



EMu Documentation

Release Notes: EMu 5.0

Document Version 1

EMu Version 5.0

EMu
Museum
Management
System



KE Software

An AXIELL Group Company

emu.axiell.com

© 2015 All rights reserved

Contents

Here you will find collected together the Release Notes for EMu 5.0, alongside all documents referenced in the notes. These release notes and documents are also available on the [EMu website](#).

This PDF document brings together a number of individually published documents: please note that page numbering below refers to this combined PDF document and not to the page numbers printed at the bottom of pages, as each individual document has its own internal numbering:

Release Notes: EMu 5.0	5
Thesaurus Browse View	15
Reporting Formats	35
Unicode Support	81

Release Notes: EMu 5.0

Release Date: 16 November 2015

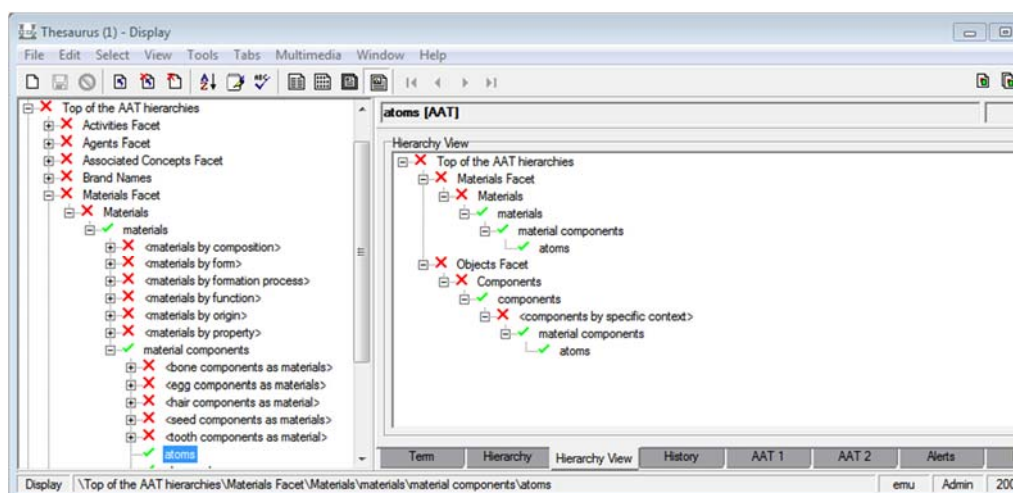
Requirements

- Windows 2003, Vista, Windows 7, Windows 8, Windows 8.1, Windows 10
- [Texpress 9.0.001](#) or later
- [TexAPI 6.0.012](#) or later
- [Perl 5.8.8](#) or later

New Features

Thesaurus Browse View The Thesaurus module Browse View option has been updated with many new functionality and usability improvements, including:

- Seamless interaction with other EMu modes (Search, Display and Edit) and views (List, Contact Sheet, Page and Details). Records selected in the Browse View are displayed in the main EMu window according to the current view.
- All of the hierarchies of the current record in EMu can be displayed in the Browse View, either automatically or at the click of a button.
- Additional options for navigating between and showing or hiding terms in the Browse View have been added.
- The new Hierarchy View tab displays all of the hierarchies for the displayed records term.



A complete description can be found in the [Thesaurus Browse View and Hierarchy View tab](#) documentation.

Reporting Formats

New report formats have been added to improve report generation and loading performance. It is now possible to report directly to an Open Database Connectivity (ODBC) data source or to an ActiveX Data Objects (ADO) Recordset object. The new report types are:

- *Crystal* - reports directly in ODBC format, removing the need for an ODBC filtering process.
- *Crystal ADO* - reports in ADO Recordsets for Crystal which are accessible via Crystal's ADO connector.
- *Microsoft ADO* - reports in ADO Recordsets for Microsoft products.

Existing Crystal Report and Microsoft (Excel, Power Point and Word) report types that currently connect to an ODBC data source can be changed to use an ADO Recordset.

Unicode Support

Full support for Unicode has been added, including:

- Support for the Unicode 8.0 character set.
- Data stored using UTF-8 encoding.
- Character folding for all Unicode characters.
- Use of the Default Unicode Collation Element Table (DUCET) for collation (default Unicode sorting order).
- Punctuation is indexed and searchable.
- Punctuation characters with special meaning (!~@\$^&*()=) must now be escaped with a leading backslash.
- Auto-phrasing where indexed terms are not separated by spaces are searched as a phrase.

A complete description can be found in the [Unicode](#) documentation.

Improvements

Performance Improvements

Performance improvements have been made to sorting, exact matching, reporting and attachment searches in EMu. Some examples of the observed performances improvements are:

- Sorting improvements:

Module	Records	Old Time	New Time	Factor Improvement
Catalogue	270,000	2:15	0:45	x3
Parties	25,000	1:20	0:02	x40
Events	22,000	2:15	0.015	x90

- The performance improvement increases with the size of the module. The modules are listed from small to large.
- Exact matching improvements:

Module	Records	Old Time	New Time	Factor Improvement
Catalogue	250,000	1:30	0.15	x6
Thesaurus	115,000	6:45	0:15	x27
Events	22,000	2:20	0.01	x80

- The performance improvement increases with the size of the module. The modules are listed from small to large.
- Reporting improvements:

Module	Records	Old Time	New Time	Factor Improvement
Bibliography	6,500	1:00	0.10	x6
Parties	70,000	10:00	1:00	x10

- The performance improvement increases with the size of the module. The modules are listed from small to large.
- Attachment searching improvements:

Module	Records	Old Time	New Time	Factor Improvement
Sites	10,000	10:00	1:50	x6
Parties	200,000	60:00	9:30	x11

