KE EMu Documentation

The Import tool

Document Version 1

KE EMu Version 3.2

Contents

The Import tool: import and update records

The Import tool can be used to batch import new records and to update existing records in an EMu module.

Import vs Batch load

When data is loaded into EMu using a batch load process (for instance, during the migration of legacy data) values are mapped from a field in the data source into a field in EMu. This is a quick way to get large amounts of data into EMu but it can result in records that are incomplete or that contain invalid data.

In contrast, the Import tool is a batch facility that emulates the manual creation of a new record in EMu. Consider that when a new record is created manually in EMu a whole range of background processes can be involved. For example:

- Default values can be added to the record.
- Unique values can be added (an incremental number, for instance), and values that should be unique are checked for uniqueness.
- A check that mandatory fields have been completed can be performed.
- Auto-fill fields will be auto filled!
- Any additional on-save processes specified by the system will be performed when the record is saved.

Typically, none of these are performed when records are batch uploaded into EMu. All of them are performed when the Import tool is used to import records or update them in EMu.

Data added to EMu using the Import tool is far *cleaner* than if it is simply batch uploaded.

Permissions

Users must have the appropriate permissions to use the Import tool (see page 81). In the first instance they must have the *daImport* permission for any module in which they need to import records. But keep in mind that importing records or data into EMu is no different from adding or editing records in EMu:

The *daImport* permission alone is not sufficient to allow users to import records into an EMu module. When using the Import tool to add or update records any permissions that would normally apply for the creation and editing of records will apply. Thus, permissions that have been set for a user will determine whether records will be created or edited during the import process. For instance, if a user does not normally have permission to add a value to a Lookup List, they will not be able to do so when using the Import tool. If they do not have permission to add a record to a module, they will not be able to do so when using the Import tool.

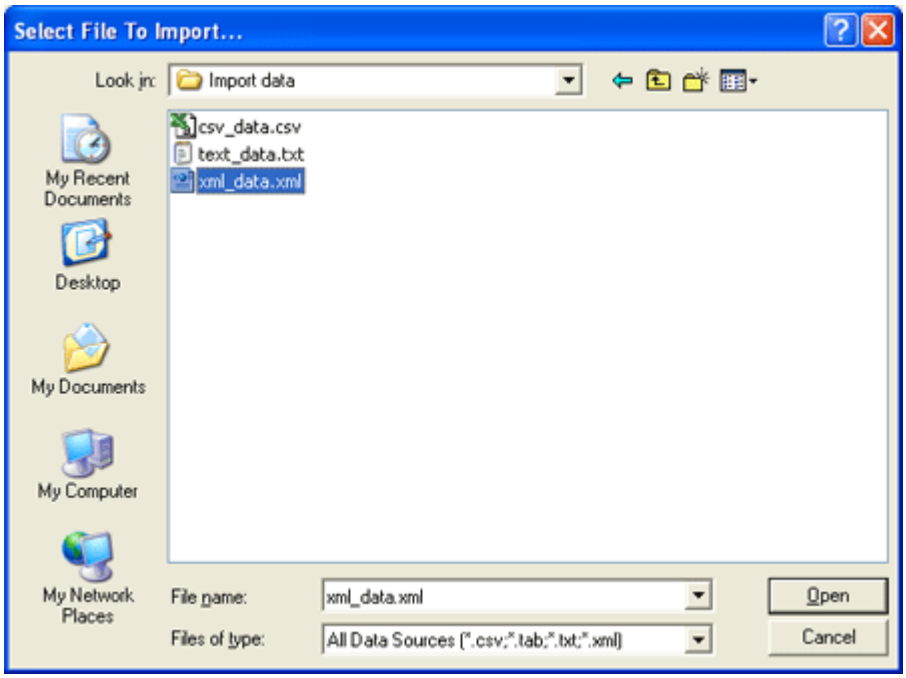
How to use the Import tool: The Import Wizard

In the module in which records are to be imported (Parties in this example):

1. Select **Tools>Import** from the Menu bar.

If the Import option is greyed out, you do not have the necessary permissions to use it (see page 81).

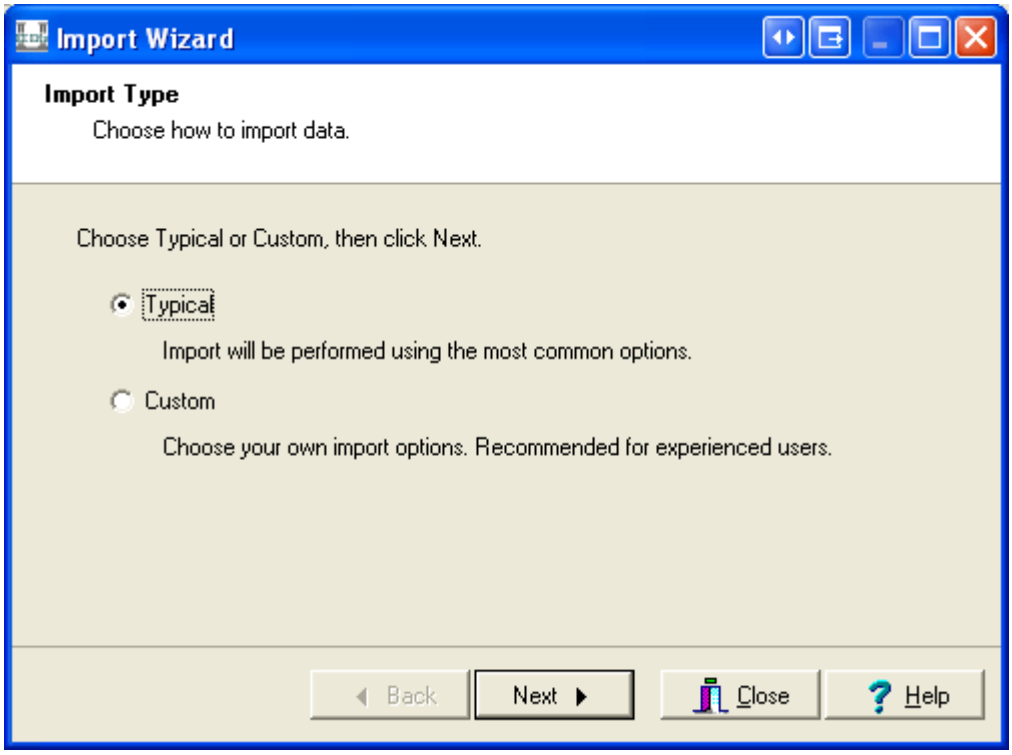
The *Select File To Import* box displays:



2. Navigate to the file that contains the data to be imported, select it and click **Open**.

   (Details about how to produce an import data file can be found on page 40.)

The Import Wizard displays:



The default option is **Typical**. Accept this option to import data using the following default import settings:

- Import data, performing format and data validation (see page 15 for details).
- If the data file references other records (see page17 for details):
  - A search is performed on all existing records in the EMu database.
  - If no match is found, a new record is added for the referenced record.
  - If one record is found, an attachment is made to that record.
  - If more than one match is found, the matching records are displayed so that the correct record can be selected manually.
- All records are imported (see page 19 for details).
- A minimal log is generated containing setup options, results and a list of errors if any occur (see page 21 for details).

3. Accept the **Typical** option

   -OR-

   Select the **Custom** radio button to change the import settings.

4. Select **Next**  to continue.

Details about a Typical import are available on page 6.

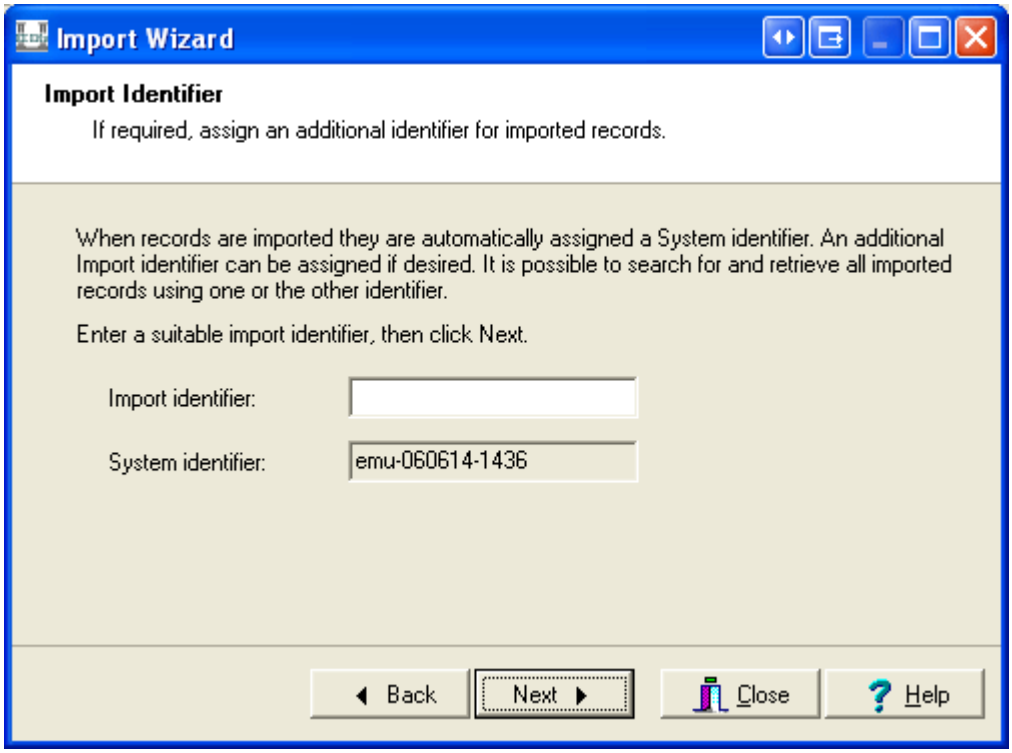
Details about a Custom import are available on page 15.

Typical import

When Typical import is selected, the Import Wizard commences with the Import Identifier screen.

Import Identifier screen

The Import Identifier screen displays:

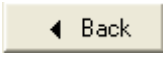


For each set of records imported a unique *System identifier* is automatically generated and saved in every record. The identifier is constructed from the user name, and the date and time (24 hour clock) the import commenced. The format of the *System identifier* is `username-yymmdd-hhmm`.

The *System Identifier* can be used to locate all records imported in a particular batch.

You have the option to add your own identifier in the *Import Identifier* field.

Although the *System Identifier* is a unique value, the *Import Identifier* can be re-used (to identify all records imported over time by the same user, for instance). For later reference, the identifiers are recorded in the log file generated with each import (typically saved in the same location as the import data file used).
See page 7 for details about the purpose and use of these identifiers.

1. Select **Back**  to return to a prior stage

-OR-

To add your own identifier for the imported records, enter it in the *Import identifier* field. Otherwise leave the field blank.

2. Select **Next** 

to continue.

   The Settings screen displays (see page 9).

System and Import identifiers

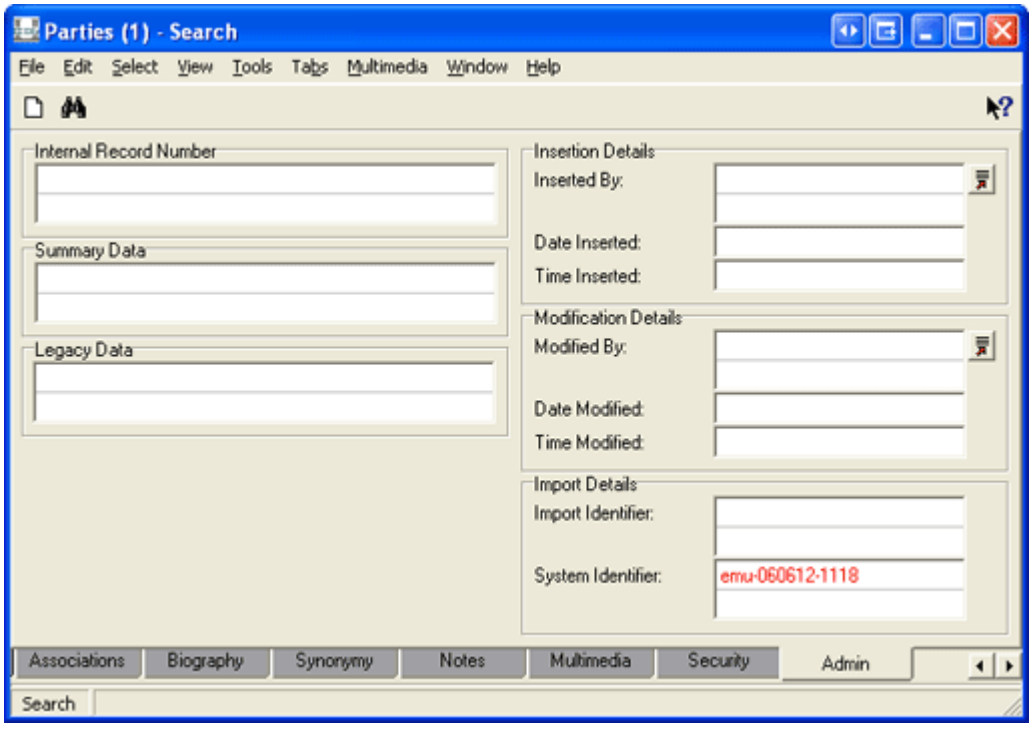
The *System* and *Import identifiers* can be used to easily identify records imported in a batch (*System Identifier*) or by whatever criteria you specify with the *Import identifier*.

The *Import identifier* can be re-used with different import batches. It could be used, for example, to identify records imported by a user (by entering the user's name in *Import identifier* for every set of records they import).

To locate imported records, search the relevant module using the *Import Details* fields on the Admin tab:

1. Enter the *System* or *Import Identifier* in the appropriate field.

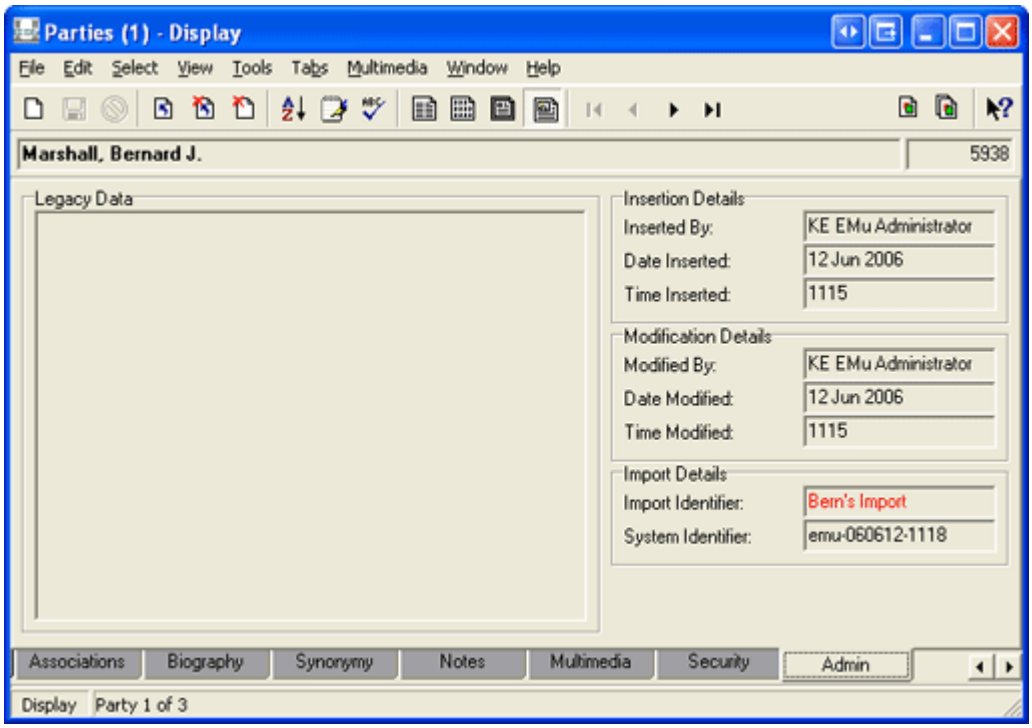
   In this example `emu-060612-1118` is entered in the *System Identifier: (Import Details)* field:



2. Run the search.

   All records imported in the `emu-060612-1118` set are returned.

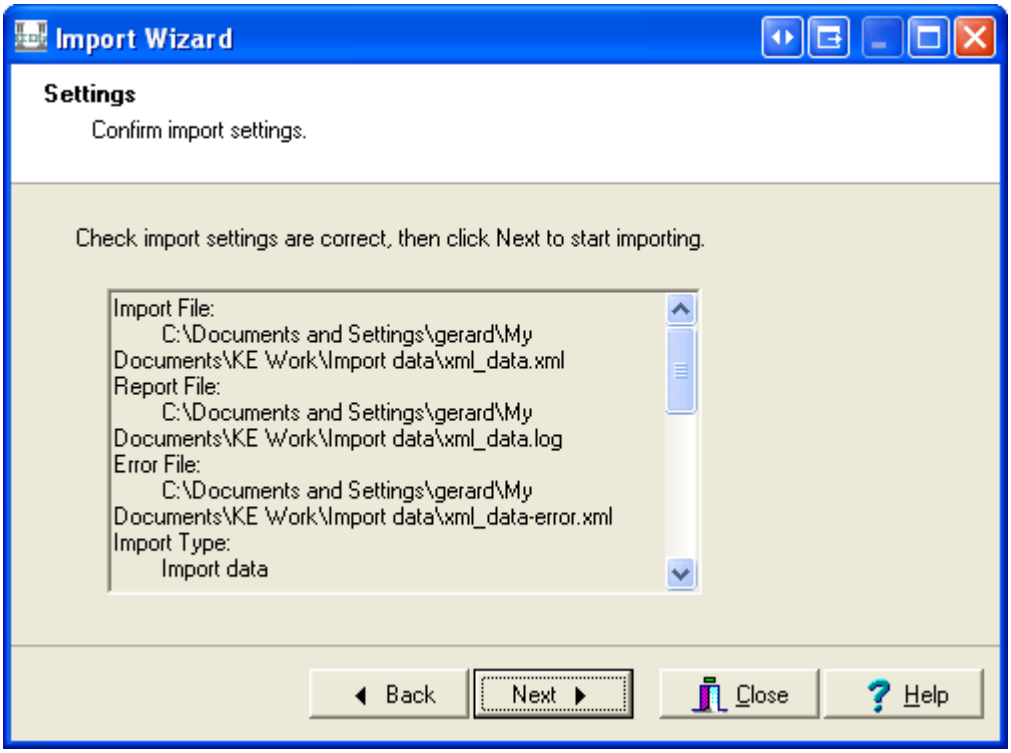
In this example an *Import Identifier* had also been added to the imported records and could have been used for the search:

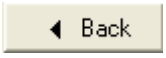


Note that once created both the *Import Identifier* and *System Identifier* are read-only and cannot be modified.

Settings screen

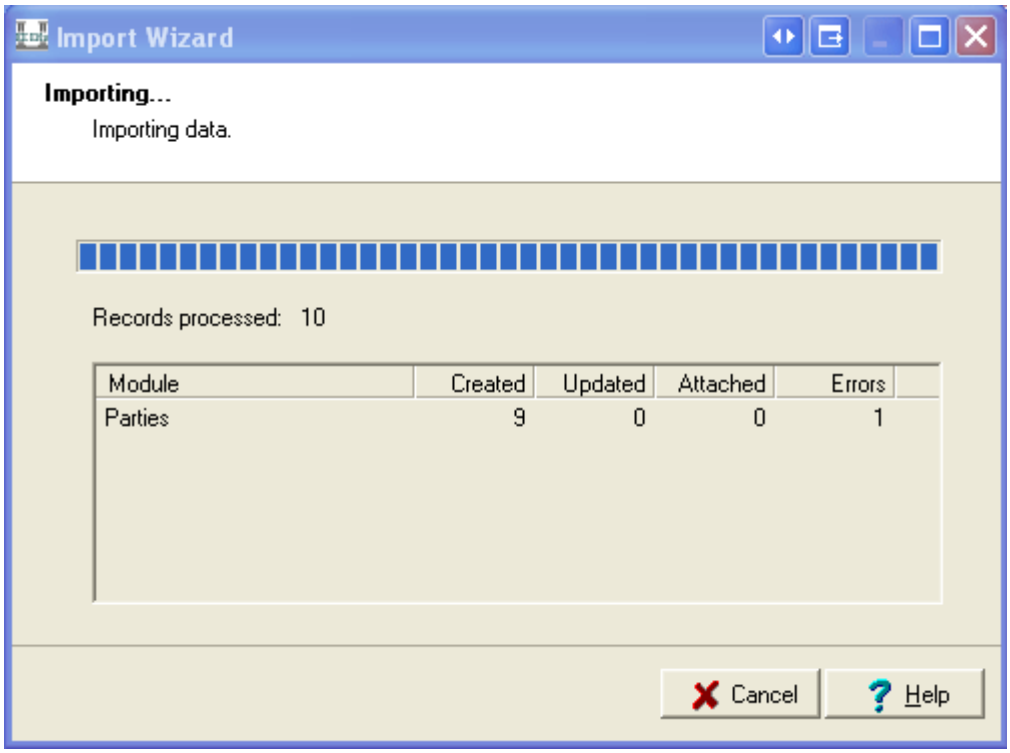
The Settings screen displays a list of the settings that will be used to import data from the selected file:



1. Scroll through the list of settings to ensure that they are correct.
2. Select **Back**  to return to a prior step and change the details
   -OR-
   Select **Next**  to proceed with the import.
   The Importing Screen displays.

