matchIT API

Frequently Asked Questions

helpIT SYSTEMS

intelligent data cleansing

The matchIT API Frequently Asked Questions

What operating systems and languages does the matchIT API support?

The API can be used by any programming language or environment that supports ActiveX controls. This includes VB.NET, C#, Visual C++, Visual Basic, Delphi, VB Scripts, Visual FoxPro etc. Database languages that support ActiveX controls, such as SQL Server stored procedures, will also support the API.

The API is supported on Windows 2000/XP/2003 operating systems. The API is also supported on various Unix and Linux systems, called from Java, Perl or C/C++ - please call your supplier for details. Platforms not currently supported can be produced on demand.

What run-time libraries does the matchIT API need?

The Windows version of the matchIT API and the SDK applications use only standard Windows runtime libraries, such as ATL, which are available on all supported Windows platforms and in any case are statically linked.

For Unix and Linux versions, the API uses only ANSII compatible C++ and no non-standard run-time libraries are required.

I get an 'Unknown error' when I start up my application that uses the matchIT API – how can I fix it?

This error is usually caused when the API has not been correctly installed, or the DAT file location is incorrect. Please run the Reactivate utility to ensure that the API has a valid activation code, and also please ensure that the DAT file location is correct and contains valid DAT files.

Do I need to activate copy protection on every machine that I install the matchIT API on?

Unless you have an enterprise or site level agreement with us, you will need to activate the API with a valid activation code on every machine.

An activation code is required on each machine that will use the API. When a code expires, the API cannot be used until the Reactivate utility is run and the API reactivated with a new activation code. Note that reinstalling a newer version of the API does not prompt for a new activation code during installation, any current code is used.

Where do I start in order to develop my application?

Start with the sample application in the SDK that is closest to your requirements. Sample applications are available for:

- on-line duplicate prevention and on-line inquiry (the findIT SDK and findIT Web)
- batch dedupe and merge/purge (the matchIT SDK and the matchIT SDK for SQL Server)
- web data entry to parse names, standardize data and screen for garbage (the cleanseIT SDK).

If the sample applications in your installation do not include the programming language(s) that you are using, please contact your supplier to check on availability of sample applications and code snippets for these languages.

What are match keys and why do I need them?

Match keys and their usage are described in the matchIT API Documentation, available from the Windows start menu, matchIT API program group.

Do I have to store match keys for all the records already in my database?

For batch processing, you should store the match keys for every record in the database, otherwise there is a large overhead in regenerating these keys every time an existing record is involved in a comparison with another record, and you will be limited in the fields that you can use to assemble groups of candidate matches.

For on-line processing for duplicate prevention or inquiry, you do not have to store match keys for every record in the database if your data is sufficiently normalized to allow effective lookups to assemble candidate matches e.g. if last name is in a separate field, you know the company name as well as contact name, postal codes are at street level and are reliable. However, storing the match keys allows a greater likelihood of matches being found using less information. For example, if you look up Bill Dayton at Zip code 10536 and on the database you have Mr W Deighton at this Zip code, you will be able to find Mr Dayton immediately if you store the phonetic Lastname key. However, if you don't store this key, you will be reliant on (say) a lower level postal code (as in the UK) or the street part of the address being keyed before you can find Mr Deighton. This is because it is not normally efficient to compare all records on the database for a 5 digit Zip code with the name being entered for that Zip code.

My database stores the data in a relational format. I've got a one-to-many relationship between addresses and people at those addresses. How can I use the matchIT API to find matches in this kind of data?

For most effective use of the matchIT API, you should store the key fields that the matchIT API generates in your database and then use these key fields (possibly in conjunction with standard fields such as postal code) to assemble candidate matching pairs which the matchIT API can "score". You can store the generated key fields either in the relevant table (address keys in the address table, people keys in the people table etc.), or you can store them in new tables that have one-to-one relationships to the existing tables. You can then assemble candidate matches by e.g. using SQL joins on the tables. Alternatively, you could create a "flat" keys table which would deliberately hold redundant keys data in that (for example) the same address key would be stored multiple times, as this could prove more efficient for the purposes of grouping candidate matches.

What match keys should I use?

This depends on what you are using the matchIT API for, the level of matching you want (one per person, company, address, etc.) and the nature of your data and your requirements. You can use any fields as match keys, but as a general rule, you should select more than one key (which will usually include at least one compound key) and there shouldn't be a common element to all the keys. For example, you should make sure that the Phonetic Lastname Key is not contained in every match key that you use, to help detect duplication no matter what data may be missing or inconsistent. You can use keys containing fields which have not been standardized by the matchIT API, but these are not usually as reliable for finding matches as those which have been standardized.