Mandatory field consistency improvements

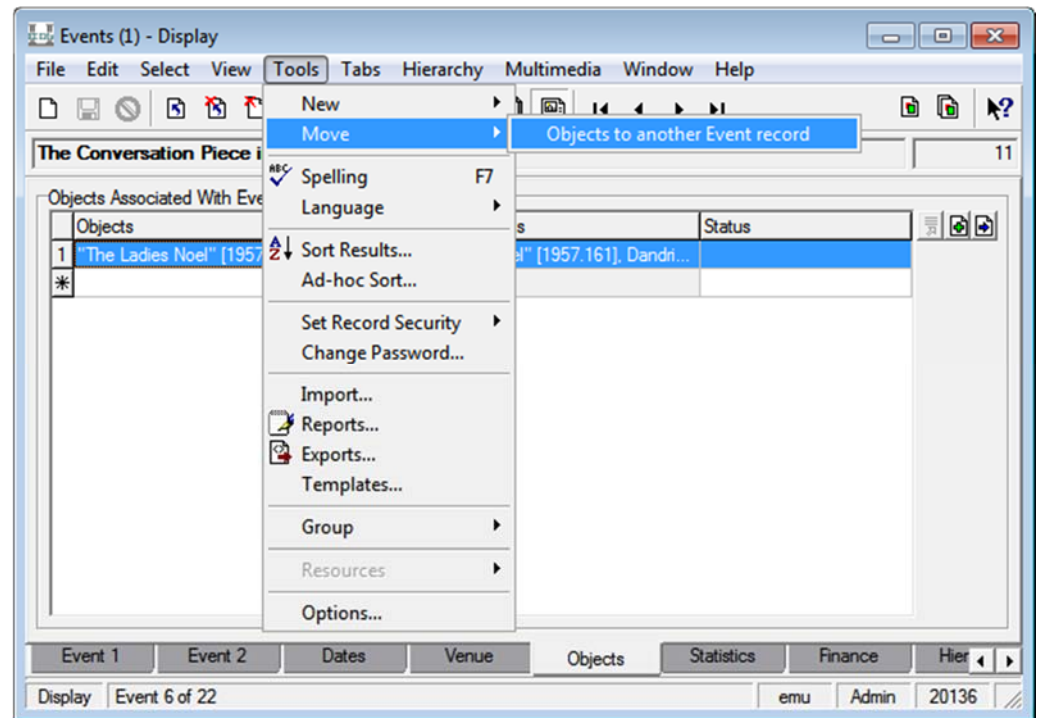
The consistency of the behaviour of grid and edit fields associated with columns that have been set as mandatory using the Mandatory Registry entry has been improved. The behaviour has been modified so that when a user tries to leave an empty edit or grid field that uses a mandatory column, the user will first be presented with the mandatory pop-up message. If the user tries to leave the field again, the mandatory pop-up message will not be presented until they re-enter and leave the empty field again or the record is saved. Prior to these changes


mandatory settings that applied to edit fields were only triggered when the record was saved.

Additionally, changes have been made so the Lookup List buttons of grid controls that are associated with mandatory columns can be used without triggering the mandatory pop-up message.

Events module Move Objects command

A command has been added to the Tools drop down menu of the Events module that uses the Exhibit Objects extensions. The command allows objects selected in the *Objects* grid on the Objects tab to be moved to another Event record. Associated Exhibit Object records are also moved:



To use the command you select a series of rows in the *Objects* grid corresponding to the objects to be moved. Once they are selected the **Tools>Move>Objects to another Event record** command can be invoked. A new instance of the Events module is displayed allowing the user to locate the record to which the objects should be moved. The  button is clicked to make the selection. Once the Event record is selected the user is asked to confirm the movement of objects to the selected Event record. Once confirmed the objects are moved one at a time. When the operation is complete, the number of objects moved is displayed.

Styling of HTML controls

A new Column CSS Registry entry has been added, making it possible to specify the default font family and font size for EMu client controls that display HTML formatted text. An example of an HTML control is the control associated with the NarNarratives column on the Narrative tab of the Narratives module.

A complete description can be found in the EMu Help documentation for the Column CSS Registry entry.

Issues Resolved

- Fixed an issue when using the Import Tool where a new line character in appended data assigns the data to the incorrect row of the specified column.
- Fixed an issue when using the Internal tab of the Movements module to update the Location of an attached object; the *irn* of the new Location would be put in the Movement Notes of the Catalogue instead of the Movements Notes of the Movements module.
- Corrected the display of the full name of the Source in Accession Lots module Summary Data.
- Changed the behaviour when adding image types that support multiple images in a single file (specifically the TIFF, TIFF/EP and DNG file formats) to the Multimedia module. Previously the "last" image from the file was selected as the Multimedia module master image; now the "first" image is used.
- Fixed an issue where a mini-multimedia component linked to a grid did not update when the grid row was changed.
- Fixed an issue where dates before 30 December 1899 entered in a control via the pop-up calendar were off by one day.
- Fixed an issue where the value of the active cell of a grid control was retained when moving between records.
- Fixed an issue where controls would appear outside of the tab boundaries when the **View>Thumbnail** menu option was specified.
- Fixed an issue where updating Holder records in the Locations module did not trigger updates to the associated fields of Catalogue records using that location.
- Fixed an issue where the Operations module could not be accessed with the error "You are not a registered user of "eoperations" table."
- Fixed an issue where Lookup List buttons were not correctly enabled or disabled when a record was set to read-only using Record Level Security.
- Fixed an issue where IMu server log files would overlap when the IMu server configuration setting *process-count* was greater than one.
- Corrected the placement of the French translation for the "Time Moved" caption in the **Tools>Relocate** dialogue.
- Fixed an issue where aborting an operation (e.g. reporting) would cause the display of a different record than the record that was displayed prior to the operation.
- Fixed the display of Unicode characters in Admin Task controls.
- Fixed an issue where the Lookup List values displayed might not be correct for Lookup hierarchies where the hierarchy included grid and edit controls and those controls were linked to another grid control.
- Fixed an issue where merging records via the **Additional Search>Merge** menu option would produce the error "TexAPI Error: (Number 0)".
- Fixed an issue where generating resolutions of a TIFF file and a Multimedia|Metadata|Embed Registry entry specified that some or all of the tags should be embedded in the generated resolution resulted in an "Access Violation" error.
- Fixed an issue where the emuoperations utility did not correctly handle date orders other than "dmy" (day, month, year).
- Fixed an issue where Lookup List hierarchy values associated with a grid were not automatically back-filled if the Lookup button was used and the cursor was not in the grid.

