

Recorder 6

Taxon name matching using the Species Dictionary

JNCC, July 2009

© JNCC, Peterborough 2009

Contents

1	Introduction.....	2
1.1	The species dictionary	2
1.2	Matching the names	3
1.3	Preferred Lists	3
2	Matching the data.....	4
2.1	Create a table to hold the names.....	4
2.2	Import the names into the table	5
2.3	Name matching to identify Taxon_Key.....	6
2.3.2	Initial matching using preferred lists (to find the Taxon_Key)	6
2.3.3	Broader matching using all lists (to find the Taxon_Key)	9
2.3.4	Matching the remaining species individually (for Taxon_key).....	11
2.4	Secondary matching to identify the Taxon_List_Item_Key from the Taxon_Key.....	14
2.4.1	Initial matching against preferred lists to find the Taxon_List_Item_Key	14
2.4.2	Broader matching against all lists to find Taxon_List_Item_Key	15

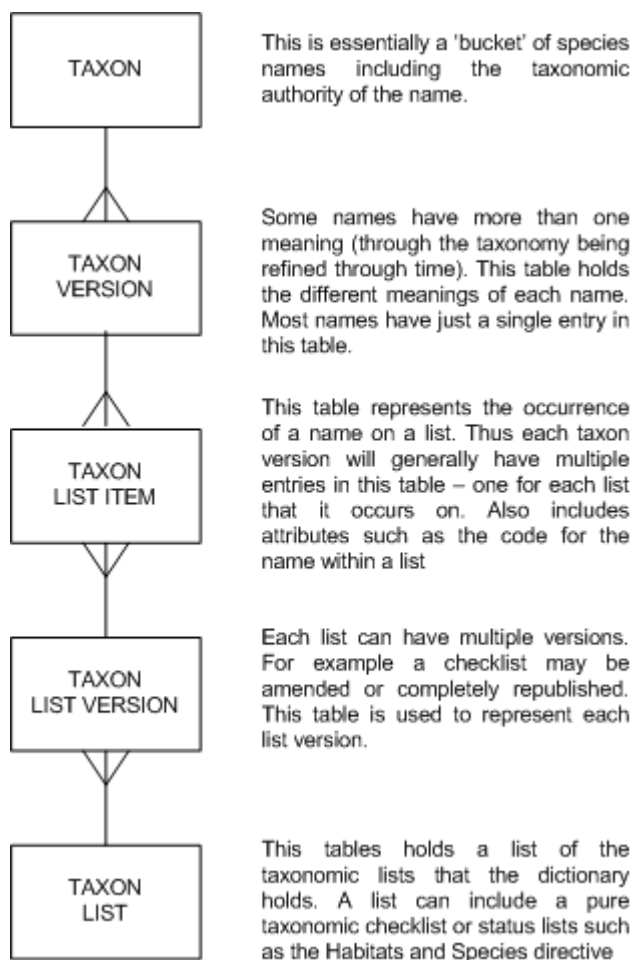
1 Introduction

This tutorial is aimed at the beginner to intermediate level: some experience of working within MS Access is assumed and a copy of MS Access is required.

Matching a list of names into the species dictionary is a fairly common task carried out by data managers, for example when manually creating a recording card. However, the complexity of the species dictionary within Recorder makes this a less than straightforward task for the uninitiated. This tutorial sets out the basic steps to match a list of names, including briefly outlining the main structure of the species dictionary. There are a couple of different methods of achieving the same results. The mechanism provided in this tutorial uses the raw species dictionary not the Index Taxon Name table.

1.1 The species dictionary

Before beginning any operation with the species dictionary you need to have a basic understanding of how it is structured within Recorder. The key concept to understand is that the species dictionary is heavily based on the concept of taxonomic lists. Thus rather than just associating a species record with a species name, the record is actually made against the instance of a name on a particular list (typically a published taxonomic checklist). The benefit of this approach is that it is less ambiguous when the taxonomy of a species changes. Generally with a particular list it is fairly clear what meaning is associated with a particular name and so tagging a record to a particular list reduces this sort of ambiguity. The five tables that make up the main 'spine' of the model are shown below:



1.2 Matching the names

The easiest way to match names into the dictionary is probably to work in the linked Access database created when Recorder installs (this is an Access database which contains linked tables from the SQL Server database). The linked Access database is called nbndata.mdb and can be found in C:\Program Files\Recorder 6\Database. If the database is not there (and you have Recorder installed and working on your computer) then you need to recreate the linked Access table. This can be done using the tool/file specifically designed for this job called 'Create Access mdb' which can be downloaded from www.recordersoftware.org. This tutorial matches the taxa from the BRC ladybird recording card (including the search codes which can be used for fast data entry) from an excel source file using Access 2007. The file used is available for download to allow you to work through the tutorial (www.recordersoftware.org).

