converted to the LACS system. The LACS system is being used for many rural route addresses and city addresses that are being modified to city style addresses so that emergency vehicles, such as police cars and ambulances, can more easily find these locations.
ncoaAcsKeyline	Retrieves the output Address Change Service (ACS) keyline. The ACS keyline is a code the USPS uses to uniquely identify any address in the United States. You can use the ACS keyline to match the records in your mailing list with the list of ACS notifications in the USPS's ACS fulfillment files.
ncoaHouseNumber	Retrieves the house number for the output street address. For example, if the output street address is "123 Main St," the house number is "123."
ncoaPreDirectional	Retrieves the pre-directional for the output street address. For example, if the output street address is "123 E Main St," the predirectional designator is "E."
ncoaStreetName	Retrieves the street name for the output street address. For example, if the output street address is "123 Main St," the street name is "Main."
ncoaStreetSuffix	Retrieves the street suffix for the output street address. For example, if the output street address is "123 Main St," the street suffix is "St."
ncoaPostDir	Retrieves the post-directional for the output street address. For example, if the output street address is "123 Main St N," the postdirectional is "N."
ncoaSUD	Retrieves the secondary unit designator (SUD) for the output street address. For example, if the output street address is "123 Main St Apt 12," the SUD is "Apt."
ncoaUnitNum	Retrieves the unit number for the output street address. For example, if the output street address is "123 Main St Apt 12," the unit number is "12."
ncoaLeftovers	Retrieves any leftover information that was part of the input street address string, but was not used for obtaining a match. Leftover information is input data that AccuMail had to discard to correct the address. For example, if the input street address is "123 Main St Rubbish Here," the output leftover information is "Rubbish Here."
ncoaPMB	Retrieves the output Public Mailbox address

ncoaDPV	<p>Retrieves the 3-byte DVP (Delivery Point Verification) match codes. The match codes indicate whether or not the address is valid (and if not, why not), whether or not the address is within a Commercial Mail Receiving Agency (CMRA), and whether or not the address was flagged as a False Positive. The "DPV" field returns a separate code in each position of the 3-byte result string, as follows:</p> <p>BYTE 1</p> <p>blank – The address was not coded by AccuMail and therefore no DPV processing was performed.</p> <p>Y – All delivery point components of the address were DPV validated.</p> <p>D – The address's building number was DPV validated, but required unit-level information is missing.</p> <p>S – The address's building number was DPV validated, but the unit number is invalid.</p> <p>N – The address's building number is invalid.</p> <p>BYTE 2</p> <p>blank – The address was not coded by AccuMail and therefore no DPV processing was performed.</p> <p>Y – The address was found in the CMRA (Commercial Mail Receiving Agency) table.</p> <p>N – The address was not found in the CMRA (Commercial Mail Receiving Agency) table.</p> <p>BYTE 3</p> <p>blank – The address was not coded by AccuMail and therefore no DPV processing was performed.</p> <p>Y – The address was found in the False Positive table.</p> <p>N – The address was not found in the False Positive table.</p>
ncoaDPVAnswer	<p>blank – The address was not coded by AccuMail and therefore no DPV processing was performed.</p> <p>Y – All delivery point components of the address were DPV validated.</p> <p>D – The address's building number was DPV validated, but required unit-level information is missing.</p> <p>S – The address's building number was DPV validated, but the unit number is invalid.</p> <p>N – The address's building number is invalid.</p>
ncoaDpvCMRA	Commercial Mail Receiving Agency
ncoaDpvFalsePositive	This is a seed table used by the Delivery Point Verification Service (DPV). It is used by the service to guard against the possibility of mailers manufacturing artificial mailing lists from the data in the DPV database of every valid delivery point in the U.S.
ncoaDpvFootnotes	<p>Retrieves the 8-byte DPV (Delivery Point Verification) Footnotes string. This field returns up to four 2-character codes that supplement the DVP field codes (described above) providing additional information about the DPV match/miss-match. Up to four of the following 2-character codes will be returned:</p> <p>AA – The address was successfully coded by AccuMail.</p> <p>A1 – The address was not successfully coded by AccuMail.</p> <p>BB – All components of the address were DPV validated.</p> <p>CC – The address's building number was DPV validated, but the unit number is invalid.</p> <p>N1 – The address's building number was DPV validated, but required unit-level information is missing.</p> <p>M1 – A building number is missing for the input address.</p> <p>M3 – The address's building number is invalid.</p> <p>P1 – The input address is missing a required PO, RR, or HC Box number.</p> <p>RR – The input address was identified by DPV as a Commercial Mail Receiving Agency (CMRA).</p> <p>R1 – The input address was identified by DPV as a Commercial Mail Receiving Agency (CMRA), but required unit-level information is missing.</p>
ncoaLastLine	Retrieves the output 'last line' string. This output 'last line' string is a formatted city/state/ZIP string that includes the correct ZIP+4 Code. For example, the 'last line'

	information for Datatech SmartSoft is "Agoura Hills, CA 91301-4301."
ncoaMoveType	Indicates the move as Family, Business, or Residential.
ncoaResult	Retrieves the output error code. AccuMail assigns an error code if the input record could not be found in the USPS National Database. This is a 2-byte code that identifies what was wrong with the input address and why AccuMail could not match it. If AccuMail matched the input address successfully, then this field returns a blank string.
ncoaCorrections	Retrieves the output correction codes string. This string consists of single character codes that indicate the corrections AccuMail had to make to the input record.
ncoaErrorMessage	Retrieves the message text associated with the error code that AccuMail assigned if the input address could not be matched. This is a descriptive sentence or paragraph that describes the reason AccuMail could not correct the input record. If AccuMail matched the input address successfully, then this field returns a blank string.
ncoaWasDigitCoded	Returns a 1 for addresses that successfully coded with a plus4. Returns a 0 for uncoded addresses.
ncoaWasCRRTCoded	Returns a 1 for addresses that has returned with a Carrier Route or a 0 if no Carrier Route was returned.
ncoaWasPlusCoded	Returns a 1 for addresses that successfully coded with a plus4. Returns a 0 for uncoded addresses.
ncoaWasDPBCoded	Retrieves the output delivery point barcode string. This string consists of a 2-byte delivery point code plus a 1-byte checksum digit. These constitute the values required for producing a Delivery Point Barcode.
ncoaWasDPVCoded	Returns a 1 for addresses that were successfully DPV coded or a 0 for addresses that did not pass DPV validation.

7.1.2.5 Reports

Three reports are created after NCOA processing:

- **CASS Certificate (PDF);**
- **NCOA Details Report** - this is a text file that will list all matches found;
- **NCOA Processing Summary Report (PDF).**

When processing has completed, the reports are moved to the reports folder (as specified by the path attribute of the outputSettings/reports node in the configuration file).

7.1.2.6 Return Codes for NCOA

There are two fields AccuMail returns for NCOA. These fields are COACode and COAFound. COAFound is a True/False flag that tells whether an address change was found. Below are the codes that could be returned in the COACode field.

Code: Return code.

Description: Explanation of return code.

Address: "Y"=New address provided; "N"=New address not provided.

How: "D"=Derived by data, returned in lieu of 11 digit; "S"=Derived by software.

Code	Description	Address	How
A	COA Match - The input record matched to a COA record. A new address	Y	D

	could be furnished. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.		
77	ANK - ANK will not return the actual new address. ANK will, however, provide users a return code indicating a probable move occurred in months 19-48, along with the move-effective date. You should suppress these records from your database or flag these records for deletion and not mail to them.	N	D
66	Daily Delete – The input record matched to a business, individual or family type COA record with an old address that is present in the daily delete file. The presence of an address in the daily delete file means that a COA with this address is pending deletion from the COA master file and that no mail may be forwarded from this address. This return code may be returned regardless of the processing mode, matching logic or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	S
00	No Match - The input record COULD NOT BE matched to a COA record. A new address could not be furnished. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: When processing in any mode and this return code is received it is required to attempt the match again using the next level of matching logic allowed by the processing mode.	N	D
01	Found COA: Foreign Move – The input record matched to a COA record but the new address was outside the USPS delivery area. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
02	Found COA: Moved Left No Address (MLNA) – The input record matched to a COA record, but the new address was not provided to USPS. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved then no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
03	Found COA: Box Closed No Order (BCNO) – The Input record matched to a COA record containing an old address of PO BOX, which has been closed without a forwarding address provided. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
04	Cannot match COA: Street Address with Secondary – In the STANDARD mode utilizing Family matching logic the input record was a potential match to a family type COA record with an old address that contained secondary information. The input record does not contain secondary information. The record is a ZIP + 4 street level match. This address match situation requires individual name matching logic to obtain a match and individual names do not match. Please Note: This return code is only obtained when processing in the STANDARD mode using Family matching logic.	N	D
05	Found COA: New 11-digit DPBC is Ambiguous – The input record matched to a COA record. The new address on the COA record could not be converted to a deliverable address because the DPBC represents more than one delivery point. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
06	Cannot Match COA: Conflicting Directions: Middle Name Related – There is more than one COA record for the match algorithm and the middle names or initials on the COAs are different. Therefore, a single match result could not be determined. This return code is only obtained when using individual matching logic. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
07	Cannot Match COA: Conflicting Directions: Gender Related –There is more than one COA record for the match algorithm and the genders of the names on the COAs are different. Therefore, a single match result could not be determined. This return code is only obtained when using individual matching logic. Please Note: If this return code is achieved, no other	N	D

	matching attempts are permitted regardless of the PROCESSING mode.		
08	Cannot Match COA: Other Conflicting Instructions – The input record was a potential match to two COA records. The two records were compared and due to differences in the new addresses, a match could not be made. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
09	Cannot Match COA: High-rise Default – The input record was a potential match to a family COA record from a High-rise address ZIP + 4 coded to the building default. This address match situation requires individual name matching logic to obtain a match and individual names do not match. Please Note: This return code is only obtained when processing in the STANDARD mode using Family matching logic.	N	D
10	Cannot Match COA: Rural Default – The input record was a potential match to a family COA record from a Rural Route or Highway Contract Route address ZIP + 4 coded to the route default. This address situation requires individual name matching logic to obtain a match and individual names do not match. Please Note: This return code is only obtained when processing in the STANDARD mode using Family matching logic.	N	D
11	Cannot Match COA: Individual Match: Insufficient COA Name for Match – There is a COA record with the same surname and address but there is insufficient first/middle name information on the COA record to produce a match using individual matching logic. This return code is only obtained when using individual matching logic. Please Note: When processing in the STANDARD mode and this return code is received utilizing Individual Logic, discontinue the Individual logic sequence and go straight to the FAMILY matching logic.	N	D
12	Cannot Match COA: Middle Name Test Failed – The input record was a potential match to a COA record. A match cannot be made because the input name contains a conflict with the middle name or initials on the COA record. This return code is only obtained when using individual matching logic. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	S
13	Cannot Match COA: Gender Test Failed – The input record was a potential match to a COA record. A match cannot be made because the gender of the name on the input record conflicts with the gender of the name on the COA record. This return code is only obtained when using individual matching logic. Please Note: When processing in the STANDARD mode and this return code is received utilizing Individual logic, discontinue the Individual logic sequence and go straight to FAMILY matching logic.	N	S
14	Found COA: New Address Would Not Convert at Run Time – The input record matched to a COA record. The new address could not be converted to a deliverable address. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	S
15	Cannot Match COA: Individual Name Insufficient – The input record was a potential match to a COA record that contains a first initial and middle initial/name [ex. C M Smith or C Mary Smith]. A match cannot be made because the input middle initial/name is missing or does not equal the middle initial/name on the COA. This return code is only obtained when using individual matching logic. Please Note: When processing in the STANDARD mode and this return code is received utilizing Individual logic, discontinue the Individual logic sequence and go straight to FAMILY matching logic.	N	S
16	Cannot Match COA: Secondary Number Discrepancy – The input record was a potential match to a street level COA record. However, a match is prohibited based on one of the following reasons: 1) There is conflicting secondary information on the input and COA record; 2) the input record contained secondary information and matched to a family record that does not contain secondary information. In item 2, this address match situation requires individual name matching logic to obtain a COA match and individual names do not match. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	S

17	Cannot Match COA: Other Insufficient Name – The input record was a potential match to a COA record that contains a full first name and full middle name. The input middle initial/name is missing or different from the middle name on the COA. A match cannot be made because the first name on the COA was truncated (drop-n flag) and the middle names must be equal in order to make this match. This return code is only obtained when using individual matching logic. Please Note: When processing in the STANDARD mode and this return code is received utilizing Individual Logic, discontinue the Individual logic sequence and go straight to FAMILY matching logic.	N	S
18	Cannot Match COA: General Delivery – The input record was a potential match to a COA record from a General Delivery address. This address situation requires individual name matching logic to obtain a match and individual names do not match. Please Note: This return code is only obtained when processing in the STANDARD mode using Family matching logic.	N	D
19	Found COA: New Address not ZIP+4 coded or New address primary number not DPV confirmable – There is a change of address on file but the new address cannot be ZIP+4 coded and therefore there is no 11-digit DPBC to store or return, or the new address primary number cannot be confirmed on DPV. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
20	Cannot Match COA: Conflicting Directions after re-chaining – Multiple COA records were potential matches to the input record. The COA records contained different new addresses and a single match result could not be determined. This return code may be returned regardless of the processing mode, matching logic, or COA type. Please Note: If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	N	D
91	COA Match: Secondary Number dropped from COA – The input record matched to a COA record. The COA record had a secondary number and the input address did not. Please Note: This return code is derived from Individual and business matching logic only. If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	Y	S
92	COA Match: Secondary Number Dropped from input address – The input record matched to a COA record. The input address had a secondary number and the COA record did not. The record is a ZIP + 4 street level match. Please Note: This return code is derived from individual and business matching logic only. If this return code is achieved, no other matching attempts are permitted regardless of the PROCESSING mode.	Y	S

7.1.2.7 Return Codes for NCOA Processing Summary Report

Pre-processes Performed:

- N = None
- Y = Yes but with no data modifications
- D = Yes, data modifications from sources other than postal data
- P = Yes, data modifications from postal data only (i.e.: ZIP+4, DPV)
- B = Yes, data modifications from postal and other sources

Concurrent Processes Performed

- N = None
- Y = Yes but with no data modifications
- D = Yes, data modifications from sources other than postal data

- P = Yes, data modifications from postal data only (ie:ZIP+4, DPV)
- B = Yes, data modifications from postal and other sources

Post-processes Performed

- N = None
- Y = Yes but with no data modifications
- D = Yes, data modifications from sources other than postal data
- P = Yes, data modifications from postal data only (LACSLink™)
- B = Yes, data modifications from postal and other sources

Standard Output Returned

- Y = All NCOALink required output returned to client
- N = Post-processes modified return information (i.e.: updates applied to list)
- B = Post-processes modified return information; however, separate file containing all required output data was also returned

Matching Logic Applied

- S = Standard (Business, Individual and Family matches allowed)
- I = Individual only
- B = Business only
- C = Individual and Business only
- R = Individual and Family only

Data Returned

- C = COA Data Returned (including footnotes and processing statistics)
- F = Footnotes (no COA data included; may include processing statistics)
- S = Statistics only (no COA data or footnotes provided)

Class of Mail

Alphanumeric. Class of mail to be used for mailings produced from customer mailing list.

- A = First-Class only
- B = Periodicals only
- C = Standard Mail only
- D = Package Services only
- E = First-Class & Periodicals
- F = First-Class & Standard Mail
- G = First-Class & Package Services
- H = Periodicals & Standard Mail
- I = Periodicals & Package Services

- J = Standard Mail & Package Services
- K = First-Class, Periodicals & Standard Mail
- L = First-Class, Periodicals & Package Services
- M = First-Class, Standard Mail & Package Services
- N = Periodicals, Standard Mail & Package Services
- O = First-Class, Periodical, Standard Mail, Package Services

7.1.2.8 Return Codes for CASS Certification

These are codes given if your address could not be certified. These codes will be written to the optional field "ncoaResult":

- Blank – The address has been successfully coded.
- 1 – The address is coded but undeliverable (i.e. on side of street known to contain no houses).
- 2 – The ZIP code was not found and the city and state cannot be used to determine a geographical area to search.
- 3 – Coding would result in changing both ZIP and city. This is illegal for PO BOX and route type addresses.
- 4 – The best match would result in too many suspicious changes.
- 5 – The street was identified as an alias but was out of the range restricted for that alias.
- 6 – No street address was given.
- 7 – There are no street name matches in the given ZIP code or in any geographically-related ZIP code.
- 8 – The street may contain superfluous components which cannot be discarded with confidence.
- 9 – The house number could not be matched.
- 10 – The best match was made to a ZIPMOVE record but was not an exact match.
- 11 – A ZIPMOVE match was made but no exact match could be found in the new ZIP.
- 12 – The Early Warning System indicates that an exact match will become available in the next database update.
- 13 – There are multiple matches with the same degree of confidence. This may indicate an inconsistency in the USPS data.
- 14 – Incorrect suffix, directional, street name, or unit resulted in multiple matches with the same degree of confidence.
- 15 – Incorrect ZIP, city, or urbanization resulted in multiple matches with the same degree of confidence.
- 16 – A corrected field was too long to fit into the supplied field.
- 17 – Media Error. The database could not be read because of a hardware or system problem.

7.1.2.9 Correction Codes

The correction codes are used to describe what was done to the address in the coding process. Each coded record will be assigned a string of one or more characters. The user may assign any size field to hold the correction codes. If the field is too short, then the codes will be truncated without error. These codes will be written to the optional field "ncoaCorrections":

- A – Normal street match.
- B – PO BOX match.
- C – Route type match.
- D – 'Unique ZIP' match.
- E – 'Small town' match.
- F – Alias match.
- G – 'Highrise alternate' match.
- H – Firm match.
- I – Highrise match.
- J – Highrise default match.
- K – Route default match.
- L – Street name corrected.
- M – Street suffix corrected.
- N – Predirectional corrected.
- O – Postdirectional corrected.
- P – City corrected.
- Q – State corrected.
- R – ZIP corrected.
- S – Urbanization corrected.
- T – ZIP+4 corrected.
- U – House number corrected.
- V – Unit number corrected.
- W – Secondary unit designator corrected.
- X – Firm corrected.
- Y – Street swapped with firm.
- Z – Street swapped with alternate.
- 0 – Dual address changed to PO BOX.
- 1 – Dual address street match.
- 2 – Input city is not preferred but is acceptable.
- 3 – Street standardized.
- 4 – Unit not verified.
- 5 – Leftovers found.
- 6 – ZIPMOVE match.
- 7 – LACSLINK match.

7.1.2.10 Security Protocol

Executive Summary:

Datatech SmartSoft provides clients with up-to-the-minute address changes for all United States Addresses via the United States Postal Service (USPS). The company and USPS maintain a strong commitment to privacy while providing addressing details to individuals who live within the continental United States.

Documentation:

The purpose of this document is to provide clear and compelling reason(s) for Datatech SmartSoft clients to allow their databases to be updated with current USPS NCOA Link move update address information. In keeping with the USPS security regulations, clients are not able to update these addresses without first:

1. **Identification:** This is accomplished through Datatech SmartSoft, Inc. NCOA Link product license keys which are uniquely generated for our clients, then verified during each Move Update job request.

2. **Communications:** Datatech SmartSoft transforms customer data into a binary format prior to data transfer. This transformation ensures that customer data is not human readable.

3. **Intrusion Detection:** Datatech SmartSoft takes the appropriate steps to ensure data is secure, from both internal and external sources.

4. **Auditing:** All accesses to the AccuMail Move™ server are stored in a relational database including details related to time, client and activity.

Physical Network Security

Physical Security and Availability of Server(s):

Our managed servers run on HP c-class blade server systems. This enterprise solution allows full remote control of power, virtual media and virtual connection to redundant networks and storage area networks. Using the c-7000 enclosures with hot-swap fans and power supplies, the typical failures resulting in downtime on servers are virtually eliminated. Blades can be instantly swapped out in the unlikely event of a failure. These high security facilities guarantee 99.9% up time. The servers are located in the Nevada NAPs, served by a one billion dollar fiber hub with access to over 100 Tier 1 backbone with separate and redundant fiber optic paths. Our servers are located with disaster recovery as a priority and in a region that is free from ice storms, tornadoes, earthquakes, power outages and other natural disasters.