- Fixed an issue while running multiple simultaneous multimedia imports in multiple instances of the EMu client where the system-generated multimedia resolutions could be assigned to the wrong multimedia record.
- Fixed an issue where data might not have been saved when adding a Lookup value to an edit field that is part of a Lookup hierarchy with a grid field and both fields are linked to another grid field.
- Fixed an issue where data might not be saved to the correct row when the user invokes undo (e.g. via the **Edit>Undo** menu option) when the cursor is within a grid that is linked to another grid.
- Added a missing entry to the crontab configuration file used for new EMu installations.
- Fixed an issue where Lookup Lists did not sort according to the user-defined Sort Order in the Lookup Lists module.
- Fixed an issue where the incorrect start and end dates values were set when a user-initiated export was run via the **Tools>Exports...** menu dialogue.

Upgrade Notes

The upgrade from EMu Version 4.3 to EMu 5.0 involves a number of steps. Please follow the instructions below carefully.

Do not skip any steps under any circumstances.

Before proceeding with the update please ensure that a complete backup of the EMu server exists and is restorable.

There are four components that require upgrading:

- Texpress (the database engine)
- TexAPI (web services)
- EMu Server (the application)
- EMu Client (the client)

The notes below detail how to upgrade all systems. Check the [Releases](#) table for Client specific notes.

In the notes below, *clientname* refers to the name of the client directory for the current installation. The term `~emu` is used to refer to user `emu`'s home directory. This is normally `/home/emu`.

Stopping EMu services

1. Log in as `emu`
2. Enter `client clientname`
3. Enter `ls -l loads/*/data* local/loads/*/data*`
4. Check that each data file is empty and that no `data.t` files exist.
If `data.t` files do exist, please wait for the loads to drain before proceeding.
5. Enter `emuload stop`
6. Enter `emuweb stop`
7. Enter `texlicstatus`
Make sure no one is using the system.
The upgrade will not complete successfully if users are accessing data.

Record Session

Each step in the upgrade process produces detailed output. In most cases this output will exceed the size of the screen. It is strongly recommended that the output of the upgrade session is recorded, so if errors occur, the output can be examined.

1. Enter `script /tmp/output-5-0`

A new shell will start and all output recorded until the shell is terminated.

Installing Texpress

Installing Texpress 9.0 is only required for the first client upgraded to EMu 5.0. Once Texpress 9.0 has been installed, this section may be skipped for subsequent upgrades.

1. Enter `cd ~emu`
2. Enter `mkdir -p texpress/9.0.xxx/install` (where xxx is the patch level number).
3. Enter `cd texpress/9.0.xxx/install`
4. Obtain the appropriate [Texpress version](#) for your Unix machine.
Save the release in `~emu/tepress/9.0.xxx/install`, calling it `tepress.sh`.
5. Enter `sh texpress.sh`
The Texpress release will be extracted.
6. Enter `. ./profile`
7. Enter `bin/texinstall ~emu/tepress/9.0.xxx`
The Texpress installation script will commence.
8. Enter `cd ~emu/tepress/9.0.xxx`
9. Enter `. ./profile`
10. Enter `bin/texlicinfo`
Obtain your Texpress licence code and place it in a file called `.licence`.
11. Enter `bin/texlicset < .licence` to install the licence.
12. Enter `\rm -fr install`
13. Enter `cd ~emu/tepress`
14. Enter `ln -s 9.0.xxx 9.0`

Upgrading TexAPI

Installing TexAPI is only required for the first client upgraded to EMu 5.0. Once TexAPI has been installed, this section may be skipped for subsequent upgrades.

1. Enter `cd ~emu/tepress`
2. Enter `mkdir 6.0.xxx` (where xxx is the patch level number).
3. Obtain the appropriate [TexAPI version](#) for your Unix machine.
Save the release in `~emu/tepress`, calling it `texapi.sh`.
4. Enter `sh texapi.sh -i ~emu/tepress/6.0.xxx` (expand the `~emu`).
5. Enter `\rm -f texapi`
6. Enter `ln -s 6.0.xxx texapi`
7. Enter `\rm -f texapi.sh`

Upgrading EMu Server

1. Enter `cd ~emu/clientname`
2. Enter `mkdir install`
3. Enter `cd install`
4. Obtain the appropriate [EMu server version bundle](#).
Save the release bundle file in `~emu/clientname/install` calling it `emu.sh`.
5. Enter `sh emu.sh`
The EMu release will be extracted.
6. Enter `. ./profile`

7. Enter `bin/emuinstall clientname`
The EMu installation script will commence.
8. Enter `cd ~emu/clientname`
9. Enter `cp .profile.parent ../.profile`
10. Enter `. ../.profile`
11. Enter `client clientname`
12. Each table will now be upgraded to Texpress 9.0. The upgrade program must be run on each table sequentially.
 1. Enter `texupgrade table`
where *table* is the name of the Texpress table to be updated.
 2. The upgrade program will apply the following changes:
 - Backup the data file(s) in the database directory (named `data.90upd.gz`, etc.).
 - Upgrade the data file from ISO-8859-1 to UTF-8, printing the changes as they occur. A copy of the output is saved in `data.90upg.log` in the database directory.
 - Lengthen any fields that need to grow to contain the UTF-8 data. A copy of the output is saved in `ins.90upg.log` in the database directory.
 - Reindex the table.
 3. If a field cannot be lengthened automatically, a listing of the field name, its old length and the required length can be found in the file `ins.90upg` in the database directory.
IMPORTANT: if a field cannot be lengthened, it **must** be extended manually by running `texdesign table` and altering the length of the field to the required value.
 4. All tables that have fields extended **must** have their Insertion Forms sent back to Axiell so they can be checked into the master tree. If this step is not performed, data in fields that were lengthened locally will be truncated when EMu is next upgraded, resulting in data loss!
13. Enter `emureindex`
14. Enter `emulutsrebuild -f -t`
15. Removal of the temporary directory (and its contents) is recommended:
Enter `\rm -fr install`
16. The client will now be upgraded to EMu 5.0. If you are upgrading from a version prior to EMu 5.0, you must run the upgrade scripts for all versions after the old version **before** running the EMu 5.0 upgrade.
17. Enter `upgrade-5-0`
18. Enter `upgrade-5-0.luts`

Starting EMu services

1. Enter `emuload start`
2. Enter `emuload status`
Check that all loads started successfully. Investigate any loads that failed to start.

Record Session

The recording of the upgrade session may now be terminated.

1. Enter `exit`

The session output is available in `/tmp/output-5-0`.