1.3 Preferred Lists

The dictionary holds numerous lists and the same species name may occur on a number of different lists. Some time ago, the concept of preferred lists was developed. In essence these are lists which are considered by the Natural History Museum (the managers of the species dictionary) to represent the correct and current taxonomy for their taxonomic area. Ultimately it is hoped these will span the full taxonomic coverage of the UK and already there is good coverage of most of the popular groups. As a general rule matching to names on these 'preferred' lists is likely to be more robust. There is a flag in the TAXON_LIST table called "PREFERRED" which, if set to true, indicates that the list represents one of these preferred lists.

2 Matching the data

Matching the data involves a number of stages. Firstly the species list needs to be transferred from Excel into Access and then the names need to be matched with those in the species dictionary to identify the `taxon_keys`. Initially matches are made using preferred lists and then subsequently using both broad and individual searching techniques. After all the `taxon_keys` are identified and issues resolved they need to be matched to `taxon_list_item_keys`, again, firstly using the preferred lists and then using a broader search.

2.1 Create a table to hold the names

First create a table that will hold the names and their matching keys. For the purposes of this tutorial the table will be called “MySpeciesList” and have the following structure:

Field	Data type	Description
SEARCH_CODE	Text 15 characters	Optional field to hold the species codes for use in quick entry in the recording card (eg BRC codes: see example)
TAXON_NAME	Text 100 characters	Used to hold the species names to be matched
TAXON_AUTHORITY	Text 100 characters	Optionally include the taxonomic authorities of the names
TAXON_KEY	Text 16 characters	Will be populated with the NBN key that represents the name
TAXON_LIST_ITEM_KEY	Text 16 characters	Will be populated with the <code>taxon_list_item_key</code> (representing the occurrence of the name on a list)

The easiest way to create the table is to click “New” (from the table view in Access) and then “Design view” and just create the fields listed above. In Access 2007 you click the Create tab and choose Table and then click View/Design View.

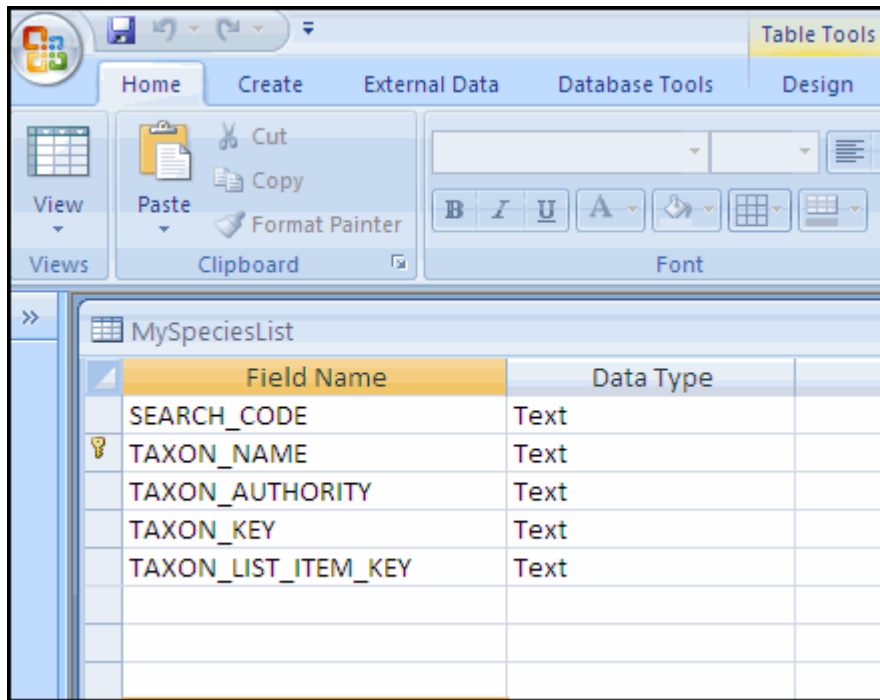


Figure 1: Creating the MySpeciesList table in design view

2.2 Import the names into the table

Assuming the names exist in some form of electronic file they can be imported into the table using the MS Access import wizard (File -> Get external data -> Import). Note that you may need to alter the “Files of type” at the bottom of the import wizard screen before your file is listed. In Access 2007 the route is External data/Excel[or other source]/Append a copy of the records to the table [MySpeciesList] and then click through the remainder of the wizard with appropriate answers for your dataset. Import the list of names and ensure that at least the TAXON_NAME field in the table is populated.