Electronic Network Security:

Electronic network security and verification is controlled by a variety of methods including binary transformation of customer data, server-side authorizations of user specific accounts and passwords, custom product licensing, database authorizations and client access auditing.

User specific Accounts and Passwords:

SmartSoft clients pass through multiple levels of security when processing each job request. The first layer is integrated operating system security where each user / password combination must have access to transfer customer binary data to the server. The second custom security layer involves validation of each users account, password and licenses prior to processing any list. These security checks ensure that the user is valid, has the correct permissions and has adequate credits to process the job. This check passes through an additional layer of security that accesses our data storage devices where all account details are stored.

Client Access Auditing:

SmartSoft tracks certain metadata related to each job processed. Tracking usage allows Datatech SmartSoft, Inc to audit individual usage across accounts, licenses and products. This does not include tracking or durational storage of specific customer data beyond what is required for job processing.

SmartSoft tracks processing information related to each job in accordance with the standard Software Performance Requirements associated with all licensees of the NCOALink® data from the United States Post Office (USPS®). This information is generally statistical in nature and summarizes monthly activity for each end user associated with each Processing Acknowledgement Form (PAF). This does not include tracking or durational storage of specific customer data beyond that required for job.

7.2 Key Generation

7.2.1 msp_CreateKeysTable

WARNING: This stored procedure has been deprecated and will be removed in a future release of matchIT SQL. The stored procedure should no longer be used – except for configurations that have a compatibility setting less than 2.0.0 – and existing processes modified accordingly.

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.

The Procedure simply creates a new table in the database (using the connection string specified in the datasource) that contains all necessary matching and key fields. The following XML settings (which can be set using the appropriate page in the Web based UI) are used during the execution of this procedure:

Setting	Description
datasources	Specifies the database connection, table and column mappings used to define the dataset being processed. The keys table that will be created when this procedure runs will have the same name as the first mapped table with a `__keys__` suffix.
generalSettings->deleteKeysTableOnGenerate	When this setting is switched on, the keys table (if it already exists) will be deleted and recreated by this procedure, prior to being re-populated with keys by the BulkGenerateKeys stored procedure.
outputSettings->keyColumns	Specifies the columns that the keys table is created with (and populated by BulkGenerateKeys). Note that all key fields used in the match keys (such as the fuzzy and exact match keys if running the relevant stored procedures) must be included in the keys table.

7.2.2 msp_AddKeyFieldsToTable

WARNING: This stored procedure has been deprecated and will be removed in a future release of matchIT SQL. The stored procedure should no longer be used, and existing processes modified accordingly.

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.
- Table name – specifies the named table that the key columns will be added to.

Simply appends all necessary matching and key fields to the specified table in the database (using the connection string specified in the datasource).

Note that to use this procedure, your XML must specify only one table in the datasources section, and this must also be marked as the keys table.

e.g.

```
<tables>  
  <table name="contacts" isKeysTable="true" />  
</tables>
```

7.2.3 msp_GenerateKeys

WARNING: This stored procedure has been deprecated and will be removed in a future release of matchIT SQL. The stored procedure should no longer be used, and existing processes modified accordingly.

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications

This procedure generates the match key field values for all records in the current database (using the connection string specified in the datasource).

We highly recommended using msp_BulkGenerateKeys, with a separate keys table, as it provides the best key generation performance.

The following settings are specific to the match key generation. These settings can be configured through the web UI or directly in the XML configuration file used during execution of the procedure:

Setting	Description
dataSources	<p>Specifies the database connection, table and column mappings used to define the dataset being processed.</p> <p>Only one table can be marked as the keys table using the isKeysTable option. The keys table that will be populated when this procedure runs will have the name specified in the name attribute of the table marked as the keys table. The recommendation is to <i>not</i> actually specify a keys table, so that the name of the keys table will be automatically generated (see the suffixes in the outputSettings section).</p> <p>If you are using msp_BulkGenerateKeys (recommended) then one table may be added in the data source section to generate a cleaned, standardised version of your input data. To configure this table, add a table with the isOutputTable property set to true. This table can be used to proper case your data, parse names and simple address validation – please see the matchIT API setting options described below.</p>
generalSettings->ensureUniqueRefIsClustered	<p>If enabled, then the uniqueRef column specified in the first table mapped in a data source is checked to ensure that it is referenced by a unique clustered index. (Note that this can be, but is not required to be, a primary key column.)</p>
outputSettings->reports	<p>Specifies whether reporting is enabled, what folder the reports will be produced into, and what report format should be used.</p>
matchITAPISettings->generate->dropExcludedWords	<p>With this property set to True, during the generate step matchIT SQL will flag any records that contain exclusion words in any of the key fields (fields such as addressee, company or the address lines). Such exclusion words include "Deceased", "Addressee" (indicating a record may be a header record) and any other Exclusion type entries in the NAMES.DAT file. Records are flagged by setting the first character of the <i>mkDataFlags</i> field in the generated keys table to "X".</p> <p>Note that the default location of the names.dat file is: C:\Program Files\Common Files\matchIT API\dat</p>
matchITAPISettings->generate->properCase	<p>If the <i>ProperCase</i> property is set to True, the fields within the configured output table created during key generation</p>

	(with the exception of premise (i.e. building) numbers and postcodes) will be correctly cased.
matchITAPISettings->generate->considerCasing	<p>If this property is set to True, then matchIT SQL will consider the casing of the incoming data when it is splitting the data up for extracting keys, proper casing, and so forth.</p> <p>For Example, with consider casing switched on the company:</p> <p>ABCD Systems Ltd</p> <p>ABCD would be considered to be an acronym rather than simply a word (i.e. on output without this option, ABCD would be output as Abcd).</p>
matchITAPISettings->generate->specialCaseMac	<p>Where a last name begins with Mac, when formatting salutations, matchIT SQL follows this with a small letter or a capital letter, depending on this property. A value of True will mean that MACLEAN will be formatted as MacLean. You can add exceptions to the rule (e.g. Maccabee, Macclesfield, MacKay, Mackie) to the NAMES.DAT file. If you invariably want to use a lower case letter following Mac, set this property to False.</p> <p>NB: Names beginning Mach are always formatted with a lower case H, e.g. Machin, Machinery. Names beginning Mc are formatted with a capital letter following, if they are greater than 3 characters long.</p>
matchITAPISettings->generate->variableKeysMaxLength	<p>This specifies the maximum length of various variable-length phonetic keys created. Such keys are PhoneticLastName, PhoneticFirstName, PhoneticMiddleName, PhoneticOrganizationName1, PhoneticOrganizationName2, PhoneticOrganizationName3, PhoneticStreet, and PhoneticTown.</p> <p>The default is eight characters.</p>
matchITAPISettings->generate->quality->enabled	<p>By default, quality scoring is disabled and all quality scores are 0.</p> <p>Enabling this feature allows matchIT SQL to generate quality scores for data fields such as:</p> <p>Names</p> <p>Emails</p> <p>Addresses</p> <p>Company Names</p> <p>These results are written to the output table configured in the Datasource being used during the GenerateKeys process.</p>
matchITAPISettings->generate->quality->address->allowBlankPostcode	If disabled (enabled by default) then addresses without a postal code are restricted to a maximum quality score of 1.
matchITAPISettings->generate->quality->email->webmailFiltering	If enabled (default) then email addresses that use webmail provider (such as Hotmail, Yahoo, & mail.com) domains are restricted to a maximum quality score of 7.
LOW LEVEL ADVANCED SETTINGS	The following settings are low level and most users will not normally need to modify the default settings.
matchITAPISettings->generate->Name->joinMarriedPrefixes	With this property set to True, multiple addressees with the same last name will be treated as married e.g. input names of "Mr John Smith and Ms. Mary Smith" or "Mr John Smith & Mary Smith" would have a Salutation generated of "Mr and Mrs Smith" and a Contact generated of "Mr and Mrs J Smith" or "Mr and Mrs John Smith".
matchITAPISettings->generate->Name->generateContact	With this property set to True, matchIT SQL will generate a contact for the input name. The contact will be structured in same way as you would expect to find its corresponding input name on e.g. the front of an envelope. For example, the input name of "John Smith" or "Mr John Smith" would result in a generated contact of "Mr J Smith".

	An accurate contact value cannot be generated when matchIT SQL is unable to determine the gender of an input name. In this situation, the generated contact would be equal to the input name. e.g. "J Smith" as an input name would result in a generated contact of "J Smith".
matchITAPISettings->generate->Name->contactFullname	Set this property to True to include the full first name of any incoming name in the CONTACT field; just the initial will be used if the property is False. For example, if the property is True, and the incoming name is "John Smith", then the generated contact will be "Mr John Smith", if it is False, then the contact will be "Mr J Smith".
matchITAPISettings->generate->Name->defaultSalutation	This property determines the default salutation, either where matchIT SQL can't determine one (for example, C Smith or Chris Smith, which could be either Mr or Ms), or where the Prefix supplied doesn't have a salutation rule. If you include the word "Dear" as at the start of the default salutation (i.e. actually specify "Dear Customer" and not just "Customer", then all the salutations derived by matchIT SQL will start with the word "Dear" unless the salutation for the type of title (or prefix) specifies "Title" only. For example, Mr J Smith will result in a salutation of "Dear Mr Smith" whereas The Bishop of Liverpool will result in a salutation of "My Lord".
matchITAPISettings->generate->Name->defaultGender	The Default Gender property is the gender to assume when matchIT SQL can't determine whether the name is male or female e.g. Chris Smith, C Smith. If you set this property to Male or Female, matchIT SQL will assume it to be male or female accordingly, and develop a salutation using Mr or Ms as the prefix.
matchITAPISettings->generate->Name->UseEquivalentName	If you set the Use Equivalent Name property to True, matchIT SQL replaces the first name with its equivalent from the NAMES.DAT file, if there is an entry for the input first name. This enables, for example, "Tony Smith" and "Anthony Smith" to be picked up as a match. The initial of the original first name is stored in the <i>Record.DataFlags</i> property to enable, for example, "Tony Smith" and "T Smith" to still be matched.
matchITAPISettings->generate->Name->EnhancedDoubleBarrelledLookup	When enabled, this property will cause an unrecognised middle name to be considered part of a non-hyphenated double-barrelled last name (for example, where the full name is John Harrington Jones, the last name will be considered Harrington -Jones because Harrington is not a recognised first name).
matchITAPISettings->generate->Name->processBlankLastName	With this property enabled, a blank lastname will cause extra processing to be performed on other input data to help detect typographical errors. For example, if a firstname was entered but not a lastname, then it'll be assumed that the firstname is in fact the lastname and match keys will be generated rather than being left blank.
matchITAPISettings->generate->Name->replaceAndWithAmpersand	By default, matchIT SQL will convert 'and' to an ampersand when outputting InputFields.Name.Addressee. Disabling this property will prevent this behaviour.
matchITAPISettings->generate->Name->parseNameElements	When enabled, this will cause input name elements (including prefix, firstnames, and lastname) to be parsed. If matchIT SQL deems any values to have been entered into an incorrect field (for example, suffixes and qualifications in the lastname field), it will reassign these values into the correct fields. This property is disabled by default, so that any such incorrect values are not reassigned.
matchITAPISettings->generate->Name->detectInverseNames	With this property enabled, matchIT SQL will attempt to identify addressee names that have been specified with the lastname preceding the firstnames, provided a comma

	delimiter follows the lastname (for example, "Smith, John" where Smith is the lastname). Without a comma, a name is assumed to be in standard left-to-right format, with the firstnames preceding the lastname.
matchITAPISettings->generate->Name->parseAsNormalizedName	When enabled, addressee names are assumed to be in a delimited normalized format similar to the NormalizedName value that's output by during Key Generation. Currently supported delimiters are spaces, commas, semicolons, and pipes (' ').
matchITAPISettings->generate->Address->Extract->premise	This will move or copy premise numbers found in the address lines into a field labelled PREMISE in the table configured as an output table during key generation.
matchITAPISettings->generate->Address->Extract->thoroughfare	This will move or copy address data recognized as the thoroughfare of the address (based on Address type entries found in the NAMES.DAT file) into a field labelled THOROUGHFARE in the table configured as an output table during key generation.
matchITAPISettings->generate->Address->Extract->town	This will move or copy address data recognized as the town or city from the address lines to a field labelled TOWN in the table configured as an output table during key generation.
matchITAPISettings->generate->Address->Extract->postTownsOnly	If this is enabled, together with Extract->Town, then only post towns (i.e. any towns found in the TOWNS.DAT file) will be moved or copied.
matchITAPISettings->generate->Address->Extract->region	This will move or copy US, Canadian or Australian states or provinces, or valid UK counties (or other regions found in the NAMES.DAT file), that are found in the address lines into a field labelled REGION in the table configured as an output table during key generation.
matchITAPISettings->generate->Address->Extract->postcode	This will move or copy UK postcodes, or US zip codes found in the address lines into a field labelled POSTCODE. Only UK postcodes with an outward half that is valid according to the MAILSORT.DAT file will be extracted.
matchITAPISettings->generate->Address->Extract->country	This will move or copy valid countries found in the address lines (based on Country type entries found in the NAMES.DAT file) into a field labelled 'COUNTRY' in the table configured as an output table during key generation.
matchITAPISettings->generate->Address->abbreviateRegion	Set this property to True if you want matchIT SQL to abbreviate States or Provinces when processing address lines e.g. to change "Pennsylvania" to "PA" within the table configured as an output table during key generation.
matchITAPISettings->generate->Address->upperCaseTown	This applies to UK addresses only. Set this property to True to convert the post town in the address to capitals within the table configured as an output table during key generation. Note that, if the <i>ProperCase</i> property is set to False, then this property is ignored.
matchITAPISettings->generate->Address->verifyPostcode	If set to True, this property verifies and corrects the format of the postcode. Numerics are changed to alphas and vice versa where appropriate. This feature makes use of the rules concerning the alphanumeric structure of the postcode. E.g. it changes "KT22 BDN" to "KT22 8DN" – it will change 0, 1, 5 and 8 to O, I, S and B, or vice versa, if that makes the postcode alphanumerically correct. matchIT SQL will not verify or correct the format of postcodes that are not in the postcode field. The cleaned postcodes are output to the table configured as an output table within the datasource.
matchITAPISettings->generate->Address->defaultThoroughfareLine	This property is used when the matchIT SQL is generating a phonetic address key, for which it needs to know the thoroughfare (e.g. street) and the town in the address. If it cannot locate a thoroughfare in the address, usually because it cannot find a word to indicate one, such as "Street", then it will be assumed that the thoroughfare is the contents of the address line indicated by this property

	(if it is greater than zero). For example, if this property is set to 2, then matchIT SQL will take the contents of address line 2 as the thoroughfare if it cannot find a thoroughfare word in the address. This property should only be used if the addresses in your data are very rigidly structured.
matchITAPISettings->generate->Address->numOfLinesToScan	<p>This property enables personal names to be extracted from address lines. It can be set to 1 or 2. If set to 1, only the first address line will be scanned for names. If set to 2, both the first and second address lines will be scanned and have names extracted from them if found. Any personal names found can then be used for the generation of Contacts and Salutations.</p> <p>If either or both of the Organization->Extract->Jobtitle and Organization->Extract->Name properties are used in conjunction with this one, matchIT SQL will not only scan the ADDRESSEE field for job titles and business names, but will also scan the corresponding number of address lines.</p>
matchITAPISettings->generate->Address->premiseFirst	When parsing an address, this Boolean property indicates whether to expect the premise or flat number to come first in address lines when the flat is not explicitly specified (e.g. "Flat 5").
matchITAPISettings->generate->Organization->Extract->jobTitle	<p>This will copy or extract job titles contained within the ADDRESSEE field into a field labelled JOB_TITLE within your output table.</p> <p>Job Titles are recognized by having a word or string defined as a Job Title in the NAMES.DAT file e.g. Director.</p>
matchITAPISettings->generate->Organization->Extract->name	<p>This will copy or extract any business names contained within the ADDRESSEE field into a field labelled COMPANY within your output table.</p> <p>Business names are recognized by having a word or string defined as a Business word in the NAMES.DAT file e.g. Ltd. Care should be taken when using this property, as words like "Bank" can be taken to indicate a Business when this isn't the case (e.g. it may be a last name or part of an address line).</p> <p>If you want Extract Company Name processing to be applied also to the first one or two lines of the address, you must Set the property <i>Generate->Address->NumOfLinesToScan</i> to either 1 or 2.</p>
matchITAPISettings->generate->Organization->joinInitials	Set this property to True if you want a group of initials separated by spaces or dots in a company name to be concatenated. For example, if this property is True, then "I B M" and "I.B.M." will be replaced by "IBM" within the company field of your output table. Note that, if the <i>Generate->ProperCase</i> property is set to False, then this property will have no effect.
matchITAPISettings->generate->Organization->useEquivalentName	<p>If this property is set to True, then the equivalent (according to the NAMES.DAT file) of words indicating a business name, such as "Motors" or "Services" are included in the <i>NormalizedOrganization</i> field in the generated keys table and the corresponding phonetic keys. This enables, for example, "Wood Green Cars" to match "Wood Green Motors" well (because "Cars" has an equivalent of "Motors"), but ensures that neither of them match "Wood Green Carpets" well.</p> <p>If you want tight legal matching turned on so that for example, 'Wood Green Cars Limited' will match 'Wood Green Cars Ltd' but 'Wood Green Cars Group' will not match 'Wood Green Cars Ltd', then in addition to setting this option to True, you will also need to modify the 'matchITAPISettings>datPath' property to the location of the tight dat files which can be found in a subfolder called 'Tight' under each region folder in C:\matchIT</p>

	<p>SQL\config\dataFiles\... .</p> <p>If you set this property to True, you should change any words in the NAMES.DAT file that you do want ignored, such as "Ltd" and "Inc" to Noise type so that they are not included in the <i>NormalizedOrganization</i> field. As a rule of thumb, if you are doing business matching on a file that is very geographically concentrated, that is, contains records mostly from the same immediate area, then set the <i>Generate->Organization->UseEquivalentName</i> property to True, otherwise set it to False.</p>
matchITAPISettings->generate->Organization->normalizationTruncation	<p>Disabled by default (i.e. set to 0) If this setting is enabled, and the organization consists of more than four words, then the third element of field NormalizedOrganization within your generated keys table will be truncated to the first N characters of each word after the first two (where N is the value of this setting).</p>
matchITAPISettings->generate->Organization->ignoreParentheses	<p>With this property enabled, any words that are enclosed with parentheses within an organization name will be excluded from the generated phonetic organization keys. This can be useful for records such as Remnel Ltd and Remnel (UK) Ltd, to ensure records with these company names are compared if the phonetic organization keys are being used as part of composite match keys.</p>
matchITAPISettings->generate->Organization->ignoreTrailingPostTown	<p>This property, when enabled, will exclude from the phonetic organization keys any trailing post town (defined in the towns.dat file) or UK county that appears at the end of a company name.</p> <p>For example, the phonetic organization keys for Handso Ltd and Handso Essex Ltd will be the same to help ensure such records will be compared.</p>

7.2.3.1 Match Key Fields

The match key table can contain the following fields:

Column Name	Description
(unique ref)	Unique reference for each record – either directly specified for the keys table in the table mappings, or taken from the first table in the mappings.
mkNameKey	Phonetic representation of the name. Optional.
mkOrganizationKey	Phonetic representation of the company name. Optional.
mkAddressKey	Phonetic representation of the address lines. Optional.
mkPhoneticStreet	Phonetic representation of the thoroughfare. Optional.
mkPhoneticTown	Phonetic representation of the town/city. Optional.
mkPostOut	First part of the postal code/zip. Optional.
mkPostIn	Second part of the postal code/zip. Optional.
mkName1	Phonetic representation of the lastname. Optional.
mkName2	Phonetic representation of the firstname. Optional.
mkName3	Phonetic representation of the middle name or initial. Optional.
mkOrgName1	Phonetic representation of the first word of the company name. Optional.
mkOrgName2	Phonetic representation of the second word of the company name. Optional.
mkOrgName3	Phonetic representation of the third word of the company

	name. Optional.
mkTelAreaCode	Telephone area code. Optional.
mkTelLocalNumber	Telephone local number. Optional.
mkFaxAreaCode	Fax area code. Optional.
mkFaxLocalNumber	Fax local number. Optional.
mkName2Found	Indicates whether the firstname was found in the names.dat file. Optional.
mkNormalizedName	Normalised version of all of the consumer name data.
mkGender	Generated gender based on the name data provided
mkSuffix	Any suffix data extracted from the name fields in the source data.
mkNormalizedOrganization	Normalised version of the Organization name.
mkPremise	Premise data extracted from the address lines. Optional.
mkFlatNo	Sub premise extracted from the address lines. Optional.
mkDataFlags	Flag field generated during key generation. Please see Appendix B.
mkMasterPriority	Master priority calculated for the record based on the completeness of the data as defined by matchIT SQL's Master Priority Matrix. Used during the grouping of matches to help determine the master record.
mkAddressLength	The calculated length of the address data contained in the source data. Used during the grouping of matches to help determine the master record.