Upgrading EMu Client

EMu 5.0 does not require the new Windows client to be installed on every machine for network installations. Updating the network server is sufficient. For standalone installations a new client is required on each machine. To upgrade the EMu Client follow the [Installing EMu Client](#) notes.

Cleanup

The EMu Server upgrade produces a number of files as part of its output. These files are located in each database directory. The files are:

<code>data.90upg.gz</code>	A backup of the data file(s) taken before the upgrade process commences. If there is more than one data partition, subsequent files will be named <code>data#001.90upg.gz</code> , etc. The backup files may be sizeable even though they are compressed.
<code>data.90upg.log</code>	A copy of the output generated as the data file(s) update to UTF-8. The log lists all non-ASCII characters encountered and whether they were upgraded or were already in UTF-8 format. The file is typically small in size.
<code>ins.90upg</code>	If the file exists, then it lists all fields that could not be lengthened as part of the upgrade process. The file lists the field name, its current length and the required length. If the upgrade was successful, the file will not exist.
<code>ins.90upg.log</code>	A copy of the output generated as the database fields are lengthened. The log lists all fields requiring resizing and whether the resize was successful.

Once the upgrade is finished and testing is complete the above files may be removed.



EMu Documentation

Thesaurus Browse View & Hierarchy View tab

Document Version 1

EMu 5.0

EMu
Museum
Management
System



KE Software

An AXIELL Group Company

emu.axiell.com

© 2015 All rights reserved

Contents

SECTION 1	Browse View	1
	The primary selection	3
	Browse View right-click menu options	4
	View menu options	8
SECTION 2	Hierarchy View tab	11
	Hierarchy View right-click menu options	12
	Index	15

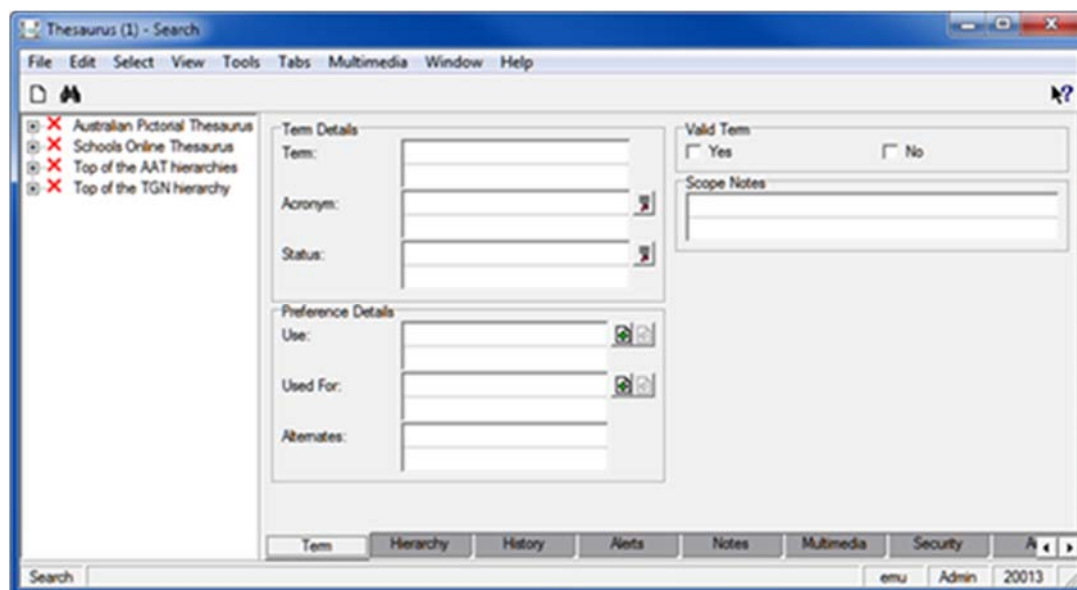
SECTION 1

Browse View

Browse View mode in the EMu Thesaurus module provides a hierarchical view of all loaded thesauri. It interacts seamlessly with other modes (Search, Display and Edit), views (List, Contact Sheet, Page and Details), and with the Hierarchy View tab (page 11).

Browse View mode is enabled by selecting **View>Browse View** from the Thesaurus module Menu bar. When the system is set to display a left-to-right language, the Browse View panel displays to the left of the main Thesaurus module window (as below); in a system displaying a right-to-left language (e.g. Arabic), the panel displays to the right of the main Thesaurus module window.

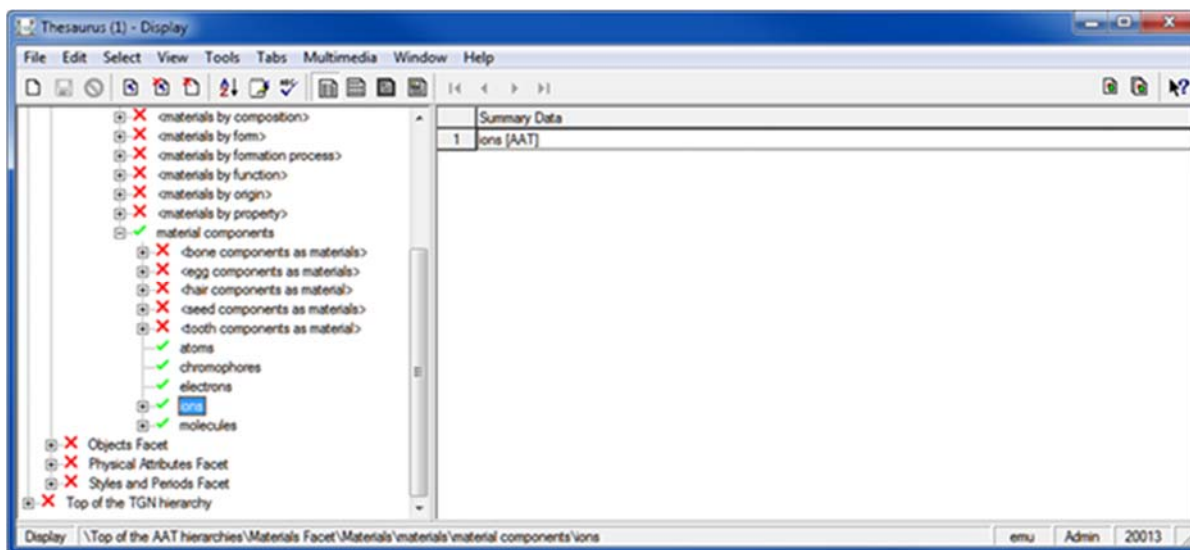
By default, the Browse View will show the top-level terms for all available thesauri when Browse View is enabled. In this example, four thesauri are available:



Users are able to navigate the thesaurus hierarchy by selecting the plus symbol (+) beside a term to reveal the next level of child terms, and so on. Terms without the plus symbol do not have any child terms.