For most purposes, we recommend default match keys as shown in the tables on the following pages. There are four tables, for:

- duplicate prevention/inquiry
- batch matching e.g. dedupe and merge/purge

with each category being further divided into recommendations depending on existing database volumes.

The variations by volume stem from the fact that the greater the volume of records in your database, the larger the groups of records with the same match key and the longer it takes to compare candidate matches. The point at which you should use the keys recommended for high database volumes depends on the nature of your data and your requirement. For example, matching data keyed for data entry against a million geographically widespread records already on the database may give perfectly good response using the "medium volume" keys, whereas batch dedupe of half a million records all from one local area may take too long unless you use the "high volume" keys.

The keys in the tables are each represented in two columns, one for countries such as the US where postal codes are typically at the town/city level and one for other countries such as the UK where postal codes are typically at the street or lower level. If your data virtually all has low level postal codes such as Zip+4 codes for the US, you can use the keys in the right hand columns.

If your file contains lots of exact duplicates, or contains very complex or badly structured data, you may need to use different keys – please contact your supplier if you would like advice on this. When using keys which differ from the recommendations, it is a good idea to test what extra matches may be picked up by the recommended keys, to ensure that the keys you want to use are effective.

Recommended match keys for duplicate prevention and inquiry - medium volumes of existing data

Data entered	US and other data with high level postal codes	UK and other data with low level postal codes
Company, contact, postal code	• PhoneticOrganizationName1 + PostOut • PhoneticLastName + PostOut • PhoneticOrganizationName1 + PhoneticLastName	• PhoneticOrganizationName1 + PostOut • PhoneticLastName + PostOut • Postcode
Company, contact only	• PhoneticOrganizationName1 + PhoneticLastName • PhoneticOrganizationName1 + PhoneticOrganizationName2 • PhoneticLastName + PhoneticFirstName	• PhoneticOrganizationName1 + PhoneticLastName • PhoneticOrganizationName1 + PhoneticOrganizationName2 • PhoneticLastName + PhoneticFirstName
Company, postal code only	• PhoneticOrganizationName1 + PostOut • PhoneticOrganizationName1 + PhoneticOrganizationName2	• PhoneticOrganizationName1 + PostOut • Postcode • PhoneticOrganizationName1 + PhoneticOrganizationName2
Contact, postal code only	• PhoneticLastName + PostOut • PhoneticLastName + PhoneticFirstName	• PhoneticLastName + PostOut • Postcode • PhoneticLastName + PhoneticFirstName
Company only	• PhoneticOrganizationName1 + PhoneticOrganizationName2	• PhoneticOrganizationName1 + PhoneticOrganizationName2
Contact only	• PhoneticLastName + PhoneticFirstName	• PhoneticLastName + PhoneticFirstName
Postal code only	• Postcode	• Postcode
When address is entered, in addition to any of the above	• AddressKey + Premise	• AddressKey

Recommended match keys for duplicate prevention and inquiry - high volumes of existing data

Data entered	US and other data with high level postal codes	UK and other data with low level postal codes
Company, contact, postal code	• PhoneticOrganizationName1 + PhoneticOrganizationName2 + PostOut • PhoneticLastName + PhoneticFirstName + PostOut • PhoneticOrganizationName1 + PhoneticLastName + PhoneticFirstName	• PhoneticOrganizationName1 + PostOut • PhoneticLastName + PostOut • Postcode
Company, contact only	• PhoneticOrganizationName1 + PhoneticLastName • PhoneticOrganizationName1 + PhoneticOrganizationName2 • PhoneticLastName + PhoneticFirstName	• PhoneticOrganizationName1 + PhoneticLastName • PhoneticOrganizationName1 + PhoneticOrganizationName2 • PhoneticLastName + PhoneticFirstName
Company, postal code only	• PhoneticOrganizationName1 + PostOut • PhoneticOrganizationName1 + PhoneticOrganizationName2	• PhoneticOrganizationName1 + PostOut • Postcode • PhoneticOrganizationName1 + PhoneticOrganizationName2
Contact, postal code only	• PhoneticLastName + PhoneticFirstName + PostOut • PhoneticLastName + PostOut (where LastName is uncommon) • PhoneticLastName + PhoneticFirstName (where name is uncommon)	• PhoneticLastName + PhoneticFirstName + PostOut • PhoneticLastName + Postcode • PhoneticLastName + PhoneticFirstName (where name is uncommon)
Company only	• PhoneticOrganizationName1 + PhoneticOrganizationName2	• PhoneticOrganizationName1 + PhoneticOrganizationName2
Contact only	• PhoneticLastName + PhoneticFirstName	• PhoneticLastName + PhoneticFirstName
Postal code only	• Postcode	• Postcode
When address is entered, in addition to any of the above	• AddressKey + Premise	• AddressKey