The optional columns can be configured in the outputSettings->keyColumns node in a configuration file; most are enabled by default, but key columns not required by any match keys can be disabled to help improve performance of key generation and deduplication.

Columns not marked as optional will always be added to the keys table and cannot be disabled.

7.2.4 msp_BulkGenerateKeys

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.

As msp_GenerateKeys, except that the matching and key field data is bulk loaded into an empty keys table. Significantly reduces the time taken for key generation, particularly when used with large databases.

7.2.5 msp_CreateIndexes

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.

- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.

Creates all the nonclustered indexes on the specified keys table required for efficient clustering of potential matches. Note that the indexes are automatically created when the keys are generated, so it's not normally necessary to execute this stored procedure.

7.3 Exact Deduplication

7.3.1 msp_FindExactMatches

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.

Finds all exact matching record pairs in the specified database (taken from the supplied datasource). The following settings are relevant to FindExactMatches:

Setting	Description
matchKeys->exactKeys	The match keys that will be used are specified in the XML within the exactKeys keys tags under the match keys section. Fields can be concatenated together to create an exact match key e.g. <key key1="mkName1" key2="mkName2" /> Which means that all records with the same phonetic forename and surnames will be recorded as matches.
dataSources	Defines the database connection, tables and columns of the dataset that is to be matched.
outputSettings->exactMatchesTable	Specifies the name of the exact_matches table that will be produced (which contains the results from the FindExactMatches processing).
outputSettings->reports	Specifies whether reporting is enabled, what folder the reports will be produced into, and what report format should be used.

If the 'excludeExactMatches' configuration option is enabled (the default), then exact matches will be automatically excluded when msp_FindMatches is run, to help increase fuzzy deduplication performance. If the level of duplication is low, however, this option should be disabled.

As the matching process runs, the results are written out to a results table within your SQL Server database (in reality the matching results are written to a temporary file and then bulk loaded on completion of the process). The Find Exact Matches process produces 1 output table as follows (the name of which can be configured through the Web UI or XML):

7.3.1.1 Exact_matches table structure

Column	Description
ID	Record ID for each matching pair.
Record1	Reference ID of the first record in the matching pair.
Record2	Reference ID of the second record in the matching pair.

7.3.2 msp_FindExactOverlap

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Main datasource ID – specifies the data source to be used from the configuration file, which contains the table and column mapping specifications of the first dataset.
- Overlap datasource ID – specifies the data source to be used from the configuration file, which contains the table and column mapping specifications of the second dataset to be used in the overlap.

Finds all records that exactly overlap the specified tables.

The 'excludeExactMatches' configuration option can be used with this stored procedure, as per msp_FindExactMatches.

Setting	Description
matchKeys->exactKeys	The match keys that will be used are specified in the XML within the exactKeys keys tags under the match keys section. Fields can be concatenated together to create an exact match key e.g. <pre><key key1="mkName1" key2="mkName2" /></pre> Which means that all records with the same phonetic forename and surnames will be recorded as matches.
dataSources	Defines the database connection, tables and columns of the datasets that are to be matched.
outputSettings->exactMatchesTable	The Overlap attribute specifies the name of the exact_matches table that will be produced (which contains the results from the FindExactMatches processing). If the Overlap attribute is empty, the table will be given the same name as that specified in the name attribute.
outputSettings->reports	Specifies whether reporting is enabled, what folder the reports will be produced into, and what report format should be used.

During processing, exact matches are written to an overlap exact_matches output table as configured within the configuration file used.

7.3.2.1 Overlap Exact_matches table structure

Column	Description
ID	Record ID for each matching pair.
Record1	Reference ID of the first record in the matching pair.
Record2	Reference ID of the record in the second supplied datasource that record1 has matched to.

7.3.3 msp_GroupExactMatches

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used from the configuration file, which contains the table and column mapping specifications of the first dataset.

After running `msp_FindExactMatches`, this will group all matching record pairs into *sets* of matching records, and will create tables that are referenced by a following instance of `msp_FindMatches`.

Setting	Description
<code>dataSources</code>	Defines the database connection, tables and columns of the datasets that are to be matched.
<code>outputSettings->exactMatchesTable</code>	Name of the table containing the exact matching pairs results which are to be grouped into matching sets.
<code>outputSettings->groupedExactMatchesTable</code>	Name of the output table to be created when the grouping runs.

7.3.3.1 Exact_matches_grouped table structure

During processing the stored procedure will output the results to the `exact_matches_grouped` table (this name can be configured – see above). The structure of the output table is as follows:

Column	Description
ID	Record ID for each matching group.
Record1	Reference ID of the first record in the matching pair.
Record2	Reference ID of the second record in the matching relationship.
MatchRef	ID of the matching group that the records belong to. Note that the MatchRef will be the ID value of the first record in the matching group.

7.3.4 msp_GroupExactOverlap

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Main datasource ID – specifies the data source to be used from the configuration file, which contains the table and column mapping specifications of the first dataset.
- Overlap datasource ID – specifies the data source to be used from the configuration file, which contains the table and column mapping specifications of the second dataset to be used in the overlap.

After running msp_FindExactOverlap, this will group all matching record pairs into sets of matching records, and will create tables that are referenced by a following instance of msp_FindOverlap.

Setting	Description
dataSources	Defines the database connection, tables and columns of the datasets that are to be used in the overlap.
outputSettings->exactMatchesTable	Name of the table containing the exact matching pairs results which are to be grouped into matching sets.
outputSettings->groupedExactMatchesTable	Name of the output table to be created when the grouping runs. Note that if the overlap attribute is blank, then the name will be taken from the Name attribute.

7.3.4.1 Overlap Exact_matches_grouped table structure

During processing the stored procedure will output the results to the exact_matches_grouped table (this name can be configured – see above). The structure of the output table is as follows:

Column	Description
ID	Record ID for each matching group.
Record1	Reference ID of the first record in the matching pair taken from the first datasource.
Record2	Reference ID of the record from the second datasource that is deemed to match Record1 From the first datasource.
MatchRef	ID of the matching group that the records belong to. Note that the MatchRef will be the ID value of the first record in the matching group and will be from the second datasource.

7.4 Fuzzy Deduplication

7.4.1 msp_FindMatches

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications.

Finds all matching record pairs in the specified table. There are quite a few configurable settings for this procedure:

Setting	Description
matchKeys->fuzzyKeys	The match keys that will be used are specified in the XML within the fuzzy keys tags under the match keys section. Fields can be concatenated together to create a match key e.g. <key key1="mkPostOut" key2="mkName1" /> Which means that all records with the same PostOut value and same phonetic surname will be compared.
generalSettings->minimumIndividualScore	The minimum threshold score required for a match to be considered an Individual level match.
generalSettings->minimumFamilyScore	The minimum threshold score required for a match to be considered a family level match.
generalSettings->minimumHouseholdScore	The minimum threshold score required for a match to be considered a household level match.
generalSettings->minimumBusinessScore	The minimum threshold score required for a match to be considered a business level match.
generalSettings->minimumCustomScore	You may decide to create your own custom match level. This setting configures the minimum threshold score required for a match to be considered a custom level match.
generalSettings->preClustering	Internal to matching algorithm, leave switched on for best performance.
generalSettings->stripPuncWhenExactMatching	matchIT SQL will ignore punctuation during exact matches when activated. This setting should only be used when using non match key columns.
generalSettings->excludeExactMatches	If this is enabled (the default) and msp_FindExactMatches/Overlap is run prior to msp_FindMatches/Overlap, then the final matches table produced will not include any exact matches. The exact matches already found are excluded to boost the performance of the fuzzy matching step and it is recommended that you use FindExactMatches if you are processing large datasets. Note that the data from the exact_matches table should be appended to the matches table before the relevant grouping and output stored procedures are run, unless Merge Exact Matches is enabled.
generalSettings->flagMatchesAtHigherLevels	If this setting is enabled, then individual level matches will always be marked as family and household level matches. Family level matches will be marked as household level matches.

outputSettings->matchesTable	<p>This setting within the configuration allows you to specify the name of the matches table that will be produced.</p> <p>Additionally, for each matching level, you can choose what level of scoring information you would like written into this table. By default, only the total scores for each matching level are enabled, but for example you could add the address component scores by setting the address property to '1'.</p>
outputSettings->largeClustersTable	<p>This setting within the configuration allows you to specify the name of the table containing the large clusters that will be produced.</p>
ADVANCED SETTINGS	The following settings are advanced and most users will not normally need to modify the default settings.
matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchGender	<p>When this property is set to True, potential matches will be disregarded if their genders differ. If however the gender is unknown in one or both of the records, the records will potentially be classed as a match.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchSuffix	<p>When this property is set to True, potential matches will be disregarded if their suffixes differ. If however the suffix is unknown in one or both of the records, the records will potentially be classed as a match.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchLocation	<p>When this property is set to True, potential matches will be disregarded if their address locations differ.</p> <p>In detail, this means that the postcodes in the two records (if present) must achieve at least a probable match with the address score at least a Possible match, or the address score must be at least a Likely match irrespective of the postcodes, or the postcodes must achieve a Sure match irrespective of the address. This is to prevent false matches where there is some match on address, but where the addresses are clearly not the same, for example "10 High Street, Bookham", and "10 High Street, Alford".</p> <p>Switch this constraint off if you want to match people or companies in different locations; you may want to match on items of data that are independent of location, such as date of birth or bank account.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchPremise	<p>When this property is set to True, potential matches will be disregarded if their premise numbers differ. If however the premise number is unknown (e.g. one record or both records may contain a premise name), the records will potentially be classed as a match.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->noOneEmptyPremise	<p>When this property is set to True, potential matches will be disregarded if one of the addresses is missing a premise number.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->allowFuzzyPremiseMatch	<p>When <i>both</i> this and MustMatchPremise are set to True, then potential matches will be disregarded if the premises are not exact matches (for example, 71 and 71) or if they're not <i>fuzzy</i> matches (for example 71 and 71A, 45 and 54, or 71 and 7).</p> <p>Note that this property has no effect if MustMatchPremise is set to False because, in that case, fuzzy premises are always allowed.</p> <p>Also note that this setting is also available for Family, Household, Business and Custom match levels.</p>

matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchDirectional	<p>When this property is set to True, potential matches will be disregarded if both addresses (i.e. typically US) have a pre- or post-directional (e.g. N, North, E, etc.) but they don't match. For example, with this constraint enabled, "N Washington Ave" and "S Washington Ave" will not be matched.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchNumericStreetName	<p>When this property is set to True, potential matches will be disregarded if both addresses (i.e. typically US) have a numeric street name but they don't match. For example, with this constraint enabled, "5th Ave" and "15th Ave" will not be matched.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchJointNames	<p>When this property is set to True, potential matches will be disregarded if one record has a joint name but the other doesn't. For example, normal behaviour will match "Mr & Mrs J Smith" with "Mr J Smith"; setting this property to True will prevent such matches.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->mustMatchBuilding	<p>When this property is set to True, potential matches will be disregarded if their building names differ. If however one or both addresses do not contain a building name, the records will potentially be classed as a match.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->constraints->noOneEmptyBuilding	<p>When this property is set to True, potential matches will be disregarded if one of the addresses is missing a building name.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->name	<p>Defines the scores produced when names are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->organization	<p>Defines the scores produced when organisations are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->address	<p>Defines the scores produced when address fields are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->postcode	<p>Defines the scores produced when postcode/zip fields are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->telephone	<p>Defines the scores produced when telephones are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->email	<p>Defines the scores produced when emails are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->dateOfBirth	<p>Defines the scores produced when date of birth fields are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>
matchITAPISettings->matchingRules->individualLevel->weights->customField1	<p>Defines the scores produced when fields defined as customField are compared.</p> <p>Note that this setting is also available for Family, Household, Business and Custom match levels.</p>

matchITAPISettings->matchingRules->individualLevel->nameMatchingMatrix	The location of the name matching matrix. Note that this setting is also available for Family, Household, Business and Custom match levels.
matchITAPISettings->matchingRules->individualLevel->organizationMatchingMatrix	The location of the Organization Matching Matrix. Note that this setting is also available for Family, Household, Business and Custom match levels.

7.4.1.1 Match Scoring

During the FindMatches step, records with the same match keys are grouped together and then compared with each other. matchIT SQL uses the scoring weights for specific fields to determine an overall match score. If the score reaches the defined threshold score for a given match level (configured in the minimum score settings – see above), then a matching pair will be written to the results table along with the score achieved.

The scoring weights have the following categories:

- Sure – generally means a match is certain (for example when scoring a name comparison, both the forenames and surnames of the records might be the same)
- Likely – There are small differences between the records for field being compared.
- Possible – There are notable differences between the records for field being compared, but the fields could still be a match.
- OneEmpty – one record has a blank entry for the field being compared, e.g. one record has no postcode when scoring a postcode/zip comparison.
- BothEmpty – both records have blank entries for the field being compared.

For example, if the minimum score setting for Individual Level matching is 80, then the combined component scores for Name, Address, Postcode, Email, Telephone, Organization and Custom must be greater than equal to 80. The component scores themselves are driven by the weights settings which dictate the score that will be achieved when specific elements are compared (depending upon how similar those elements are).

For example, by default when two names are compared at individual level, should they be considered to be a Sure match, then the name component of the overall score would be 60. If the postcodes when compared were considered to only be Possible matches, then the default weight indicates that the postcodes would score 15.

7.4.1.1.1 Name Comparisons

matchIT SQL uses a name matching matrix to decide on whether when comparing names, records should achieve Sure, Likely, Possible etc. Upon installation of matchIT SQL, the following folder will be created

C:\matchIT SQL\config\matchingMatrices

Which contains matching matrices for the various matching levels. To illustrate how the matrix works, consider the following XML:

```
<lastnames match="equal">
  <firstnames match="equal">
    <middlenames match="equal">sure</middlenames>
    <middlenames match="both_empty">sure</middlenames>
```



```

<middlenames match="one_empty">sure</middlenames>
<middlenames match="approx">likely</middlenames>
<middlenames match="contains">likely</middlenames>
<middlenames match="unequal">possible</middlenames>
</firstnames>

```

This indicates that where the last names for 2 records being compared are the same and the first names are also the same, then the ultimate result of the comparison would depend upon the data in the middle name fields, e.g. for Middle names that are also equal, then the result is sure. The actual score that this sure match would be worth, would depend on the matching weights defined in your configuration file for the sure match on name (at the corresponding matching level). There are similar matching matrices for comparing organization names.

For non name data, the matching result is determined as follows:

7.4.1.1.2 Postcode/Zip Comparisons

The following rules are applied during comparisons of postcode/zip fields:

- equal -> SURE MATCH
- part equal (e.g. 5 digit zips match) -> LIKELY MATCH
- fuzzy differences (1-char insertion/deletion/replacement, 2-char transposition) -> POSSIBLE
- not equal -> NO MATCH

7.4.1.1.3 UK Specific Postcodes

The following rules are applied during comparisons of UK postcode fields:

- Equal (postout & postin present) -> SURE MATCH
- fuzzy differences (1-char insertion/deletion/replacement, 2-char transposition) where both postout and postin are present -> LIKELY MATCH
- Equal postouts, but both records missing postin -> LIKELY MATCH
- Equal postouts, but one record missing postin -> POSSIBLE MATCH
- fuzzy differences in the postout sections (1-char insertion/deletion/replacement, 2-char transposition), but both records missing postins -> POSSIBLE MATCH
- Other differences -> NO MATCH

7.4.1.1.4 Address Comparisons

Address scores are calculated as a percentage of the address Sure weight.

Likely and Possible are really just thresholds and are not used the same way that they are for names, companies, postcodes etc.

An address that scores less than Possible will score 0.

If mustMatchLocation is enabled (the default), then a record that scores 0 on postcode must score at least Likely for address; a record that doesn't score Sure on postcode must get at least Possible for address; otherwise (if the record scores Sure on postcode) then the address can score anything.

7.4.1.1.5 Dates of Birth Comparisons

The following rules are applied during comparisons of date of birth fields:

- equal -> SURE MATCH
- fuzzy differences (1-char insertion/deletion/replacement, 2-char transposition) -> LIKELY MATCH
- containment (if at least 10 chars) -> POSSIBLE
- not equal -> NO MATCH

7.4.1.1.6 Email Comparisons

The following rules are applied during comparisons of email fields:

- equal -> SURE MATCH
- fuzzy (1-char insertion/deletion/replacement, 2-char transposition) -> LIKELY MATCH
- not equal -> NO MATCH

7.4.1.1.7 Custom Field Comparisons

The following rules are applied during comparisons of custom fields:

- equal -> SURE MATCH
- fuzzy differences (1-char insertion/deletion/replacement, 2-char transposition) -> LIKELY MATCH
- containment (if at least 10 chars) -> POSSIBLE
- not equal -> NO MATCH

7.4.1.2 matches table

As the matching process runs, the results are written out to tables within your SQL Server database. The Find Matches process produces 2 output tables, the first being the matches table which is described below (the names of which can be configured through the Web UI or XML):

This table contains the matching pairs that have been identified as a result of the fuzzy matching process.

Column	Description
ID	Record ID for each matching pair.
Record1	Reference ID of the first record in the matching pair.
Record2	Reference ID of the record that is the second record in the matching pair.
Level	<p>The Level column indicates the matching level(s) at which a match was found. If it contains a 1 then the two records match at the Individual level; if 2, then Family level; if 4, then Household level; and if 8, then Business level. Multiple levels are indicated by summing values – for example, 9 would indicate a match at both Individual and Business levels (1+8), and 15 a match at all four levels (1+2+4+8).</p> <p>By default, the Level column is followed by the total score for the four matching levels. These columns are fully configurable within a configuration file. Component scores (for name, organisation, address, etc.) can also be output for any level(s).</p>
IndividualScore	Individual level total match score.
FamilyScore	Family level total match score.
HouseholdScore	Household level total match score.
BusinessScore	Business level total match score.
These following columns relate either to master record identification or bridging prevention and shouldn't be used for any other purpose; they are subject to change in future versions of matchIT SQL.	
MatchFlags	The MatchFlags column is only used when Bridging Prevention is enabled (see GroupMatches).
MasterPriority1	Used for Master Record Identification.
MasterPriority2	Used for Master Record Identification.
AddressLength1	Used for Master Record Identification.
AddressLength2	Used for Master Record Identification.
Key	Indicates through which match key (as specified in your configuration file) this matching pair was found.

7.4.1.3 large_clusters table

This table lists the clusters that contain too many records (i.e. the Maximum Cluster Size has been exceeded). Processing the cluster will therefore be skipped to avoid the stored procedure potentially requiring a significant amount of processing time.

Column	Description
ID	Record ID.
KeyIndex	The composite key being processed.
Search	The current composite key value that identifies the current cluster (for example, the value of mkPostOut+mkName1 – note that a pipe character separates each key value).
Records	The total number of records in the cluster.
MaxRecords	The maximum cluster size constant.

7.4.2 msp_FindOverlap

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Main datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications. This datasource will be considered the Main datasource.
- Overlap datasource ID – specifies the data source to be used within the configuration file, which contains the table and column mapping specifications. This datasource will be considered the Overlap datasource.

The FindOverlap procedure is essentially the same as FindMatches (and allows for the same settings), however the outputted results table contains matching pairs where a record in the update table has matched against a record in the main table.

As the matching process runs, the results are written out to tables within your SQL Server database. The FindOverlap process produces 2 output tables as follows (the names of which can be configured through the Web UI or XML).

7.4.2.1 Overlap matches table

This table contains the matching pairs that have been identified as a result of the fuzzy matching process.