SEARCH_CODE	TAXON_NAME	TAXON_AUTHORITY
59502	Adalia 10-punctata	
59501	Adalia 2-punctata	
59101	Adonia variegata	
60001	Anatis ocellata	
59201	Anisosticta 19-punctata	
59301	Aphidecta oblitterata	
59901	Calvia 14-guttata	
58801	Chilocorus bipustulatus	
58802	Chilocorus renipustulatus	
58101	Clitostethus arcuatus	
57901	Coccidula rufa	
57902	Coccidula scutellata	
59605	Coccinella 11-punctata	
59603	Coccinella 5-punctata	
59604	Coccinella 7-septempunctata	
59602	Coccinella hieroglyphica	
59601	Coccinella magnifica	
58901	Exochomus quadripustulatus	
60301	Halyzia sedecimguttata	
59701	Harmonia 4-punctata	
59702	Harmonia axyridis	
94201	Henosepilachna argus	
59001	Hippodamia 13-punctata	
58601	Hyperaspis pseudopustulata	
60101	Myrrha 18-guttata	
60201	Myzia oblongoguttata	

Figure 2: MySpeciesList populated with the ladybird data.

3 Name matching to identify Taxon_Key

3.1 Initial matching using preferred lists (to find the Taxon_Key)

The first step is to match the imported names against the main dictionary. This is generally best done with at least a degree of manual intervention. Many names appear in the dictionary more than once (e.g. many occur both with and without authorities) but attempts to match on both authority and name are likely to produce poor matches, particularly due to minor discrepancies in the authority.

Create a query [Create/Query Design] and change the view to SQL and simply copy and paste query 1 into the SQL screen and press RUN (red exclamation mark). This query checks whether there is more than one dictionary entry for each species name. If no rows are returned this means that there are no species names with multiple entries in the dictionary. If there are rows returned then you will need to resolve these species manually (see section 3.3.2).

Query 1: Query to view initial matches of imported names in the species dictionary but filtering to those names that appear on 'preferred lists' and return rows that have more than one taxon key available.

```
SELECT MySpeciesList.TAXON_NAME, TAXON.ITEM_NAME,
TAXON_LIST.PREFERRED, Count(TAXON.TAXON_KEY) AS CountOfTAXON_KEY
FROM (((MySpeciesList INNER JOIN TAXON ON MySpeciesList.TAXON_NAME =
TAXON.ITEM_NAME) INNER JOIN TAXON_VERSION ON TAXON.TAXON_KEY =
TAXON_VERSION.TAXON_KEY) INNER JOIN TAXON_LIST_ITEM ON
TAXON_VERSION.TAXON_VERSION_KEY =
TAXON_LIST_ITEM.TAXON_VERSION_KEY) INNER JOIN TAXON_LIST_VERSION
ON TAXON_LIST_ITEM.TAXON_LIST_VERSION_KEY =
TAXON_LIST_VERSION.TAXON_LIST_VERSION_KEY) INNER JOIN TAXON_LIST ON
TAXON_LIST_VERSION.TAXON_LIST_KEY = TAXON_LIST.TAXON_LIST_KEY
GROUP BY MySpeciesList.TAXON_NAME, TAXON.ITEM_NAME,
TAXON_LIST.PREFERRED
HAVING (((TAXON_LIST.PREFERRED)=True) AND ((Count(TAXON.TAXON_KEY))>1));
```

3.1.1 Where no rows of data are returned by query 1.

Where query 1 returned no rows there are no species names with multiple entries in the dictionary with those species on the preferred lists. Therefore the matches from query 1 (without the count of taxon_key field) can be used to update the MySpeciesList table.

In the design view remove the count of the taxon_key field (figure 3) by highlighting the column (click at the top) and press <delete>.

Field:	TAXON_NAME	ITEM_NAME	PREFERRED	TAXON_KEY
Table:	MySpeciesList	TAXON	TAXON_LIST	TAXON
Total:	Group By	Group By	Group By	Count
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:			True	>1
or:				

Figure 3: Remove the count of Taxon_key field from the query in design view. Note that if you re-run this query at this point to view the data the Preferred column shows '-1'. This is just another way MSAccess uses to present 'True' ('False'='0').

Change the query type to Update query (figure 4) and add the two fields which are to be updated with the details of how they are to be updated. Add Taxon_Authority and Taxon_Key from the MySpeciesList table and include the details [Taxon].[Authority] and [Taxon].[Taxon_Key] on the update row respectively. In addition alter the 'criteria' row for each of these new columns to "Is null" (figure 5). This ensures that the fields are only updated where they do not already contain a value (so if you have already matched anything manually this will not be lost).