Importing screen

The Importing Screen displays and processing commences:

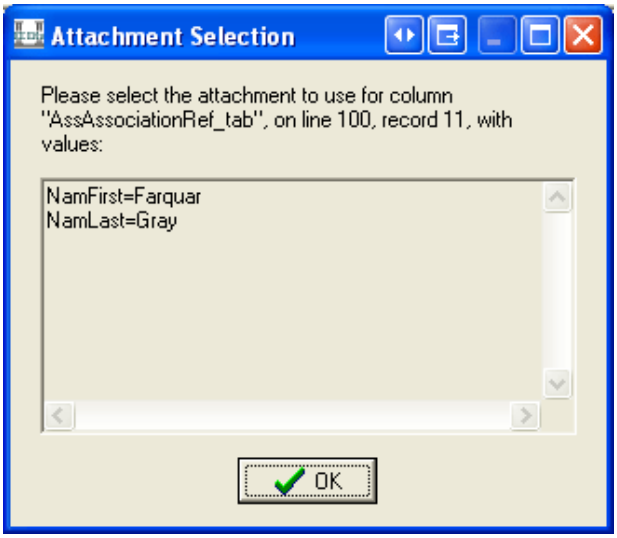


If there are references in the data file to other records (attachments), you will typically need to perform steps 1 and 2 below. If you choose a Custom Import option, it is possible to bypass these steps (see page 4).

When there is a reference in the import data file to another record, a search for the attachment record is performed using the details provided.

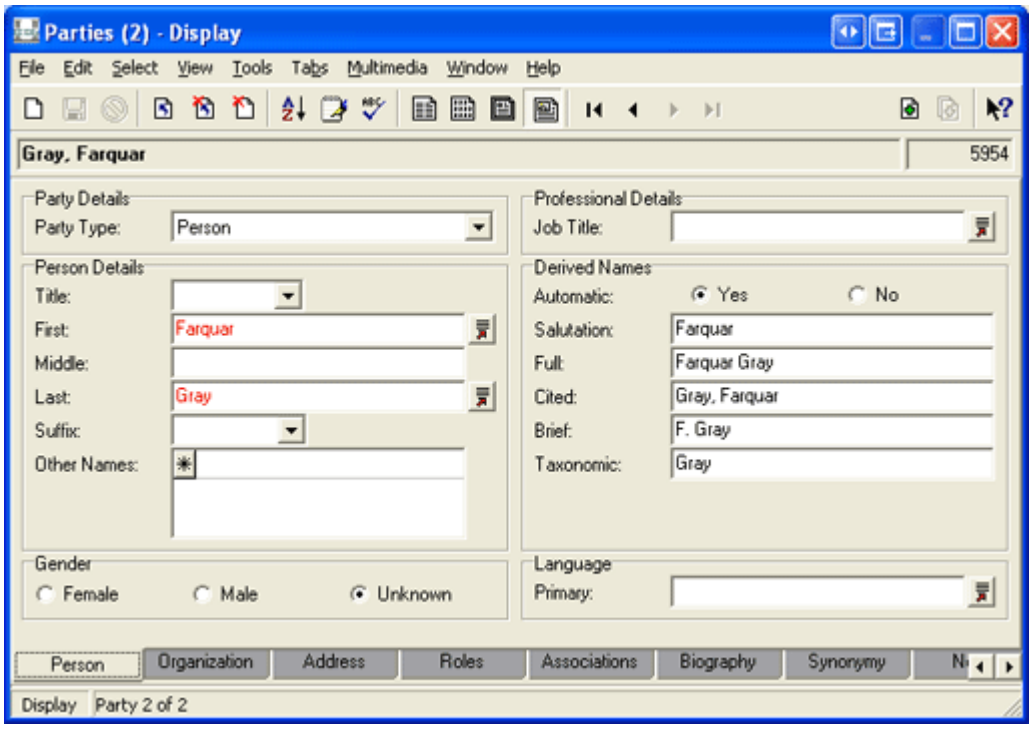
By default a search of all records, including those being imported, is made.

If more than one match is found, the Attachment Selection box displays:



In this case a record being imported seeks to attach to another Parties record with the first name `Farquar` and the last name `Gray`. Against the odds, there are two Parties records with the same details!

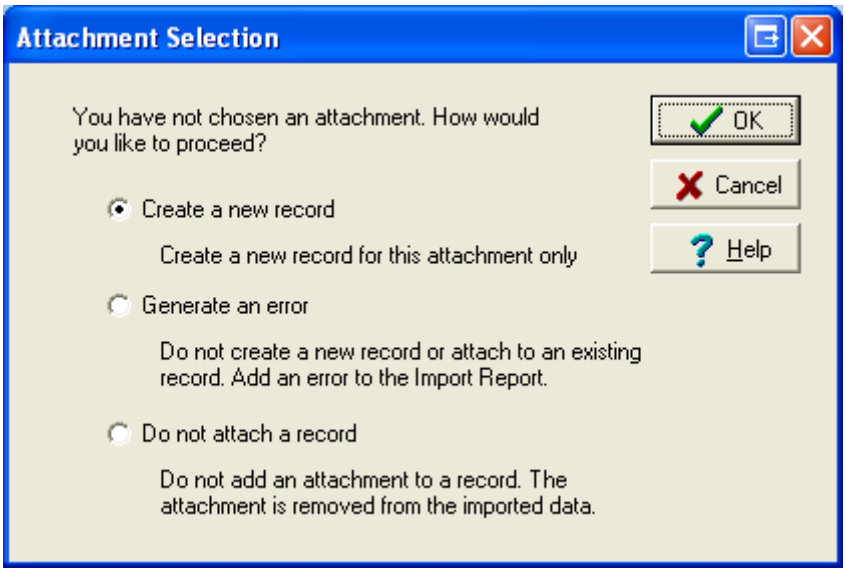
1. Select **OK**  to view the matching Parties records:



Because records being imported could reference both of these Parties records, it will probably be necessary to open the import data file, locate the record that is seeking to make this attachment and confirm that you have identified the correct attachment relationship. The record number and line number of the record in the import data file are displayed in the Attachment Selection box.

2. Identify the correct attachment record and select the **Attach Current Record**  button in the Toolbar.

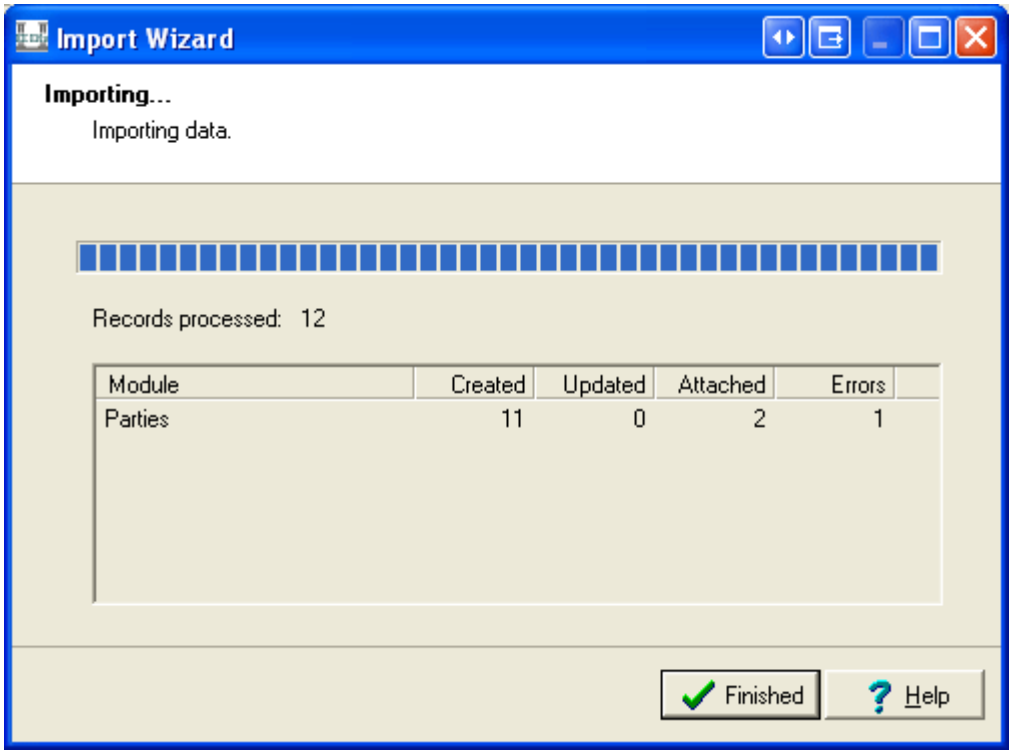
If none of the listed records is the correct attachment select the **Close** button in the module's Title bar. The Attachment Selection box displays with three options:



Select an option (the default is to create a new record) and click **OK**.

This process repeats until each attachment is resolved.

When the import process is complete, details about the import are shown, including how many records were created, how many attachments were made and the number of errors:



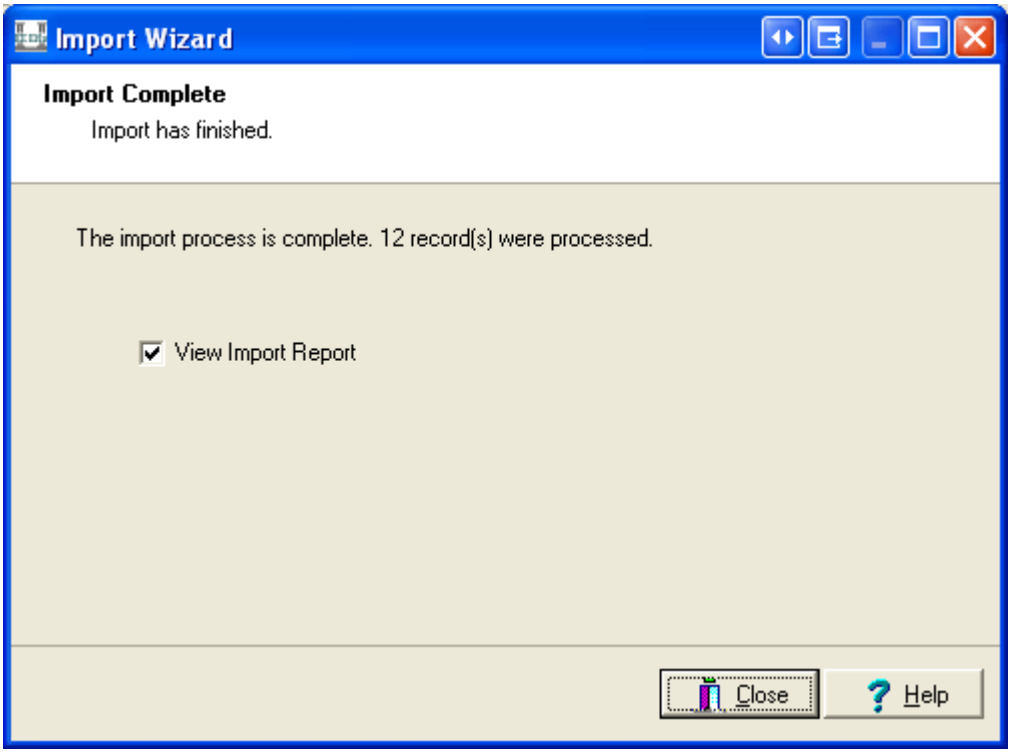
3. Select **Finished** 

.

The Import Complete screen displays.

Import Complete

The Import Complete screen displays a summary of the number of records processed:

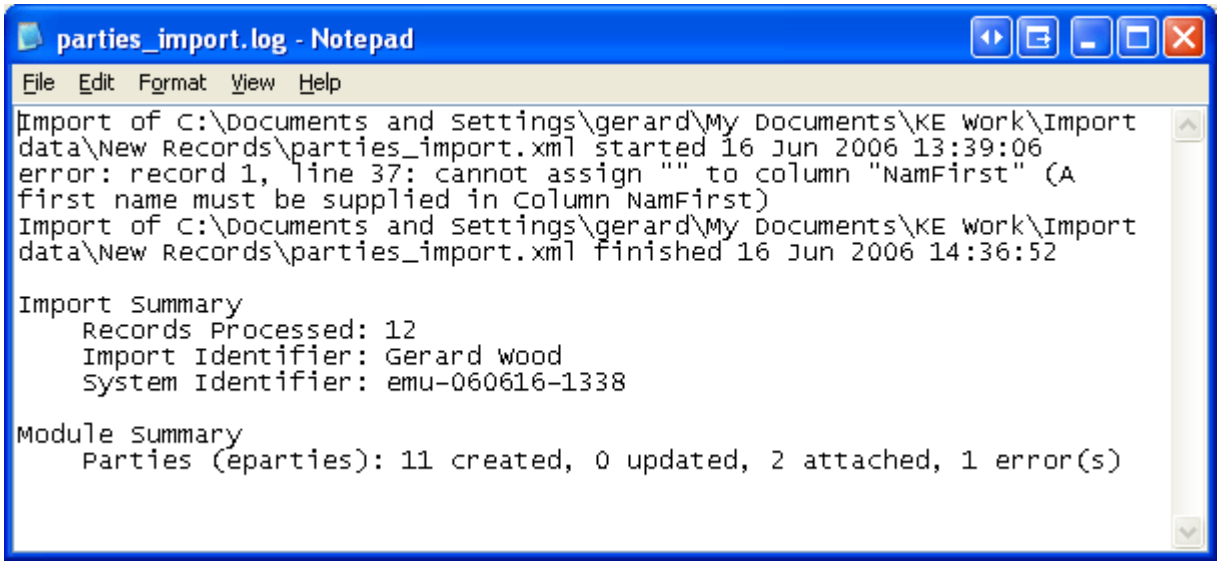


By default the *View Import Report* checkbox is checked.

1. Uncheck the *View Import Report* checkbox and select **Close**  to exit the Import Wizard **without** viewing the report

   -OR-

   Select **Close**  to exit the Import Wizard and display the *Import Report* log:



All records without errors are imported into EMu. Any records with errors are identified in the *Import Report* and are output to an error file (which is in the same format as the original import data file). Both files are typically saved to the same location as the original import data file.

The imported records will display in the module window.

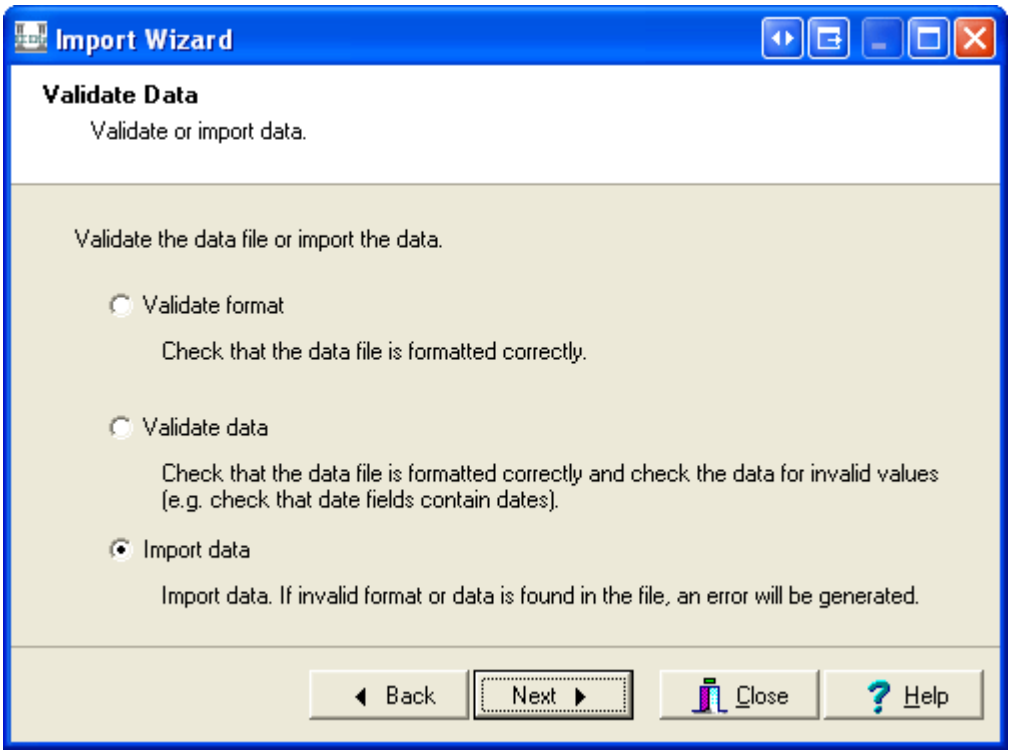
Any records listed or displaying prior to the import will be discarded: only the imported records will display.

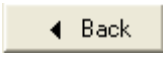
Custom import

When Custom import is selected, it is possible to change the default Import settings (see page 4). The Import Wizard commences with the Validate Data screen.

Validate Data screen

The Validate Data screen displays with three options:



1. Select **Back** 

to return to a prior stage

   -OR-

   Accept the default option (Import data)

   -OR-

   Select the radio button beside the required option:

   - **Validate format**

     Select this option only to check that the format of the import data file is valid (e.g. that the correct column names have been used). The records in the data file will not be imported into EMu. Instead a report is generated.

     You would choose this option to ensure that the format of the import data file will not generate any errors when you do choose to import it.

     When **Next** 

is selected the Records screen displays (details on page 19).

- **Validate data**

  As for **Validate format** but also checks that the data in the data file is valid (e.g. that date fields contain dates). The records in the data file will not be imported into EMu. Instead a report is generated.

  You would choose this option to ensure that the data and format of the import data file will not generate any errors when you do choose to import it.

  When **Next**  is selected the Records screen displays (details on page 19).

- **Import data**

  This is the default option. When selected the data and format of the data file will be validated and the records will be imported. If there is invalid data or format in the file:

  i. An error will be generated and recorded in the log file.

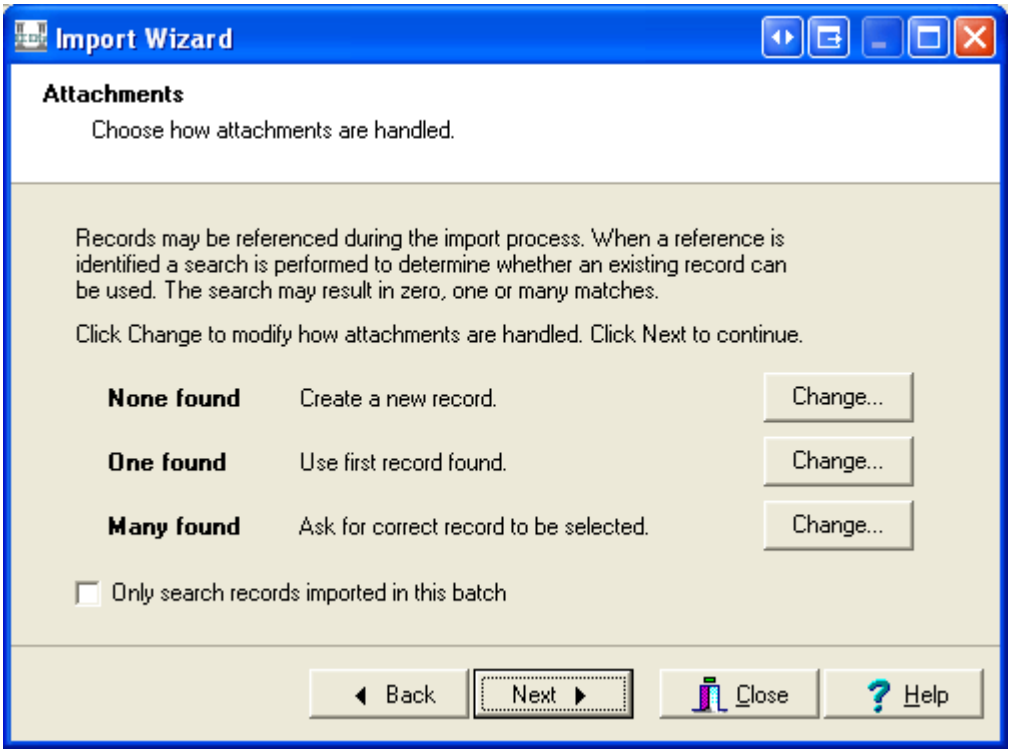
  ii. An additional file is auto-generated containing any erroneous record. This file is in the same format as the original import data file (.csv or .xml for instance) and, once the errors have been corrected, can be used to complete the import of the remaining record(s).

When **Next**  is selected the Attachments screen displays (details on page 17).

2. Select **Next**  to continue.

Attachments screen

The Attachments screen provides options to determine how attachments should be handled during the import:



About this screen

When records are imported into EMu it is possible for an attachment to other records to be specified in the import data file.

The attachment record can exist in EMu prior to the import or can be created during the import.

When an attachment is identified, a search is performed to locate the attachment record. There are three possible results:

Result	The default action is:	All options available by selecting Change:
None found	Create a new record	Create a new record
		Generate an error
One found	Use first record found	Create a new record
		Generate an error
		Use first record found
		Ask for correct record to be selected
Many found	Ask for correct record to be selected	Create a new record
		Generate an error
		Use first record found
		Ask for correct record to be selected

Either accept the default actions or select the **Change** Change... button beside a Result. The *Select Attachment Type* box displays (details on page 19).

Only search records imported in this batch checkbox

You would select this checkbox to restrict the search for attachment records to those being imported (or created) in the current batch. The default (unchecked) is to search all records.

In other words, if the box is checked only records imported in this import can be used as attachments.