Column	Description
ID	Record ID for each matching pair.
Record1	Reference ID of the first record in the matching pair.
Record2	Reference ID of the record that is the second record in the matching pair. This record belongs to the second datasource specified in the parameters.
Level	<p>The Level column indicates the matching level(s) at which a match was found. If it contains a 1 then the two records match at the Individual level; if 2, then Family level; if 4, then Household level; and if 8, then Business level. Multiple levels are indicated by summing values – for example, 9 would indicate a match at both Individual and Business levels (1+8), and 15 a match at all four levels (1+2+4+8).</p> <p>By default, the Level column is followed by the total score for the four matching levels. These columns are fully configurable within a configuration file. Component scores (for name, organisation, address, etc.) can also be output for any level(s).</p>
IndividualScore	Individual level total match score.
FamilyScore	Family level total match score.
HouseholdScore	Household level total match score.
BusinessScore	Business level total match score.
These following columns relate to either master record identification or bridging prevention and shouldn't be used for any other purpose; they are subject to change in future versions of matchIT SQL.	
MatchFlags	The MatchFlags column is only used when Bridging Prevention is enabled (see GroupMatches).

MasterPriority1	Used for Master Record Identification.
MasterPriority2	Used for Master Record Identification.
AddressLength1	Used for Master Record Identification.
AddressLength2	Used for Master Record Identification.
Key	Indicates through which match key (as specified in your configuration file) this matching pair was found.

7.4.2.2 large_clusters table

This table lists the clusters that contain too many records (i.e. the Maximum Cluster Size has been exceeded). Processing the cluster will therefore be skipped to avoid the stored procedure potentially requiring a significant amount of processing time.

Column	Description
ID	Record ID
KeyIndex	The composite key being processed.
Search	The current composite key value that identifies the current cluster (for example, the value of mkPostOut+mkName1 – note that a pipe character separates each key value).
Records	The total number of records in the cluster.
MaxRecords	The maximum cluster size constant.

7.4.3 msp_GroupMatches

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains the database connection string.
- Level – Matching Level at which to group the records (Individual, Family, Household, Business, Custom).

After running msp_FindMatches, this will group all matching record pairs into *sets* of matching records. If you have previously run FindExactMatches, then we recommend that you copy the matching pairs data from your exact_matches table into you matches table prior to running this grouping step, unless Merge Exact Matches is enabled.

Setting	Description
generalSettings->preventBridgedMatches	When enabled matchIT SQL will attempt to stop match groups containing bridged records such as: J Smith John Smith Julian Smith In the scenario above, both John Smith and Julian Smith match with J Smith, but not with each other.
generalSettings->masterRecordIdentification	When this setting is active, matchIT SQL will use the MasterPriority matrix to determine which record in a matching group should be marked as the master record (i.e. the best record). When this setting is off, the record with the lowest unique_reference will be chosen as the master record.
dataSources->ConnectionString	Connection string used to connect to the database during processing.
outputSettings->groupedMatchesTable	Name of the group matches output table that will be produced during the processing of this procedure.
outputSettings->matchesTable	Name of the matches table containing the matching pairs that will be used as the input source for this procedure.
generalSettings->excludeExactMatches	The merge attribute of this setting indicates whether to merge all exact matches, only those for the grouping level being used, or none. The records are then merged prior to the grouping.

7.4.3.1 Master Record Identification

As part of the grouping process, matchIT SQL will intelligently choose the master record for a matching group based on the data contained in the records in the matching set. The record with the best data is designated as the master record for the matching group.

matchIT SQL uses a master priority table to determine how to score the quality of the data held within the matching records. The default location of this file is as follows:

- C:\matchIT SQL\config\masterPriorities\default.xml

The table itself dictates scoring rules for each type of matching field. For example, by default if the phonetic last name field is empty (mkName1), then the master priority score will have 99 points subtracted from it. The overall score is the sum of the scores for each field listed in the matrix.

Note that you can also add Customfields (if you have mapped any in your datasources) into the matrix for example consider the following rule:

- `<rule field="CustomField1" test="value" pos="8" operation="equal" value="X" score="-33" />` - this rule means that when the fields mapped as CustomField1 is scored, if the value of the character in position 8 contains an X, then the total master priority score will have 33 subtracted from it.

Note that when testing the length or value of a field, the following operations are permitted:

- equal
- notEqual
- greater
- notGreater
- less
- notLess

In the event that you have 2 records both scoring the same master priority score, and both having the highest score in the matching group, then the record with the biggest address length (i.e. number of characters in the address fields) will be marked as the master record.

The record in the matching group with the highest master priority score will be designated as the Master Record.

7.4.3.2 matches_grouped table structure

During processing the stored procedure will output the results to the matches_grouped table (this name can be configured – see above). The structure of the output table is as follows:

Column	Description
ID	Record ID for each matching group.
Record1	Reference ID of the first record in the matching pair.
Record2	Reference ID of the record that is deemed to match Record1 From the first datasource.
Score	The Score column is copied from the relevant level's total score (grouping can only take place on <i>one</i> matching level; to group using multiple levels requires multiple runs of GroupMatches/GroupOverlap).
MatchRef	Indicates the unique reference of the master record in the group. If Master Record Identification is enabled, then this will indicate the 'best' record in the group; if not, then the MatchRef will simply be set to the lowest unique reference of all the records in the group (note that the column is a char column, not an integer column, so a unique ref of 100 will be deemed 'lower' than 20 because a left-to-right character-based comparison is used).
BaseScore	Indicates the lowest score of all the matches in the group, or, if blank, this will indicate a merged exact match

This table lists the records from the matches table after they have been placed into groups of matches. For example, if record A matches B and B matches C, then all three records will be placed into the same group.

7.4.4 msp_GroupOverlap

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string.
- Level – Matching Level at which to group the records (Individual, Family, Household, Business, Custom).

After running msp_FindOverlap, this will group all matching record pairs into *sets* of matching records. If you have previously run FindExactOverlap, then we recommend that you copy the matching pairs data from your exact_matches table into you matches table prior to running this grouping step, unless Merge Exact matches is enabled.

Setting	Description
generalSettings->preventBridgedMatches	When enabled matchIT SQL will attempt to stop match groups containing bridged records such as: J Smith John Smith Julian Smith In the scenario above, both John Smith and Julian Smith match with J Smith, but not with each other.
generalSettings->masterRecordIdentification	When this setting is active, matchIT SQL will use the MasterPriority matrix to determine which record in a matching group should be marked as the master record (i.e. the best record). When this setting is off, the record with the lowest unique_reference will be chosen as the master record.
dataSources->ConnectionString	Connection string used to connect to the database during processing.
outputSettings->groupedMatchesTable	Name of the group matches output table that will be produced during the processing of this procedure. Note that if the overlap attribute is empty, then the name in the name attribute will be used.
outputSettings->matchesTable	Name of the matches table containing the matching pairs that will be used as the input source for this procedure. Note that if the overlap attribute is empty, then the name in the name attribute will be used.

7.4.4.1 Overlap matches_grouped table structure

During processing the stored procedure will output the results to the matches_grouped table (this name can be configured – see above). The structure of the output table is as follows:

Column	Description
ID	Record ID for each matching group.
Record1	Reference ID of the first record in the matching pair (from the Main database)
Record2	Reference ID of the record that is deemed to match Record1 From the Overlap datasource.
Score	The Score column is copied from the relevant level's total score (grouping can only take place on <i>one</i> matching level; to group using multiple levels requires multiple runs of GroupMatches/GroupOverlap).
MatchRef	Indicates the unique reference of the master record in the group. In the case of an overlap, the MatchRef column indicates the unique reference of the record from the overlap table; in effect, it's a simple copy of the Record2 column.
BaseScore	Indicates the lowest score of all the matches in the group.

This table lists the records from the matches table after they have been placed into groups of matches. For example, if record A matches B and B matches C, then all three records will be placed into the same group.

7.5 Output

7.5.1 msp_OutputMatchingPairs

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string.

After running msp_FindMatches, this will output all matching record pairs to the 'matching_pairs' table, ordered by the Score column.

Setting	Description
outputSettings->matchingPairsTable	Specifies the name of the matching pairs table that will be produced.
outputSettings->matchesTable	Specifies that name of the matches table that will be required to generate the matching pairs table.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches table to produce the matching_pairs table.

7.5.1.1 Matching_pairs table

Column	Description
Score	Match Score for the matching pair.
ID_1	Reference ID of the first record in the matching pair.
ID_2	Reference ID of the second record in the matching pair.
MatchRef	

Note that the matching_pairs table also contains the source fields for each record mapped in the datasource within the configuration file, allowing you to view the actual data that has matched.

7.5.2 msp_OutputMatchingGroups

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string.

After running msp_GroupMatches, this will output all groups of matching records to the 'matching_groups' table, ordered by the MatchRef column.

Setting	Description
outputSettings->matchingGroupsTable	Specifies the name of the matching groups table that will be produced.
outputSettings->groupedMatchesTable	Specifies that name of the matches_grouped table that will be required to generate the matching_groups table.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches_grouped table to produce the matching_pairs table.

7.5.2.1 Matching_groups table

Column	Description
MatchRef	Reference ID for the matching group
ID	ID of record

Note that the matching_groups table also contains the source fields for each record mapped in the datasource within the configuration file, allowing you to view the actual data that has matched.

7.5.3 msp_OutputDuplicates

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string.

After running msp_GroupMatches, this will output all the non-master duplicate records (i.e. all records where the unique ref is different from the MatchRef) that are to be removed from the source table(s).

Setting	Description
outputSettings->duplicatesTable	Specifies the name of the duplicates table that will be produced.
outputSettings->groupedMatchesTable	Specifies that name of the matches_grouped table that will be required to generate the matching_groups table.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches_grouped table to produce the duplicates table.

7.5.3.1 Duplicates table

This table contains the non-master duplicate records following the matching process. The structure of the table is determined by the fields mapped in the datasource (i.e. it contains an ID field, but also the fields that you have mapped in the datasource).

7.5.4 msp_OutputDedupedTable

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string.

This effectively produces the opposite of msp_OutputDuplicates. All records from source table(s) are output, except for the identified non-master duplicate records.

Setting	Description
outputSettings->dedupedTable	Specifies the name of the deduped table that will be produced.
outputSettings->groupedMatchesTable	Specifies that name of the matches_grouped table that will be required to generate the matching_groups table.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches_grouped table to produce the deduped table.

7.5.4.1 Deduped table

This table contains only the master records following the matching process. The structure of the table is determined by the fields mapped in the datasource (i.e. it contains an ID field, but also the fields that you have mapped in the datasource).

7.5.5 msp_TagMatchingResultsWithGroupLevel

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string
- Level – can be Individual, Family, Household, Business or Custom.

This procedure renames any of the grouping related tables produced by the preceding four procedures and the msp_GroupMatches Procedure. Each table is renamed by giving it a suffix matching the value passed in the level parameter.

Setting	Description
outputSettings->duplicates	Name of the duplicates table to be renamed.
outputSettings->dedupedTable	Name of the deduped table to be renamed.
outputSettings->matchingGroupsTable	Name of the matching_groups table to be renamed.
outputSettings->groupedMatchesTable	Specifies that name of the matches_grouped table to be renamed.
dataSources	Specifies the datasource containing the connection string to the database.

7.5.6 msp_OutputOverlapMatchingPairs

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Main datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Overlap datasource ID – specifies the second data source to be used within the configuration file, which contains database connection string and table/column mappings.

After running msp_GroupOverlap, this will output all record pairs that match between the two datasources to the 'matching_pairs' table, ordered by the Score column.

Setting	Description
outputSettings->matchingPairsTable	The Overlap attribute specifies the name of the overlap matching pairs table that will be produced. If the Overlap attribute is empty, then the name attribute will be used.
outputSettings->matchesTable	The Overlap attribute specifies that name of the overlap matches table that will be required to generate the matching pairs table. If the Overlap attribute is empty, then the name attribute will be used.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches table to produce the overlap matching_pairs table.

7.5.6.1 Overlap Matching_pairs table

Column	Description
Score	Match Score for the matching pair.
ID_1	Reference ID of the first record in the matching pair.
ID_2	Reference ID of the second record in the matching pair which will be a record from the second datasource.
MatchRef	

Note that the overlap matching_pairs table also contains the source fields for each record mapped in the datasource within the configuration file, allowing you to view the actual data that has matched.

7.5.7 msp_OutputOverlapMatchingGroups

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Main datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Overlap datasource ID – specifies the second data source to be used within the configuration file, which contains database connection string and table/column mappings.

After running msp_GroupOverlap, this will output all groups of matching records that match between the two datasources to the 'matching_groups' table, ordered by the MatchRef column.

Setting	Description
outputSettings->matchingGroupsTable	The Overlap attribute specifies the name of the matching groups table that will be produced. If the Overlap attribute is empty, then the name attribute will be used.
outputSettings->groupedMatchesTable	The Overlap attribute specifies that name of the matches_grouped table that will be required to generate the matching_groups table. If the Overlap attribute is empty, then the name attribute will be used.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches_grouped table to produce the matching_pairs table.

7.5.7.1 Overlap Matching_groups table

Column	Description
MatchRef	Reference ID for the matching group; this will reference a record in the second datasource that is considered to be the master record for the matching group.
ID_1	ID of record from the first datasource.
ID_2	ID of record from the second datasource.

Note that the overlap matching_groups table also contains the source fields for each record mapped in the datasources within the configuration file, allowing you to view the actual data that has matched.

7.5.8 msp_OutputOverlapDuplicates

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Main datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Overlap datasource ID – specifies the second data source to be used within the configuration file, which contains database connection string and table/column mappings.

After running msp_GroupOverlap, this will output all duplicate records from datasource2 that overlap with datasource1.

Setting	Description
outputSettings->duplicatesTable	The Overlap attribute specifies the name of the duplicates table that will be produced. If the Overlap attribute is empty, then the name attribute will be used.
outputSettings->groupedMatchesTable	The Overlap attribute specifies that name of the matches_grouped table that will be required to generate the duplicates table. If the Overlap attribute is empty, then the name attribute will be used.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches_grouped table to produce the duplicates table.

7.5.8.1 Overlap Duplicates table

This table contains the non master duplicate records following the matching process. The structure of the table is determined by the fields mapped in the datasource (i.e. it contains an ID field, but also the fields that you have mapped in the datasource).

7.5.9 msp_OutputOverlapDedupedTable

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Main datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Overlap datasource ID – specifies the second data source to be used within the configuration file, which contains database connection string and table/column mappings.

This effectively produces the opposite of msp_OutputOverlapDuplicates. All records from datasource2 are output, except for the duplicate records.

Setting	Description
outputSettings->dedupedTable	The Overlap attribute specifies the name of the deduped table that will be produced. If the Overlap attribute is empty, then the name attribute will be used.
outputSettings->groupedMatchesTable	The Overlap attribute specifies that name of the matches_grouped table that will be required to generate the deduped table. If the Overlap attribute is empty, then the name attribute will be used.
dataSources	Specifies the datasource containing the specification for the source data that will be combined with the matches_grouped table to produce the deduped table.

7.5.9.1 Overlap Deduped table

This table contains the master records following the matching process. The structure of the table is determined by the fields mapped in the datasource (i.e. it contains an ID field, but also the fields that you have mapped in the datasource).

7.5.10 msp_TagOverlapMatchingResultsWithGroupLevel

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Level – can be Individual, Family, Household, Business or Custom.

This procedure renames any of the grouping related tables produced by the preceding four procedures and the msp_GroupOverlap Procedure. Each table is renamed by giving it a suffix matching the value passed in the level parameter.

Setting	Description
outputSettings->duplicates	Overlap attribute specifies the name of the duplicates table to be renamed. If the Overlap attribute is empty then the name attribute will be used.
outputSettings->dedupedTable	Overlap attribute specifies the name of the deduped table to be renamed. If the Overlap attribute is empty then the name attribute will be used.
outputSettings->matchingGroupsTable	Overlap attribute specifies the name of the matching_groups table to be renamed. If the Overlap attribute is empty then the name attribute will be used.
outputSettings->groupedMatchesTable	Overlap attribute specifies the name of the matches_grouped table to be renamed. If the Overlap attribute is empty then the name attribute will be used.
dataSources	Specifies the datasource containing the connection string to the database.

7.6 Triggers

7.6.1 msp_CreateTableTriggers

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.

Creates dynamic key update triggers on each table defined in the specified data source in the specified configuration file. This means that the keys table for the selected datasource will automatically be updated should records in the source data be modified in some way.

7.6.2 msp_DeleteTableTriggers

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.

Deletes dynamic key update triggers on each table defined in the specified data source in the specified configuration file.

7.6.3 msp_GenerateSingleKeys

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Record ID – unique ref of the record to update the keys for.
- Table Name – name of the keys table to be updated.
- Action – "update" or "delete"

This method is called by the key update triggers on tables defined in the data source. It updates, creates or deletes the keys entry(s) for the specified record in the specified data source.

7.7 Miscellaneous

7.7.1 msp_CreateCustomMatchesTable

WARNING: This stored procedure has been deprecated and will be removed in a future release of matchIT SQL. The stored procedure should no longer be used, and existing processes modified accordingly.

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Table name suffix – table in which to create the primary key column.

Creates a custom schema of a matches table from the standard matches table as defined in the configuration object for the specified data source. Relies on a matches table existing in the specified data source. The table name defined in the configuration can be given a suffix as the third parameter of the procedure if creating multiple instances to distinguish between them.

7.7.2 msp_CreateCustomGroupedMatchesTable

WARNING: This stored procedure has been deprecated and will be removed in a future release of matchIT SQL. The stored procedure should no longer be used and existing processes modified accordingly.

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Table name suffix – table in which to create the primary key column.

Creates a custom schema of a grouped matches table from the standard matches_grouped table as defined in the configuration file for the specified data source. Relies on a matches_grouped table existing in the specified data source. The table name defined in the configuration can be given a suffix as the third parameter of the procedure if creating multiple instances to distinguish between them.

7.7.3 msp_CreateUniqueRefField

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Table name – table in which to create the primary key column.
- Column name – name of the column to create.

Creates a primary key column with a specified name within the named table in the data source in the given datasource.

7.7.4 msp_SingleRecordMatch

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this procedure is run.
- Datasource ID – specifies the data source to be used within the configuration file, which contains database connection string and table/column mappings.
- Fullname
- Company
- Address1
- Town
- Region
- Postcode

This procedure finds all records that match a generated record made up from the parameters supplied as arguments for the procedure (Namely full name, company, address1, town, region, postcode), in the specified data source in the specified configuration.

The following results are returned:

Column	Description
RecordID	Reference ID for the matching record within the specified datasource.
Level	The Level indicates the matching level(s) at which a match was found. If it contains a 1 then the two records match at the Individual level; if 2, then Family level; if 4, then Household level; and if 8, then Business level. Multiple levels are indicated by summing values – for example, 9 would indicate a match at both Individual and Business levels (1+8), and 15 a match at all four levels (1+2+4+8). By default, the Level column is followed by the total score for the four matching levels. These columns are fully configurable within a configuration file. Component scores (for name, organisation, address, etc.) can also be output for any level(s).
IndividualScore	Individual level score.
FamilyScore	Family level score.
HouseholdScore	Household level score.
BusinessScore	Business level score.

7.7.5 mfn_SingleRecordMatch

This TVF (table-valued function) is identical to the msp_SingleRecordMatch stored procedure, except that its output is a temporary table that can be queried using a SELECT statement – for example, SELECT * FROM mfn_SingleRecordMatch(*arguments*).

The output is as msp_SingleRecordMatch – i.e. the total score for the four matching levels (see section 7.7.4 above).

7.7.6 mfn_SingleRecordMatch2

This TVF (table-valued function) is identical to the mfn_SingleRecordMatch TVF, except that its input is an XML query string.

The XML query string has the following general format:

```
<query>
  <inputs>
    <fieldName>value</fieldName>
    <fieldName>value</fieldName>
    <fieldName>value</fieldName>
    ...
  </inputs>
</query>
```

Where fieldname is any valid matchIT input field. For example:

```
<query>
  <inputs>
    <fullName>gill laughton</fullName>
    <organization>gretch ltd</organization>
    <address1>165 clock tower road</address1>
    <town>isleworth</town>
    <region>middlesex</region>
  </inputs>
</query>
```


7.7.7 mfn_SingleRecordMatchEx

As the mfn_SingleRecordMatch TVF, except that *all* available score columns are output – i.e. the total score plus all component scores for each level.

7.7.8 mfn_SingleRecordMatchEx2

As the mfn_SingleRecordMatchEx TVF, except that its input is an XML query string (see section 7.7.6 above).