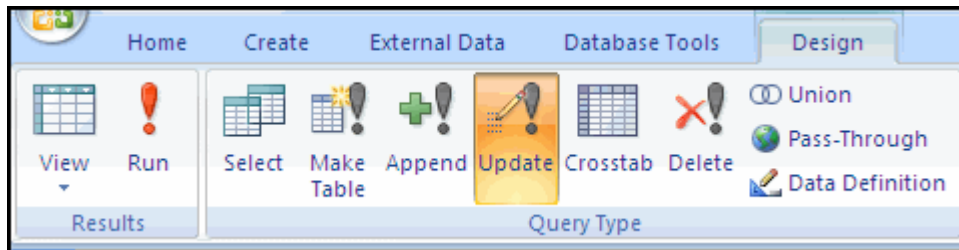


Figure 4: Change the query type to an update query.

Field:	TAXON_NAME	ITEM_NAME	PREFERRED	TAXON_AUTHORITY	TAXON_KEY
Table:	MySpeciesList	TAXON	TAXON_LIST	MySpeciesList	MySpeciesList
Update To:				[Taxon].[Authority]	[Taxon].[Taxon_Key]
Criteria:			True	Is Null	Is Null
or:					

Figure 5: Add the fields that need to be updated (Taxon_Authority and Taxon_Key in the MySpeciesList table).

You will be asked whether you want to update 22 rows (figure 6), click yes.

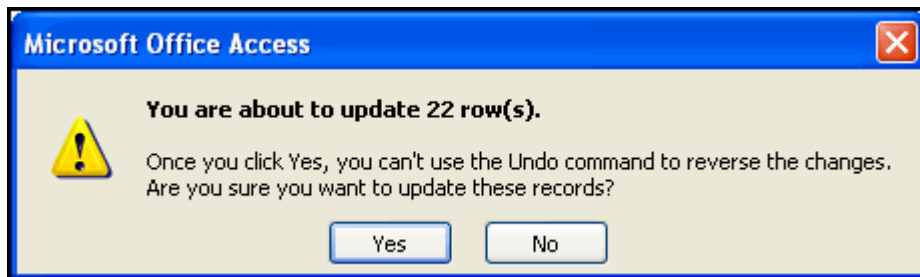


Figure 6: Warning message.

Using the example data your MySpeciesList table should look like figure 7.

SEARCH_CODE	TAXON_NAME	TAXON_AUTHORITY	TAXON_KEY
59502	Adalia 10-punctata		
59501	Adalia 2-punctata		
59101	Adonia variegata		
60001	Anatis ocellata	(Linnaeus, 1758)	NBNSYS0000008327
59201	Anisosticta 19-punctata		
59301	Aphidecta oblitterata	(Linnaeus, 1758)	NBNSYS0000008317
59901	Calvia 14-guttata		
58801	Chilocorus bipustulatus	(Linnaeus, 1758)	NBNSYS0000008309
58802	Chilocorus renipustulatus	(Scriba, 1791)	NBNSYS0000008310
58101	Clitostethus arcuatus	(Rossi, 1794)	NBNSYS0000008293
57901	Coccidula rufa	(Herbst, 1783)	NBNSYS0000008290
57902	Coccidula scutellata	(Herbst, 1783)	NBNSYS0000008291
59605	Coccinella 11-punctata		

Figure 7: Part of MySpeciesList table in datasheet view. Twenty-two rows have been updated.

3.1.2 Where rows of data are returned by query 1

For those species where there are multiple entries in the dictionary you must look at the options for each species where there are duplicates and make a choice about which one to choose. Including the Authority in the query from the Taxon table should help you to make some of the choices ie some entries may not have an authority or simply differ by a date. Once you have made your choice paste the authority and taxon_key into the MySpeciesList table. For a working example see section 3.2.1.

3.2 Broader matching using all lists (to find the Taxon_Key)

Up to this point we have matched a portion of the species – namely those that appear on preferred taxonomic lists. The remaining names probably exist in the dictionary on non-preferred lists. The search needs to be broadened to find further matches by removing the criterion to limit matches to preferred lists from the SQL. Paste query 2 into the SQL view and run. Again this searches for any species where there is more than one match using the count of Taxon_Key. If there is only one match per species the query will return no rows, if there are rows returned you will have to choose one of the options and paste the taxon_key into the relevant MySpeciesList table. The query will only return rows where there are multiple matches AND the Taxon_key in MySpeciesList is empty (ie unmatched).