The green tick (✓) and red cross (✗) indicate a term's validity as specified by the **Valid Term** options on the Term tab. Terms with a green tick have a *Valid Term* value of *Yes*, those with a red cross have a *Valid Term* value of *No*.

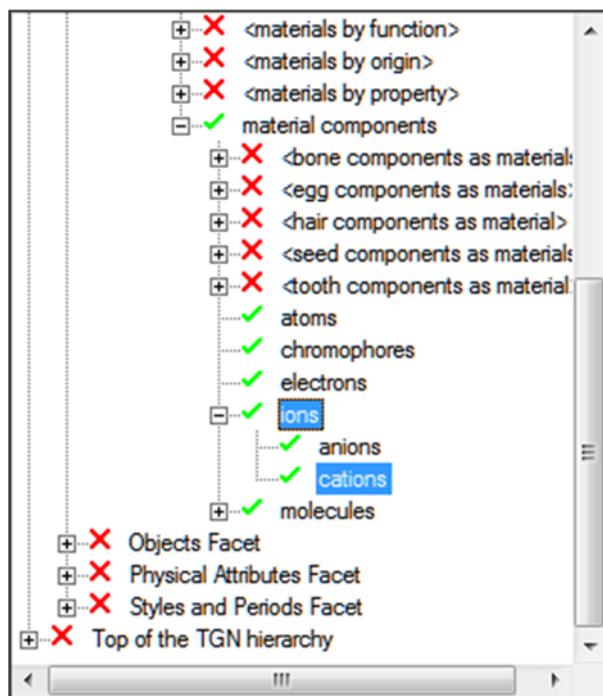
Selecting a term in the Browse View panel displays its record in the main Thesaurus module window using the currently selected display view (e.g. List, Details, etc.). If EMu is in Search mode when a term is selected, the system will automatically shift to Display mode. In this example, selecting the **ions** term shifts us from Search to Display mode and displays the **ions** record in List View:



Multiple terms can be selected using the Control (^) or Shift (⇧) keys while clicking with the mouse. The Control key allows multiple individual terms to be selected (or deselected) anywhere. The Shift key allows all of the terms between two terms to be selected.

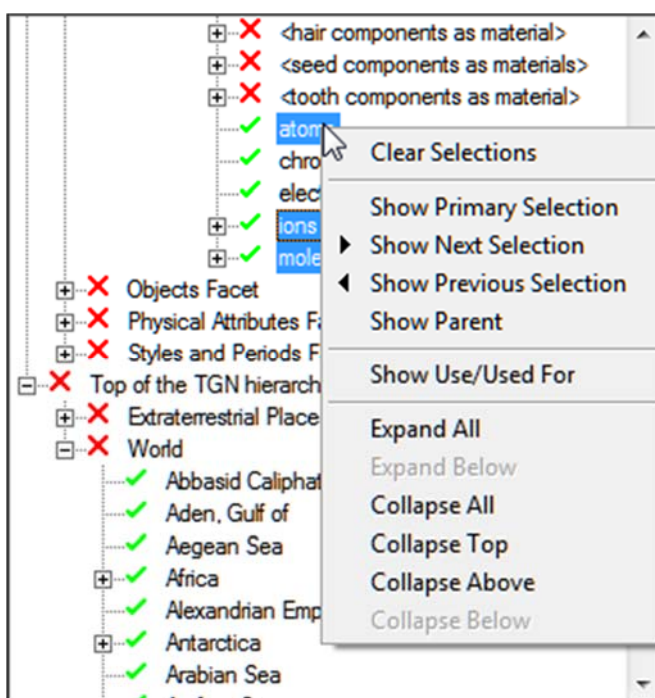
The primary selection

It is possible to select one or more terms in the Browse View (page 4) and on the Hierarchy View tab (page 11). The term that was last selected is called the *primary selection*. You can identify the *primary selection* by its dotted border. In this example two terms have been selected in Browse View, *ions* and *cations*, and *ions* is the *primary selection* term:



Browse View right-click menu options

Right-click the Browse View to display a menu with options for interacting with a term and, more generally, the hierarchy. In this example the `atoms` term has been right-clicked to display the menu:



If an option is grayed out, it is not currently valid.

It is not necessary to click a specific term in order to access the Browse View menu (you can click anywhere in the Browse View to access the menu); however some options are only valid when right-clicking a term.

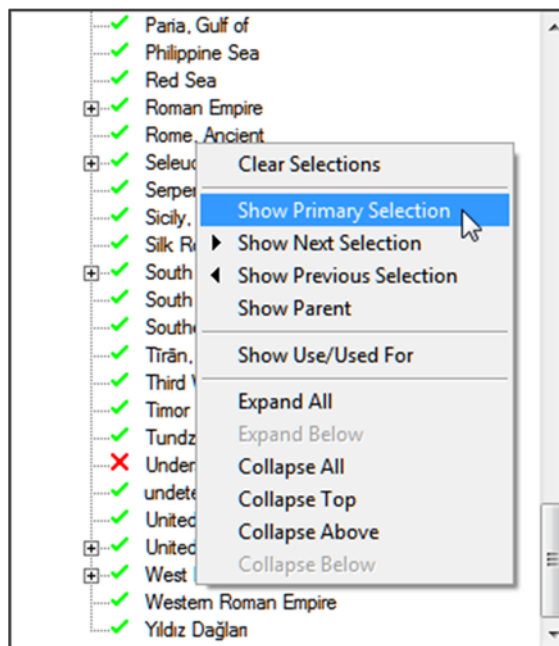
The menu options are:

Menu option	Description
Clear Selections	Clears the selection of any and all selected terms.
Show Primary Selection	Locate and display the term that is the <i>primary selection</i> . This option is useful if you have scrolled away from the <i>primary selection</i> term in the Browse View. Right-click anywhere in the Browse View and select Show Primary Selection to locate the <i>primary selection</i> term in the Browse View. In this example the <i>primary selection</i> (<code>ions</code>) is not displayed in the Browse View. Right-click anywhere in

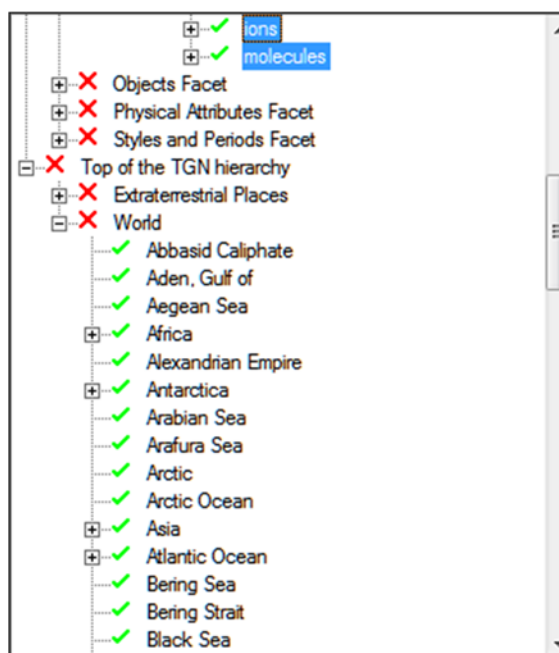
Menu option

Description

the Browse View and select **Show Primary Selection**:



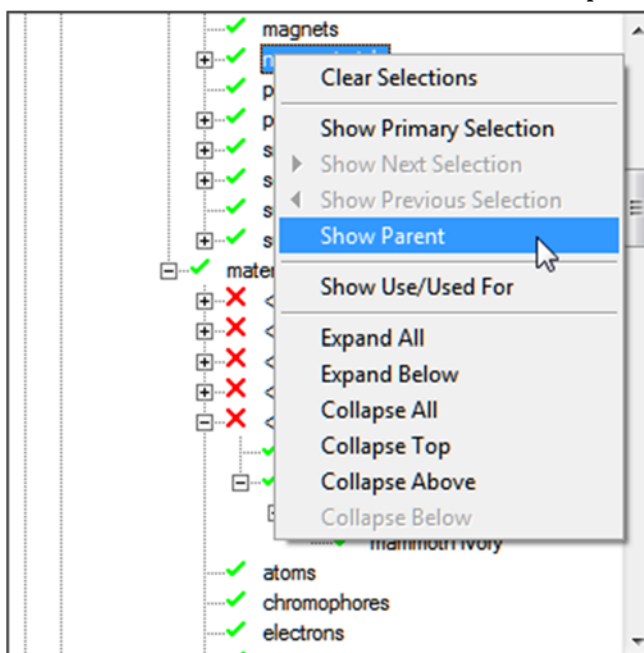
ions is located and displayed at the top of the Browse View:

**Show Next Selection**

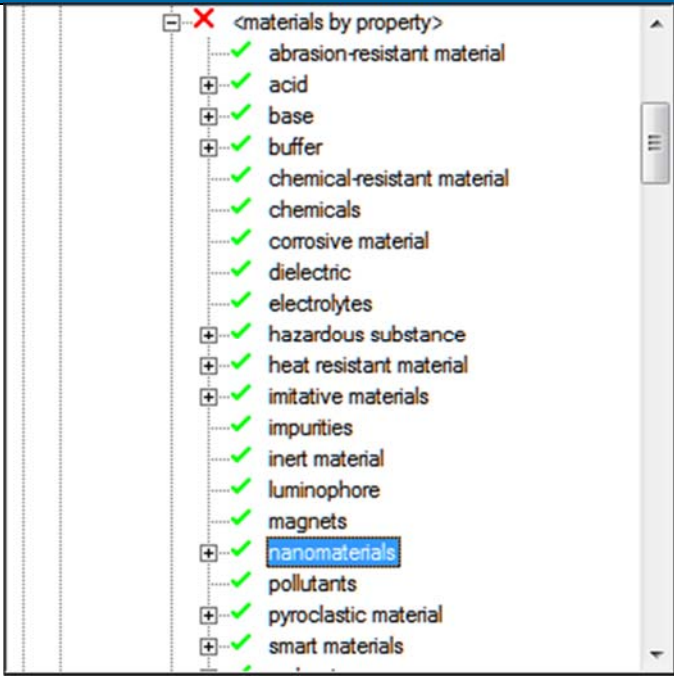

Move to the next selected term and make it the *primary selection*.

In the example above, *ions* is the *primary selection*. Right-click anywhere in the Browse View and select

Menu option	Description
	<p>Show Next Selection to make <code>molecules</code> the <i>primary selection</i>.</p> <p>If you were to select Show Next Selection again, the <i>primary selection</i> would loop back to the first selected term, <code>atoms</code> in this case.</p>
Show Previous Selection	<p>Move to the previous selected term and make it the <i>primary selection</i>.</p> <p>In the example above, <code>ions</code> is the <i>primary selection</i>. Right-click anywhere in the Browse View and select Show Previous Selection to make <code>atoms</code> the <i>primary selection</i> and display its details in the main module window.</p> <p>If you were to select Show Previous Selection again, the <i>primary selection</i> would loop back to the last selected term, <code>molecules</code> in this case.</p>
Show Parent	<p>Right-click a child term to reveal its immediate parent term. The parent term is displayed at the top of the Browse View.</p> <p>When we select Show Parent in this example:</p>