On this screen:

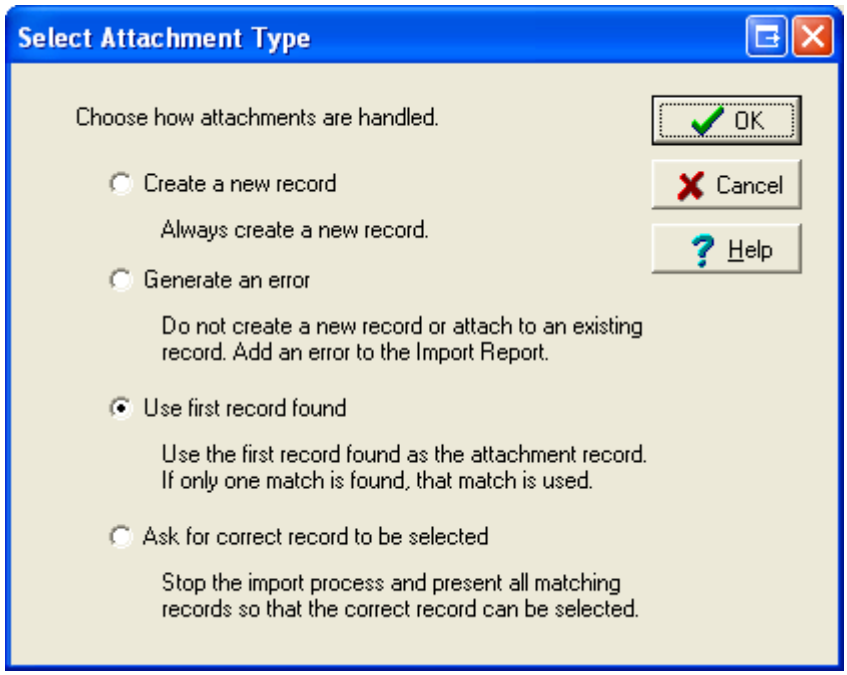
1. Accept the Default actions for each possible Result
   -OR-
   Select the **Change** Change... button beside each Result to be modified.
   The Select Attachment Type box displays (details on page 19).
   When a selection has been made, you are returned to this screen.
2. Select **Back** ◄ Back to return to a prior stage
   -OR-
   Select **Next** Next ► to continue.

The Records screen displays (details on page 19).

Select Attachment Type

The Select Attachment Type box displays when a Change button is selected on the Attachments screen (see page 17).

Options that do not apply will be greyed out and unavailable.



It is possible to change the default action when a reference is identified in the import data file:

Option	Whenever a reference is identified in the data import file...
Create a new record	...but no record is found in EMu, always create a new record and attach to it.
Generate an error	...but no record is found in EMu, do not create a new record. Instead add an error to the log file.
Use first record found	...and one or more records are found in EMu, attach to the first record found. If only one record is found, that one is used.
Ask for correct record to be selected	...and one or more records are found in EMu, display the matching record(s) so that the correct one can be selected.

On this screen:

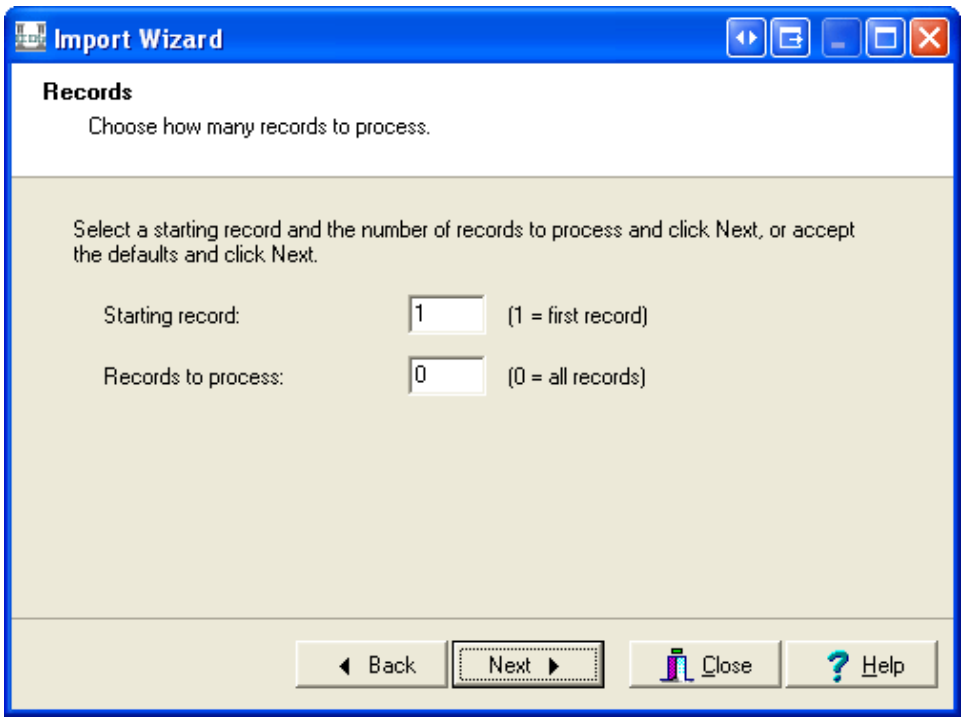
1. Select the radio button for the required option.

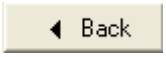
Click OK to return to the Attachments screen (see page 17).

Records screen

The Records screen provides options to specify:

- The record in the import data file from which to commence the import (the Starting record); and
- The number of records in the data file to import.



1. Select **Back**  to return to a prior stage

   -OR-

   To import all records in the data file, do not change the default settings

   -OR-

   Specify the following:

   - **Starting record**

     Enter the number of the record in the import data file from which to commence the import.

Note that import data files are said to be *one-based*, that is the first record is number 1.

     To import all records from this one on, leave *Records to process* as 0.

   - **Records to process**

     Enter the number of records to process from the *Starting record*.

If a number other than 1 is entered in the *Starting record* field and you wish to import all records from that one onwards, leave *Records to process* as 0.

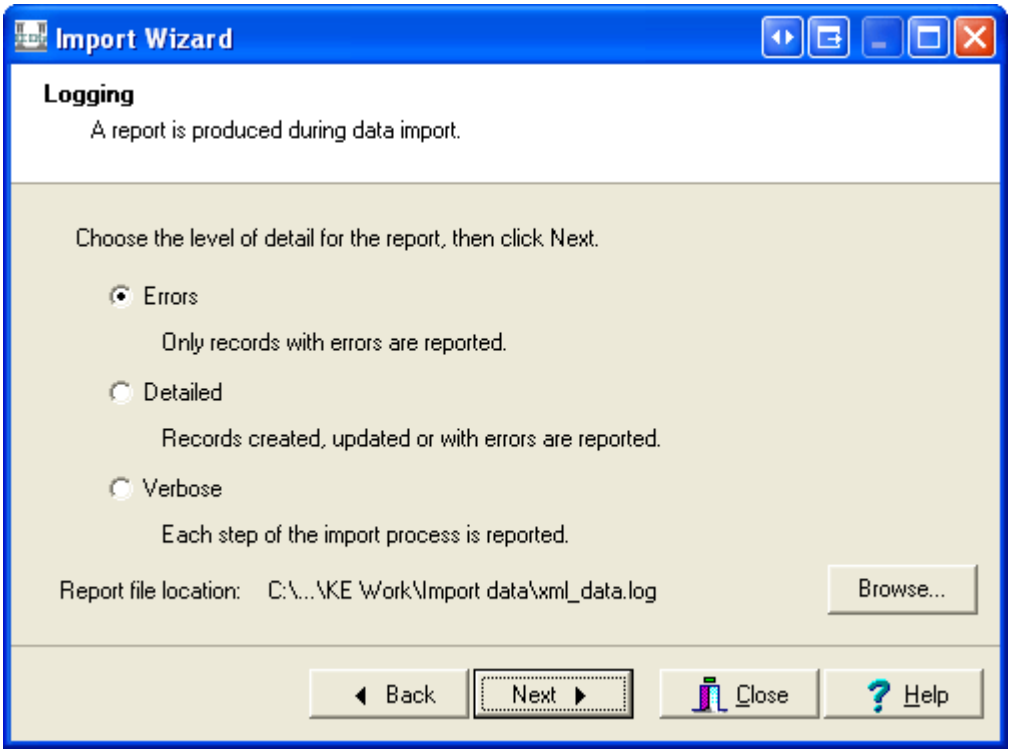
2. Select **Next**  to continue.

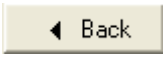
   The Logging screen displays.

Logging screen

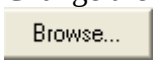
The Logging screen provides options to specify:

- How much detail to include in the auto-generated *Import Report*.
  By default only import settings, import results and errors are reported.
- Where to save the auto-generated Import Report and the error file (which is generated if there are errors).
  By default the files are saved in the same location as the data file used for the import.



1. Select **Back**  to return to a prior stage
   -OR-
   Select a radio button beside a report type:
   - **Errors**
     This is the default option. If errors occur, they are recorded in the *Import Report*. The file name for the report is the same as the import data file used, but with a *.log* extension, e.g. *import_data.log*.
   - **Detailed**
     A more detailed report will be generated, including which records were created, which were updated and which records have errors.
   - **Verbose**
     Every step of the import is reported.

 The *Import Report* will display (by default) at the end of the import process.

2. Change the save location for the *Import Report* by selecting **Browse**
   

   -OR-
   Accept the default save location.
3. Select **Next** 

   to continue.
   (The remaining steps are the same as for the Typical Import.)
   The Import Identifier screen displays (details on page 6).

Example Imports (with attachments to other records)

In the following examples an XML data file is used to import records into the Parties module. The steps described would be the same for importing a .csv or .txt data file.

Both a Typical (page 23) and Custom (page 30) import are demonstrated.

Typical import

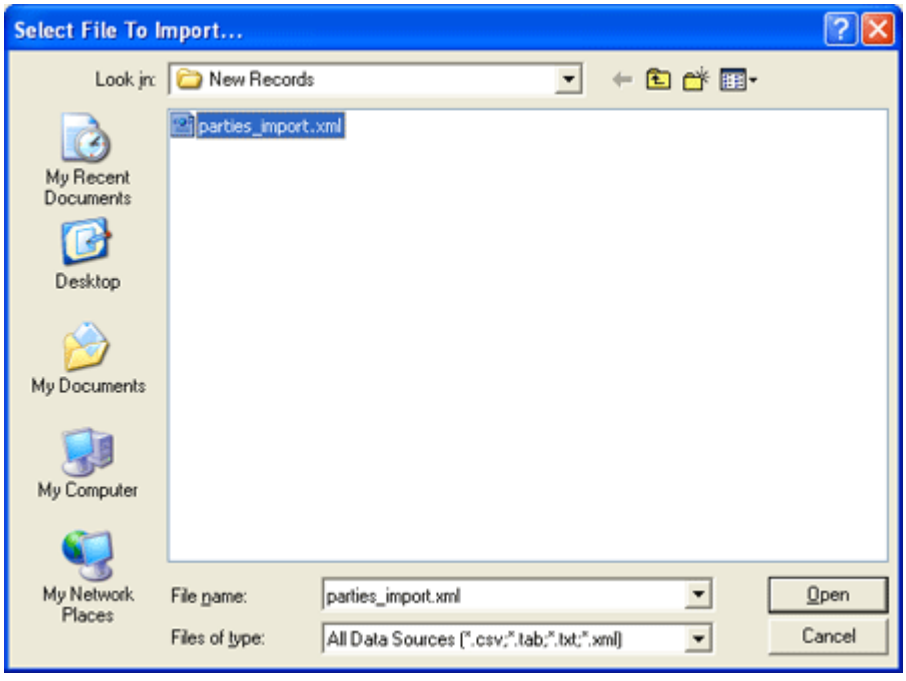
In this example the records to be imported are specified in an XML file (*parties_import.xml*), and are imported into the Parties module using the Typical import settings. Each step is described in detail in *How to use the Import tool* on page 4.

1. In the Parties module, select **Tools>Import** from the Menu bar.

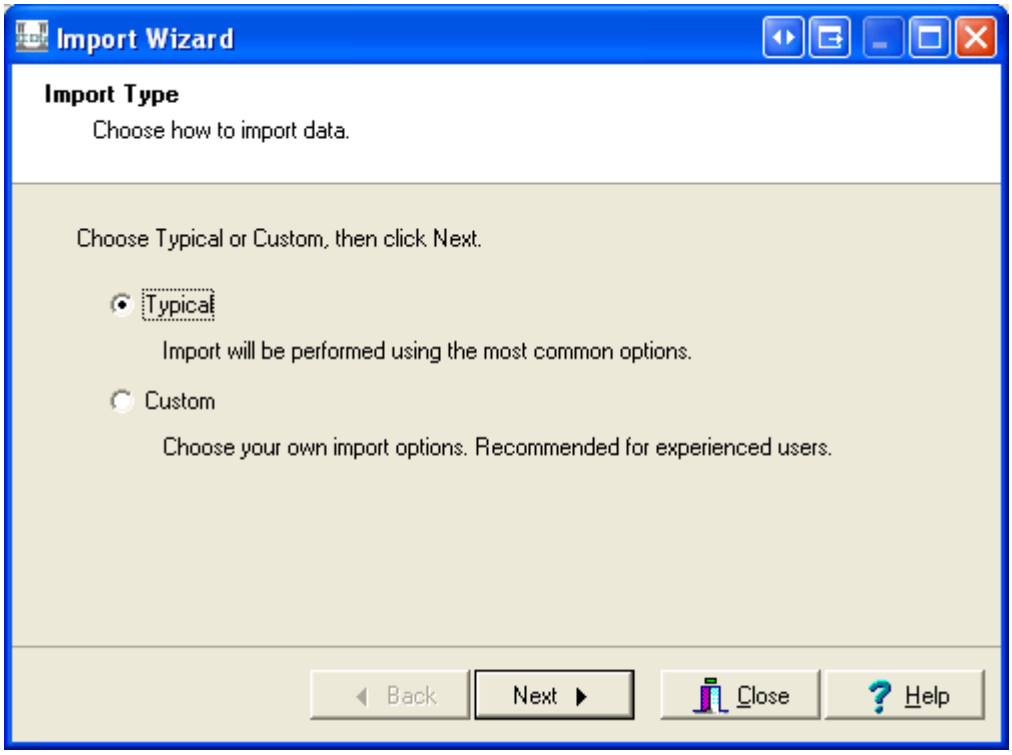
If the Import option is greyed out, you do not have the necessary permissions to use it (see page 81).

The *Select File To Import* box displays.

2. Navigate to the file that contains the data to be imported, select it and click **Open**:



The Import Wizard displays:

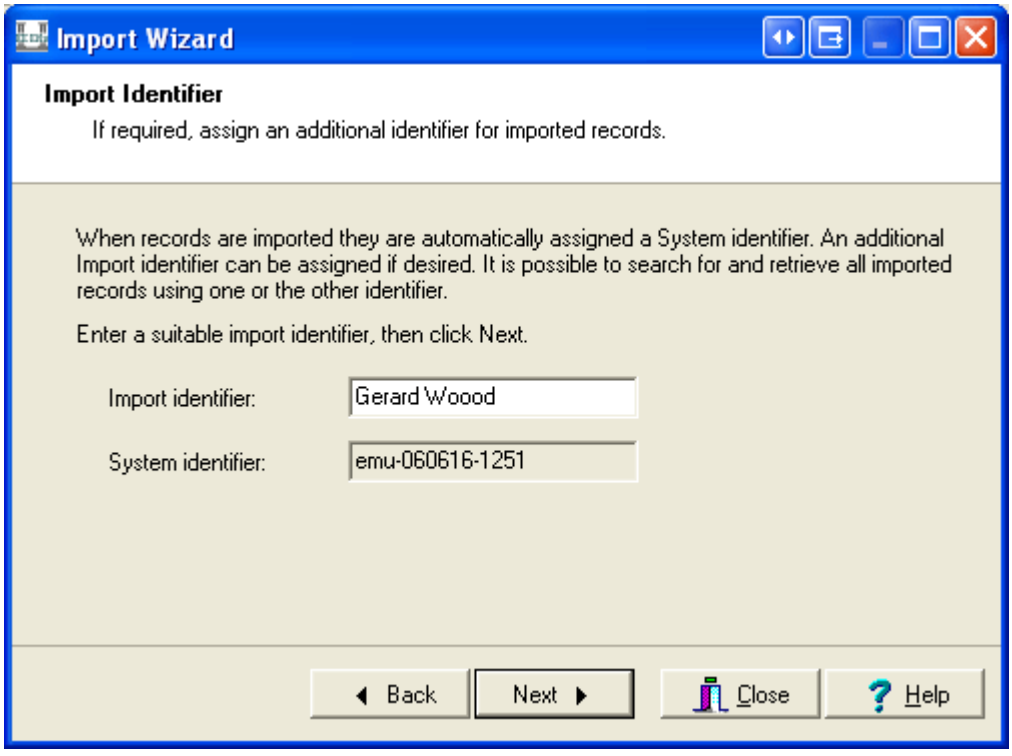


The default option is **Typical**. Accept this option to import data using the default import settings (described on page 4).

3. Select **Next**  to continue.

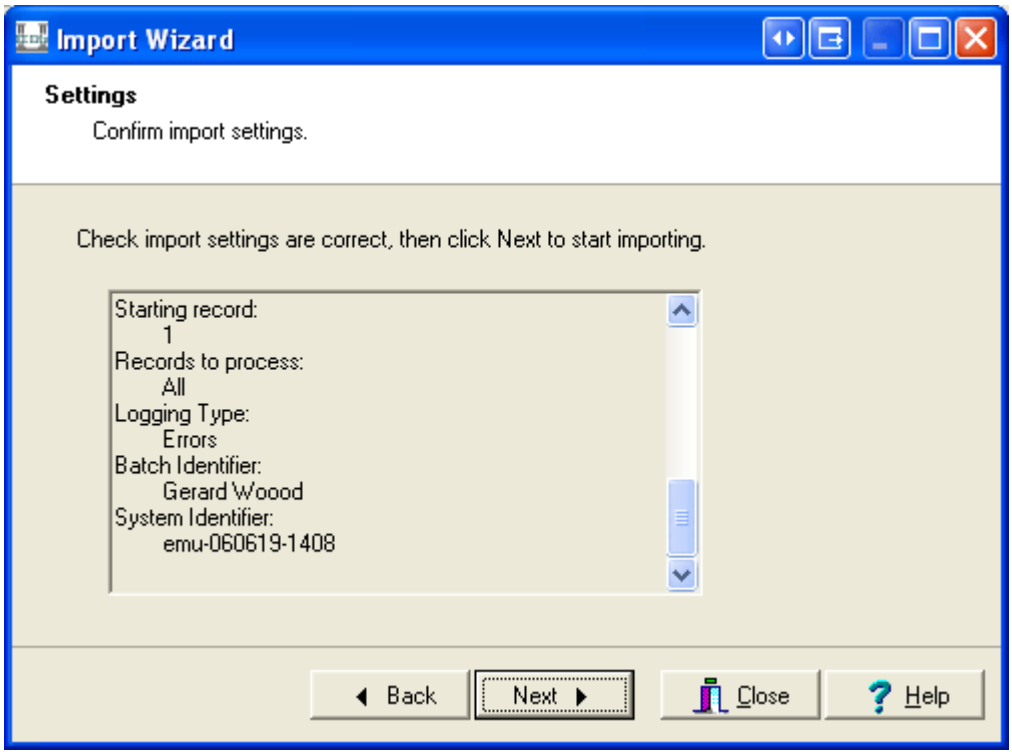
   The Import Identifier screen displays.

4. In this example an *Import identifier* is added to more easily identify who imported the records in this batch:

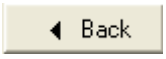


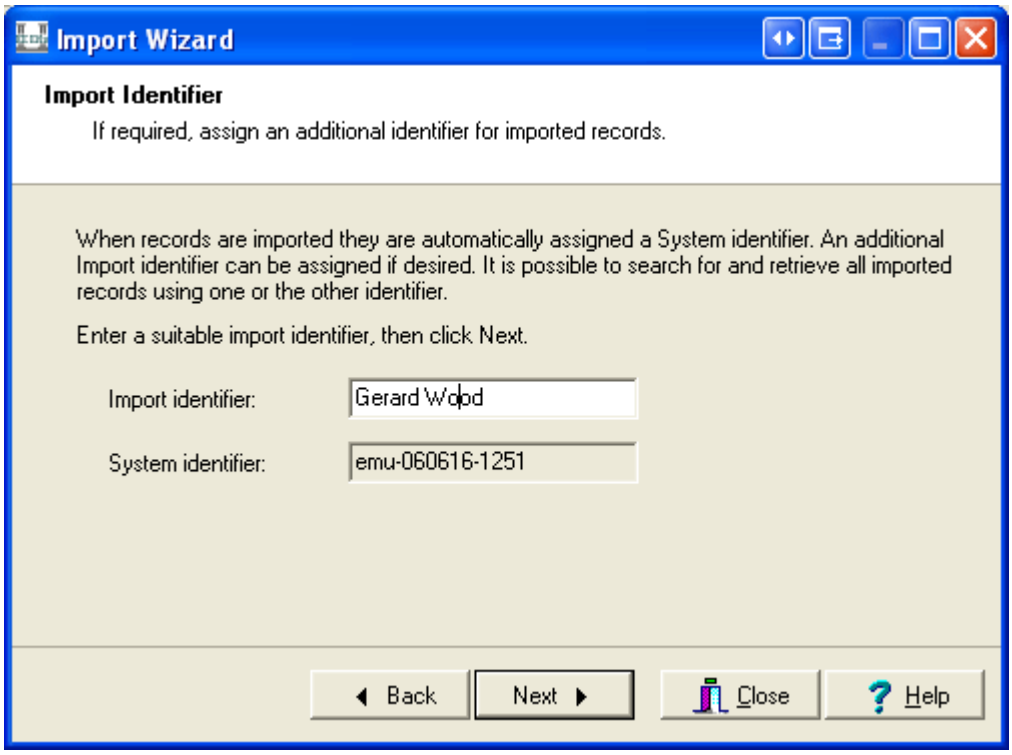
5. Select **Next**  to continue.