Query 2: Query to view matches of imported names into the species dictionary (non-preferred lists).

```
SELECT MySpeciesList.TAXON_NAME, TAXON.ITEM_NAME,
Count(TAXON.TAXON_KEY) AS CountOfTAXON_KEY
FROM MySpeciesList INNER JOIN TAXON ON MySpeciesList.TAXON_NAME =
TAXON.ITEM_NAME
WHERE (((MySpeciesList.TAXON_KEY) Is Null))
GROUP BY MySpeciesList.TAXON_NAME, TAXON.ITEM_NAME
HAVING (((Count(TAXON.TAXON_KEY))>1))
ORDER BY MySpeciesList.TAXON_NAME;
```

3.2.1 Dealing with multiple matches (where there are rows of data returned by query 2)

In this example query 2 has returned two rows of data (figure 8). These two species have more than one match in the species dictionary and we will need to look at these individually (with a slightly broader query – query 3) to choose which match is the most suitable.

TAXON_NAME	ITEM_NAME	CountOfTAXON_KEY
Adonia variegata	Adonia variegata	2
Scymnus frontalis	Scymnus frontalis	2

Figure 8: Species with more than one match in the species dictionary which have not already been matched in the MySpeciesList table.

You need to broaden the categories to include the authority and the taxon key from the taxon table and remove both the count of taxon_key and the where is null of taxon_key (figure 9 shows how to change the query in design view or you can paste the equivalent SQL in query 3 into the SQL view).

Field:	TAXON_NAME	ITEM_NAME	AUTHORITY	TAXON_KEY
Table:	MySpeciesList	TAXON	TAXON	TAXON
Total:	Group By	Group By	Group By	Group By
Sort:	Ascending			
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		"Scymnus frontalis" Or "Adonia variegata"		
or:				

Figure 9: Change to query 2 as described in design view.

Query 3: Query to show the authority and taxon_key for the two species that need to be matched manually.

```
SELECT MySpeciesList.TAXON_NAME, TAXON.ITEM_NAME, TAXON.AUTHORITY,
TAXON.TAXON_KEY
FROM MySpeciesList INNER JOIN TAXON ON MySpeciesList.TAXON_NAME =
TAXON.ITEM_NAME
GROUP BY MySpeciesList.TAXON_NAME, TAXON.ITEM_NAME, TAXON.AUTHORITY,
TAXON.TAXON_KEY
HAVING (((TAXON.ITEM_NAME)="Scymnus frontalis" Or
(TAXON.ITEM_NAME)="Adonia variegata"))
ORDER BY MySpeciesList.TAXON_NAME;
```

Your data will look like figure 10. You can see that the first row of data for each species has a more complete authority than the second row for each species. These authorities and their equivalent taxon keys should be manually pasted into your MySpeciesList table (those shown in the red boxes). Continue on to section 3.3.2 to match those species with single matches

TAXON_NAME	ITEM_NAME	AUTHORITY	TAXON_KEY
Adonia variegata	Adonia variegata	(Goeze, 1777)	NHMSYS0000570413
Adonia variegata	Adonia variegata	(Goeze)	NBNSYS0000141011
Scymnus frontalis	Scymnus frontalis	(F., 1787)	NBNSYS0000008296
Scymnus frontalis	Scymnus frontalis		NHMSYS0020083529

Figure 10: Resulting data from query 3

3.2.2 Matching those species with only a single match in the dictionary (where there are no rows of data returned by query 2 or after multiple matches in section 3.2.1 are completed)

To match those species with a single match in the species dictionary view query 2 in design view:

- remove the count of taxon_key field
- change query 2 to an update query
- add the two fields which are to be updated in the MySpeciesList table see figure 11

The SQL for these changes to query 2 can be seen in query 4). This time you will update 20 rows and your table will have 44 rows matched.

Field:	TAXON_NAME	ITEM_NAME	AUTHORITY	TAXON_KEY	TAXON_AUTHORITY	TAXON_KEY
Table:	MySpeciesList	TAXON	TAXON	TAXON	MySpeciesList	MySpeciesList
Update To:					[Taxon].[Authority]	[Taxon].[Taxon_Key]
Criteria:					Is Null	Is Null
or:						

Figure 11: Change the query type to an update query and make the changes shown in the red box to the query design.

Query 4: The equivalent SQL for the changes in figure 11.

```
UPDATE MySpeciesList INNER JOIN TAXON ON MySpeciesList.TAXON_NAME =
TAXON.ITEM_NAME SET MySpeciesList.TAXON_AUTHORITY = [Taxon].[Authority],
MySpeciesList.TAXON_KEY = [Taxon].[Taxon_Key]
WHERE (((MySpeciesList.TAXON_AUTHORITY) Is Null) AND
((MySpeciesList.TAXON_KEY) Is Null));
```

There are now only five species in our example without a taxon_key and authority.

3.3 Matching the remaining species individually (for Taxon_key)

There are five species still to be matched in our example. Two of these simply have no authority (*Adalia 10-punctata* and *Propylea 14-punctata*) whilst the other three (*Coccinella 7-septempunctata*, *Scymnus impexus* and *Scymnus interruptus*) need to be matched individually using query 6.