Match keys to use for batch matching of names and addresses - medium data volumes		
Matching level	**US and other data with high level postal codes**	**UK and other data with low level postal codes**
Person level	• PostOut + PhoneticLastName • PhoneticLastName + street part of AddressKey (char 5-8) • AddressKey + Premise	• PostOut + PhoneticLastName • PhoneticLastName + street part of AddressKey (char 5-8) • Postcode
Family level	Same as Person level	Same as Person level
Address level	• PostOut + PhoneticLastName • PostOut + street part of AddressKey + Premise • AddressKey + Premise	• PostOut + PhoneticLastName • Postcode + street part of AddressKey + Premise • Postcode
Organization level	• PostOut + PhoneticOrganizationName1 • PhoneticOrganizationName1 + street part of AddressKey • AddressKey + Premise	• PostOut + PhoneticOrganizationName1 • PhoneticOrganizationName1 + street part of AddressKey • Postcode
If items of data other than name and address are available e.g. Telephone, Email, Account	Use these keys as individual match keys, possibly in addition to those above	Use these keys as individual match keys, possibly in addition to those above

Match keys to use for batch matching of names and addresses - high data volumes		
Matching level	**US and other data with high level postal codes**	**UK and other data with low level postal codes**
Person level	• AddressKey + PhoneticLastName • PostOut + street part of AddressKey + Premise (where Premise is non-blank) • Postcode + PhoneticLastName (where Postcode is at street or lower level)	• Postcode + PhoneticLastName • AddressKey + PhoneticLastName • Postcode + Premise (or Delivery Point Suffix if available)
Family level	Same as Person level	Same as Person level
Address level	• AddressKey + PhoneticLastName • PostOut + street part of AddressKey + Premise (where Premise is non-blank) • Postcode + Premise (or Delivery Point code if available) (where Postcode is at street or lower level and Premise is non-blank)	• AddressKey + PhoneticLastName • Postcode + Premise (or Delivery Point Suffix if available)
Organization level	• AddressKey + PhoneticOrganizationName1 • Postcode + PhoneticOrganizationName1 + PhoneticOrganizationName2 • Postcode + street part of AddressKey + Premise (where Premise is non-blank)	• Postcode + PhoneticOrganizationName1 • AddressKey + PhoneticOrganizationName1 • Postcode + Premise (or Delivery Point Suffix if available)
If items of data other than name and address are available e.g. Telephone, Email, Account	Use these keys as individual match keys, possibly in addition to those above	Use these keys as individual match keys, possibly in addition to those above

Can I use other fields as match keys apart from the standard ones?

Yes you can, although you must remember that there is no normalization of these fields performed by e.g. SQL Select or Index statements, so you must make sure that casing, punctuation and white space differences do not affect the match key, or are handled by using the API's generated match keys.

What matching weights should I use for my data?

Again, this is dependant on the data, but the default weights that the matchIT API allocates are adequate for most purposes:

- Residential (one record per person or family)

Description	**Name**	**Address**	**Postcode**
Sure	60	40 (US) 30 (UK)	30
Likely	40	30 (US) 22 (UK)	20
Possible	20	20 (US) 15 (UK)	15
One Empty	15	5	10
Both Empty	25	5	10

  The different Address weights for US (or high level postal codes) and UK (or low level postal codes) stem from the fact that the Sure weight on Postcode is reserved for postcodes that are (as far as can be determined) at street or lower level. For example, for US Zip codes, the Sure weight will only be allocated to full Zip+4 codes that are equal, whereas 5 digit Zip codes that are equal will only achieve the Likely weight, as will a 5 digit Zip which matches the first 5 digits of a Zip+4 code.

- Residential (one record per address – household matching)

  As for one record per person or family, but with zero weights on Name.

- Business (one record per person or contact)

  This is the same as for Personal Matching, as by default the matchIT API ignores company names when matching business contacts – this is because it is very common for companies to change their name, individuals to work for more than one company in the same group etc. Of course, you can choose to put a weight on the company name as well, if you wish.

- Business (one record per company)

  As for one record per person or family, but with the same weights on Organization Name instead of Name.

How do I set up the API to use a particular matching level?

The matchIT API has four matching levels (individual, family, household, and business). When two records are compared, the scores are automatically calculated for all matching levels; no prior set up is required beforehand.

After comparing two records, for example, the Name score for the individual matching level can be retrieved using 'Scores.IndividualLevel.Name' and the Organization score for the business matching level can be retrieved using 'Scores.BusinessLevel.Organization'.

What architecture would you suggest for web-based data entry forms using the matchIT API?

We recommend that you implement a multi-tier architecture where the business tier (usually a server in its own right) has the API implemented on it. In this way, web queries can be sent to this machine. This tier will then carry out the various casing and parsing functions on the data entered, and return the results to the web client.