   The Settings screen displays with a summary of the settings to be used for this import. Scroll through the list to confirm the settings are correct:



   In this case we note that the Import Identifier has been misspelled (`Gerard Woood`).

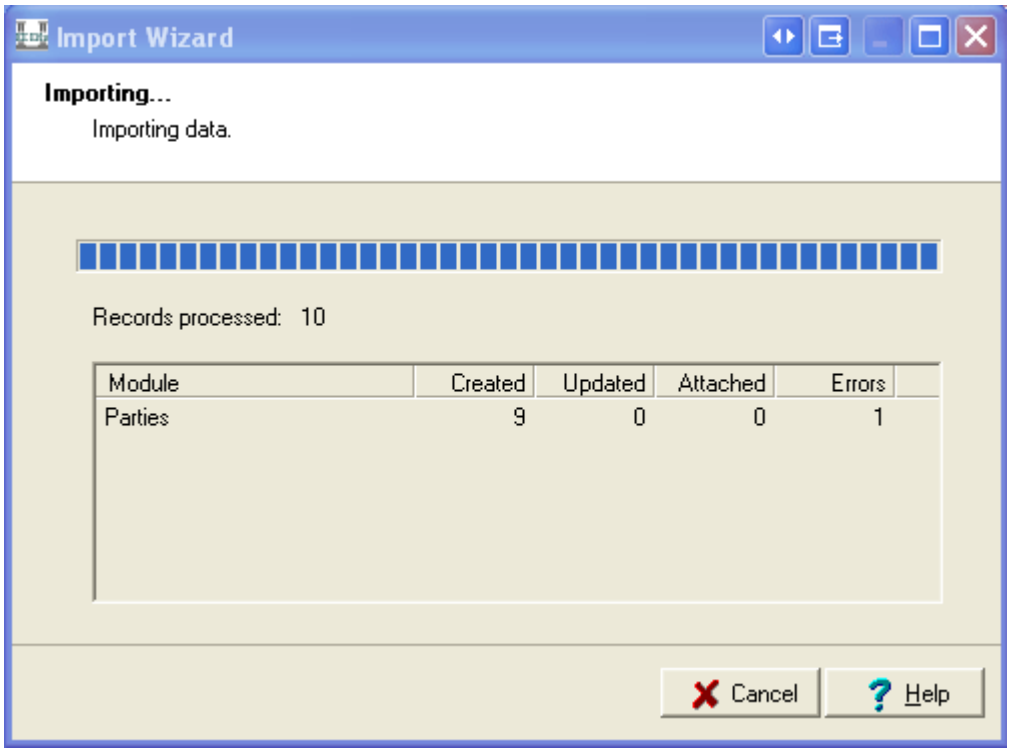
6. Select **Back**  to return to a prior step and correct the details:



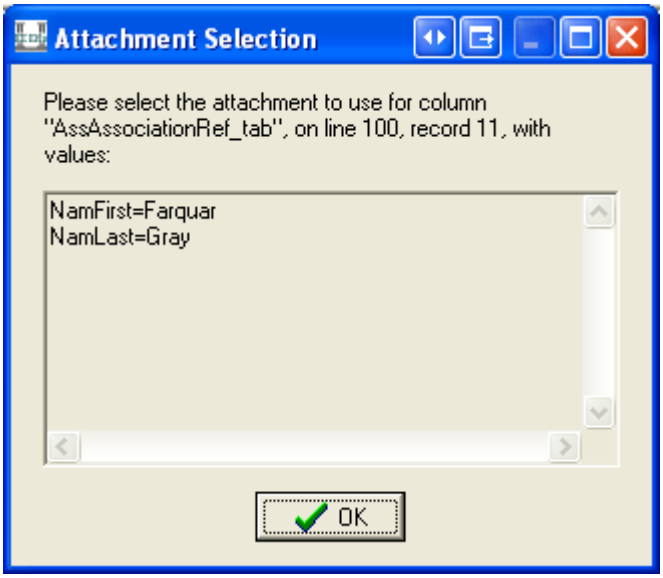
7. Select **Next**  to return to the Settings screen.

8. On the Settings screen select **Next**  to begin processing the import.

   The Importing Screen displays and processing commences:

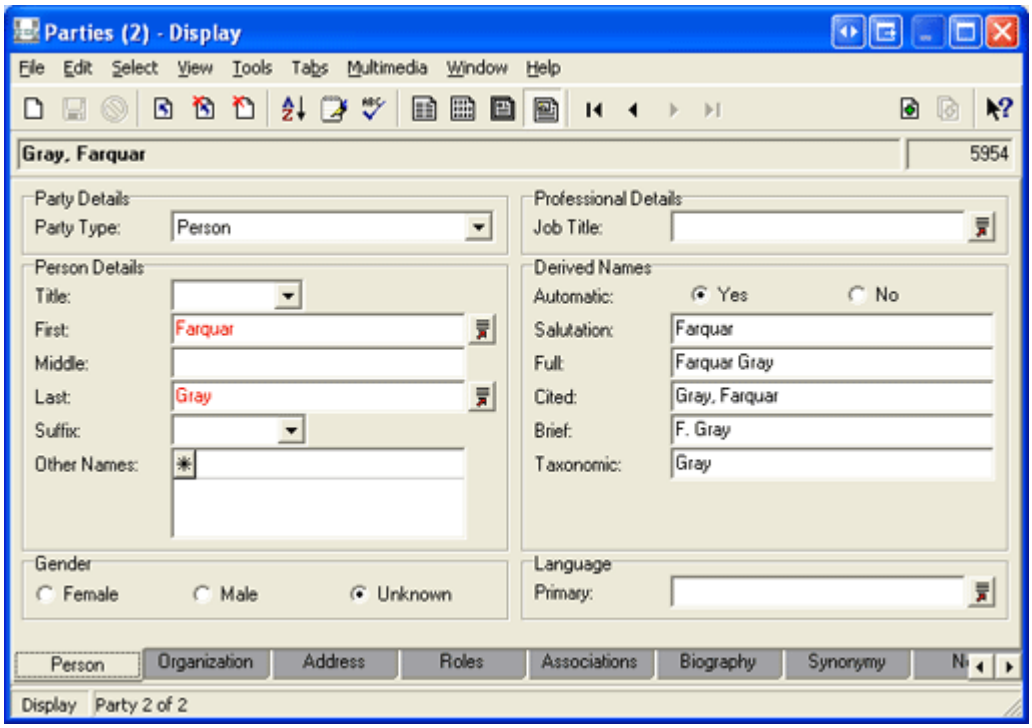


   If attachments are specified in the data file a search is performed using the details provided. If more than one match is found for the attachment record, the default action is for processing to stop and the matching records to be displayed in the appropriate module, allowing you to select the correct match:



In this case a record being imported seeks to attach to another Parties record with a first name `Farquar` and a last name `Gray`. There are two parties records with the same details.

i. Select **OK**  to view the matching records in the Parties module:

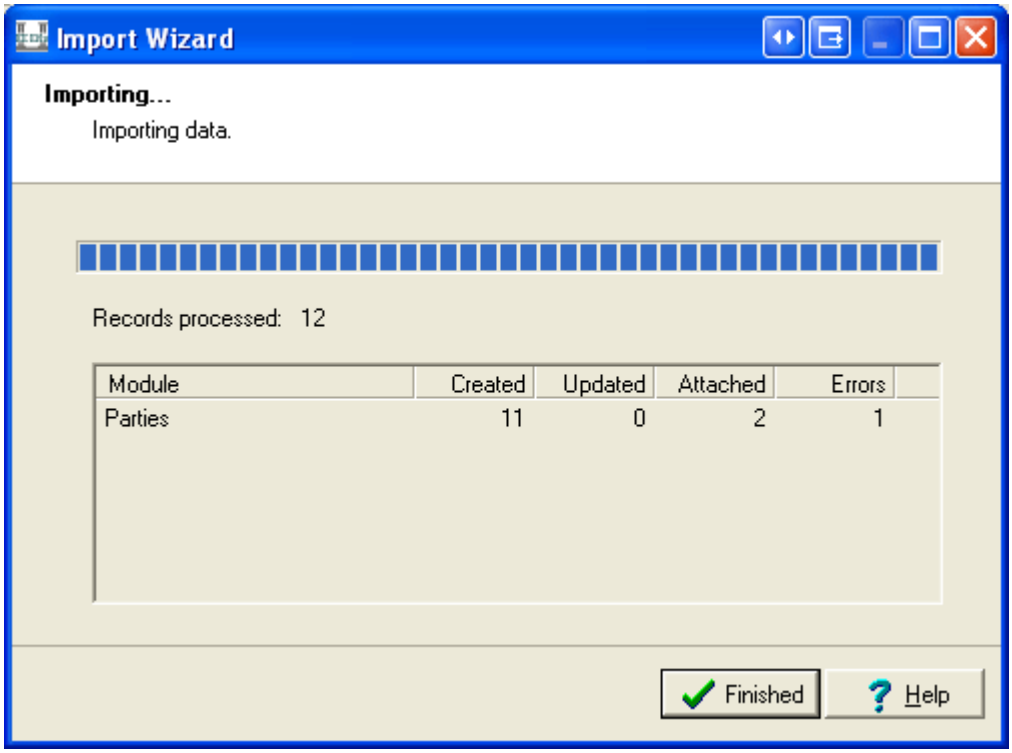


Because records being imported could reference both of these Parties records, it will probably be necessary to open the import data file, locate the record that is seeking to make this attachment and confirm that you have identified the correct attachment relationship. The record number and line number of the record in the import data file are displayed in the Attachment Selection box (shown above) to assist you to locate this record in the import data file.

ii. Identify the correct record in the attachment module and select the **Attach Current Record** button in the Toolbar.

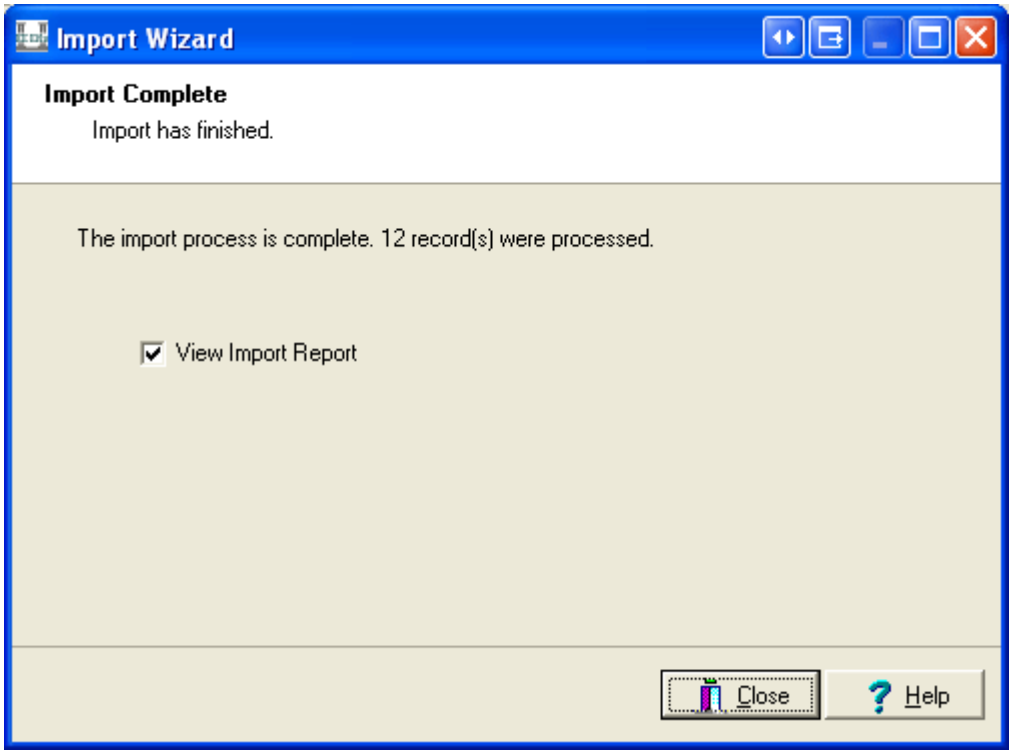
This process will repeat until all attachments are resolved.

When the import process is complete, details about the import are shown, including how many records were created, how many attachments were made and the number of errors:

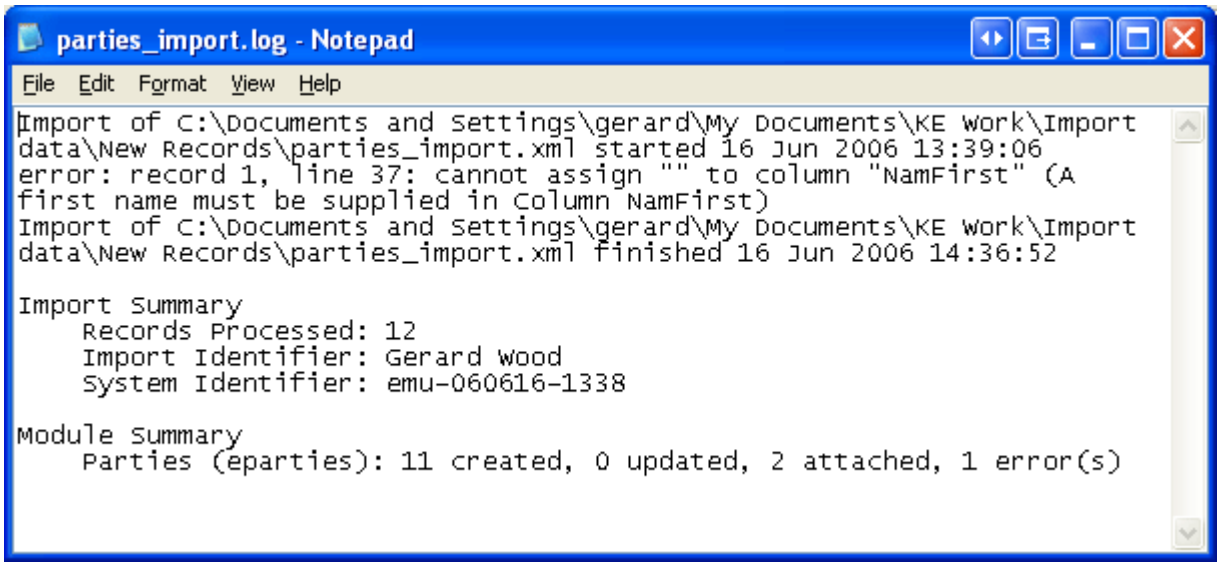


9. Select **Finished** Finished.

The Import Complete screen displays:



10. Select **Close**  to exit the Import Wizard and display the *Import Report* log:



About the auto-generated log and error file

A report will be generated during the import process. In the event of any errors, a data file will also be generated (`filename_error.xml` for example) in the same format as the file used in the data import. Typically these files are saved in the same location as the original import data file.

By default the report contains:

1. A log of the settings and the success of the import.

2. A description of any errors generated during the import.

The error file is a copy of the data file used for the import but containing only erroneous records. It can be used for iterative debugging. If there are any errors:

1. Locate the error in the error file and fix it.
2. Try the import again using the modified error file.
   If there are still errors, another version of this file will be generated.

Custom import

In this example the records to be imported are specified in an XML file (*parties_import.xml*), and are imported into the Parties module using the Custom Import settings.

Select this option to change any of the default settings used to import data into EMu. The default settings are listed on page 4.

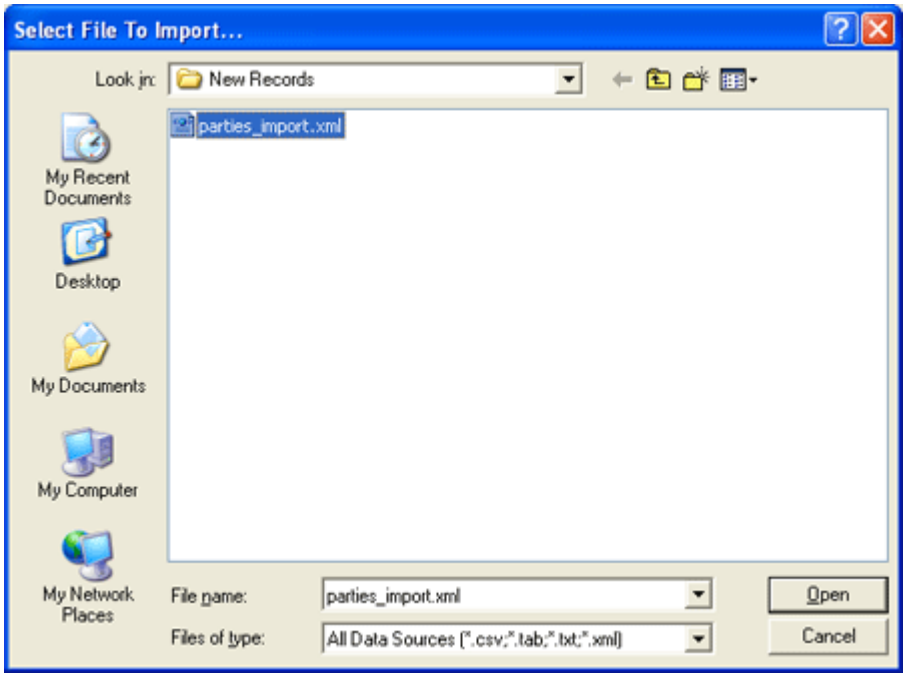
Each step is described in detail in *How to use the Import tool* from page 4 onwards.

1. In the Parties module, select **Tools>Import** from the Menu bar.

If the Import option is greyed out, you do not have the necessary permissions to use it (see page 81).

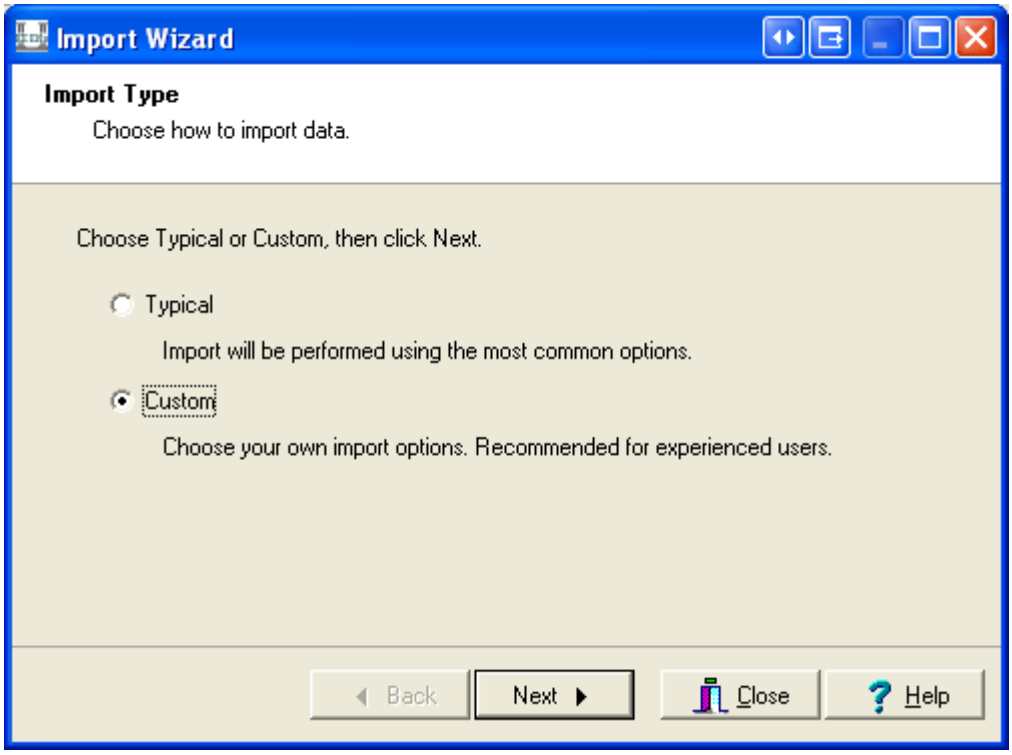
The *Select File To Import* box displays.

2. Navigate to the file that contains the data to be imported, select it and click **Open**:



The Import Wizard displays.