3.3.1 Matching species with no authority

You may have already noticed that *Adalia 10-punctata* and *Propylea 14-punctata* have in fact been matched to the species dictionary in query 2. These have a taxon_key but no authority. You can either match these manually using the information in section 3.3.2 or you can use update query 5 (which is simply query 4 without updating the taxon_authority field). This will update the two rows of data.

Query 5: Updating matched species which only have a taxon_key (no authority)

```
UPDATE MySpeciesList INNER JOIN TAXON ON MySpeciesList.TAXON_NAME =  
TAXON.ITEM_NAME SET MySpeciesList.TAXON_KEY = [Taxon].[Taxon_Key]  
WHERE (((MySpeciesList.TAXON_KEY) Is Null));
```

3.3.2 Matching the remaining three names

There are now only three species that do not match the species dictionary, these are: *Coccinella 7-septempunctata*, *Scymnus impexus* and *Scymnus interruptus*. To find matches for these we can use the following query (6) for each species.

Query 6: Query to look for entries in the TAXON table that match a particular text string (in this case 'interruptus'). Paste the text in the box into the SQL screen of an Access query.

```
SELECT TAXON.TAXON_KEY, TAXON.ITEM_NAME, TAXON.AUTHORITY  
FROM TAXON  
WHERE TAXON.ITEM_NAME Like "*interruptus*"  
ORDER BY TAXON.ITEM_NAME;
```

Your result will return 27 rows; the highlighted row, in figure 12, is the specie required. Copy the taxon_key returned here into your taxon_key field against this species in your MySpeciesList table (and also the authority).

TAXON_KEY	ITEM_NAME	AUTHORITY
NBNSYS0000142007	Aphrodes interruptus	(Scott)
NHMSYS0000456674	Bromus interruptus	(Hack.) Druce
NBNSYS0000002587	Bromus interruptus	(Hackel) Druce
NHMSYS0100000819	Bromus mollis var. interruptus	Hack.
NBNSYS0100002257	Chlorops interruptus	Meigen, 1830
NBNSYS0000146538	Ectemnius interruptus	
NHMSYS0000874188	Ectemnius interruptus	
NBNSYS0100002964	Eristalis interruptus	(Poda, 1761)
NBNSYS0000010439	Evacanthus interruptus	(Linnaeus)
NHMSYS0001714391	Hydrochus interruptus	
NBNSYS0000149038	Hydrochus interruptus	von Heyden, 1870
NBNSYS0000149771	Laccophilus interruptus	(Panzer, 1795)
NBNSYS0000008522	Leptocerus interruptus	(F., 1775)
NHMSYS0020442229	Leptocerus interruptus	(Fabricius, 1775)
NBNSYS0000009352	Leptothorax interruptus	(Schenck)
NHMSYS0000874535	Leptothorax interruptus	(Schenck, 1852)
NBNSYS0100022513	Musca interruptus	Poda, 1761
NBNSYS0000152228	Necrophorus interruptus	Stephens, 1830
NBNSYS0000023037	Nicrophorus interruptus	Stephens, 1830
NHMSYS0000874754	Nysson interruptus	(Fabricius, 1798)
NBNSYS0000009580	Nysson interruptus	(Fabricius)
NBNSYS0000154777	Potamogeton interruptus	Kit.
NHMSYS0020150834	Scymnus (Scymnus) interruptus	(Goeze, 1777)
NHMSYS0000874984	Temnothorax interruptus	(Schenck, 1852)
NHMSYS0000874986	Temnothorax tuberointerruptus	Bondroit, 1918
NHMSYS0001716253	Trichodes interruptus	Kraatz, 1894
NHMSYS0020190349	Tydeus interruptus	Thor, 1932

Figure 12: The highlighted row is the species required.

When you repeat this query with *Coccinella* and *impexus* there are several further issues:

Coccinella 7-septempunctata

The name as seen in the imported spreadsheet (*Coccinella 7-septempunctata*) is different to that held in the species dictionary (*Coccinella 7-punctata*) but can still be matched to this species without question. The name in the spreadsheet appears to be incorrect whilst the species dictionary entry is correct. This can be double-checked using current checklists and other official, comprehensive sources available on the internet. If you think that there is a mistake in the species dictionary or a species missing you can post your opinion on the NBN Forum in the species dictionary category: <http://forums.nbn.org.uk/viewforum.php?id=9>.