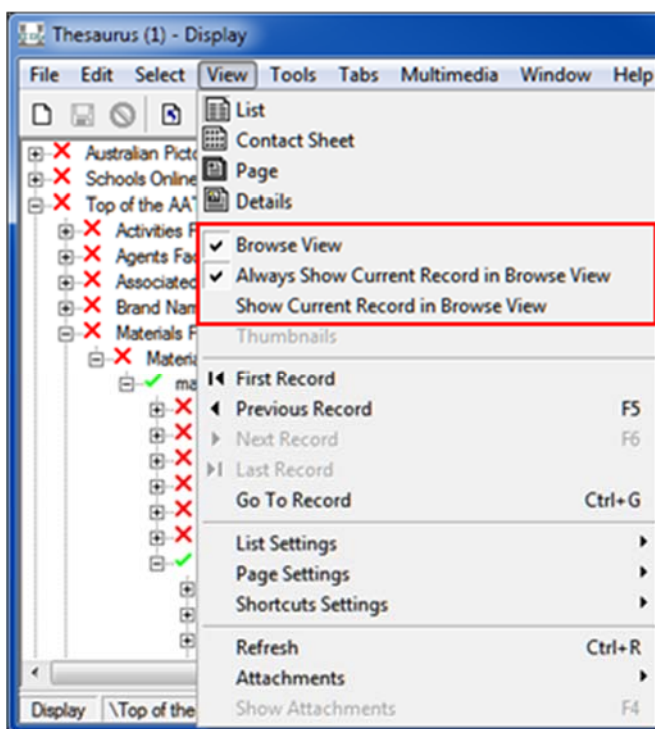


the parent for `nanomaterials` will display at the top of the Browse View:

Menu option	Description
	
Show Use/Used For	<p>Select to show / hide the display of <i>Use</i> and <i>Used For</i> terms. Simultaneously toggles the display of <i>Use</i> and <i>Used For</i> terms on the Hierarchy View tab.</p> <p>See <i>Hierarchy View Tab</i> (page 11) for details.</p>
Expand All	<p>Expand all terms that have previously been expanded.</p> <div data-bbox="714 1220 1482 1421" style="background-color: #e6f2ff; padding: 10px;">  <p>Only previously expanded terms are expanded because a thesaurus can contain more terms than can be loaded and displayed in the Browse View in a reasonable length of time.</p> </div>
Expand Below	<p>Expand the right-clicked term to reveal its immediate child terms.</p>
Collapse All	<p>Collapse all expanded terms.</p>
Collapse Top	<p>Collapse the hierarchy to the highest level of the term that is right-clicked.</p>
Collapse Above	<p>Collapse the hierarchy to the immediate parent term of the term that is right-clicked.</p>
Collapse Below	<p>Collapse any children terms of the term that is right-clicked</p>

View menu options

The View menu has a number of options specific to the Browse View:



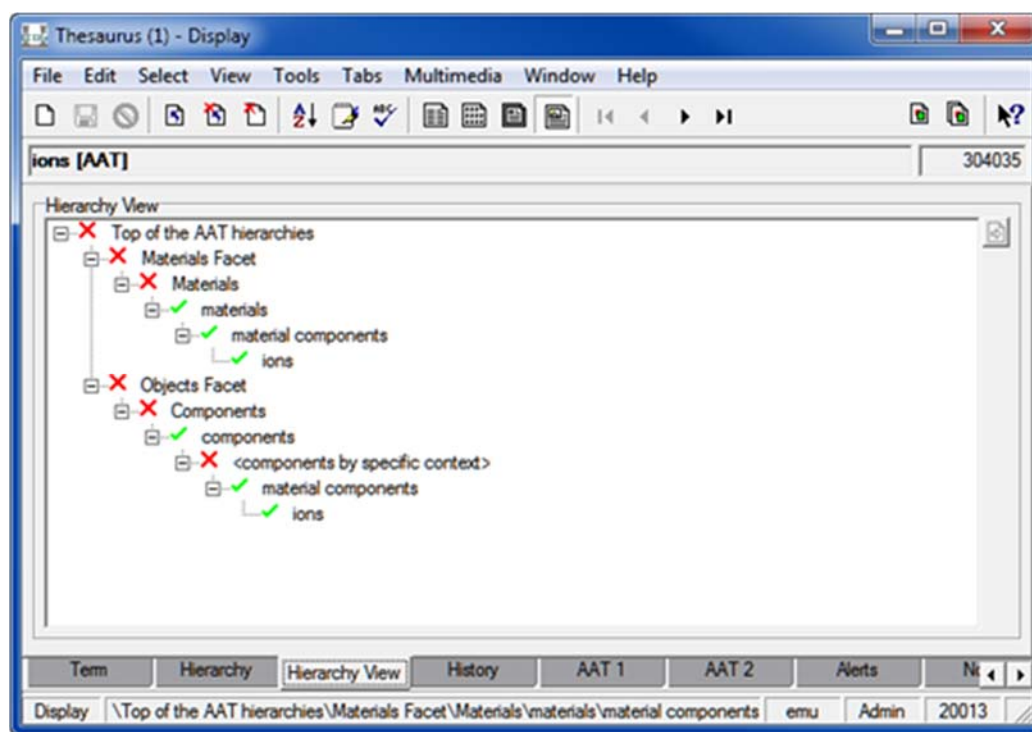
Menu option	Description
Browse View	Show/hide the Browse View panel.
Always Show Current Record in Browse View	<p>While this option is enabled (as shown above), as the current record changes in the main module window (e.g. by clicking a row in List View, or navigating through records in Details View), the term for the current record will be located and shown in the Browse View hierarchy and will become the <i>primary selection</i>.</p> <p>Select again to turn off this feature (removing the tick from beside the menu option).</p> <p>If the term appears in multiple branches in the Browse View hierarchy, each instance of the term will be selected.</p>

Menu option	Description
Show Current Record in Browse View	<p>When clicked, the term for the current record in the main module window will be located and shown in Browse View and will become the <i>primary selection</i>.</p> <p>If the term appears in multiple branches in the Browse View hierarchy, each instance of the term will be selected.</p> <p>This option is also available by right-clicking a row in List View.</p>

SECTION 2

Hierarchy View tab

The Hierarchy View tab displays the expanded thesaurus hierarchy for the term of the current record. In this example, we see the full hierarchy of the *ions* term:



The Hierarchy View tab provides a visual representation of the hierarchy of a single term in a thesaurus. As such the Hierarchy View tab:

- Does not display the current term's child terms
- AND-
- Only displays *Use* and *Used for* terms for the current term.