7.7.9 mfn_SingleGenerateKeys

Input Parameters:

- Configuration file – the file path of the configuration file to be used when this function is run.
- Datasource ID – specifies the data source to be used within the configuration file.
- XML – specifies one or more data items for which to generate keys, in an XML-formatted string.

This function generates the keys only for the data passed into the function, and outputs a table containing a single row – for example, `SELECT * FROM dbo.mfn_SingleGenerateKeys(arguments)`.

The XML data must be specified in the format `<data attribute="value" ... />` where *attribute* must be one of the standard matchIT API field types:

fullName	jobTitle	postIn
prefix	address1-9	country
lastName	flatNo	deliveryPoint
firstNames	premise	telephone
initials	thoroughfare	fax
qualification	town	dateOfBirth
suffix	region	email
organization	postcode	customField1-9
department	postOut	

The XML data can contain any number of attribute=value pairs. Here's an example:

```
SELECT * FROM dbo.mfn_SingleGenerateKeys('config', 'datasource',  
'<data fullName="John Smith" organization="helpIT systems" />')
```

7.8 General Progress Logging Table

The following table is produced during processing which logs each matchIT SQL process that runs.

7.8.1 log

The log table is written to, by the following procedures:

- BulkGenerateKeys
- GenerateKeys
- FindMatches
- FindOverlap
- FindExactMatches
- FindExactOverlap
- OutputMatchingPairs
- OutputMatchingGroups
- OutputDedupedTable
- OutputDuplicates
- OutputOverlapMatchingPairs
- OutputOverlapMatchingGroups
- OutputOverlapDuplicates
- OutputOverlapDedupedTable

and contains progress information from these potentially long-running stored procedures.

Column	Description
ID	Reference ID for the matching group; this will reference a record in the second datasource that is considered to be the master record for the matching group.
Proc	Column identifies a running stored procedure.
Time	Time is the date and time of the log message.
Code	<p>The Code column indicates the type of XML-formatted value that is found in the Data column:</p> <p>10 – Started: the stored procedure has started.</p> <p>20 – KeyStarted: processing of a composite key has started (FindMatches/FindOverlap/FindExactMatches/FindExactOverlap).</p> <p>25 – RecordsSelected: the SELECT statement has completed.</p> <p>30 – PreClustered: pre-clustering has completed (FindMatches/FindOverlap).</p> <p>50 – Progress: indicates processing progress.</p> <p>70 – KeyFinished: the current composite key has finished (FindMatches/FindOverlap/FindExactMatches/FindExactOverlap).</p> <p>90 – Finished: the stored procedure has finished.</p> <p>(Note that future versions of matchIT SQL will likely log additional information.)</p>
Data	Progress message description.

8 Summary Reporting

matchIT SQL is able to produce a summary report on completion of certain stored processes. These reports can be output to a file using one of several available formats (note that the 'type' attribute is used within XML configuration files to specify the reports' format):

- PDF (Portable Document Format; type=pdf).
- RTF (Rich Text Format; type=rtf).
- Microsoft Excel (97-2003; type=xls).
- Microsoft Excel (97-2003; raw data only; type=xls2).
- HTML 3.2 (tables; type=html3).
- HTML 4.0 (type=html4).
- Crystal Report (type=rpt).

Additionally, report statistics are output to a new uniquely-named table within the data source.

Reports are produced for the following stored procedures. In most cases the collected data is either self-explanatory or uses standard helpIT terminology, unless detailed below:

- GenerateCorrectedAddresses.
- GenerateKeys – Note excluded records can be identified if the first character of a mkDataFlags value is an 'X'.
- BulkGenerateKeys - See GenerateKeys.

Note that you can use the setting outputSettings->reports in your configuration file to specify whether reports should be generated or not.

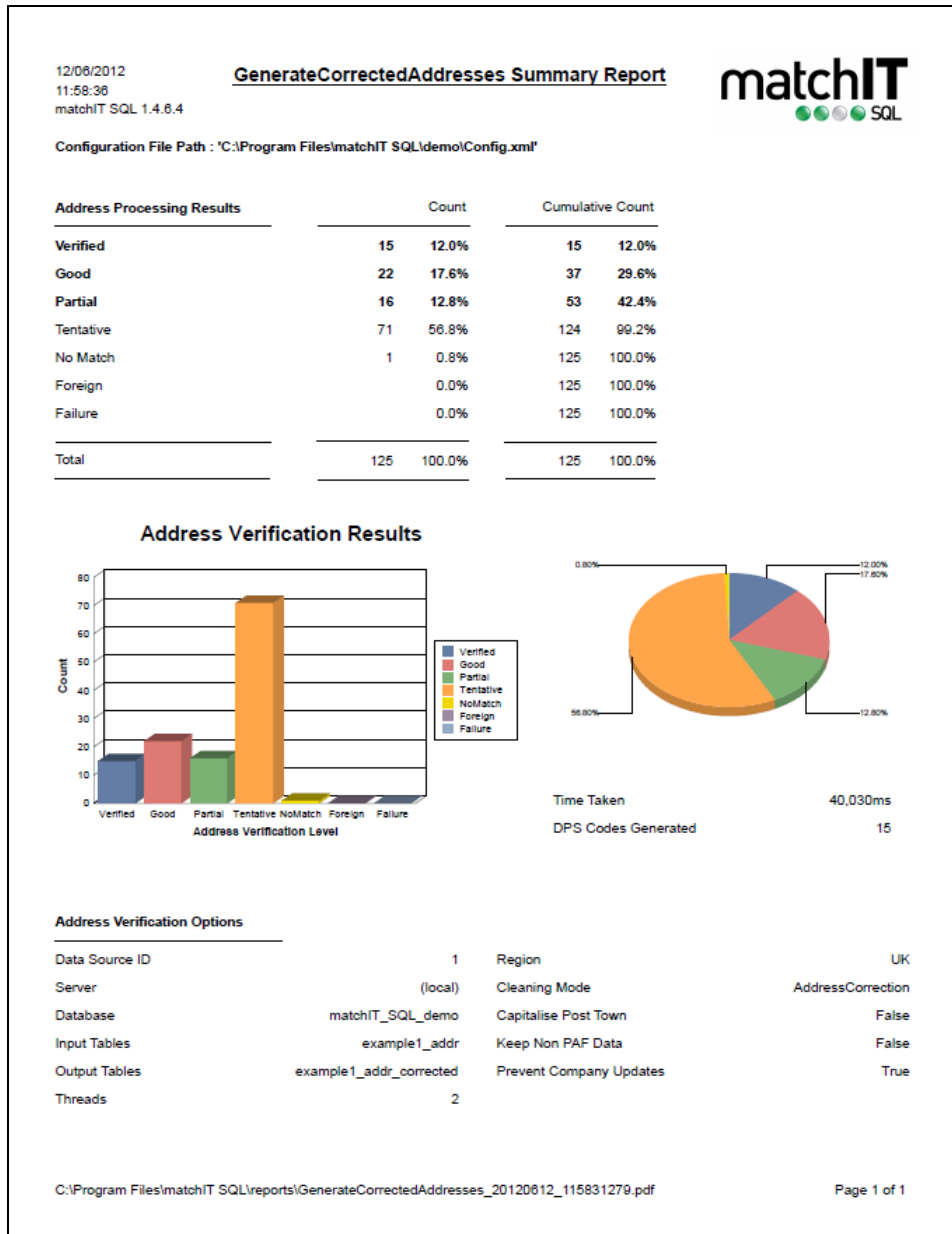
It is possible to incorporate a custom logo into the reports in place of the default matchIT SQL logo that displays in the top right of the first page of each report. To do this, simply save your logo as "ReportLogo.jpg" and place it in the matchIT SQL bin directory (which by default is C:\Program Files\matchIT SQL\bin). Once in this location, the report creation process will automatically detect and use the custom logo you have provided. With regards to the dimensions of the logo, to match the size of the default matchIT SQL logo you should create the logo on a canvas of 371px wide by 106px high.

As well as incorporating a custom logo into the report, it is also possible to include a website address and contact name (each of which appear in that order under the logo on the top right of the report). To do this, simply add a 'contactWebsite' and 'contactName' to the 'reports' node in the xml configuration file that is being called by the stored procedures (or the template being used in the case of SSIS). An example of this can be seen below. Note that the attribute names are case sensitive.

```
<reports enabled="true" path="C:\matchIT SQL\reports" format="pdf" schema="reports"
index="index" maxIndexEntries="1000" contactName="John Smith"
contactWebsite="www.domain.com">
```

8.1 GenerateCorrectedAddresses Summary Report

The GenerateCorrectedAddresses Summary report is produced at the end of the GenerateCorrectedAddresses process and will look similar to the following screenshot.



The report provides general figures reflecting the number of records falling into each verification level. These figures are also displayed in the form of a bar graph and a pie chart, to make general inspection easier to read and any problematic results easier to notice.

8.2 FindMatches Summary Report

The FindMatches Summary report is produced at the end of the FindMatches process and will look similar to the following screenshot.

14/07/2011 16:00:03 matchIT SQL 1.3.4		<u>FindMatches Summary Report</u>	matchIT ●●○○●SQL
Info	Config	C:\Program Files\matchIT SQL\demo\Config.xml	
Inputs	DataSource	1	
	Server	(local)	
	Database	matchIT_SQL_demo	
	Table	example1	
	Table	example1_addr	
	Table	example1_keys	
Outputs	Table	example1__exact__	
	Table	matches	
Key 1	Table	large_clusters	
	Columns	[mkPostOut] + [mkName1]	
	Total records	125	
	Records read	18	
	Comparisons	9	
	Duplicates	9	
	Clusters	9	
	Large clusters	0	
	Errors	0	
	Time taken	328ms	
Key 2	Columns	[mkName1] + [mkPhoneticStreet]	
	Total records	125	
	Records read	19	
	Comparisons	11	
	Duplicates	11	
	Clusters	9	
	Large clusters	0	
	Errors	0	
	Time taken	78ms	
Key 3	Columns	[Postcode]	
	Total records	125	
	Records read	49	
	Comparisons	144	
	Duplicates	110	
	Clusters	12	
	Large clusters	0	
	Errors	0	
	Time taken	328ms	

Totals	Total records	375
	Records read	86
	Comparisons	164
	Duplicates	130
	Clusters	30
	Large clusters	0
	Errors	0
	Time taken	1,031ms
Matches	Total matches	117
	Matches at individual level	13
	Matches at family level	18
	Matches at household level	93
	Matches at business level	0
	Matches at custom level	0

The FindMatches process produces the following information:

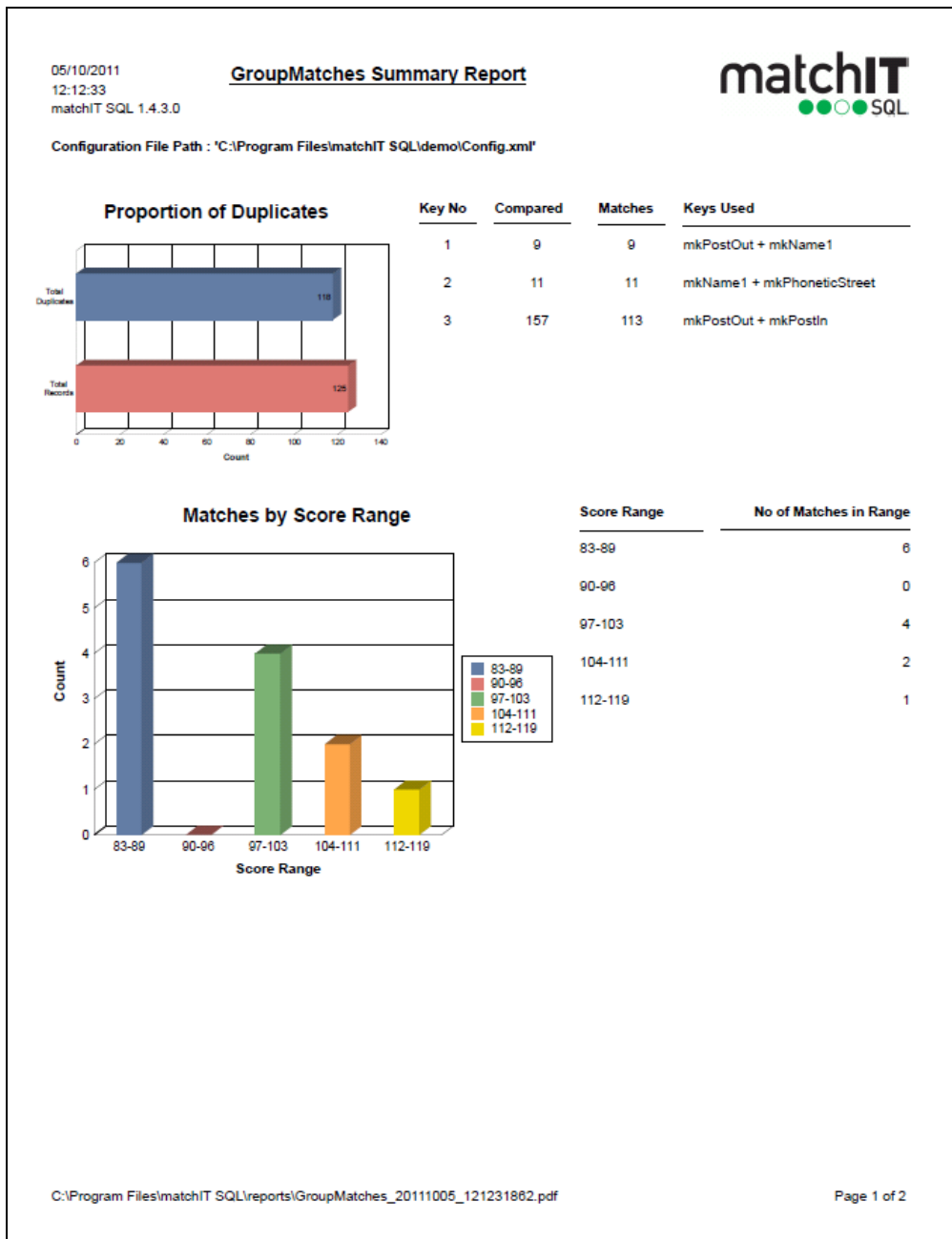
Reporting Category	Description
Total Records	'Total records' refers to both the total number of records in the table, for each key, and the sum of these.
Records Read	'Records read' refers to the number of records that are considered for further comparison by the matchIT API. If pre-clustering is enabled (the default) then this figure will usually be substantially lower than total number of records in the table.
Comparisons	'Comparisons' refers to the number of record pairs that are compared by the matchIT API.
Duplicates	'Duplicates' refers to both the number of duplicates found by a particular key and to the sum of these.
Matches	'Matches' refers to a final number of unique duplicates (for example, one particular match might be found using two different keys, causing a duplicate count of 2 but a correct match count of just 1).
Large clusters	'Large clusters' refers to groups (clusters) of potential matches that are too big to be processed (i.e. the number of records exceeds the maximum cluster size).
Errors	'Errors' refers to processing errors within the matchIT API; details of such records are logged (wherever possible, please forward these records to helpIT systems).

8.3 FindOverlap Summary Report

See FindMatches.

8.4 GroupMatches Summary Report

This report is produced at the end of the GroupMatches process and will look similar to the following screenshot.



Notes:

- Proportion of Duplicates – indicates the percentage of records that matchIT SQL has identified as being duplicated records.
- Matches by Score Range - indicates the number of matches in each scoring range, based on the minimum and maximum matching scores.

8.5 GroupOverlap Summary Report

See GroupMatches.

8.6 FindExactMatches Summary Report

See FindMatches. Note, however, that records are *not* compared using the matchIT API, two records are deemed exact matches simply if their composite key values are identical.

Notes:

- 'Records skipped' refers to records that have a blank composite key value; blank key components are indeed permitted, but they can't all be blank. (For example, if the composite key is 'mkNameKey+mkAddressKey+mkPostOut+mkPostIn', then a record will be skipped if *all* of these keys are blank; two records will be considered an exact match if any key is *not* blank and their composite key values are identical.)

8.7 FindExactOverlap Summary Report

See FindExactMatches.

8.8 GroupExactMatches Summary Report

See GroupMatches.

Note, however, that there is no 'scores' section (exact matches are not given a score, they either exactly match or they don't).

8.9 GroupExactOverlap Summary Report

See GroupExactMatches.

9 Address Correction

9.1 Installation

For matchIT SQL to be able to generate corrected addresses using the `msp_GenerateCorrectAddresses` stored procedure, a valid activation code must have been issued by helpIT systems. Use this activation code when installing matchIT SQL.

If matchIT SQL was previously installed without addressing support, then matchIT SQL will need to be reinstalled. Run Reactivate (Start Menu->Programs->matchIT SQL->Utilities) to apply the new activation code provided by helpIT, then re-run the matchIT SQL installer.

During installation, an Address Correction page will be shown. Ensure the option "Install addressing components" is checked to add addressing support. For UK addressing, there will be an additional option, Download data files that, if checked, will allow automatic download of the addressing data. For US and International addressing, helpIT will provide installation media that will install the data separately.

9.2 Setup

Additional configuration steps must be performed before the addressing stored procedure can be used (the following steps show how to configure addressing with the example database, `matchIT_SQL_demo`).

Firstly, the built-in Local Service account must be given write access to the database. This is because the address correction actually occurs outside the stored procedure; it is run under the context of the matchIT SQL Service which, by default, uses the Local Service account (a low privilege Windows service account).

Lastly, the account must be given the following roles for the `matchIT_SQL_demo` database: `db_datareader`, `db_datawriter`, and `db_ddladmin`. Alternatively, the account can be given the `db_owner` role instead of these three roles.

Note that the matchIT SQL Service can be changed to run under any user account, including domain accounts. In this case, that user account will instead require both the `bulkadmin` role and write access to the database.

Please refer to the matchIT SQL Installation and Deployment Guide for further information.

9.3 Usage

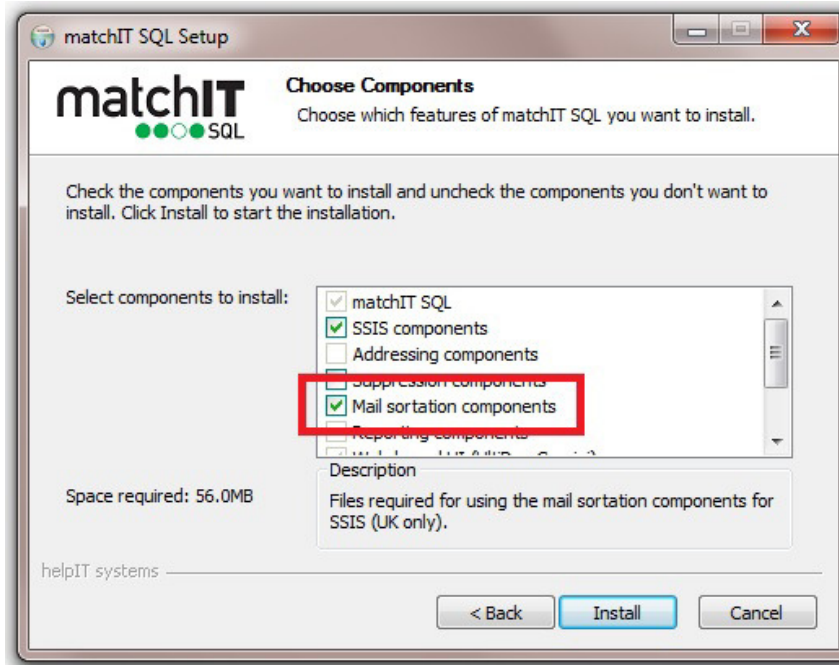
The stored procedure requires an appropriate Addressing section in an XML configuration file, which indicates the input columns to be used for address correction, plus the output table and columns that will be populated. To configure this, either modify a copy of the installed template configuration, or set up a new configuration via the matchIT SQL web-based UI; alternatively, configure a `GenerateCorrectedAddresses` task in an SSIS package.

Refer to `msp_GenerateCorrectedAddresses` for further details.