Scymnus impexus

This species is not currently in the species dictionary. The name could have changed or it might simply have not been added to the centrally held species dictionary yet for another reason. You can search the internet and use the NBN Gateway to find out why. This particular species is a non-established introduction. All sightings, according to the A.G.Duff of the

Checklist of Beetles of the British Isles 2008, have been of non-breeding introductions and are therefore not considered part of the fauna of the UK. This species has therefore not been added to the species dictionary. You can add this species to your version of the species dictionary in Recorder if you wish to record your sightings in the following way: Dictionaries/Edit Taxon Details then choose the list you want to add to (List of additional names is recommended so that the information is not overwritten with the next upgrade to the dictionary) and then simply click Add and fill in the details of the species.

4 Secondary matching to identify the Taxon_List_Item_Key from the Taxon_Key

Up to this point we have matched all the names on MySpeciesList to a taxon_version_key. However recorder actually links to specific instances of a name on a list so we must now match the name and taxon_version_key to a taxon_list_item_key. Again this involves first matching against preferred lists and then broadening the search to include non-preferred lists.

4.1 Initial matching against preferred lists to find the Taxon_List_Item_Key

The first step in this part of the process is to match the Taxon_Keys to the Taxon_List_Item_Keys using the preferred lists. Paste query 7 into the SQL screen. This query checks to see whether there is more than one Taxon_List_Item_Key for the species on preferred lists. It should return no rows of data. If rows are returned you will need to manually investigate the duplicate keys to ascertain which is the most up-to-date (and copy and paste them into your MySpeciesList table).

Query 7: Identifies the Taxon_List_Item_Key using the preferred lists.

```
SELECT MySpeciesList.TAXON_NAME, MySpeciesList.TAXON_KEY, TAXON_LIST.PREFERRED,
Count(TAXON_LIST_ITEM.TAXON_LIST_ITEM_KEY) AS CountOfTAXON_LIST_ITEM_KEY
FROM TAXON_LIST INNER JOIN ((MySpeciesList INNER JOIN (TAXON_LIST_ITEM INNER JOIN
TAXON_VERSION ON TAXON_LIST_ITEM.TAXON_VERSION_KEY =
TAXON_VERSION.TAXON_VERSION_KEY) ON MySpeciesList.TAXON_KEY =
TAXON_VERSION.TAXON_KEY) INNER JOIN TAXON_LIST_VERSION ON
TAXON_LIST_ITEM.TAXON_LIST_VERSION_KEY =
TAXON_LIST_VERSION.TAXON_LIST_VERSION_KEY) ON TAXON_LIST.TAXON_LIST_KEY =
TAXON_LIST_VERSION.TAXON_LIST_KEY
GROUP BY MySpeciesList.TAXON_NAME, MySpeciesList.TAXON_KEY,
TAXON_LIST.PREFERRED
HAVING (((TAXON_LIST.PREFERRED)=True) AND
((Count(TAXON_LIST_ITEM.TAXON_LIST_ITEM_KEY))>1));
```

4.1.1 Where no rows of data are returned from query 7

Where no rows are returned this means that there is a single entry in the dictionary for the Taxon_List_Item_Key and that this data can be used in an update query to update your MySpeciesList table. To update your MySpeciesList table with this data you need to amend the query in design view as follows (or paste query 8 into the SQL view):

- remove the CountOfTaxon_List_Item_Key

Taxon name matching using the Species Dictionary

re-run the query to check the data that is returned is what you expected, then:

- change the query to an update query
- add the field that needs to be updated: Taxon_List_tem_Key from the MySpeciesList table and include Is Null in the criteria row as shown in figure 13.

Running this query will update 23 rows.

Field:	TAXON_NAME	TAXON_KEY	PREFERRED	TAXON_LIST_ITEM_KEY
Table:	MySpeciesList	MySpeciesList	TAXON_LIST	MySpeciesList
Update To:				[Taxon_List_Item].[Taxon_List_Item_Key]
Criteria:			True	Is Null
or:				

Figure 13: Details that need to be amended in order to change query 7 to an update query.

Query 8: Query to update the Taxon_List_Item_Key field in the MySpeciesList table.

```
UPDATE TAXON_LIST INNER JOIN ((MySpeciesList INNER JOIN
(TAXON_LIST_ITEM INNER JOIN TAXON_VERSION ON
TAXON_LIST_ITEM.TAXON_VERSION_KEY =
TAXON_VERSION.TAXON_VERSION_KEY) ON MySpeciesList.TAXON_KEY =
TAXON_VERSION.TAXON_KEY) INNER JOIN TAXON_LIST_VERSION ON
TAXON_LIST_ITEM.TAXON_LIST_VERSION_KEY =
TAXON_LIST_VERSION.TAXON_LIST_VERSION_KEY) ON
TAXON_LIST.TAXON_LIST_KEY = TAXON_LIST_VERSION.TAXON_LIST_KEY SET
MySpeciesList.TAXON_LIST_ITEM_KEY = [Taxon_List_Item].[Taxon_List_Item_Key]
WHERE (((TAXON_LIST.PREFERRED)=True) AND
((MySpeciesList.TAXON_LIST_ITEM_KEY) Is Null));
```

4.1.2 Where rows of data returned from query 7

Where there are multiple rows returned by query 7 you will need to choose which Taxon_List_Item_Key to use and paste this into your MySpeciesList table.