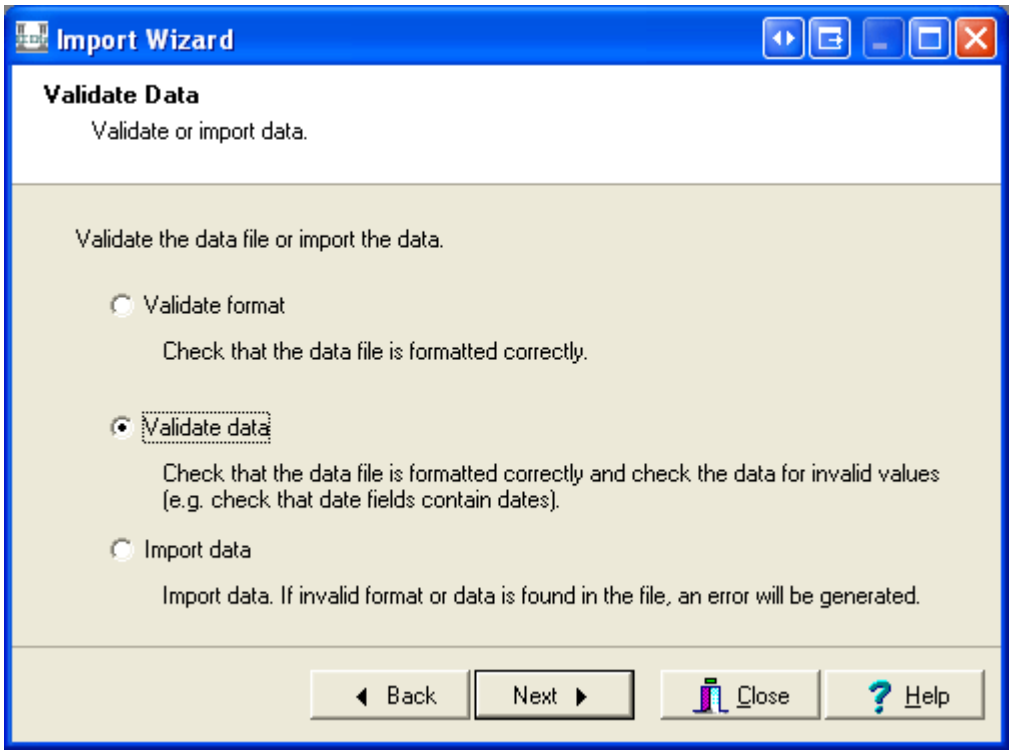
3. Select the **Custom** radio button:



4. Select **Next**  to continue.

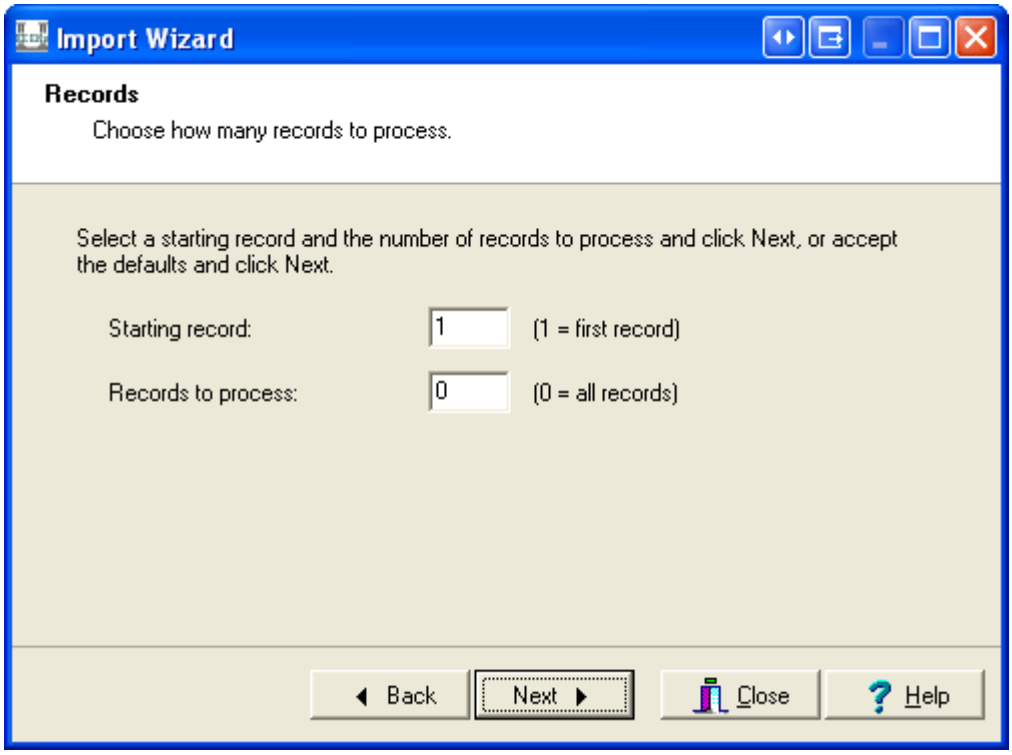
The Validate Data screen displays. In this example we'll first check that the format and data in the import data file are valid.

5. Select the **Validate Data** option:



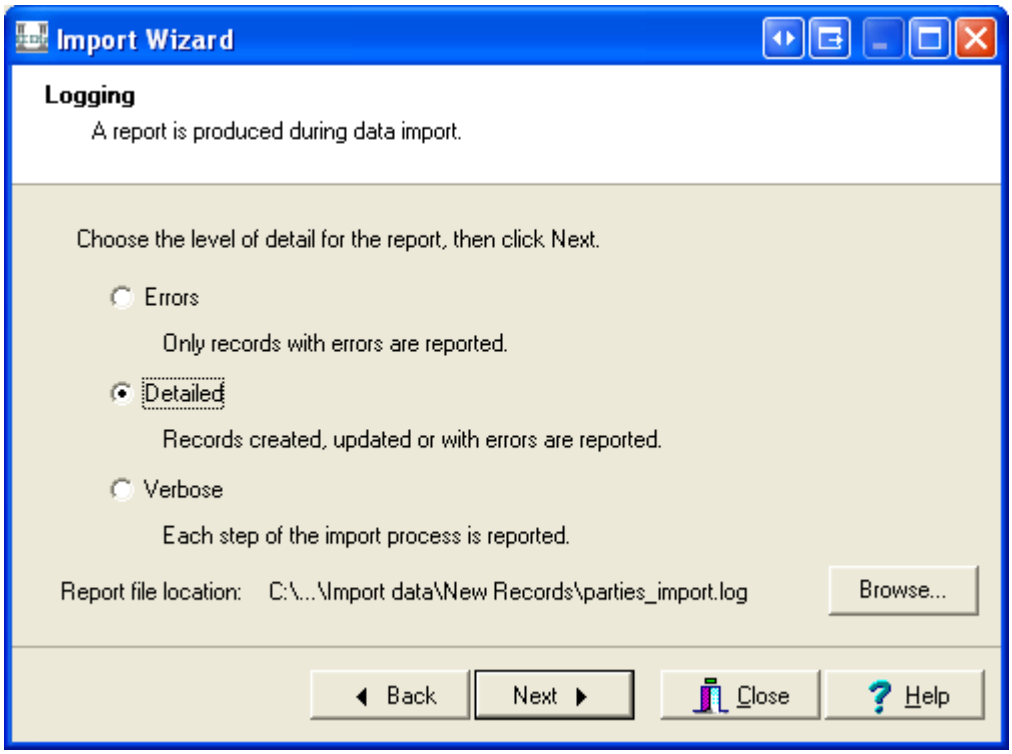
6. Select **Next**  to continue.

The Records screen displays:



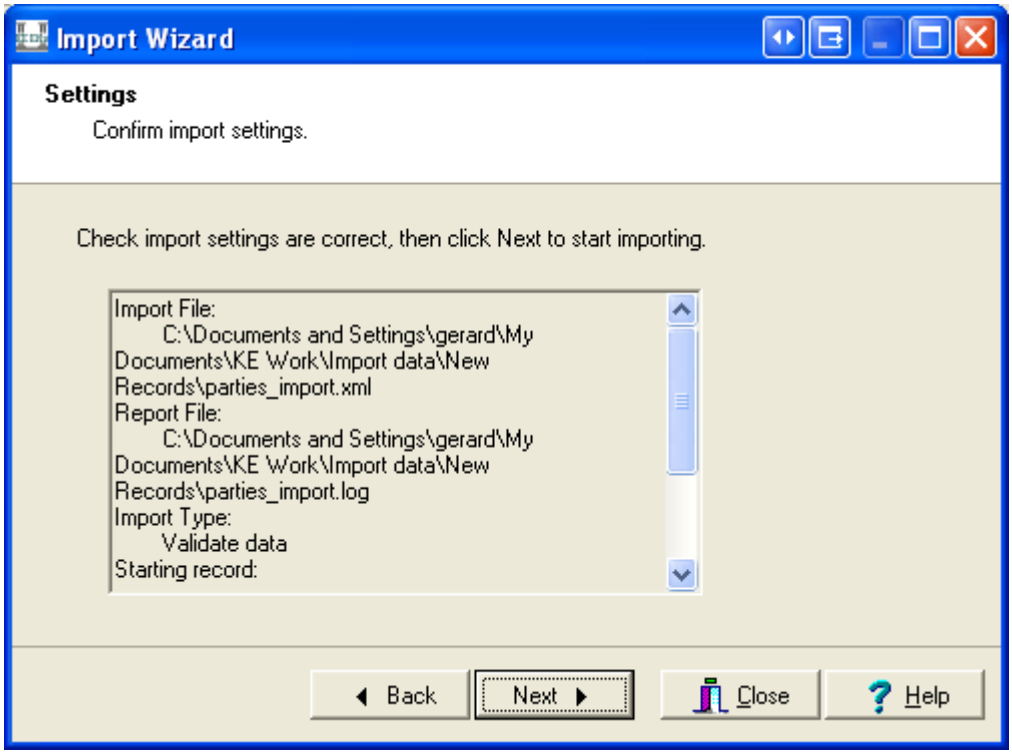
7. We want to validate all records in the import data file, so accept the default settings and select **Next**  to continue.

The Logging screen displays.
8. Select the **Detailed** option:

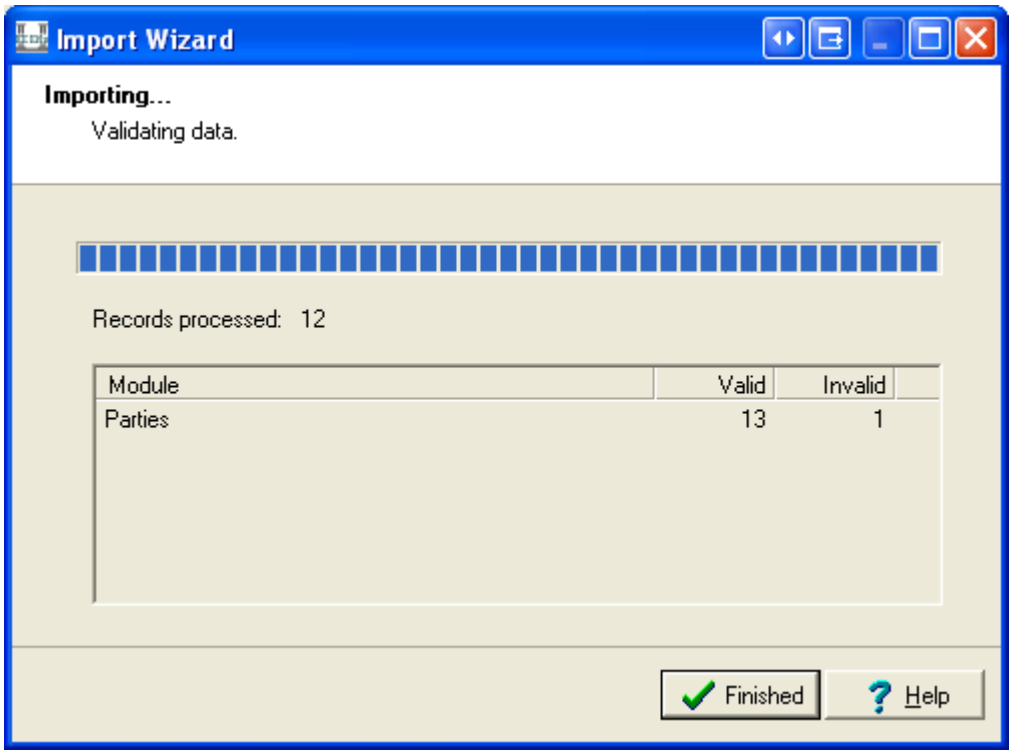


9. Select **Next**  to continue.

The Settings screen displays:



10. Scroll through the list of settings to ensure that they are correct.
    These settings are fine.
11. Select **Next**  to proceed with the import.
    The validation is processed and the Importing Screen displays with a summary of the validation results:



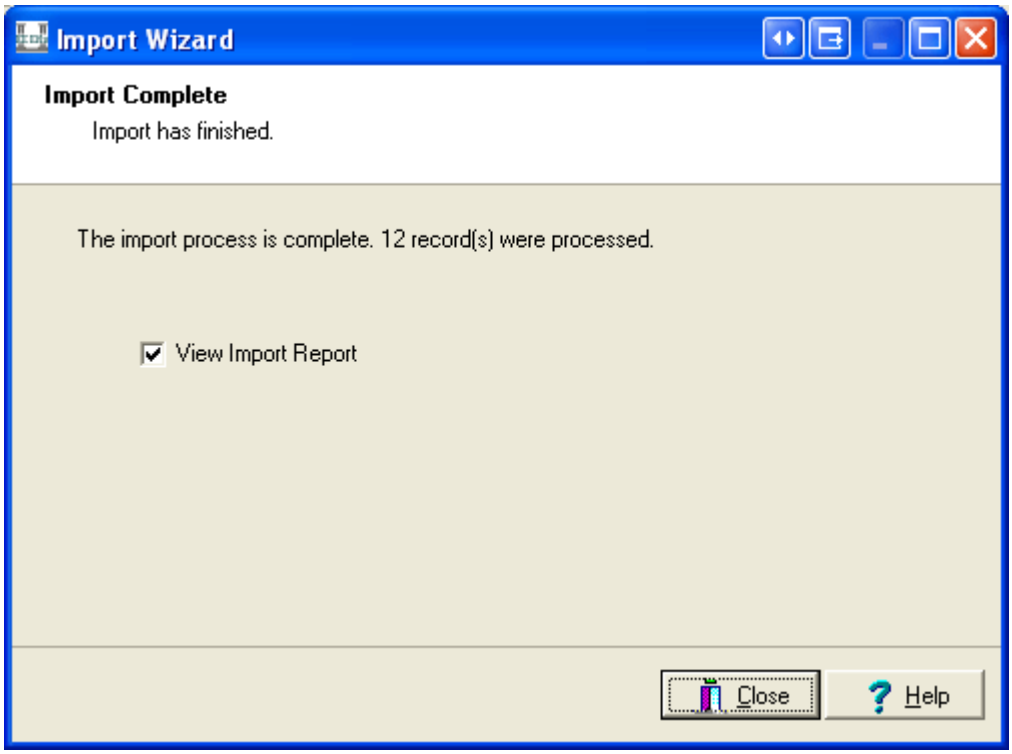
The *Records processed* count is the number of complete records processed in the import data file (a complete record can contain references to one or more attachment records). Notice that the total number of records listed for the module is greater. This count includes any attachment records processed in EMu as a consequence of the import.

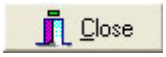
Note that one record is found to be invalid.

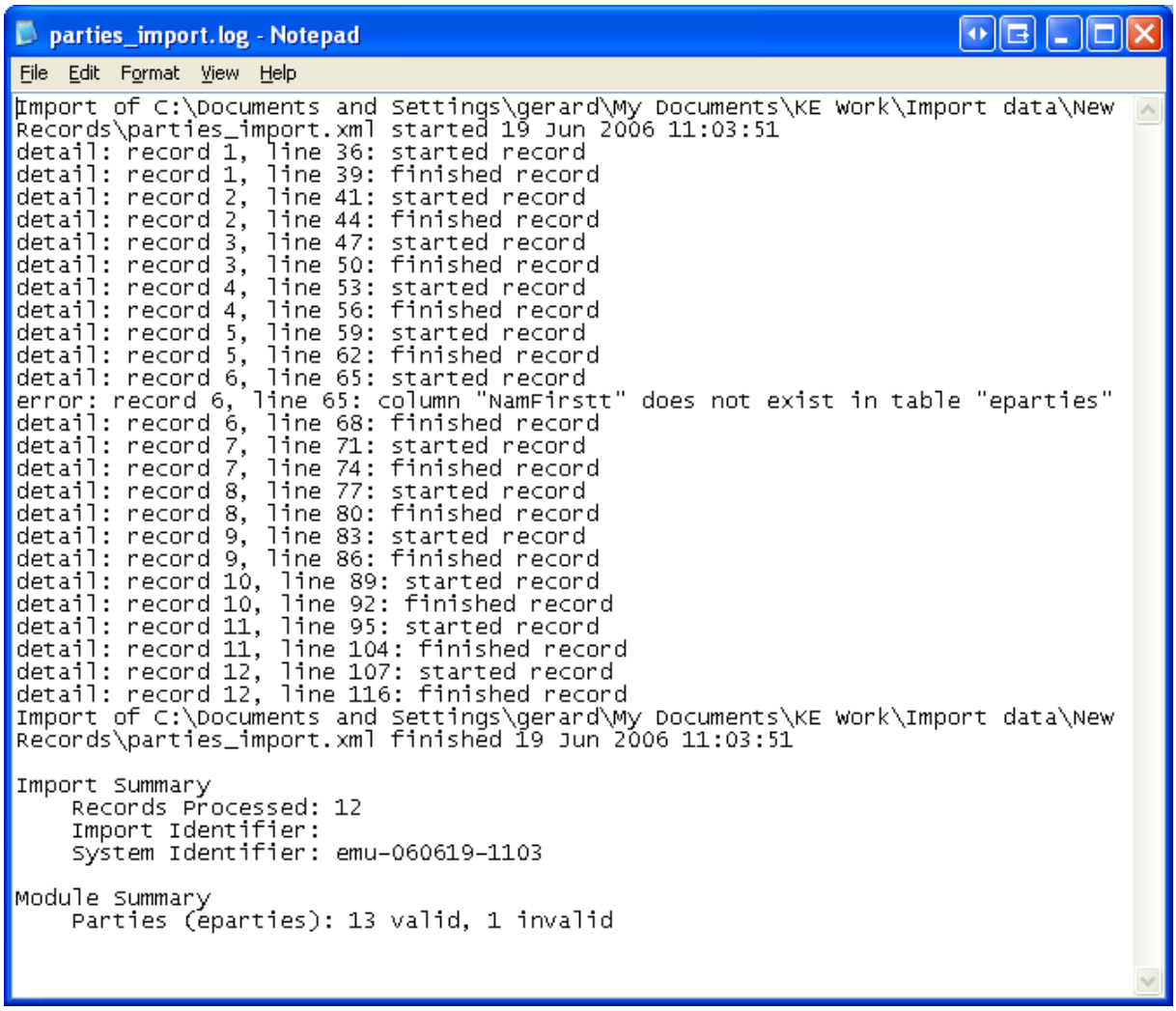
12. Select **Finished** 

.

The Import Complete screen displays:



13. Select **Close**  to exit the Import Wizard and display the *Import Report* log. The error is clearly indicated:



14. Locate and open the import data file (*parties_import.xml*.in this example) and correct the error (in this case a typo in the column name *NamFirst*).

> Another file, called *parties_import-error.xml* is auto-generated by EMu and placed in the same directory as the original import data file. If you had been importing records rather than validating them, all of the valid records would have been imported and the erroneous records would have been placed in this error file. You could then correct the errors in the error file and perform a data import using it.
> In this case, no records have been imported, so we need to correct the original data file.