This has the following advantages:-

- the API is installed in only one place, so maintenance of the application is easy;
- the API does not need to be downloaded as an embedded control in the web form, which could cause a significant time overhead.

Can I use the sample application code in my own programs?

We encourage you to use any relevant code from the sample applications in your own systems. These programs are intended to provide real world, scalable applications, which are free for you to amend as necessary and use in any way you wish.

How can I change the standard names and words lookups that the matchIT API recognises?

Please consult the Names and Words Table section in the matchIT API Documentation, available from the Windows start menu, matchIT API program group.

Does the matchIT API incorporate zip/postcode/address verification?

The matchIT API does not, but the findIT SDK sample data entry application does, using the rapid addressing API from QAS. Both API's are available from your supplier.

Batch applications incorporating address verification are also available from your supplier.

I don't want to use ActiveX in my application. Is the matchIT API available as a non-component standard DLL instead?

A non-ActiveX version of the matchIT API library can be produced for large scale deployments.

How well does the matchIT API work on non-English language data?

The matchIT API will work very well on non-English data, providing the character set is essentially the Roman alphabet. There are users of the matchIT API in many non-English speaking countries and using non-English data from many countries, including several European languages.

Although the soundIT phonetic algorithm is tuned for English data, a good result is achieved for the kinds of errors that arise in practise on foreign data. Combining this with the intelligent name & address processing (which copes with keying and reading errors as well as phonetic matches), the matchIT API becomes a very useful tool for processing non-English data sets. We recommend that you use the "Loose soundIT" setting for non-English language data.

What's the difference between soundIT and Soundex?

soundIT is a proprietary phonetic algorithm developed exclusively by helpIT systems. For phonetic matching, you need an accurate algorithm – one that matches words together when they sound the same, and does not match when they sound different. soundIT works by splitting a name or word into syllables, working out the sound of each syllable, with greater emphasis on the stressed syllable. This contrasts with Soundex, which disregards all vowels, plus the letters Y, W and H, unless they form the first letter of the word. This means that Soundex gets the same key for quite different names such as Brady, Broad and Beard, whilst missing names which soundIT matches, like Deighton and Dayton.

How should I set the matchIT API up for foreign data?

The API has a 'Nationality' options setting within it, Engine.Settings.NationalityOfData. If you expect the majority of your data to be from one country, just select that country, otherwise select "Other". This alters the address processing for different address and postcode formats.

We recommend that you use the "Loose soundIT" setting for non-English language data.

Does the matchIT suite use the matchIT API?

The standalone matchIT batch data cleansing suite and the matchIT API use the same core code, which is the matchIT API core code. Forthcoming versions of matchIT will use the API itself i.e. at the same level as the sample applications available in the API SDKs. This means that you can use the same options in the matchIT API as in matchIT and get the same results, which can be useful in testing different options – it is quicker to test different options in matchIT because it provides specific screens to set all the options, reports and views on selected results etc. When the options are stable, they can then be implemented using the matchIT API without having to provide the same degree of flexibility as the matchIT suite provides.

Why doesn't the matchIT API do any kind of reporting, or direct database access?

The matchIT API doesn't do any reporting or database access, as this would limit the operating systems and databases for which it can be used. If reporting or ADO database access was included in the control, this would limit the code that could be used cross-platform. However, the Windows SDK sample applications do provide database access code using ADO for common database managers.

Apart from the Name, Address, Postcode/Zip, are there any other fields that the matchIT API has special processing for?

Yes, other fields include job title, department, telephone and fax. The matchIT API does not do as much with these fields as with the Name and Address data – more information is given in the matchIT API Documentation.

In a forthcoming release of the matchIT API, you will be able to use these fields for matching by setting weights on these fields – this will be limited to non-phonetic comparisons, looking for single character differences, transpositions etc.

The matchIT API also processes fields that could be included as part of other fields – for example, Suffix, Qualification, Premise, Town, County, State, Country. These fields are mainly used for data extraction.

Can the matchIT API match email addresses?

A forthcoming version of the matchIT API will include intelligent matching and validation of email addresses – please contact your supplier for further details.

What happens if I pass data in a property where it doesn't belong?

It will be ignored or processed incorrectly. If for example, you pass a company name in a personal name (e.g. fullname) property, then the matchIT API expects to see a person's name there and will generate either a blank personal name key, or an incorrect personal name key without allowing for the different nature of company names. Matches can still be found, but the hit rate and effectiveness will be reduced.

It is possible to use the Generate method to attempt to clean up data which sometimes has items that are wrongly located e.g. you can extract data items like full name, company, town, etc from address lines into their proper fields, if they can be recognized for what they are by a word in the Names table such as **Mr** or **Ltd**.