10 Mail Sortation

10.1 Installation

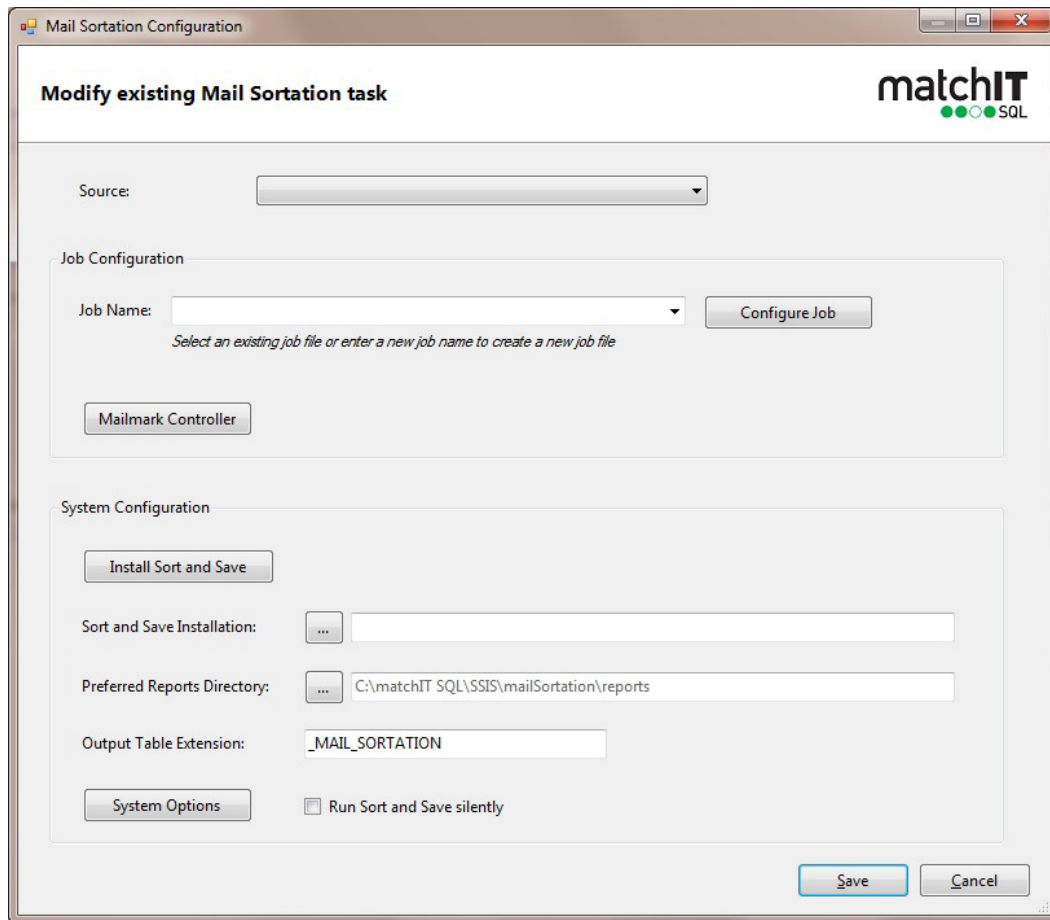
During installation of matchIT SQL, ensure that you have the Mail sortation components box checked. By checking this, the MailSortation SSIS task will be installed.



10.2 Usage

Mail Sortation is performed by separate software called Sort and Save, by BBS Ltd. This software is installed and configured via the matchIT SQL MailSortation SSIS task. When the task runs as part of your SSIS package, your SQL data is extracted and passed to Sort and Save, for sortation. Sort and Save will generate various Mail Sortation data fields as well as the reports that are required by the Mail Sortation carriers.

Upon opening the MailSortation task, you will see the following options. These must be configured before the task can be run as part of your SSIS package:



- **Source:** There must be a preceding task within your SSIS package, where the SQL source data table(s) are specified and the address and postcode fields are mapped. This can either be a **GenerateKeys** task or a **GenerateCorrectedAddresses** task. This Source dropdown menu is used to specify the preceding task.
- **Job Name:** Job files (text files with a .job extension) are used to store Mail Sortation configuration settings, E.g. item weight, mail format, delivery speed, etc. You can use the Job Name option either to select an existing job file or to create a new one.
 To create a new job file, simply type the name of the file into the Job Name text box (no file path required) and click on the Configure Job button. If you have not yet specified the location of your Sort and Save installation, you will be prompted to do so before your job file can be created.
- **Configure Job:** This will open the current job file in Sort and Save's Job Configuration mode. Here you can configure all options specific to the job, E.g. item weight, mail format, delivery speed, etc. You can also specify the carrier, E.g. Royal Mail or a Downstream Access carrier, as well as reporting and Mailmark options. See Advanced Automation for information on updating job files manually.
- **Mailmark Controller:** This will launch the Sort and Save Mailmark Controller. The Mailmark Controller gives you full control over any Mailmark jobs that have been run. These jobs can be uploaded, split, rescheduled or cancelled, as required.
- **Install Sort and Save:** This will launch the Sort and Save Installer, which will download and install the latest version of the Sort and Save software onto your PC.
- **Sort and Save Installation:** Here, you can map your installation of Sort and Save. This option is automatically populated when you use the Install Sort and Save button to perform the install. If you already have Sort and Save installed, you can click on the selection button to map this installation.

- **Preferred Reports Directory:** This option lets you choose where the generated Mail Sortation reports are copied to after each Mail Sortation job is run.
- **Output Table Extension:** When a Mail Sortation job is run, the sortation data, E.g. selection codes, barcodes, Mailmark barcodes, bag breaks, etc, are copied into a new SQL table known as the Mail Sortation output table. The output table is created in the same SQL database as the source table (the table mapped in the preceding GenerateKeys or GenerateCorrectedAddresses task), with the same name as the source table, but with an additional table name extension. This option lets you modify that extension, which by default is "_MAIL_SORTATION".
- **System Options:** This launches the Sort and Save System Options. Here, you can configure advanced settings for the available carriers as well as your Mailmark login details (required in order to perform a Mailmark sortation).
- **Run Sort and Save Silently:** Check this if you do not wish for Sort and Save to be displayed on screen when it is sorting your data.

10.3 Advanced Automation

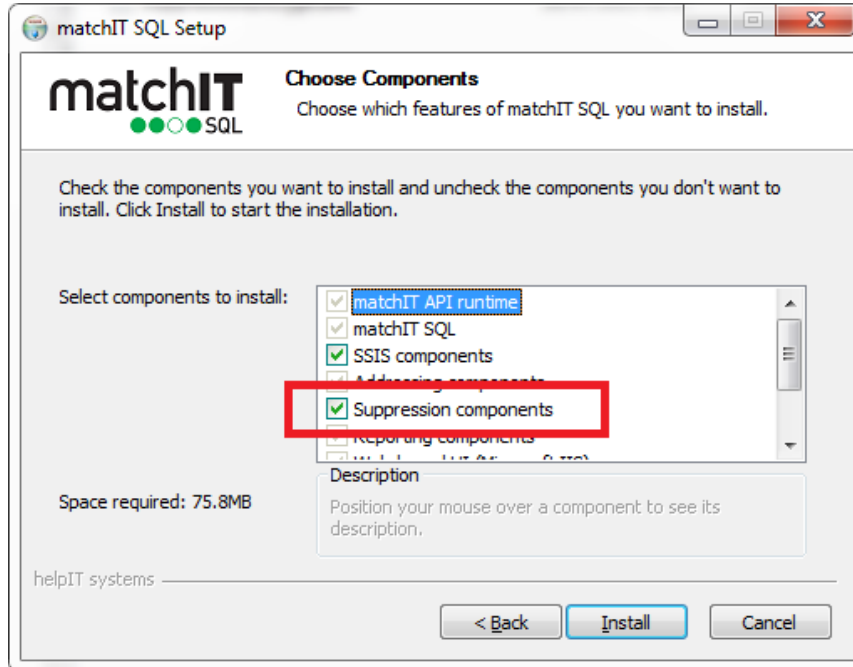
In some scenarios, particularly where lots of jobs with different requirements are being run in succession, you may find it beneficial to implement a system that dynamically configures a job file that is permanently specified in the SSIS task, before running your SSIS package. This would prevent you having to manually configure the job file between sortations, making the whole Mail Sortation process far more time efficient. For detailed information on the layout of job files, please refer to the Sort and Save documentation, located in the SYSTEM folder of your Sort and Save installation.

In addition to dynamically configuring the job file, you could potentially load the source data into a generic SQL table that is permanently mapped in the GenerateKeys or GenerateCorrectedAddresses task that precedes your Mail Sortation task. This would further reduce the need for manual configuration between jobs.

11 Suppression Processing

11.1 Installation

During installation of matchIT SQL, ensure that you have the suppression components box checked:



11.2 Utilising the SSIS package.

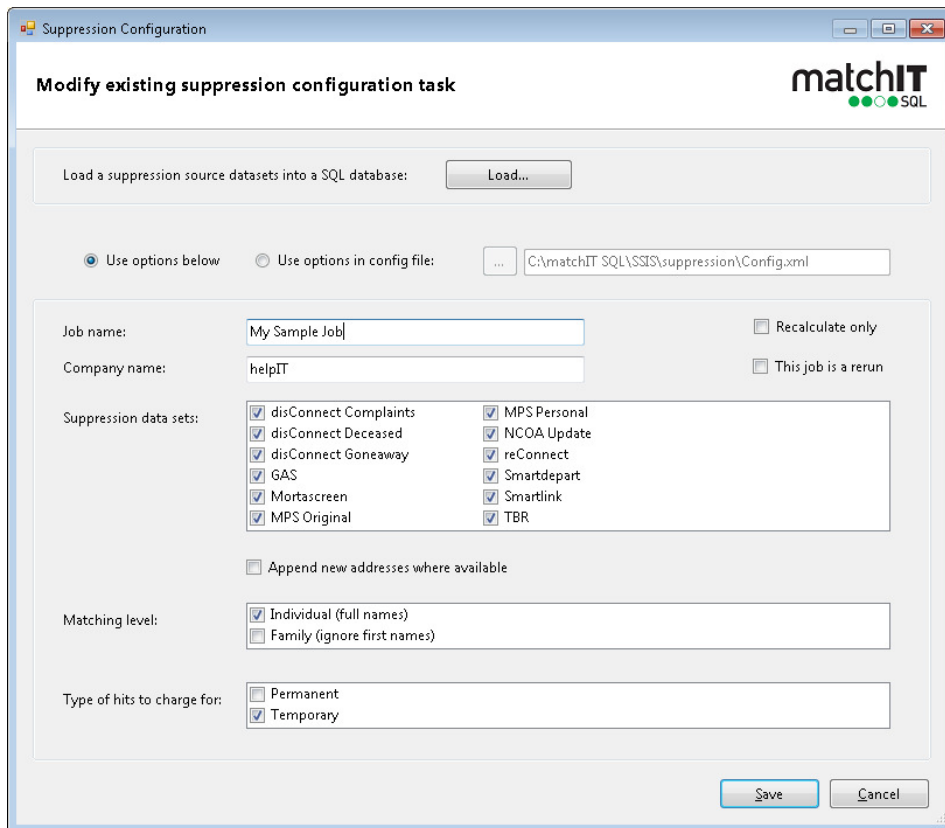
If you are enabled for suppression processing, matchIT SQL comes installed with a pre-configured suppression processing SSIS package. By default, you will find this here:

C:\matchIT SQL\SSIS\suppression

We recommend using the package as is, since you may break the reporting side if you try to customise it too much. If you must make changes though, please make a copy of the package and work using that, since otherwise future software updates will overwrite the initial package etc. We recommend copying the entire C:\matchIT SQL\SSIS\suppression folder when copying the package to avoid issues relating to task IDs. Note that the package was implemented in such a way that it's compatible from Visual Studio 2005 and later.

Within the package, you will see that the first task is called 'Suppression Configuration' – which is a special task that allows you to pre-configure your suppression process to use specific suppression datasets – either by selecting what you want from the checkbox options, or by loading from a configuration file. The latter option is there for those that want to call the process in an automated way, e.g. through the SQL Server Job Agent.

Opening up the Suppression Configuration task will display the following options:



- Every job that you run, should be given a job name to identify its processing results.
- Company name – enter your own company name or the name of the client that you are running a job for.
- Suppression data sets – select which of the standard suppression files you want to match your data against. **Note that if you want to perform an Equifax Goneaway suppression, then you will need to map both disConnect Goneaway and reConnect since for performance reasons we have split the Equifax records with COA references into a separate table; only use Append New Addresses if you actually want the data to be submitted to Equifax for reconnect processing.**
- Append new addresses where available – will automatically output a table called MISQL_newaddress_output containing new address lines for suppression datasets that have new addresses (e.g. Smartlink). Equifax new addresslines are output to a special table called DisConnectCOA__newaddress_output due to needing to be sent to Equifax for additional processing.
- Matching Level – selects whether to match at individual or family (surname only) against the suppression datasets
- Type of hits to charge for – select Permanent if you or your client will be updating a database based on the results (i.e. keeping a permanent record of the output), or temporary if this is for a single mailing and no database will be updated. Note that hits against new address databases where you have opted to output the new addresses will always be charged based on the permanent costs.
- Recalculate only – only run the result hit tables are re-generated; the matching is not repeated – see 11.2.6.

11.2.1 Suppression Priorities

Within your matchIT_SQL_suppression database, you will see that there is a table called MISQL_SuppressionPriorities which contains a list of priorities and costs for each suppression dataset. These are used to provide a summary of the cost you will incur when you utilise the suppression datasets. Note that prices change, so this table will only be an indication and should not be relied on for accurate pricing unless you are certain that the costs are accurate.

Field	Description
suppressiontype	maps directly to the container names for the suppression datasets in the SSIS package
Priority	1 = highest priority
TempCost	Cost of each temporary hit
PermCost	Cost of each permanent hit
ResultTableName	Maps to the output from the GroupOverlap task in the suppression container within the SSIS package
HasNewAddresses	indicates if the suppression dataset has new addresses for movers
SuppressionTableName	Name of the table containing the suppression data
NewAddressTableOutput	special for Equifax coaref output
isEquifax	special for Equifax datasets that need new address data to be uploaded to Equifax.
helpITLog	'1' indicates that hits against this suppression dataset will be output in the log file for upload to helpIT systems support.

If you need to extend the SSIS package to handle additional suppression datasets, then you will need to adding additional records in this table corresponding to the new datasets – if you don't then the reporting process within the package will break.

11.2.2 Using the options config.

The default config file is C:\matchIT SQL\SSIS\suppression\config.xml and contains the following XML which is self explanatory:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

```
<GeneralSettings>
```

```
  <JobName></JobName><!-- Enter the name of the suppression job -->
```

```
  <CompanyName></CompanyName><!-- Enter the name of your company -->
```

```
  <Rerun>>false</Rerun><!-- Set to true if this job is a rerun, otherwise set to false -->
```

```
  <RecalcOnly>>false</RecalcOnly><!-- Set to true if you wish to recalculate the costs with different options (e.g. different hit type or with fewer suppression files in use), without re-running the entire suppression step, otherwise set to false -->
```

```
  <SuppressionFiles></SuppressionFiles><!-- List the suppression data sets required for this job, separated with commas e.g. GAS, TBR, disconnect Complaints etc-->
```

```
  <IncludeNewAddresses>>false</IncludeNewAddresses><!-- Set to true to include new addresses when suppressing against goneaway datasets that offer new addresses, otherwise set to false -->
```

```
  <MatchingLevel>Individual</MatchingLevel><!-- Set to either Individual or Family -->
```

```
  <PermOrTemp>Perm</PermOrTemp><!-- Set to Perm if suppressions are to be permanently removed, otherwise set to Temp -->
```

</GeneralSettings>

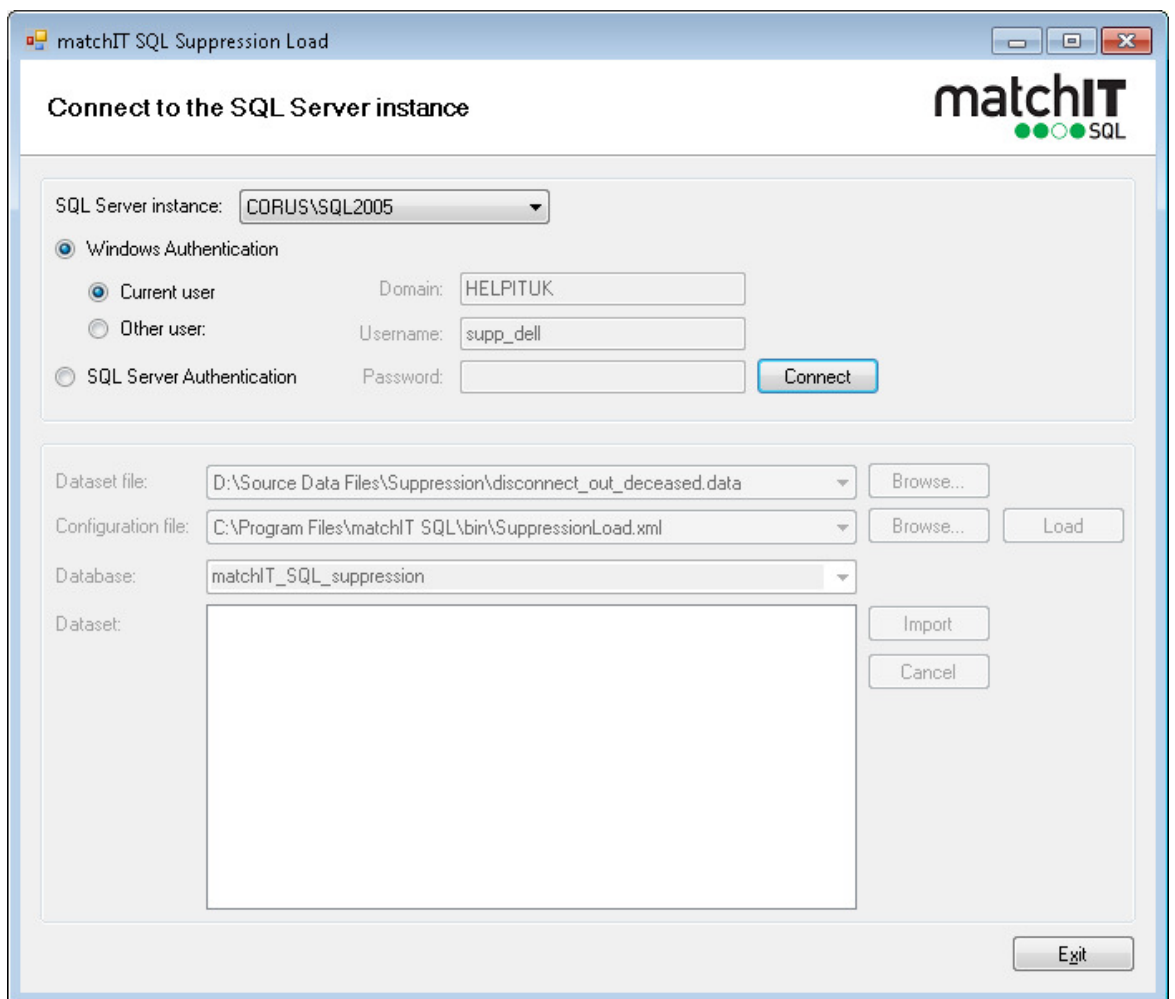
Note that the items listed in the SuppressionFiles section of the XML should map to the container names in the SSIS package for each suppression file. There is a property set for the GenerateKeys task in each Suppression File container called IsSuppressionTask that indicates to the Suppression Configuration task to list it as an option.

This means you can add your own containers – as long as you also ensure that your customer suppression datasets are also listed in the MISQL_SuppressionPriorities table in the suppression database.

Note that if you leave any settings blank, then you will get an error when trying to load the configuration file.

11.2.3 Suppression Loader

From the Suppression Configuration task, you can launch the Suppression File Loader by clicking on the Load button:



This screen allows you to easily load any of the suppression files that you are receiving from helpIT systems. You can also extend this to work with suppression files that you source directly by adding to C:\Program Files\matchIT SQL\bin\SuppressionLoad.xml

To use the loader:

1. Simply specify the database connection details.
2. Point the dataset file setting at the source data that you need to load.

3. Load the available suppression datasets by clicking load on the selected configuration file.
4. Choose the database to load the files into. Note that if you change this, then you will need to update the other Connection Managers within the SSIS package.
5. Select the dataset that you wish to load the source data into.

The Loader will automatically load the datafile into your SQL Server Database and generate the required matchIT SQL match keys at the same time.

If you are not using one or more of the suppression containers within the package (because you do not source that data), then we strongly recommend disabling those containers in the package.