4.2 Broader matching against all lists to find Taxon_List_Item_Key

There are now 27 taxon_list_item_keys to match. If you paste query 9 into the SQL view and run it you will see those species that have more than one match for taxon_list_item_key in the dictionary (all lists; preferred and non-preferred) and where there is not already a match in the MySpeciesList table.

Query 9: Those species with more than one taxon_list_item_key match where taxon_list_item_key in the MySpeciesList table is empty.

Taxon name matching using the Species Dictionary

```

SELECT MySpeciesList.TAXON_NAME, MySpeciesList.TAXON_KEY,
TAXON_VERSION.TAXON_VERSION_KEY,
Count(TAXON_LIST_ITEM.TAXON_LIST_ITEM_KEY) AS
CountOfTAXON_LIST_ITEM_KEY
FROM TAXON_LIST INNER JOIN ((MySpeciesList INNER JOIN (TAXON_LIST_ITEM
INNER JOIN TAXON_VERSION ON TAXON_LIST_ITEM.TAXON_VERSION_KEY =
TAXON_VERSION.TAXON_VERSION_KEY) ON MySpeciesList.TAXON_KEY =
TAXON_VERSION.TAXON_KEY) INNER JOIN TAXON_LIST_VERSION ON
TAXON_LIST_ITEM.TAXON_LIST_VERSION_KEY =
TAXON_LIST_VERSION.TAXON_LIST_VERSION_KEY) ON
TAXON_LIST.TAXON_LIST_KEY = TAXON_LIST_VERSION.TAXON_LIST_KEY
WHERE (((MySpeciesList.TAXON_LIST_ITEM_KEY) Is Null))
GROUP BY MySpeciesList.TAXON_NAME, MySpeciesList.TAXON_KEY,
TAXON_VERSION.TAXON_VERSION_KEY
HAVING (((Count(TAXON_LIST_ITEM.TAXON_LIST_ITEM_KEY))>1));

```

In effect it does not matter which of the multiple `taxon_list_item_keys` you choose for each of these species you simply need to choose one and populate your `MySpeciesList` table with the data. However, you may find that adding the following fields to your query may assist you in making a decision: `Long_Name` from `Table_List_Type` and `Item_Name` from `Taxon_List`.

Re-run query 9 as an update query without the count of `Taxon_List_Item_Key` and it will update 16 rows. The design view for this change to query 9 can be seen in figure 14 and the SQL in Query 11.

Field:	TAXON_NAME	TAXON_KEY	TAXON_VERSION_KEY	TAXON_LIST_ITEM_KEY	TAXON_LIST_ITEM_KEY
Table:	MySpeciesList	MySpeciesList	TAXON_VERSION	MySpeciesList	MySpeciesList
Update To:					[Taxon_List_item].[Taxon_List_Item_Key]
Criteria:				Is Null	Is Null
or:					

Figure 14: Design view of query 11.

Query 11: SQL to change query 9 to an update query

```

UPDATE TAXON_LIST INNER JOIN ((MySpeciesList INNER JOIN
(TAXON_LIST_ITEM INNER JOIN TAXON_VERSION ON
TAXON_LIST_ITEM.TAXON_VERSION_KEY =
TAXON_VERSION.TAXON_VERSION_KEY) ON MySpeciesList.TAXON_KEY =
TAXON_VERSION.TAXON_KEY) INNER JOIN TAXON_LIST_VERSION ON
TAXON_LIST_ITEM.TAXON_LIST_VERSION_KEY =
TAXON_LIST_VERSION.TAXON_LIST_VERSION_KEY) ON
TAXON_LIST.TAXON_LIST_KEY = TAXON_LIST_VERSION.TAXON_LIST_KEY SET
MySpeciesList.TAXON_LIST_ITEM_KEY = [Taxon_List_item].[Taxon_List_Item_Key]
WHERE (((MySpeciesList.TAXON_LIST_ITEM_KEY) Is Null));

```

There will be one row in your `MySpeciesList` table that remains unmatched (unless you have added this manually as described in section 3.3.2). The unmatched species is *Scymnus impexus* which we discovered earlier is not considered native and is therefore not in the species dictionary (see section 3.3.2)

Your data should now be complete for use elsewhere.