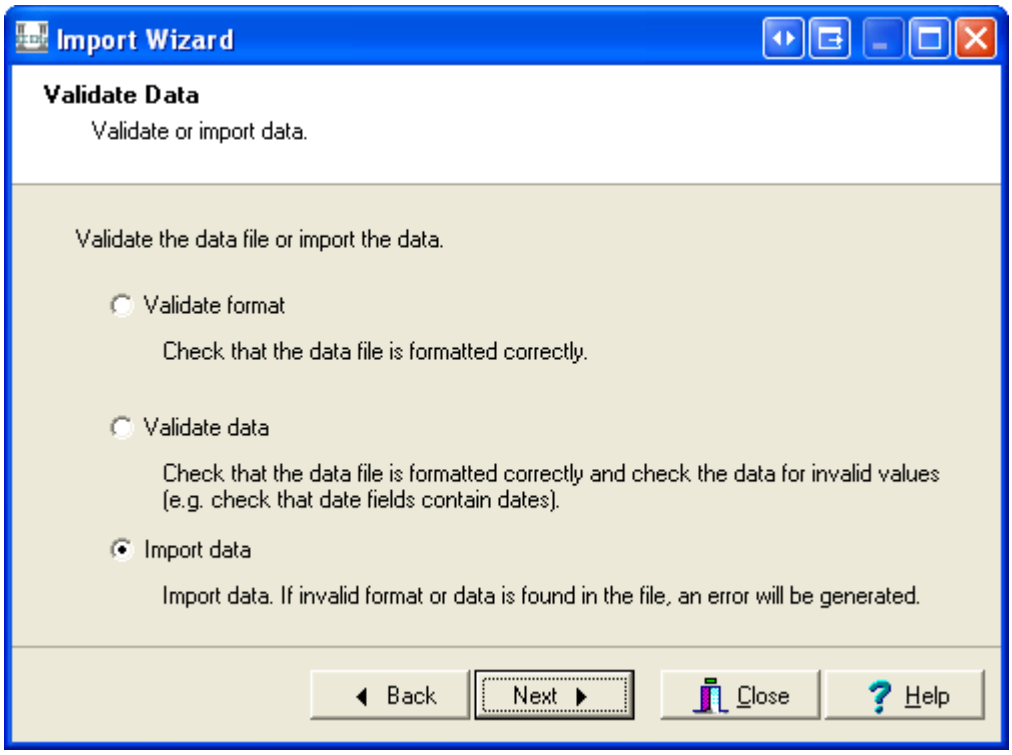
15. You could repeat this process to double check that the corrected file is valid

    -OR-

    You could proceed with the import:

16. Repeat steps 1 to 4.

17. When the Validate Data screen displays, accept the default option to Import data:



18. Select **Next**  to continue.

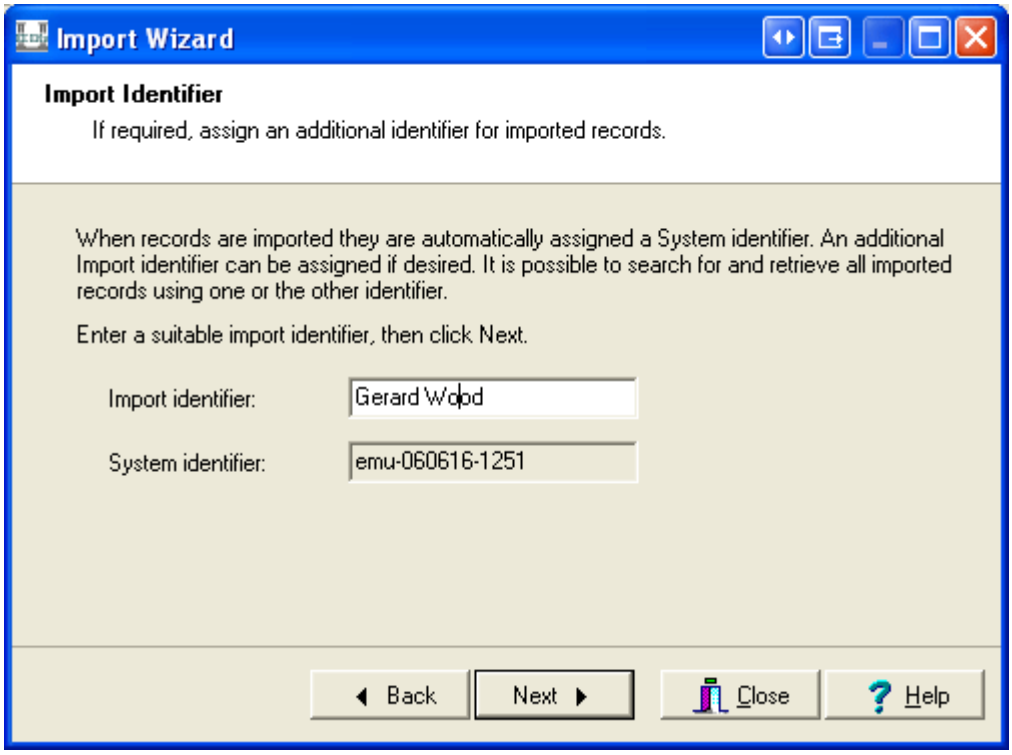
This time the Attachments screen displays.
19. Accept the default settings and select **Next**  to continue.

The Records screen displays.
20. Accept the default settings (to import all records) and select **Next**  to continue.

The Logging screen displays.
21. Accept the default settings (to log errors) and select **Next**  to continue.

The Import Identifier screen displays.

22. Add an Import identifier to facilitate locating these imported records:



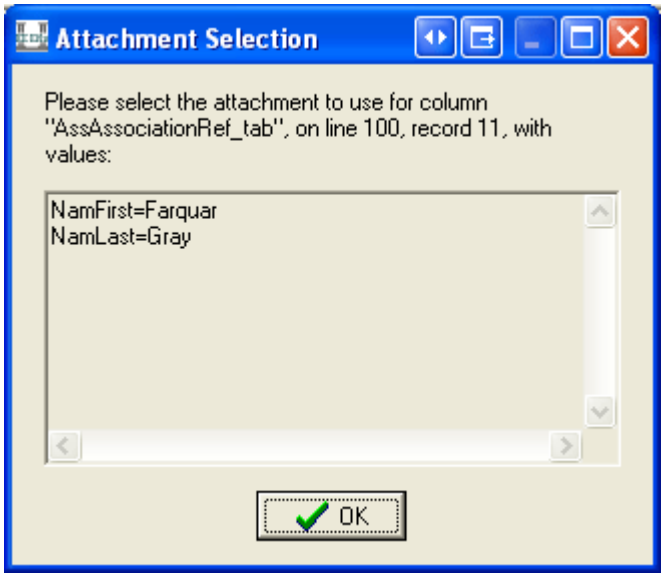
23. Select **Next**  to continue.

The Settings screen displays.

24. Ensure the settings are correct and select **Next**  to proceed with the import.

The Importing Screen displays and processing commences.

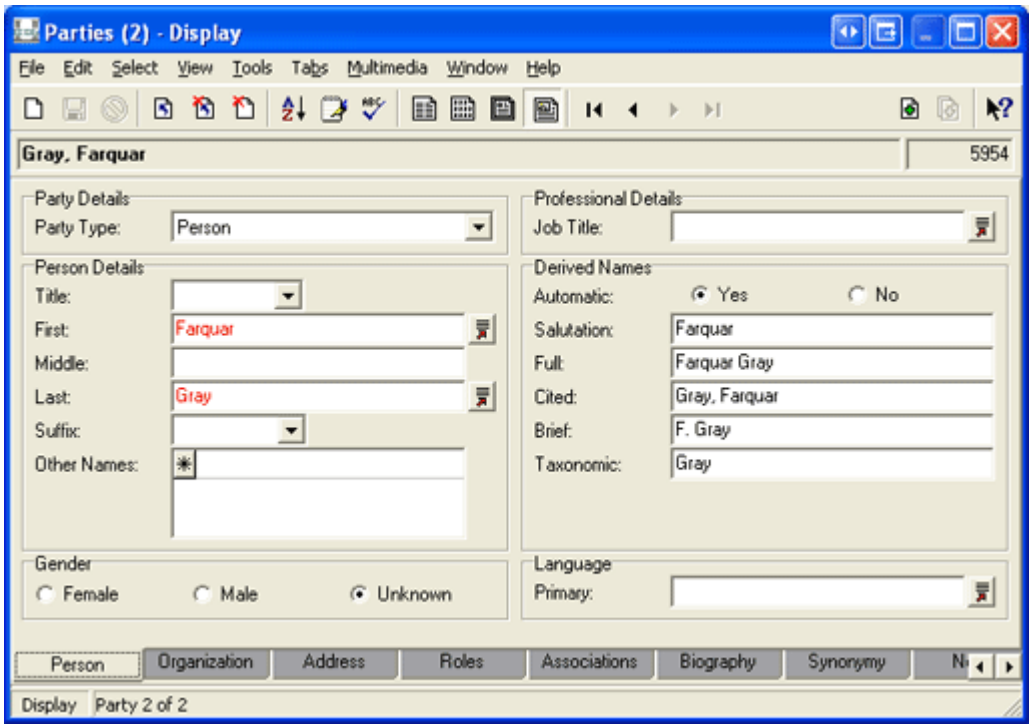
If there are references in the data file to other records (attachments) a search is performed using the details provided. If more than one match is found for the attachment record, the default action is for processing to stop and the matching records to be displayed in the appropriate module, allowing you to select the correct match:



In this case a record being imported seeks to attach to another Parties record with a first name `Farquar` and a last name `Gray`. There are two parties records with the same details.

25. Select **OK** 

to view the matching records in the Parties module:



Because records being imported could reference both of these Parties records, it will probably be necessary to open the import data file, locate the record that is seeking to make this attachment and confirm that you have identified the correct attachment relationship. The record and line number of the record in the import data file are displayed in the Attachment Selection box (shown above) to assist you to locate this record in the import data file.

26. Identify the correct record in the attachment module and select the **Attach Current Record**  button in the Toolbar.

This process will repeat until all attachments are resolved. See page 10 for more details.

When the import process is complete, details about the import are shown, including how many records were created, how many attachments were made and the number of errors.

27. Select **Finished** 

.

The Import Complete screen displays.

28. Select **Close** 

to exit the Import Wizard and display the *Import Report* log.

How to construct an import data file

Import data file format

Data can be imported into EMu using one of three file formats:

- Tab delimited or TSV (.txt or .tab file type)
- Comma delimited or CSV (.csv file type)
- eXtensible Markup Language or XML (.xml file type)

Details of the three types are available on page 79.

Field types

When importing data into EMu there are six types of field that can be specified in an import data file:

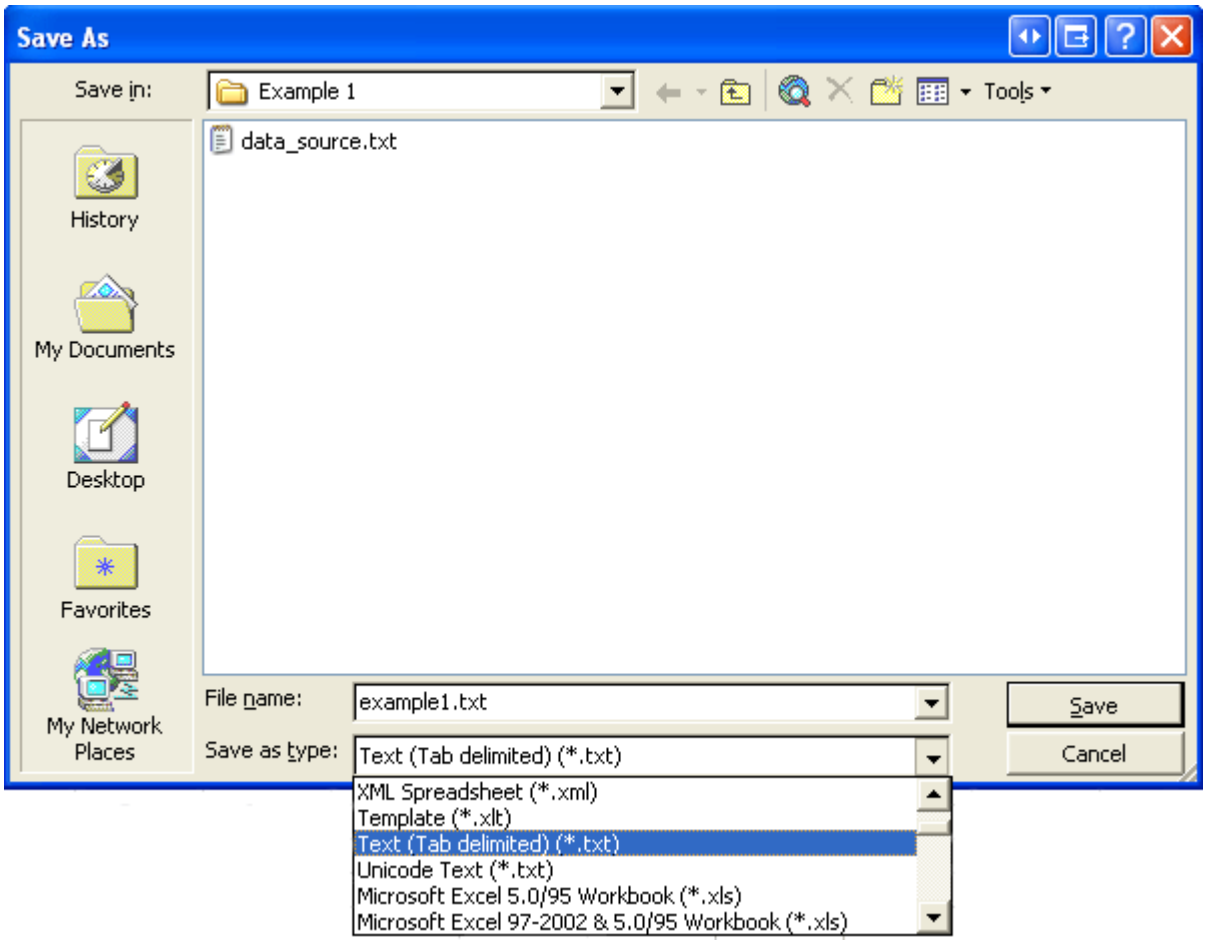
1. Atomic (a single value field) (see page 42)
2. Table (see page 50)
3. Nested Table (see page 71)
4. Atomic reference (see page 58)
5. Table reference (see page 64)
6. Nested Table reference

4, 5 and 6 are variations of the first three and specify attachments to atomic, table and nested table fields (that is, the fields do not contain data but references to other records).

This section demonstrates how to construct an import data file in the three supported formats. The approach is to use examples, each one building on the previous. It is therefore suggested that the examples are worked through consecutively.

.txt and .csv

In these examples, MS Excel is used to create both .txt and .csv file types. In each case the file is simply saved as .txt or .csv using the **Save as type** option in the Save As dialogue box:



Note the following about Date fields in MS Excel:

MS Excel will attempt to apply its default date format to values that are formatted as a Date data type. This can have undesirable consequences if Excel's default date format is different from the format you require. It is therefore recommended that date columns are formatted as text rather than Date data types. To avoid any ambiguity, it is recommended that the date values are also formatted as, e.g. 24-Nov-06 (using the text version of a month).

Finding the back-end column name

Whether constructing a .txt, .csv or .xml data file, it is necessary to use the back-end name for EMu fields (see the EMu Online Help for details about locating the back-end field name).

Example 1: Atomic

This first example demonstrates how to import or update records with data in the following single value (atomic) fields:

Value	Back-end name
First name	NamFirst
Middle name	NamMiddle
Last name	NamLast
Party Type	NamPartyType
Notes	NotNotes

It also demonstrates how to specify data that contains Carriage Return / New Lines, as in the following Notes field:

```
Joe has been with the institution for a number of years.
The level of commitment he has shown has been terrific!
He should be considered for promotion at the next staff
review.
```

Tab / Comma Separated Values

1. The first line in every TSV / CSV file **must** contain the field names as column headings:

NamPartyType	NamFirst	NamMiddle	NamLast	NotNotes

2. Each subsequent row contains a single record. The values are simply entered in the appropriate column:

NamPartyType	NamFirst	NamMiddle	NamLast	NotNotes
Person	Joe	J	Jackson	
Person	Michael		Williamson	
Person	Wilbur		Withers	A useful saxophone player.

3. To add a note with carriage returns in MS Excel:
    1. Enter the first line: `Joe has been with the institution for a number of years`.
    2. Press ALT+ENTER.
    3. Enter the next line, and so on.

NamPartyType	NamFirst	NamMiddle	NamLast	NotNotes
Person	Joe	J	Jackson	Joe has been with the institution for a number of years. The level of commitment he has shown has been terrific! Should be considered for promotion at the next staff review.
Person	Michael		Williamson	
Person	Wilbur		Withers	A useful saxophone player.

XML

The XML for this example is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
   <!-- First record-->
   <tuple>
        <atom name="NamPartyType">Person</atom>
        <atom name="NamFirst">Joe</atom>
        <atom name="NamMiddle">J</atom>
        <atom name="NamLast">Jackson</atom>
        <atom name="NotNotes">Joe has been with the institution
        for a number of years.
   The level of commitment he has shown has been terrific!
   He should be considered for promotion at the next staff
   review.</atom>
   </tuple>
   <!--Second record-->
   <tuple>
        <atom name="NamPartyType">Person</atom>
        <atom name="NamFirst">Michael</atom>
        <atom name="NamLast">Williamson</atom>
   </tuple>
   <!--Third record-->
   <tuple>
        <atom name="NamPartyType">Person</atom>
        <atom name="NamFirst">Wilbur</atom>
        <atom name="NamLast">Withers</atom>
        <atom name="NotNotes">A useful saxophone player.</atom>
   </tuple>
</table>
```

Note:

1. Except for the XML declaration (called a processing instruction) at the very top of the code (`<?xml version="1.0" encoding="ISO-8859-1"?>`), XML tags occur as pairs where each pair must have an opening and closing tag, e.g. <table> and </table>.

Commented code:

<!--This is the XML declaration and should appear at the beginning of any XML document; it identifies the document as XML and specifies the release of XML that it conforms to. The encoding defines the character set of the raw data; ISO-8859-1 specifies the Latin 1 character set used by EMu-->

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

<--! The opening and closing table tags must surround the data to be imported-->

```
<table>
```

```
<!--First record-->
```

<!--Each tuple represents a single record. A tuple can include atoms, tuples and tables-->

```
<tuple>
```

<!--Each atom represents a single value. The format for an atom is:

<atom name="colname">value</atom>

-->

```
<atom name="NamPartyType">Person</atom>
<atom name="NamFirst">Joe</atom>
<atom name="NamMiddle">J</atom>
<atom name="NamLast">Jackson</atom>
```

<!--Text and paragraph formatting entered between the atom tags is retained when the XML is imported into EMu. In this example, three lines of text are specified in the XML, and three lines will be imported into the Notes field-->

```
<atom name="NotNotes">Joe has been with the institution for a
number of years.
The level of commitment he has shown has been terrific!
He should be considered for promotion at the next staff review.</atom>
</tuple>
<!--Second record-->
<tuple>
```

<!--When specifying atoms, only fields that contain values are specified. In the previous record (tuple) five values were specified; this one only has three-->

```
<atom name="NamPartyType">Person</atom>
<atom name="NamFirst">Michael</atom>
<atom name="NamLast">Williamson</atom>
```

```
    </tuple>
    <!--Third record-->
    <tuple>
         <atom name="NamPartyType">Person</atom>
         <atom name="NamFirst">Wilbur</atom>
         <atom name="NamLast">Withers</atom>
         <atom name="NotNotes">A useful saxophone player.</atom>
    </tuple>
```

<!--Don't forget to close off all of the tags-->

```
</table>
```

Example 2: Atomic

This example extends the last one (on page 42) by adding in Organisation details. This demonstrates how to load different record types within the same data file: in this case, both Person and Organisation records will be created using the same data file.

The fields loaded are:

Value	Back-end name
First name	NamFirst
Middle name	NamMiddle
Last name	NamLast
Party Type	NamPartyType
Organisation	NamOrganisation
Department	NamDepartment
Address	AddPhysStreet
City	AddPhysCity
State	AddPhysState
Country	AddPhysCountry
Zip Code	AddPhysPost

This example also demonstrates how to specify an ampersand character in XML.

Tab / Comma Separated Values

For display purposes only, the import data below is presented with column headings listed vertically rather than horizontally. The first row of any tab or comma delimited file **must** include the column names. The appropriate layout is:

NamPartyType	NamFirst	NamMiddle	NamLast
Person	Joe	J	Jackson
Person	Michael		Williamson
Organisation			

The import data is:

Column Name (must appear as the first row of the import data file)	Record 1	Record 2	Record 3
NamPartyType	Person	Person	Organisation
NamFirst	Joe	Michael	
NamMiddle	J		
NamLast	Jackson	Williamson	
NamOrganisation	Seeing Eye Dog School	Accounting & Assoc	"Clocks and Clocks"
NamDepartment		Income Tax	
AddPhysStreet	2435 Westside Street	Level 23 1214 Hill Street	
AddPhysCity	Westside	Eastside	Smithville
AddPhysState	Central Territory	Central Territory	Central Territory
AddPhysCountry	Salsam		
AddPhysPost	123 ABC		

When this data is imported, three records will be created in the Parties module: two will be Person Party Types, the other will be an Organisation Party Type.

XML

The XML for this example is:

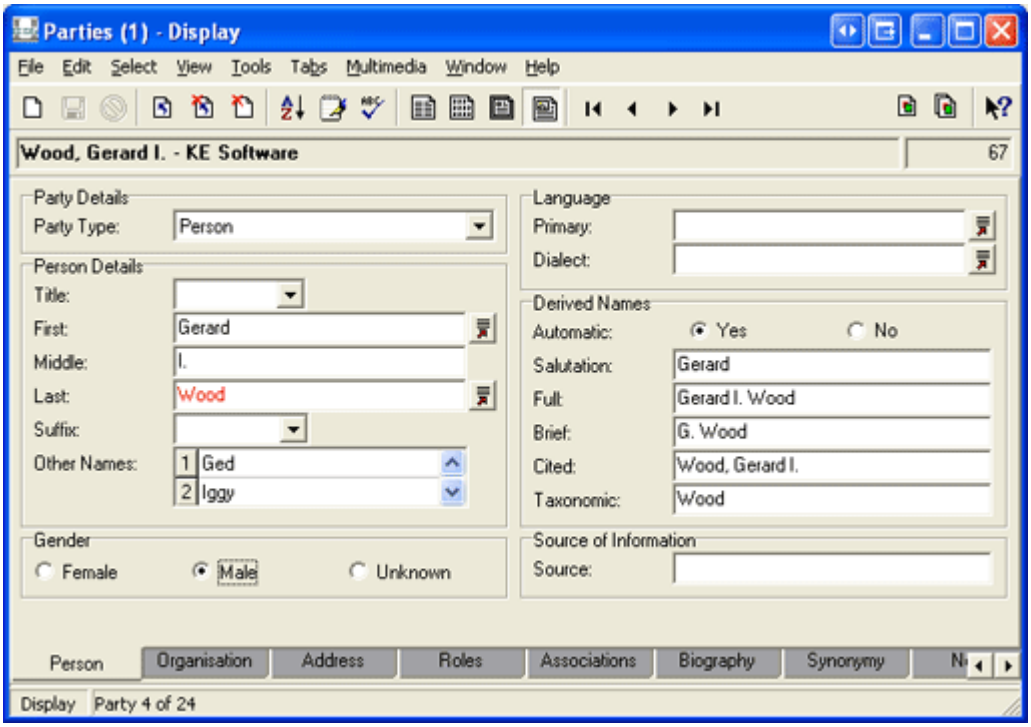
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
        <!--First record-->
        <tuple>
                <atom name="NamPartyType">Person</atom>
                <atom name="NamPartyType">Person</atom>
                <atom name="NamFirst">Joe</atom>
                <atom name="NamMiddle">J</atom>
                <atom name="NamLast">Jackson</atom>
                <atom name="NamOrganisation">Seeing Eye Dog School</atom>
                <atom name="AddPhysStreet">2435 Westside Street</atom>
                <atom name="AddPhysCity">Westside</atom>
                <atom name="AddPhysState">Central Territory</atom>
                <atom name="AddPhysCountry">Salsam</atom>
                <atom name="AddPhysPost">123 ABC</atom>
        </tuple>
        <!--Second record-->
        <tuple>
                <atom name="NamPartyType">Person</atom>
                <atom name="NamFirst">Michael</atom>
                <atom name="NamLast">Williamson</atom>

                <!--Because an ampersand (&) is a special character in XML, it
                must be specified as:
                &amp;
                if it is to appear in text
                An angled bracket < can also be escaped using:
                &lt;
                -->
                <atom name="NamOrganisation">Accounting &amp;
                Assoc</atom>
                <atom name="NamDepartment">Income Tax</atom>
                <atom name="AddPhysStreet">Level 23
        1214 Hill Street</atom>
                 <atom name="AddPhysCity">Eastside</atom>
                 <atom name="AddPhysState">Central Territory</atom>
```

```
        </tuple>
        <!--Third record-->
        <tuple>
                <atom name="NamPartyType">Organization</atom>
                <atom name="NamOrganisation">"Clocks and Clocks"</atom>
                <atom name="AddPhysCity">Smithville</atom>
                <atom name="AddPhysState">Central Territory</atom>
        </tuple>
</table>
```

Example 3: Table

This example extends the previous two by including references to fields that can accept more than one value, in other words, tables. In this screenshot of the Parties module, we see that the *Other Names: (Person Details)* field can take multiple values:



Other Names: (Person Details) is a table of values (a column) and each row can hold a name. Its back-end field name further identifies it as a table with the addition of a *_tab* suffix:

NamOtherNames_tab

In this example we create three Person Party records that include *Role* and *Other Name* values (both of which are tables).