11.2.4 Select the data to be suppressed

Within the suppression package is a task labelled 'Source Data' – this is where you map your input table that you wish to match against the suppression datasets that you have loaded.

If you are creating an automated process, then we recommend creating a generic table structure and map this here. Then import records into this table prior to calling the suppression package with the appropriate configuration settings for the job that you wish to execute.

11.2.5 Suppression Output

The suppression package produces 3 output tables at the end of processing:

1. MISQL_MergedHits – this is a work table produced during the suppression matching process containing every hit; this is then used to populate MISQL_PrioritisedHits.
2. MISQL_PrioritisedHits – this lists any of the reference IDs from your source data that have matched against a suppression file. Where a record matched multiple suppression file, only the highest priority (i.e. lowest priority value) is recorded here. Additionally the suppression reference, type and hit cost are recorded.
3. MISQL_SuppressionLog_Month– this table contains summary level information for each job processed in the last month. At the start of a new month, a new table is generated. The original tables are overwritten after 12 months. For example, the log table for August will be called MISQL_SuppressionLog_Aug.

Additionally a log file is produced for any hits that need to be communicated back to helpIT systems support. By default the file is output to the following directory:

c:\matchIT SQL\Suppressions

if you install matchIT SQL to a non default directory, then you will need to amend the SSIS package variable to contain the new path of where this log file should get written.

11.2.6 Re-generating Output without Re-running suppressions

If you have run data through against all of your installed suppression data, but then wish to produce the results against a subset (but without re-running all of the processing), then this can be achieved as follows:

Within the Suppression Configuration task – or its config file, choose a subset of the suppression files you used previously. Set the Recalculate Only setting on. Now when the package processes, as long as you did originally generate results for the suppression datasets you now have selected, then the results will be re-generated without repeating the processing.

11.2.7 Cancelling a job from the log

If you run a suppression job, but decide to not proceed with it, then you will need to cancel the job from the log, or you could be charged for the hits. To cancel the job, simply set the following variables in the CancelJob SSIS package and run the package.

- JobName – Unique name of the job that you need to cancel.
- LogTable – log table containing the job results e.g. MISQL_SuppressionLog_Oct

11.2.8 Hardware Recommendations

If you are working with numerous suppression datasets, then it's strongly recommended that you have suitable hardware to get the most out of the suppression package.

For reference, running through 80,000 records against 11 suppression datasets (including 3 large goneaway datasets making around 300,000 million suppression records in total) takes around 13 minutes on a 20 core machine, 64Gb RAM and a dedicated REVO drive.

You may find that running too many suppression files at once on lesser spec machines will slow processing.

12 SSIS Tasks

12.1 Installation

The SSIS Tasks are installed with matchIT SQL by default. During installation matchIT SQL detects which version(s) of SQL server you have installed, and, providing SQL Server Integration Services are installed, registers the appropriate assemblies to the Global Assembly Cache, as well as copying them to the DTS folder of the installation. Note that all of the following SSIS tasks use the same file (Template.xml located in the SSIS sub folder within the matchIT SQL install folder) as their base for creating XML configuration files for their processes. This template contains the paths to the matching matrices and master priorities files that are used in the processes, which can be manually amended if needed to point to different files. As well as the Tasks, a demo SSIS package is installed to the demo folder in matchIT SQL, under an SSIS sub folder. There are versions for SQL 2005, SQL 2008 and SQL 2012, all of which are configured to work with the demo database provided with matchIT SQL. Once the demo database is installed and the setup step below for the tasks has been completed, the packages should be able to run out of the box – The only things that may need amending are references to connection strings, which can be done quite easily in BIDS. If the connection strings do need to be amended, you may also need to check that the mappings in the Generate Keys components are persisted.

12.2 Setup

Before the tasks can be used in BIDS or Visual Studio, they need to be added to the toolbox in the control flow pane (Note this is not applicable to BIDS / VS 2010). You can do this by right-clicking on the toolbox and selecting 'Choose Items...'. The 'Choose Toolbox Items' dialog will appear after a while, in which you need to select the 'SSIS Control Flow Items' tab. In this tab you should see the matchIT SQL tasks currently unselected (they are identifiable by the fact that they are named using the prefix 'MISQL.'). Simply select the tasks by checking the check box next to each one, and click 'OK'.

12.3 Usage

Below is a brief description of each task and what it does. For a visual representation of some typical SSIS processes and some of the tasks themselves, please see Appendix A.

MISQL.GenerateCorrectedAddresses – This task should be the first task in a sequence if it is to be used at all. Its function is to generate corrected addresses from a given source and output them in a specified output format. This task basically wraps the same core functionality as the procedure msp_GenerateCorrectedAddresses. The specified output table in this task is persisted in a variable at the SSIS package level at design time so that it can be picked up and used in the next task.

MISQL.GenerateNCOAAddresses – This task will usually follow the GenerateCorrectedAddresses task, but can also be run standalone or even following the GenerateKeys task to allow for maximum flexibility. Its purpose is to identify individuals, families, and companies that have changed their address (i.e. moved), to help keep a database up to date. Please see section 7.1.2 for further information on the msp_GenerateNCOAAddresses stored procedure, which provides the same functionality as this task. Note that this service is currently only available for US addresses, to licensed users only.

MISQL.GenerateKeys – This task should be the first task in a sequence unless any of the previous tasks are also being used. This task generates the match keys required to be used in the matching tasks from the specified source data. Mappings are made from source columns to their relevant matchIT API field. This task encapsulates the same core functionality as msp_CreateKeysTable and msp_BulkGenerateKeys. By setting up a GenerateKeys task, you are effectively setting up a 'Data Source' with an ID, which is used as a reference in following matching and grouping tasks.

MISQL.FindMatches – This task should follow on from a GenerateKeys task. It is used to set up and execute fuzzy matching based on specified match keys and minimum score thresholds, and is pointed at a data source set up by a GenerateKeys task. The core functionality used by this task is the same as the procedure msp_FindMatches.

MISQL.GroupMatches – This task should follow on from a FindMatches task. It is used to group the results produced by a FindMatches task. Different types of tables can be selected to be output and the names of the tables can be specified. The core functionality of this task encapsulates the procedures msp_GroupMatches, msp_OutputMatchingPairs, msp_OutputMatchingGroups, msp_OutputDuplicates and msp_OutputDedupedTable.

MISQL.FindExactMatches – This task should again follow on from a GenerateKeys task. It is basically the same as the FindMatches task, only instead of fuzzy matching, exact matching is applied using the keys specified. The core functionality used is the same as msp_FindExactMatches.

MISQL.GroupExactMatches – This task should follow on from a FindExactMatches task. It is used to group the results produced by a FindExactMatches task. The core functionality is the same as the procedure msp_GroupExactMatches.

MISQL.FindOverlap – This task should follow on from two GenerateKeys tasks. It is used to set up and execute fuzzy overlapping between the two data sources set up in the preceding GenerateKeys tasks. The match keys and score thresholds to be used can be configured. The core functionality used by this task is the same as the procedure msp_FindOverlap.

MISQL.GroupOverlap – This task should follow on from a FindOverlap task. It is used to group the results produced by a FindOverlap task. Different types of tables can be selected to be output and the names of the tables can be specified. The core functionality of this task encapsulates the procedures msp_GroupOverlap, msp_OutputOverlapMatchingPairs, msp_OutputOverlapMatchingGroups, msp_OutputOverlapDuplicates and msp_OutputOverlapDedupedTable.

MISQL.FindExactOverlap – This task is basically the same as the FindOverlap task, only instead of fuzzy matching, exact matching is applied using the keys specified. The core functionality used is the same as msp_FindExactOverlap.

MISQL.GroupExactOverlap – This task should follow on from a FindExactOverlap task. It is used to group the results produced by a FindExactOverlap task. The core functionality is the same as the procedure msp_GroupExactOverlap.

13 Troubleshooting

This section describes common error messages that you may encounter while getting your SQL scripts and matchIT SQL Stored Procedures working.

13.1 A .NET Framework error occurred during execution of user-defined routine or aggregate "msp_GenerateKeys":

Msg 6522, Level 16, State 1, Procedure msp_GenerateKeys, Line 0

A .NET Framework error occurred during execution of user-defined routine or aggregate "msp_GenerateKeys":

System.Data.DBConcurrencyException: Concurrency violation: the UpdateCommand affected 0 of the expected 1 records.

13.1.1 Possible Cause:

The keys table exists and is not empty, and msp_GenerateKeys previously failed.

Subsequently running msp_GenerateKeys means that, because the table isn't empty, SQL Update commands are used to write data to the key fields rather than an Insert command (if the table was empty). But the exception is thrown when trying to update a row that doesn't exist.

To get around this, delete the keys table then re-run the key generation. Also, please consider using msp_BulkGenerateKeys, as that stored procedure is much less likely to leave the table in an incomplete state.

13.2 A .NET Framework error occurred during execution of user-defined routine or aggregate msp_GenerateCorrectedAddresses

Msg 6522, Level 16, State 1, Procedure msp_GenerateCorrectedAddresses, Line 0

A .NET Framework error occurred during execution of user-defined routine or aggregate "msp_GenerateCorrectedAddresses":

System.Exception: Failure in Addressing module; exit code was 1. Please check the matchIT SQL temp directory for an error log file.

13.2.1 Possible Cause:

An error occurred during execution of the external Addressing Process.

Any errors that occur in this process are written to the progress log file that you defined in the xml configuration through the UI. Look in the progress log file for a description of what went wrong.

Note that the referenced error log file might contain this error:

- Login failed for user 'NT AUTHORITY\LOCAL SERVICE'

If this is the case, it'll be necessary to give the Local Service account read-write access to the database being processed via this stored procedure. (This is because the Addressing runs under the context of the matchIT SQL Service – outside SQL Server – and requires direct access to the database. By default, the Service runs using the Local Service account.) Additionally, the account will need to be given the 'bulk admin' server role.

Please refer to the Address Correction section for further information.

13.3 Column '<column>' has invalid type: <type>

Msg 6522, Level 16, State 1, Procedure msp_OutputDuplicates, Line 0

A .NET Framework error occurred during execution of user-defined routine or aggregate "msp_OutputDuplicates":

core.DataException: Column '<column>' has invalid type: <type>

13.3.1 Cause:

matchIT SQL currently provides support for the following column data types: bit, tinyint, smallint, int, bigint, char, varchar, nchar, nvarchar, datetime, xml, uniqueidentifier, float, real, decimal, numeric.

Other data types (such as money, text, and date) can still be mapped in XML configuration files, but they will cause this exception in stored procedures such as msp_OutputDuplicates and msp_OutputDedupedTable. Please contact helpIT systems if you encounter such an exception.

Appendix A – SSIS Screenshots

The SSIS tasks have been designed to be intuitive and simple to use, however we have included screenshots below of some typical processes one would run with the tasks, as well as some screenshots of the GenerateCorrectedAddresses and GenerateKeys tasks.

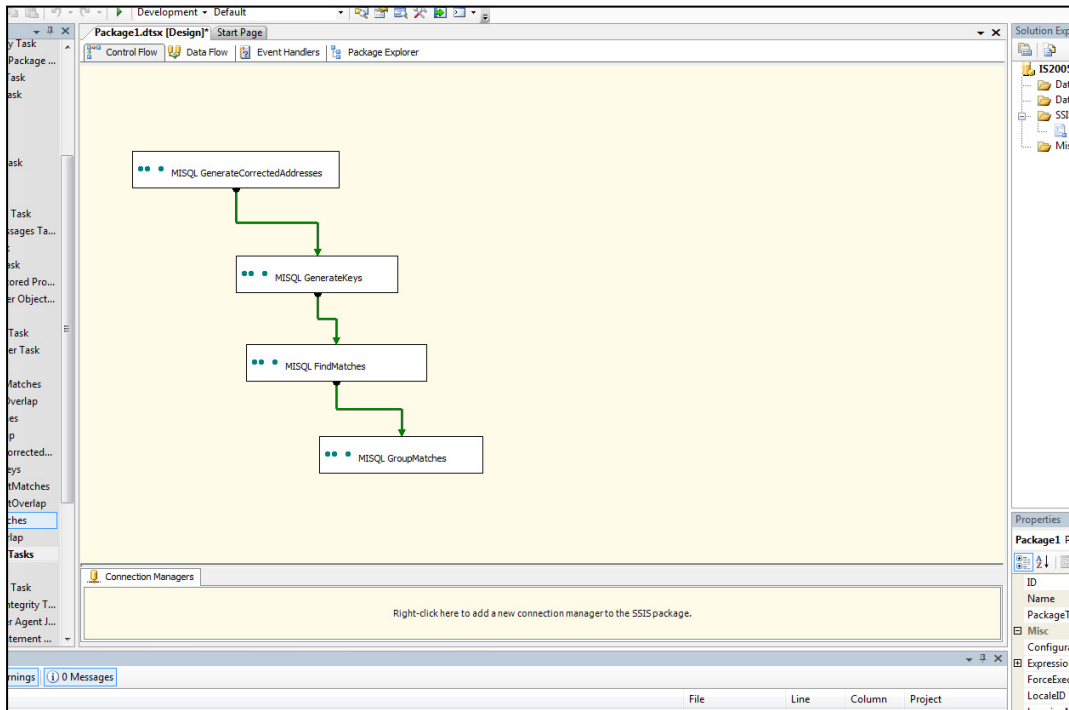


Figure 1 – Matching Process

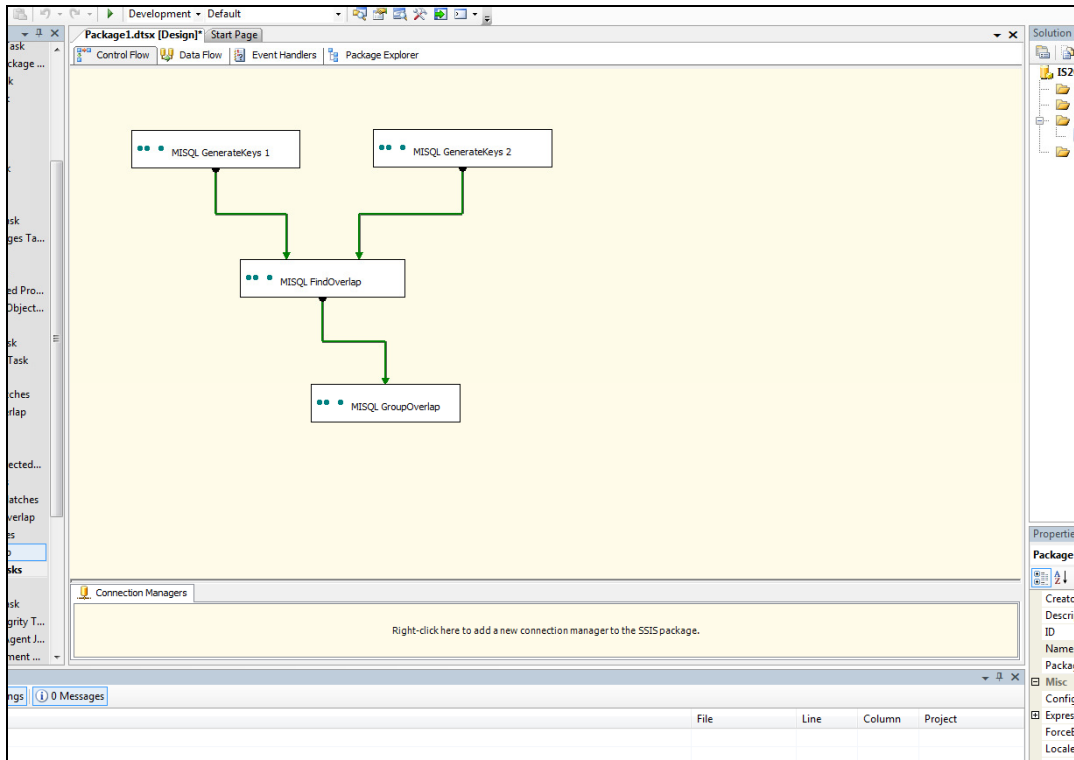
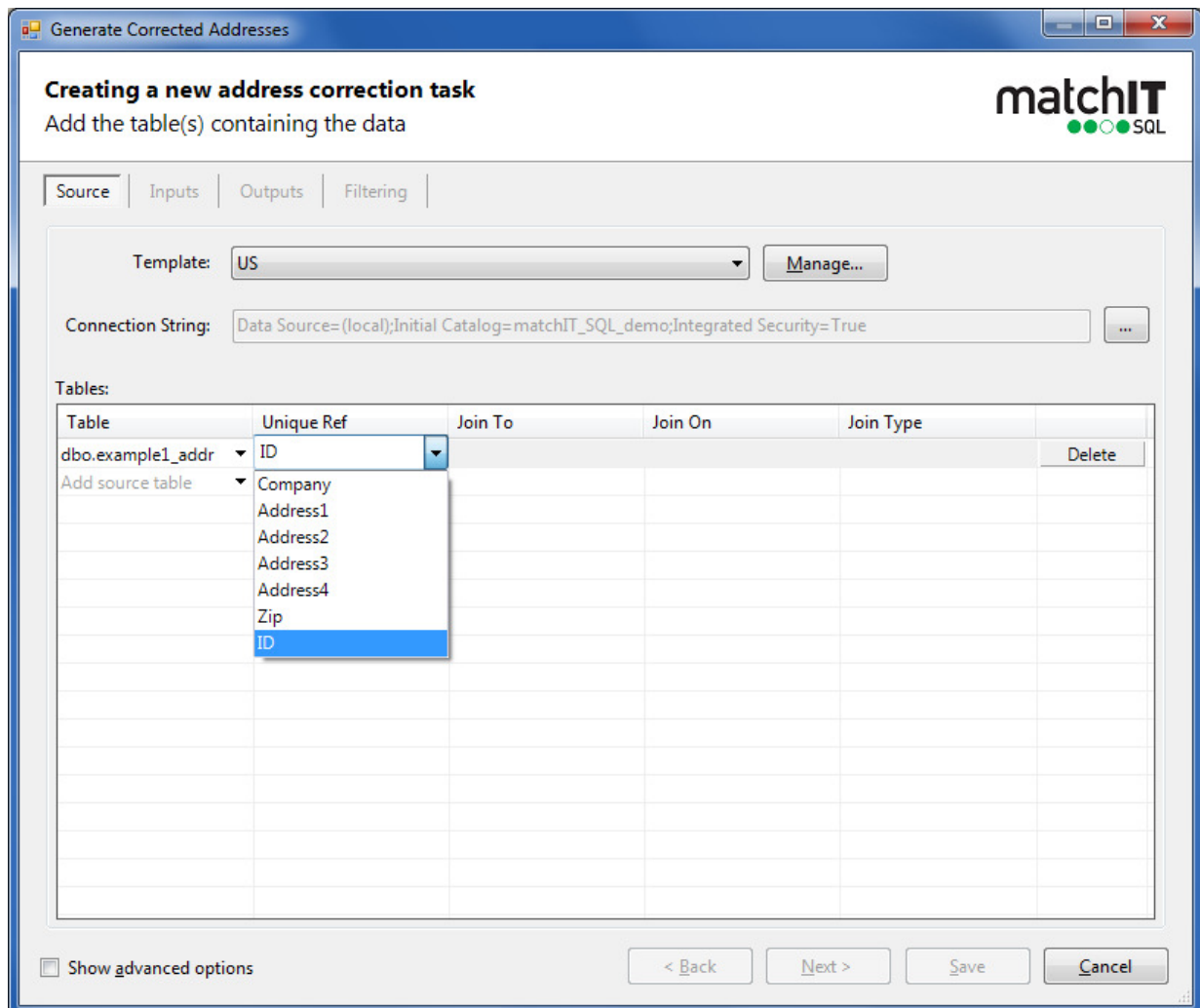


Figure 2 – Overlap Process

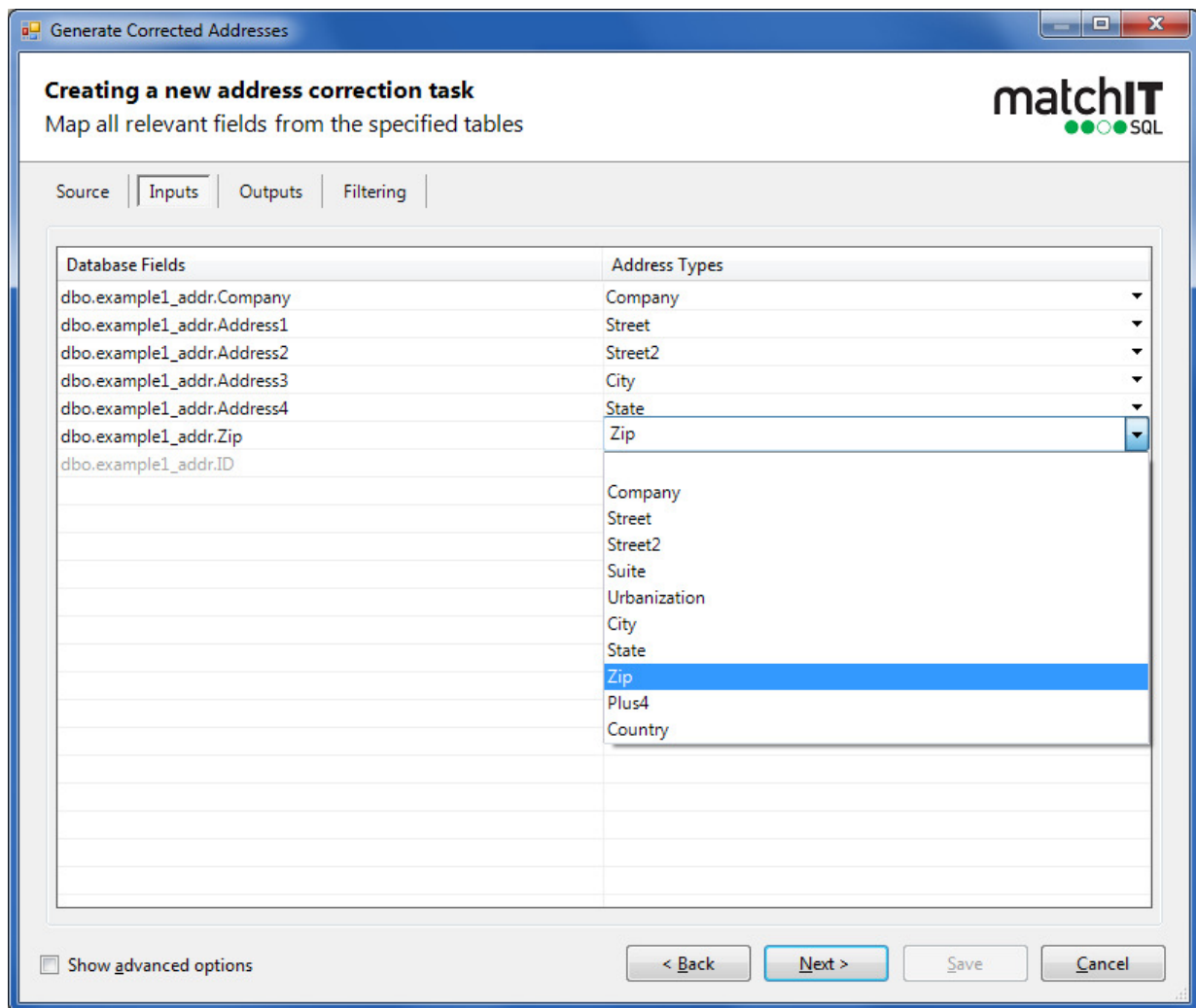
Generate Corrected Addresses Task

Source Tables



This tab is used to specify the source data tables. It is possible to add and join across multiple tables, as well as setting conditions for the joins / select statement.

Input Mappings



This tab is used to map specific fields in your source data to their corresponding address types for the input.

Output Mappings

Generate Corrected Addresses

Creating a new address correction task
The task can now be saved, or further configured

matchIT SQL

Source | Inputs | **Outputs** | Filtering

Name of output table: Default (dbo.example1_addr_addr_)

Columns:

Output	Source	Name	
Company	Company	Company_corrected	Delete
StreetOnly	Street	StreetOnly_corrected	Delete
Suite		Suite_corrected	Delete
Street2	Street2	Street2_corrected	Delete
Urbanization		Urbanization_corrected	Delete
City	City	City_corrected	Delete
State	State	State_corrected	Delete
ZipPlus4	Zip	ZipPlus4_corrected	Delete
Alternates		Alternates_corrected	Delete
Leftovers		Leftovers_corrected	Delete
Add output column			

This specifies the data item from the corrected address that's written to the corrected addresses table.

Show advanced options

< Back Next > Save Cancel

This tab is used for defining the structure and content of the corrected output table. Here, you map the corrected values to the relevant columns, as well as specifying the default values from the input address that should be populated in the output columns, should an address fail to be corrected.

API Settings

The screenshot shows a software window titled "Generate Corrected Addresses" with a "matchIT SQL" logo in the top right. The main heading is "Creating a new address correction task" with a sub-message: "The task can now be saved, or further configured". Below this are navigation tabs: "Source", "Inputs", "Outputs", "Filtering", and "Advanced" (which is selected). The "Advanced" tab contains a table of settings:

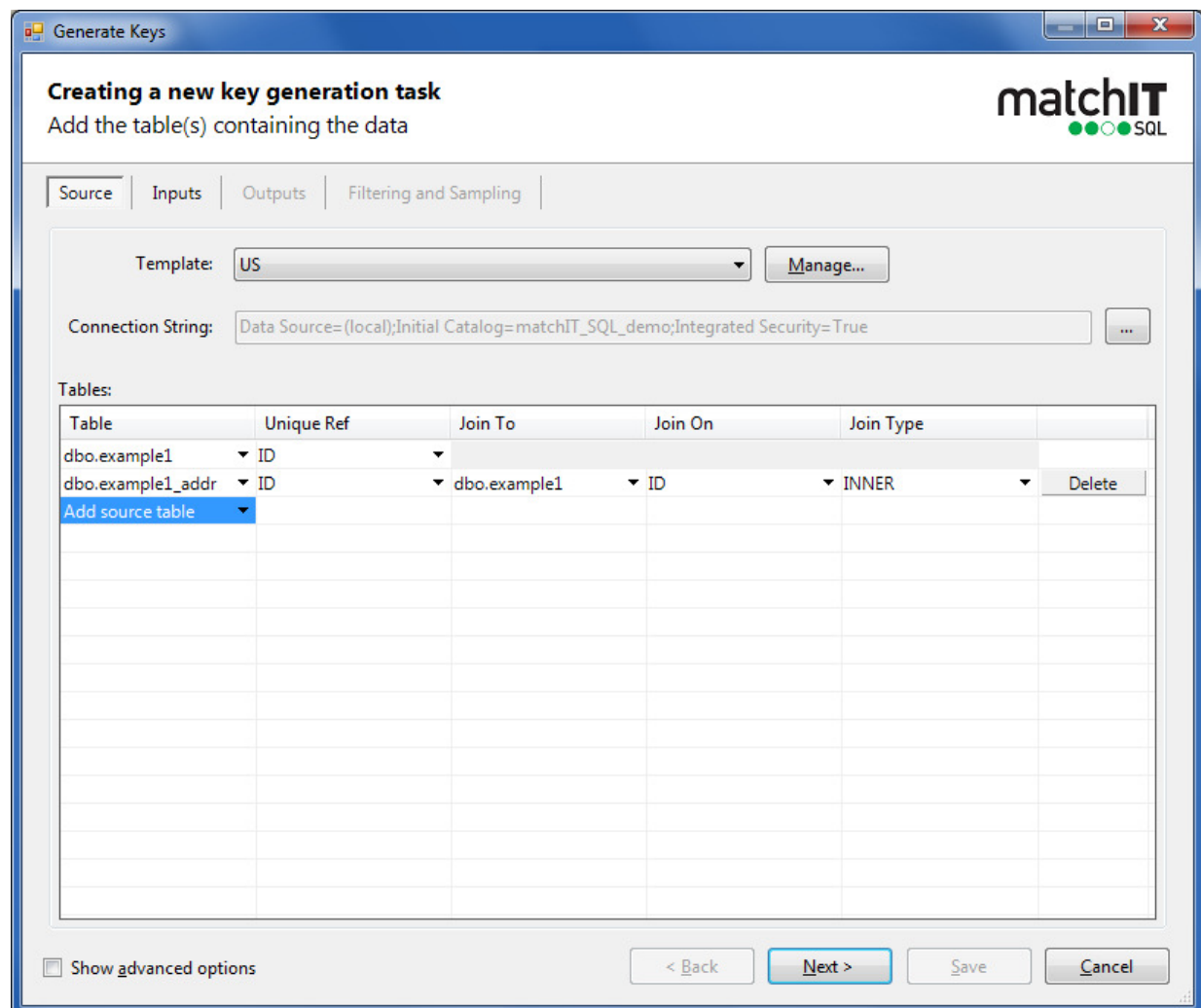
Name	Value
Progress log file path	
Progress interval	5000
US Addressing Settings	
Processing	PreferBox
Use mixed	yes
Use alias	no
International Addressing Settings	
Output casing	Title
ConfidenceThreshold	90
MinimumMatchscore	0
MinimumPostcode	0
MinimumVerificationLevel	4

Below the table, there is a text prompt: "Click a setting to view its description." At the bottom left, there is a checked checkbox labeled "Show advanced options". At the bottom right, there are four buttons: "< Back", "Next >", "Save", and "Cancel".

This tab is used to set the API-specific settings for the process.

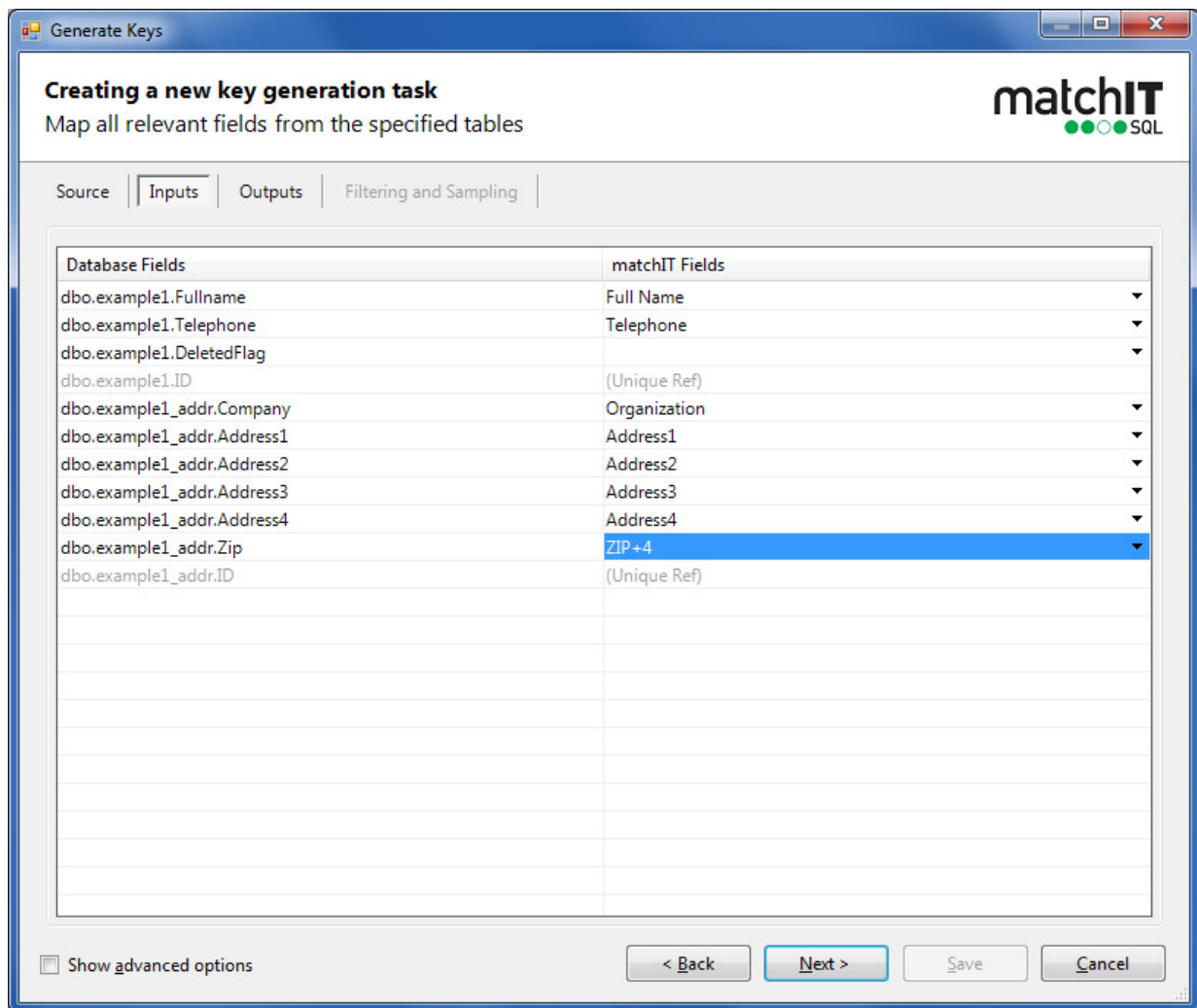
Generate Keys Task

Source Tables



This tab is used to set up the connection to the data source and define any source tables that contain the data to be de-duped. It is possible to add and join across multiple tables, as well as setting conditions for the joins / select statement.

Input Mappings



This tab is used to set mappings between the source data and matchIT Record fields

Output Settings

The screenshot shows a software window titled "Generate Keys" with a blue border. The main content area is titled "Creating a new key generation task" and includes the matchIT logo and the text "The task can now be saved, or further configured". Below this is a navigation bar with four tabs: "Source", "Inputs", "Outputs" (which is selected), and "Filtering and Sampling".

Under the "Outputs" tab, there are several configuration options:

- "Name of keys table:" with a dropdown menu set to "Default (dbo.example1_keys_)" and an empty text input field below it.
- A checked checkbox labeled "Overwrite existing keys table".
- A checked checkbox labeled "Output cleaned data".
- "Name of output table:" with a dropdown menu set to "Default (dbo.example1_output_)" and an empty text input field below it.

Below these options, there is explanatory text: "The cleaned data table will consist of cleaned, normalized and, optionally, cased data." followed by "Examples of output columns include:" and a list of examples:

- fullnames split into title, first name, last name, and suffix;
- streets, cities, regions, and postcodes extracted from address lines;
- email addresses split into username and domain;
- quality scores calculated for names, addresses, and emails.

At the bottom left, there is a checkbox labeled "Show advanced options". At the bottom right, there are four buttons: "< Back", "Next >", "Save", and "Cancel".

This tab is used to set the output settings for the data source, such as the table name and whether to overwrite the keys if they have already exist.

Filtering and Sampling Settings

Generate Keys

Creating a new key generation task
The task can now be saved, or further configured

matchIT
SQL

Source | Inputs | Outputs | **Filtering and Sampling** | Advanced

Filtering

Table	Type	Column	Condition	Value	Data Type
Add filter					

Sampling

Use this section to define the sampling settings. Hover over the labels below for more information on each setting.

Enabled

Overwrite Existing Sample

Type Percentage N in M Range Random N Max From Top

Percentage

N Value M Value

Range Field Lower Upper

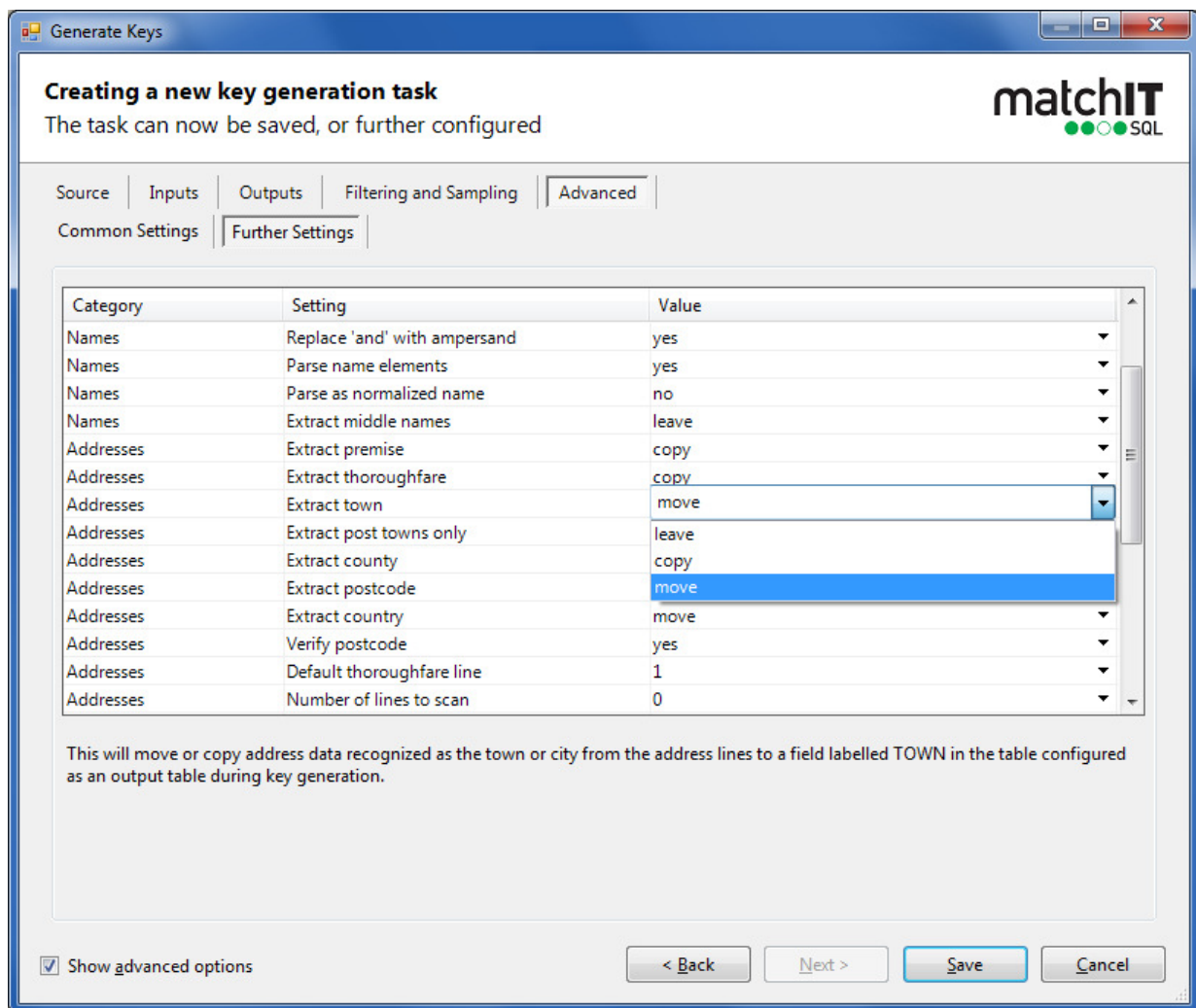
Limit

Show advanced options

< Back Next > Save Cancel

This tab is used to set the output settings for the data source, such as the table name and whether to overwrite the keys if they have already exist.

Advanced Settings



This tab is used to set API settings for the Record / Key generation process.

Appendix B - Key Generation Flag field

This field is generated during key generation in the table you have defined as the keys table. The following values may be produced dependent upon your data:

13.3.1.1 Data Flags Usage

<i>Position</i>	<i>DataFlags Usage</i>	<i>Values</i>
1	Exclusion Status	X if exclusion words are found, blank otherwise
2	Company Extracted	C if company name has been extracted
3	Company Acronym	This is set to 1,2 or 3 if any part of the company name is an acronym
4	Job Title/ Department	J if job title extracted, D if department extracted
5	Verified Postcode	V if postcode was verified OK
6	Extracted Postcode	E if postcode was extracted OK
7	Not used by the matchIT API	
8	Generated Prefix status	P - generated prefix, Q - changed prefix, S - used supplied prefix blank - no personal name processing
9	Salutation status	S - default salutation generated G - non-default (success!) salutation generated blank - no salutation field
10	Original Initial	Set to the first letter of the input forename field
11	Original Initial #2	First letter of second input forename, if any

12	Second Name Sex	Set to sex of second name if supplied & possible to calculate
13	Foreign status	F if record is foreign
14	Not used by the matchIT API	
15	Not used by the matchIT API	
16	Premise status	X if premise extracted, C if premise copied
17	Zip status	X if zip extracted, C if zip copied
18	Town status	X if town extracted, C if town copied
19	County status	X if county extracted, C if county copied
20	Country status	X if country extracted, C if country copied.