The fields loaded are:

Value	Back-end name
First name	NamFirst
Middle name	NamMiddle
Last name	NamLast
Party Type	NamPartyType
Roles	NamRoles_tab
Other Names	NamOtherNames_tab

Tab / Comma Separated Values

For display purposes only, the import data below is presented with column headings listed vertically rather than horizontally. The first row of any tab or comma delimited file **must** include the column names. The appropriate layout is:

NamPartyType	NamFirst	NamMiddle	NamLast
Person	Joe	J	Jackson
Person	Michael		Williamson
Person	Wilbur		Withers

The import data is:

Column Name (must appear as the first row of the import data file)	Record 1	Record 2	Record 3
NamPartyType	Person	Person	Person
NamFirst	Joe	Michael	Wilbur
NamMiddle	J		
NamLast	Jackson	Williamson	Withers
NamRoles_tab(1)	Trainer		
NamRoles_tab(2)	Teacher		Musician
NamOtherNames_tab(1)	Joey	Mike	
NamOtherNames_tab(2)		Willo	
NamOtherNames_tab(3)		Willy	

To specify the first row in a table the format is:

```
TableName_tab(1)
```

And the second row is:

```
TableName_tab(2)
```

And so on.

In this example three records would be created (or updated). For each record, the Roles table would have up to two entries (two rows) and the Other Names table would have up to three rows.

Updating records that include table fields

The Import tool can be used to update existing records in EMu (details on page 67). Be aware of the following however:

The entire table is replaced when the Import tool is used.

1. Notice that in this TSV/CSV file three rows are specified for NamOtherNames_tab. The record for Joe Jackson (Record 1) only contains a value in NamOtherNames_tab(1), while NamOtherNames_tab(2) and NamOtherNames_tab(3) are blank.

   If there was already a record for Joe Jackson in EMu and it had a value in NamOtherNames_tab(1) and another value in NamOtherNames_tab(2), the value in NamOtherNames_tab(2) would be overwritten when this data file was imported.

2. Consider also that if this data file was imported, the record for Michael Williamson (Record 2) would have three rows in the *Other Names* table.

   If you then updated these records with the following data file (i.e. excluding NamOtherNames_tab(3)):

NamParty Type	Nam First	Nam Middle	Nam Last	NamRoles _tab(1)	NamRoles _tab(2)	NamOther Names_tab(1)	NamOther Names_tab(2)
Person	Joe	J	Jackson	Trainer	Teacher	Joey	
Person	Michael		Williamson			Mike	Willo
Person	Wilbur		Withers		Musician		

   the existing value in NamOtherNames_tab(3) would be lost.

XML

The XML for this example is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
   <!--First record-->

   <!--As well as including atomic references for single values, a tuple (which as we've seen defines a single record) can also include a table reference-->
   <tuple>
          <atom name="NamPartyType">Person</atom>
          <atom name="NamFirst">Joe</atom>
          <atom name="NamMiddle">J</atom>
          <atom name="NamLast">Jackson</atom>

          <--!The table is referenced using table tags-->
          <table name="NamRoles_tab">

                <!--A table element is used to represent a table of values. A value may be atomic, a tuple or another table. Each row in the table is defined by a tuple.

                Note that the data structure for tables in EMu only allows one value per tuple. One value could be an atom, tuple or a table (demonstrated in Example 8 on page 71). In other words, a tuple could not contain two atoms. Each atom would need to be defined by its own tuple (as in this example).

                In this example, two rows are added to the Roles table-->
                <tuple>
                      <atom>Trainer</atom>
                </tuple>
                <tuple>
                      <atom>Teacher</atom>
                </tuple>
          </table>

          <!--In this example, one row is added to the Other Names table-->
          <table name="NamOtherNames_tab">
                <tuple>
                      <atom>Joey</atom>
                </tuple>
```

```
        </table>
    </tuple>
    <!--Second record-->
    <tuple>
        <atom name="NamPartyType">Person</atom>
        <atom name="NamFirst">Michael</atom>
        <atom name="NamLast">Williamson</atom>
        <table name="NamOtherNames_tab">
                        <tuple>
                        <atom>Mike</atom>
                        </tuple>
                        <tuple>
                         <atom>Willo</atom>
                        </tuple>
                        <tuple>
                         <atom>Willy</atom>
                        </tuple>
        </table>
    </tuple>
    <!--Third record-->
    <tuple>
        <atom name="NamPartyType">Person</atom>
        <atom name="NamFirst">Wilbur</atom>
        <atom name="NamLast">Withers</atom>
        <table name="NamRoles_tab">
```

<!--Even if there are empty values in a table list, the tuple must be specified-->

```
                <tuple>
                </tuple>
                <tuple>
                        <atom>Musician</atom>
                </tuple>
        </table>
    </tuple>
</table>
```

Example 4: Multimedia

When loading multimedia with the Import tool it is necessary to use the virtual *Multimedia* field to specify the pathway to the multimedia file to be imported. This example builds on the last one (on page 50) by including multimedia in the import, as well as a table.

The fields loaded are:

Value	Back-end name
Title	MulTitle
Description	MulDescription
Creator	MulCreator_tab
Multimedia	Multimedia

Tab / Comma Separated Values

For display purposes only, the import data below is presented with column headings listed vertically rather than horizontally. The first row of any tab or comma delimited file **must** include the column names. The appropriate layout is:

MulTitle	MulDescription	MulCreator_tab(1)
An image of a house	A classic 1950's design. Needs some work around the edges and a coat of paint.	John Smith
An image of a cow.		Unknown
A 1950's Car!		

The import data is:

Column Name (must appear as the first row of the import data file)	Record 1	Record 2	Record 3
MulTitle	An image of a house.	An image of a cow.	A 1950's Car!
MulDescription	A classic 1950's design. Needs some work around the edges and a coat of paint.		
MulCreator_tab(1)	John Smith	Unknown	
MulCreator_tab(2)	Bill Wilson		
Multimedia	C:\Images\House Image.jpg	C:\Images\Cow in paddock.jpg	C:\Images\Car 1950.jpg

XML

The XML for this example is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
   <!--First record-->
   <tuple>
          <atom name="MulTitle">An image of a house.</atom>
          <atom name="MulDescription">A classic 1950's design.
          Needs some work around the edges and a coat of paint.</atom>
          <table name="MulCreator_tab">
                 <tuple>
                        <atom>John Smith</atom>
                 </tuple>
                 <tuple>
                        <atom>Bill Wilson</atom>
                 </tuple>
          </table>
          <atom name="Multimedia">C:\Images\House Image.jpg</atom>
   </tuple>
   <!--Second record-->
   <tuple>
          <atom name="MulTitle">An image of a cow.</atom>
          <table name="MulCreator_tab">
                  <tuple>
                        <atom>Unknown</atom>
                  </tuple>
          </table>
          <atom name="Multimedia">C:\Images\Cow in paddock.jpg</atom>
   </tuple>
   <!--Third record-->
   <tuple>
          <atom name="MulTitle">A 1950's Car!</atom>
          <atom name="Multimedia">C:\Images\Car 1950.jpg</atom>
   </tuple>
</table>
```

The thoroughness of the Import process (compared to a simple batch load) is highlighted when loading multimedia. When importing a .jpg image, for example, much data is automatically generated by EMu: MIME type is automatically determined and if any additional resolutions have been defined in the Registry, these are automatically generated.

Example 5: Atomic Reference

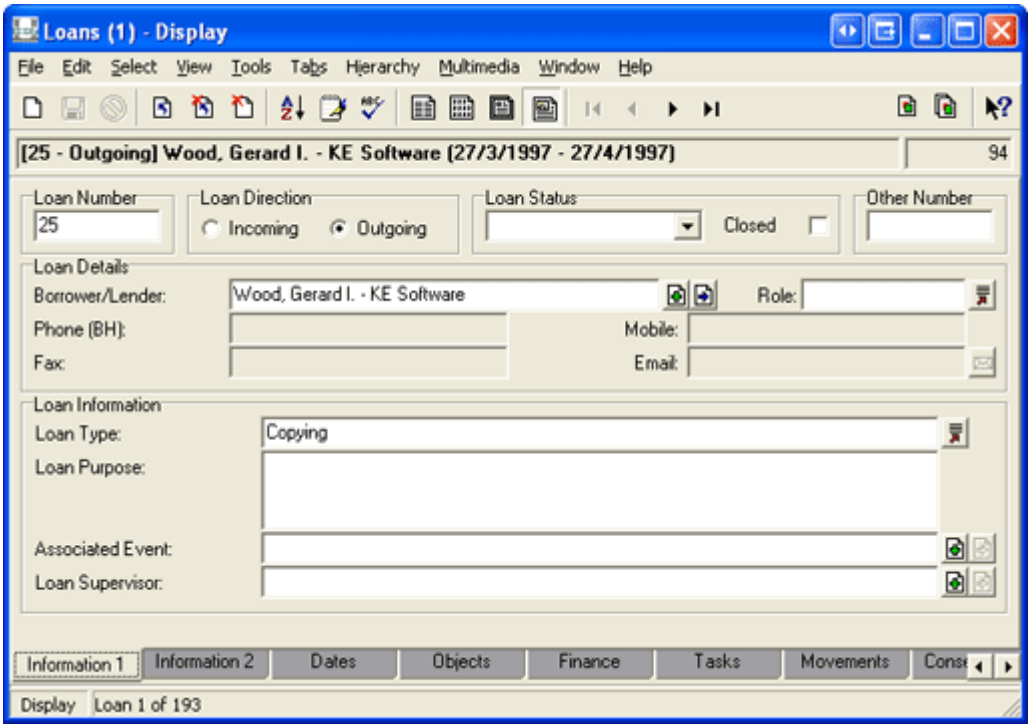
This example demonstrates how to specify an attachment to another record. The import data specifies a record in the Loans module that attaches to a Parties record (the borrower).

This example also specifies a table (Business Telephone Number) in the Parties module.

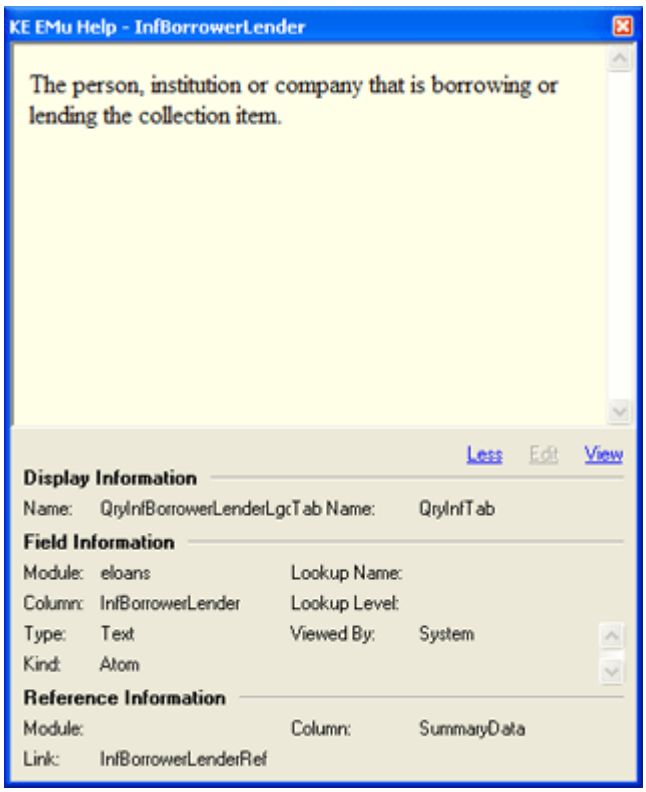
The fields loaded are:

Value	Back-end name
Loan Direction	InfDirection
Loan Purpose	InfLoanPurpose
Commencement Date	DatLoanCommencementDate
Due Date	DatLoanDueDate
Borrower/Lender	InfBorrowerLender_Ref

The attachment field in the Loans module is *Borrower/Lender: (Loan Details)*:



If we check the Field Level information for this field, the following displays:



The back-end name is *InfBorrowerLender*, but when specifying an attachment field we use its Link name: *InfBorrowerLenderRef*.

Tab / Comma Separated Values

For display purposes only, the import data below is presented with column headings listed vertically rather than horizontally. The first row of any tab or comma delimited file **must** include the column names. The appropriate layout is:

InfDirection	InfLoanPurpose	DatLoanCommencementDate
Outgoing	The loan is for research purposes.	15-Jan-06
Incoming	The loan is for an upcoming exhibition.	10-Mar-06
Outgoing	A loan without a borrower or due date!	17-Mar-66

The import data is:

Column Name (must appear as the first row of the import data file)	Record 1	Record 2	Record 3
InfDirection	Outgoing	Incoming	Outgoing
InfLoanPurpose	The loan is for research purposes.	The loan is for an upcoming exhibition.	A loan without a borrower or due date!
DatLoanCommencementDate	15-Jan-06	10-Mar-06	17-Mar-66
DatLoanDueDate	15-Jan-07	10-Jul-06	
InfBorrowerLenderRef.NamPartyType	Person	Organisation	
InfBorrowerLenderRef.NamFirst	John		
InfBorrowerLenderRef.NamLast	Smith		
InfBorrowerLenderRef.NamOrgaisation		The Art Gallery of Salsam	
InfBorrowerLenderRef.NamBusiness_tab(1)		61393470011	

The attachment reference is constructed by combining the Link name for the attachment field with a back-end field name from the attached module.

For example, to specify that the *Borrower/Lender* field in the Loans module should attach to a record for John Smith in the Parties module, you might specify:

```
InfBorrowerLenderRef.NamFirst
InfBorrowerLenderRef.NamLast
```

And to specify a row in a table in the attachment module, simply designate the row number (as we've already seen when specifying a table in the same module). For example, the first row in *Business: (Telephone Numbers)* is specified as:

```
InfBorrowerLenderRef.NamBusiness_tab(1)
```

When the import is processed a search for a record for John Smith is initiated in the Parties module; if a record is found it can be used for the attachment. If no record is found, one can be created in the Parties module using the values specified in the data file, and then the two records will be linked together.

XML

The XML for this example is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
   <!--First record-->
   <tuple>

      <!--First the atom fields in the Loans module are specified-->
      <atom name="InfDirection">Outgoing</atom>
      <atom name="InfLoanPurpose">The loan is for research
      purposes.</atom>
      <atom name="DatLoanCommencementDate">15 January 2006</atom>
      <atom name="DatLoanDueDate">15 January 2007</atom>

      <!--Next the attachment field in the Loans module is specified, followed by
      each atom field in the Parties module-->
      <tuple name="InfBorrowerLenderRef">
                        <atom name="NamPartyType">Person</atom>
                        <atom name="NamFirst">John</atom>
                        <atom name="NamLast">Smith</atom>
      </tuple>
   </tuple>
   <!--Second record-->
   <tuple>
      <atom name="InfDirection">Incoming</atom>
      <atom name="InfLoanPurpose">The loan is for an upcoming
      exhibition.</atom>
      <atom name="DatLoanCommencementDate">10 March 2006</atom>
      <atom name="DatLoanDueDate">10 July 2006</atom>
      <tuple name="InfBorrowerLenderRef">
            <atom name="NamPartyType">Organization</atom>
            <atom name="NamOrganisation">The Art Gallery of
            Salsam</atom>

            <!--As well as specifying atomic fields in the attachment module,
            tables can also be specified. In this example, the first row in the
            Business: (Telephone Numbers) table is referenced-->
            <table name="NamBusiness_tab">
                  <tuple>
                        <atom>+61 3 9347 0011</atom>
```

```
                        </tuple>
                </table>
        </tuple>
    </tuple>
    <!--Third record-->
    <tuple>
        <atom name="InfDirection">Outgoing</atom>
        <atom name="InfLoanPurpose">A loan without a borrower or due
        date!</atom>
        <atom name="DatLoanCommencementDate">17 March 1966</atom>
    </tuple>
</table>
```

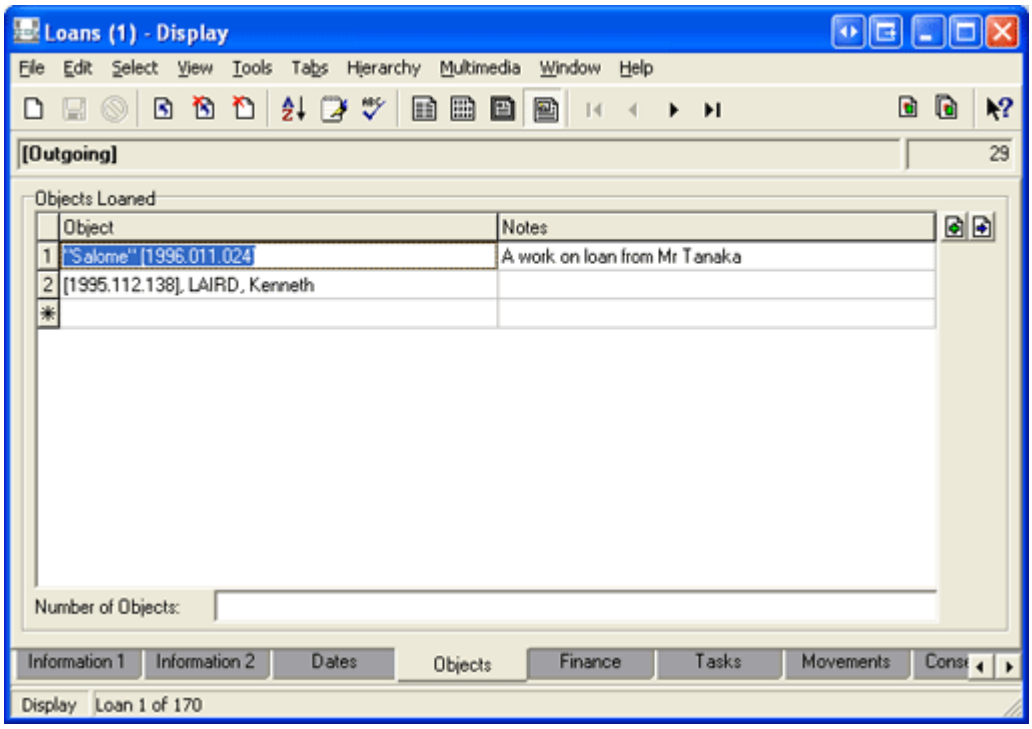
What happens when this data is imported

When this data is imported into EMu using the Typical import options, the following occurs:

1. The Parties module is searched using the details in the specified Parties fields (e.g. *NamFirst*). If no matching record is found, a record is created in the Parties module and attached to the Loans module using the *InfBorrowerLender* field. If there is a match, the new Loans record (created next) is attached to the matching Parties record.
2. A record in the Loans module is created using the values in the Loans fields.

Example 6: Specifying a table of attachments

In this example we examine how to specify a table of attachments. In this case a record in the Loans module covers the loan of a number of objects in the Catalogue module:



Objects: (Object Loaned) attaches to the Catalogue module and is a table of attachments. *Notes: (Object Loaned)* is an associated table in the Loans module.

The fields loaded are:

Value	Back-end name
Loan Direction	InfDirection
Loan Purpose	InfLoanPurpose
Commencement Date	DatLoanCommencementDate
Due Date	DatLoanDueDate
Objects	ObjObjectsLoanedRef_tab
Objects Notes	ObjObjectNotes_tab

Tab / Comma Separated Values

For display purposes only, the import data below is presented with column headings listed vertically rather than horizontally. The first row of any tab or comma delimited file **must** include the column names. The appropriate layout is:

InfDirection	InfLoanPurpose	DatLoanCommencementDate
Outgoing	The loan is for research purposes.	15-Jan-06
Incoming	The loan is for an upcoming exhibition.	10-Mar-06

The import data is:

Column Name (must appear as the first row of the import data file)	Record 1	Record 2
InfDirection	Outgoing	Incoming
InfLoanPurpose	The loan is for research purposes.	The loan is for an upcoming exhibition.
DatLoanCommencementDate	15-Jan-06	10-Mar-06
DatLoanDueDate	15-Jan-07	10-Jul-06
ObjObjectsLoanedRef_tab(1).TitAccessionNo	1996.011.024	1995.112.138
ObjObjectNotes_tab(1)	A work on loan from Mr. Tanaka	
ObjObjectsLoanedRef_tab(2).TitAccessionNo	1996.011.042	1995.112.145
ObjObjectNotes_tab(2)		
ObjObjectsLoanedRef_tab(3).TitAccessionNo	1995.112.061	
ObjObjectNotes_tab(3)	A donations from Mr Sutcliffe	

ObjObjectsLoaned is both an attachment field (giving us *ObjObjectsLoanedRef*) and a table (giving us *ObjObjectsLoanedRef_tab*).

Thus *ObjObjectsLoanedRef_tab(1).TitAccessionNo*:

1. Specifies the first row in *Object: (Objects Loaned).*
2. When Record 1 is processed a search of the Catalogue will be initiated for an object with an Accession Number of `1996.011.024`.

   If found, an attachment from the first row in *Object: (Objects Loaned)* will be made to it.

`ObjObjectNotes_tab(1)` specifies the first row in *Notes: (Objects Loaned).*

XML

The XML for this example is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
   <!--First record-->
   <tuple>

      <!--First the atom fields in the Loans module are specified-->
      <atom name="InfDirection">Outgoing</atom>
      <atom name="InfLoanPurpose">The loan is for research
      purposes.</atom>
      <atom name="DatLoanCommencementDate">15 January 2006</atom>
      <atom name="DatLoanDueDate">15 January 2007</atom>

      <!--Next the Object: (Object Loaned) attachment field is specified. This
      field attaches to one or more records in the Catalogue module (so it's a
      table).
      Don't forget that when specifying tables there can be only one value per
      tuple-->
      <table name="ObjObjectsLoanedRef_tab">
            <tuple>
                  <atom name="TitAccessionNo">1996.011.024</atom>
            </tuple>
            <tuple>
                  <atom name="TitAccessionNo">1996.011.042</atom>
            </tuple>
            <tuple>
                  <atom name="TitAccessionNo">1995.112.061</atom>
            </tuple>
      </table>

      <!--This specifies the Notes: (Object Loaned) table within the Loans
      module. Again, even if there are empty values in a list, the tuple must be
      specified-->
      <table name="ObjObjectNotes_tab">
            <tuple>
                  <atom>A work on loan from Mr. Tanaka</atom>
            </tuple>
            <tuple>
```

```
                </tuple>
                <tuple>
                      <atom>A donations from Mr Sutcliffe</atom>
                </tuple>
          </table>
    </tuple>
    <!--Second record-->
    <tuple>
          <atom name="InfDirection">Incoming</atom>
          <atom name="InfLoanPurpose">The loan is for an upcoming
          exhibition.</atom>
          <atom name="DatLoanCommencementDate">10 March 2006</atom>
          <atom name="DatLoanDueDate">10 July 2006</atom>
          <table name="ObjObjectsLoanedRef_tab">
                <tuple>
                      <atom name="TitAccessionNo">1995.112.138</atom>
                </tuple>
                <tuple>
                       <atom name="TitAccessionNo">1995.112.145</atom>
                </tuple>
          </table>
    </tuple>
</table>
```

Example 7: Update records

In this example the Import tool is used to update records in EMu. Updating records only requires that a unique field (or combination of unique fields) is used to identify an existing record. This example uses IRN as well as Accession Number to identify records in the Catalogue and update a located record's *Condition* fields: if unique fields are specified in the data file and a matching record is identified in EMu, the EMu record is updated.

It is only possible to use the Import tool to update records if "Unique Strict" is set for the unique field(s). For multi-column unique values (where a combination of fields is used to specify uniqueness) it is necessary in the EMu Registry to enable “Unique Strict” on each of the fields.

Note that if a record with an IRN or Accession Number specified in the data file does not exist in EMu, a new record will be created.

In this example, the updates will only work if Accession Number has been set up as a Unique field in the Catalogue.

Unique fields cannot be updated.

Rules

1. If a single record in the data file matches a record in EMu using a field or fields configured in EMu to be unique (e.g. an IRN), the record is updated.
2. If a record in the data file includes a field or fields configured in EMu to be unique (e.g. an IRN) and it does not match a record in EMu or there is more than one match, a new record is created.

In this example the fields updated are:

Value	Back-end name
IRN	irn
Accession Number	TitAccessionNo
Condition Status	ConConditionStatus
Date Checked	ConDateChecked
Checked By	ConCheckedByRef
Condition Details	ConConditionDetails

Tab / Comma Separated Values

For display purposes only, the import data below is presented with column headings listed vertically rather than horizontally. The first row of any tab or comma delimited file **must** include the column names. The appropriate layout is:

irn	TitAccessionNo	ConConditionStatus
484		Excellent
	1996.011.042	Poor
	1995.112.061	Good

The import data is:

Column Name (must appear as the first row of the import data file)	Record 1	Record 2	Record 3
irn	484		
TitAccessionNo		1996.011.042	1995.112.061
ConConditionStatus	Excellent	Poor	Good
ConDateChecked	15-Aug-05	16-Aug-05	17-Aug-05
ConCheckedByRef.NamFirst	Joe	Joe	Joe
ConCheckedByRef.NamLast	Jackson	Jackson	Jackson
ConCheckedByRef.NamRoles_tab(1)	Condition Checker	Condition Checker	Condition Checker
ConConditionDetails	The work is in excellent condition. Loans can be approved (subject to normal conditions).	Due to the poor condition of this work, loans should not be approved.	

`irn` and `TitAccessionNo` are both unique fields in this Catalogue; when the import data file is processed, if a record is found with a matching IRN or Accession Number, it will be updated. If no matching record is found, one will be created using the values provided.

XML

The XML for this example is:

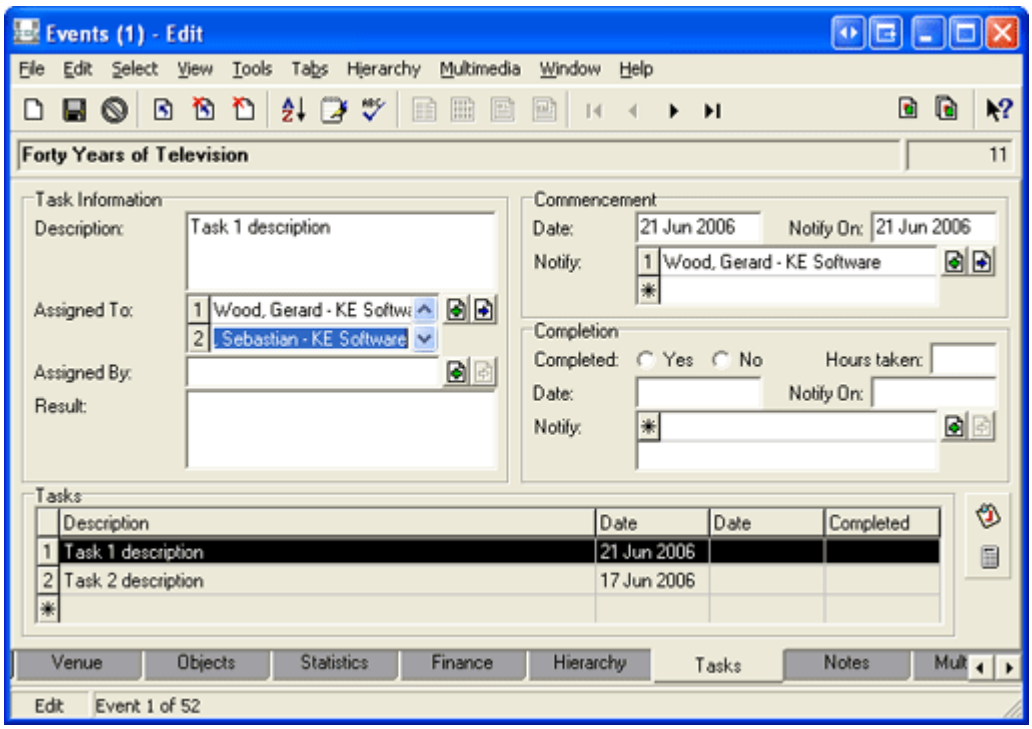
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
   <!--First record -->
   <tuple>
          <atom name="irn">484</atom>
          <atom name="ConConditionStatus">Excellent</atom>
          <atom name="ConDateChecked">15 August 2005</atom>
          <tuple name="ConCheckedByRef">
                  <atom name="NamFirst">Joe</atom>
                  <atom name="NamLast">Jackson</atom>
                  <table name="NamRoles_tab">
                         <tuple>
                                <atom>Condition Checker</atom>
                         </tuple>
                  </table>
          </tuple>
          <atom name="ConConditionDetails">The work is in excellent
          condition.
          Loans can be approved (subject to normal conditions).</atom>
   </tuple>
   <!--Second record -->
   <tuple>
          <atom name="TitAccessionNo">1996.011.042</atom>
          <atom name="ConConditionStatus">Poor</atom>
          <atom name="ConDateChecked">15 August 2005</atom>
          <tuple name="ConCheckedByRef">
                <atom name="NamFirst">Joe</atom>
                <atom name="NamLast">Jackson</atom>
                <table name="NamRoles_tab">
                      <tuple>
                             <atom>Condition Checker</atom>
                      </tuple>
                </table>
          </tuple>
          <atom name="ConConditionDetails">Due to the poor condition of
          this work, loans should not be approved.</atom>
   </tuple>
```

```
    <!--Third record -->
    <tuple>
           <atom name="TitAccessionNo">1995.112.061</atom>
           <atom name="ConConditionStatus">Good</atom>
           <atom name="ConDateChecked">15 August 2005</atom>
           <tuple name="ConCheckedByRef">
                 <atom name="NamFirst">Joe</atom>
                 <atom name="NamLast">Jackson</atom>
                 <table name="NamRoles_tab">
                       <tuple>
                             <atom>Condition Checker</atom>
                       </tuple>
                 </table>
           </tuple>
    </tuple>
 </table>
```

Example 8: Nested Tables and an update

In this example a series of tasks is updated (based on the supplied IRN) in the Events module, demonstrating how to load data into nested tables.

Nested table: an example



When a row is selected in the *Tasks* group of fields at the bottom of the module window, values in the fields above are updated. *Assigned To: (Task Information)* and *Assigned By: (Task Information)* are also tables - called nested tables in EMu - containing lists of Party names (they are thus both a table and an attachment field).

The fields used are:

Value	Back-end name
IRN	irn
Description	TasDescription_tab
Assigned To	TasPersonAssignedToRef _nesttab
Assigned By	TasTaskAssignerRef_tab
Commencement Date	TasCommencementDate0
Notify	TasStartNotifyDate0
Completed	TasCompleted_tab

Tab / Comma Separated Values

For display purposes only, the import data below is presented with column headings listed vertically rather than horizontally. The first row of any tab or comma delimited file **must** include the column names. The appropriate layout is:

irn	TasDescription_tab (1)	TasDescription_tab (2)	TasDescription_tab (3)
11	Check the event dates are available.	Organise a committee to overlook the event development.	Confirm that all works are available for exhibiting.

The import data is:

Column Name (must appear as the first row of the import data file)	Description	Record 1
irn		11
TasDescription_tab(1)	Specifies a value for *Description: (Tasks)* in the first row in the Tasks group of fields.	Check the event dates are available.
TasDescription_tab(2)		Organise a committee to overlook the event development.
TasDescription_tab(3)		Confirm that all works are available for exhibiting.
TasDescription_tab(4)		Talk to finance department about funding.
TasPersonAssignedToRef_nesttab(1:1).NamFirst	This is a nested table. The first 1 in (1:1) makes the association with the first row in the outer table: TasDescription_tab(1). The second 1 in (1:1) specifies the first row in the nested table. This is an attachment field too and specifies the *First: (Person Details)* field in the Parties module.	Joe

TasPersonAssignedToRef_nesttab(1:1).NamLast	This is a nested table. The first 1 in (1:1) makes the association with the first row in the outer table: TasDescription_tab(1). The second 1 in (1:1) specifies the first row in the nested table. This is an attachment field too and specifies the *Last: (Person Details)* field in the Parties module.	Jackson
TasPersonAssignedToRef_nesttab(1:2).NamFirst		Jim
TasPersonAssignedToRef_nesttab(1:2).NamLast		Johnson
TasPersonAssignedToRef_nesttab(2:1).NamFirst	This is a nested table. The 2 in (2:1) makes the association with the second row in the outer table: TasDescription_tab(2). The 1 in (2:1) specifies the first row in the nested table. This is an attachment field too and specifies the *First: (Person Details)* field in the Parties module.	Joe
TasPersonAssignedToRef_nesttab(2:1).NamLast		Jackson
TasPersonAssignedToRef_nesttab(2:2).NamFirst		Jim
TasPersonAssignedToRef_nesttab(2:2).NamLast		Johnson
TasPersonAssignedToRef_nesttab(2:3).NamFirst		Paul
TasPersonAssignedToRef_nesttab(2:3).NamLast		Smith
TasPersonAssignedToRef_nesttab(3:1).NamFirst		Paul
TasPersonAssignedToRef_nesttab(3:1).NamLast		Smith
TasPersonAssignedToRef_nesttab(4:1).NamFirst		Jim
TasPersonAssignedToRef_nesttab(4:1).NamLast		Johnson
TasTaskAssignerRef_tab(1).NamFirst		Rachael
TasTaskAssignerRef_tab(1).NamLast		Albost
TasTaskAssignerRef_tab(2).NamFirst		Rachael
TasTaskAssignerRef_tab(2).NamLast		Albost

TasTaskAssignerRef _tab(3).NamFirst		Rachael
TasTaskAssignerRef _tab(3).NamLast		Albost
TasTaskAssignerRef _tab(4).NamFirst		Rachael
TasTaskAssignerRef _tab(4).NamLast		Albost
TasCommencementDate0(1)		23-Mar-04
TasCommencementDate0(2)		30-Mar-04
TasCommencementDate0(3)		10-Apr-04
TasCommencementDate0(4)		14-Apr-04
TasStartNotifyDate0(1)		14-Mar-04
TasStartNotifyDate0(2)		23-Mar-04
TasStartNotifyDate0(3)		3-Apr-04
TasStartNotifyDate0(4)		7-Apr-04
TasCompleted_tab(1)		No
TasCompleted_tab(2)		No
TasCompleted_tab(3)		No
TasCompleted_tab(4)		No

XML

The XML for this example is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
      <!--First record -->
      <tuple>
            <atom name="irn">11</atom>

            <!--Four rows (tuples) in the Tasks group of fields are specified.
            This is the outer level of the nested table pair. For each tuple specified
            here there will be a tuple in the inner table specified next-->
            <table name="TasDescription_tab">
                  <tuple>
                        <atom>Check the event dates are available.</atom>
                  </tuple>
                  <tuple>
                        <atom>Organise a committee to overlook the event
                        development.</atom>
                  </tuple>
                  <tuple>
                        <atom>Confirm that all works are available for
                        exhibiting.</atom>
                  </tuple>
                  <tuple>
                        <atom>Talk to finance department about
                        funding.</atom>
                  </tuple>
            </table>

            <!--The inner table of the nested table pair is specified-->
            <table name="TasPersonAssignedToRef_nesttab">

                  <!--The first tuple corresponds to the first tuple specified in the
                  outer level table above. There are two rows specified-->
                  <tuple>
                        <table>
                              <tuple>
                                    <atom name="NamFirst">Joe</atom>
                                    <atom name="NamLast">Jackson</atom>
```

```
            </tuple>
            <tuple>
                  <atom name="NamFirst">Jim</atom>
                  <atom name="NamLast">Johnson</atom>
            </tuple>
      </table>
</tuple>
```

<!--The second tuple corresponds to the second tuple specified in the outer level table above. There are three rows specified. And so on-->

```
<tuple>
      <table>
            <tuple>
                  <atom name="NamFirst">Joe</atom>
                  <atom name="NamLast">Jackson</atom>
            </tuple>
            <tuple>
                  <atom name="NamFirst">Jim</atom>
                  <atom name="NamLast">Johnson</atom>
            </tuple>
            <tuple>
                  <atom name="NamFirst">Paul</atom>
                  <atom name="NamLast">Smith</atom>
            </tuple>
      </table>
</tuple>
<tuple>
      <table>
            <tuple>
                  <atom name="NamFirst">Paul</atom>
                  <atom name="NamLast">Smith</atom>
            </tuple>
      </table>
</tuple>
<tuple>
      <table>
            <tuple>
                  <atom name="NamFirst">Jim</atom>
                  <atom name="NamLast">Johnson</atom>
            </tuple>
```

```
            </table>
      </tuple>
</table>
<table name="TasTaskAssignerRef_tab">
      <tuple>
            <atom name="NamFirst">Rachael</atom>
            <atom name="NamLast">Albost</atom>
      </tuple>
      <tuple>
            <atom name="NamFirst">Rachael</atom>
            <atom name="NamLast">Albost</atom>
      </tuple>
      <tuple>
            <atom name="NamFirst">Rachael</atom>
            <atom name="NamLast">Albost</atom>
      </tuple>
      <tuple>
            <atom name="NamFirst">Rachael</atom>
            <atom name="NamLast">Albost</atom>
      </tuple>
</table>
<table name="TasCommencementDate0">
      <tuple>
            <atom>23 March 2004</atom>
      </tuple>
      <tuple>
            <atom>30 March 2004</atom>
      </tuple>
      <tuple>
            <atom>10 April 2004</atom>
      </tuple>
      <tuple>
            <atom>14 April 2004</atom>
      </tuple>
</table>
<table name="TasStartNotifyDate0">
      <tuple>
            <atom>14 March 2004</atom>
      </tuple>
      <tuple>
```

```
                        <atom>23 March 2004</atom>
                  </tuple>
                  <tuple>
                        <atom>3 April 2004</atom>
                  </tuple>
                  <tuple>
                        <atom>7 April 2004</atom>
                  </tuple>
            </table>
            <table name="TasCompleted_tab">
                  <tuple>
                        <atom>No</atom>
                  </tuple>
                  <tuple>
                        <atom>No</atom>
                  </tuple>
                  <tuple>
                        <atom>No</atom>
                  </tuple>
                  <tuple>
                        <atom>No</atom>
                  </tuple>
            </table>
      </tuple>
</table>
```

Supported File Formats

Three file formats are supported for the import of data into EMu:

Comma Separated Values or CSV (.csv)

Can be generated using a product like MS Excel and saving files as .csv.

An excellent definition of the CSV file format is available at http://www.edoceo.com/utilis/csv-file-format.php

Tab Separated Values or TSV(.txt or .tab)

Can be generated using a text tool, such as Notepad, or a product like MS Excel and saving files as .txt.

The same rules that apply to .csv files apply to .txt with the exception that values are separated using tabs rather than commas.

eXtensible Markup Language or XML (.xml)

Provides the greatest flexibility in specifying the data to be imported and is recommended when using the Import tool to import more than the most basic data structures (for instance, when specifying records with attachments, nested tables, etc.).

An easy method to generate the correctly structured XML for the fields you wish to import or update in EMu is to create an XML Report in EMu:

1. In the module in which data is to be imported or updated, create an *XML Document* report and include the fields to be imported or updated.
2. Run the report. An XML document is generated. The format of the report is the same as that required for an import.

Import tool Registry settings

To be able to use the Import tool to add a new record in an EMu module a user must have (or be a member of a group that has) the *daImport* permission set for the table (or tables if the record has attachments) involved in the import.

Thus *daImport* allows a user to import data that would result in the creation of a new record. When a user attempts to do so a check is made to determine whether the imported record matches an existing record; if it is a new record, the system then checks whether the user has the *daImport* permission. If they do, the record is added to EMu.

Note the following however:

daImport sits above any other permissions that would normally apply in the creation or edit of a record in a module (such as *daInsert*). A user must have all appropriate permissions to edit / add data to a record in the affected module(s).

In other words, when using the Import tool to add or update records any permissions set for a user will apply. For instance, if a user does not normally have permission to add a value to a Lookup List, they will not be able to do so when using the Import tool. If they do not have permission to add a record to a module, they will not be able to do so when using the Import tool.

The *daImport* permission alone is not sufficient to allow users to import records into an EMu module.

daImport

To be able to use the Import tool a user must have (or be a member of a group that has) the *daImport* permission.

The following is an example of a Registry entry that enables the Import tool for members of the Admin group on all modules:

Field	Description	Example
Key 1		**Group**
Key 2	Name of group	**Admin**
Key 3		**Table**
Key 4	Table(s) to which the operation permissions apply	**Default**
Key 5		**Operations**
Value	Access permissions to apply	**daQuery** **daDisplay** **daEdit** **daInsert** **daDelete** **daReplace** **daDefaults** **daReport** **daDesign** **daEditQuery** **daEditHelp** **daSecurity** **daImport**

If the operation permissions are not the same for all tables, enter the table names that permissions apply to in the *Key 4* field instead of using the **Default** value.