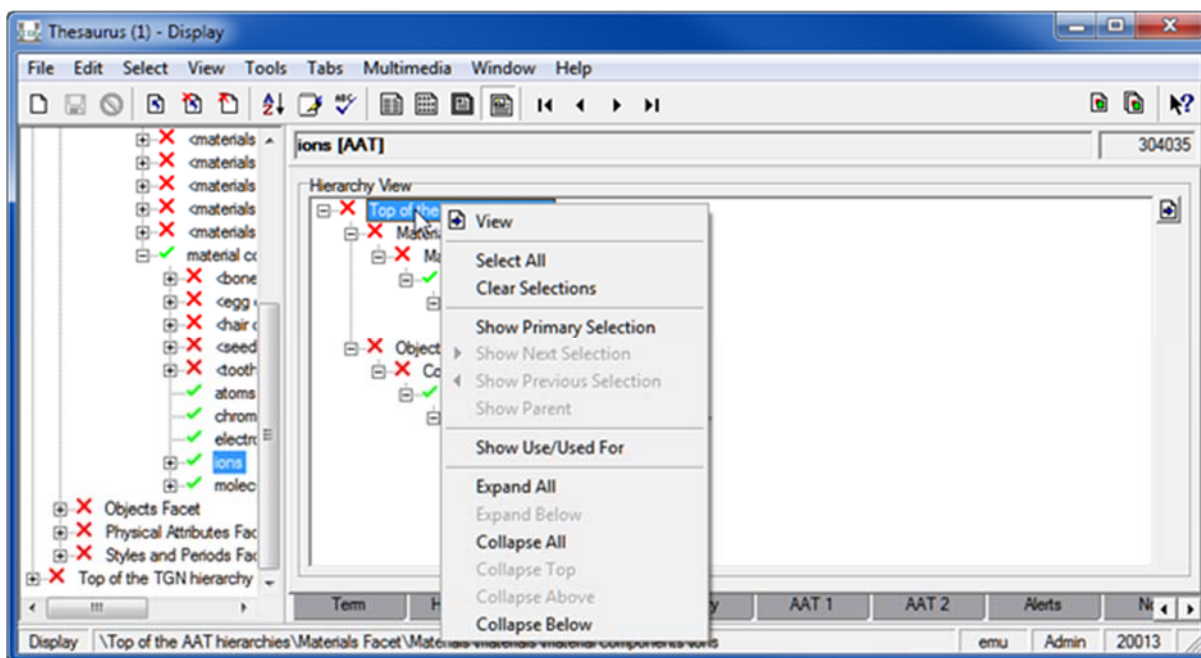


The Browse View does provide these two features.

As with the Browse View, multiple terms can be selected on the Hierarchy View tab using the Control (^) or Shift (⇧) keys while clicking with the mouse.

Hierarchy View right-click menu options


Right-click the Hierarchy View to display a menu with options for interacting with a term and, more generally, the hierarchy:

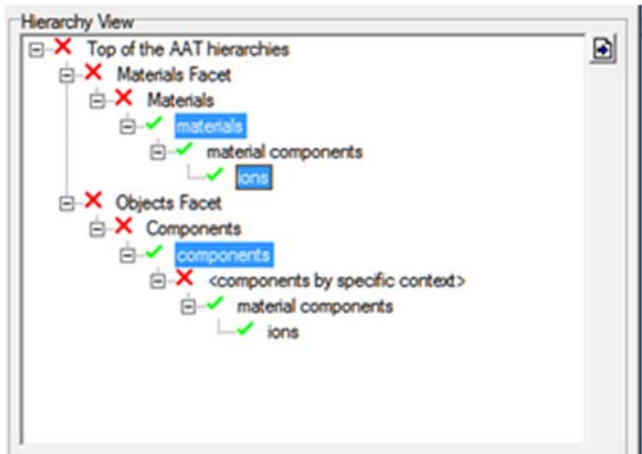


If an option is grayed out, it is not currently valid.

It is not necessary to click a specific term in order to access the Browse View menu (you can click anywhere in the Browse View to access the menu); however some options are only valid when right-clicking a term.

The menu options are:

Menu option	Description
View	Select View (or use the  button) to open another instance of the Thesaurus module and display details of the selected term(s).
Select All	Selects all terms on the Hierarchy View tab.
Clear Selections	Clear the selection of terms on the Hierarchy View tab.
Show Primary Selection	Locate and display the term that is the <i>primary selection</i> (page 3). This option is useful if you have scrolled away from the <i>primary selection</i> term on the Hierarchy View

Menu option	Description
Show Next Selection	<p>tab. Right-click anywhere in the Hierarchy View and select Show Primary Selection to locate the <i>primary selection</i> term on the Hierarchy View tab.</p> <p>Move to the next selected term and make it the <i>primary selection</i>.</p> <p>When several terms have been selected this option makes the next selected term the <i>primary selection</i>. In the following example, <i>ions</i> is the <i>primary selection</i>. Right-click anywhere in the Hierarchy View and select Show Next Selection to make <i>Components</i> the <i>primary selection</i>:</p>  <p>If you were to select Show Next Selection again, the <i>primary selection</i> would loop back to the first selected term, <i>materials</i> in this case.</p>
Show Previous Selection	<p>Move to the previous selected term and make it the <i>primary selection</i>.</p> <p>When several terms have been selected this option makes the previous selected term the <i>primary selection</i>. In the example above, <i>ions</i> is the <i>primary selection</i>. Right-click anywhere in the Hierarchy View and select Show Previous Selection to make <i>materials</i> the <i>primary selection</i>.</p> <p>If you were to select Show Previous Selection again, the <i>primary selection</i> would loop back to the last selected term, <i>components</i> in this case.</p>
Show Parent	<p>Right-click a child term to reveal its immediate parent term. The parent term is displayed at the top of the Hierarchy View.</p>

Menu option	Description
Show Use/Used For	Select to show / hide the display of <i>Use</i> and <i>Used For</i> terms. Simultaneously toggles the display of <i>Use</i> and <i>Used For</i> terms on the Browse View panel.
Expand All	Expand all terms.
Expand Below	Expand the right-clicked term to reveal immediate child terms.
Collapse All	Collapse all expanded terms.
Collapse Top	Collapse the hierarchy to the top-level of the term that is right-clicked.
Collapse Above	Collapse the hierarchy to the immediate parent of the term that is right-clicked.
Collapse Below	Collapse any children terms of the term that is right-clicked

Index

B

Browse View • 1

Browse View right-click menu options • 3, 4

H

Hierarchy View right-click menu options • 12

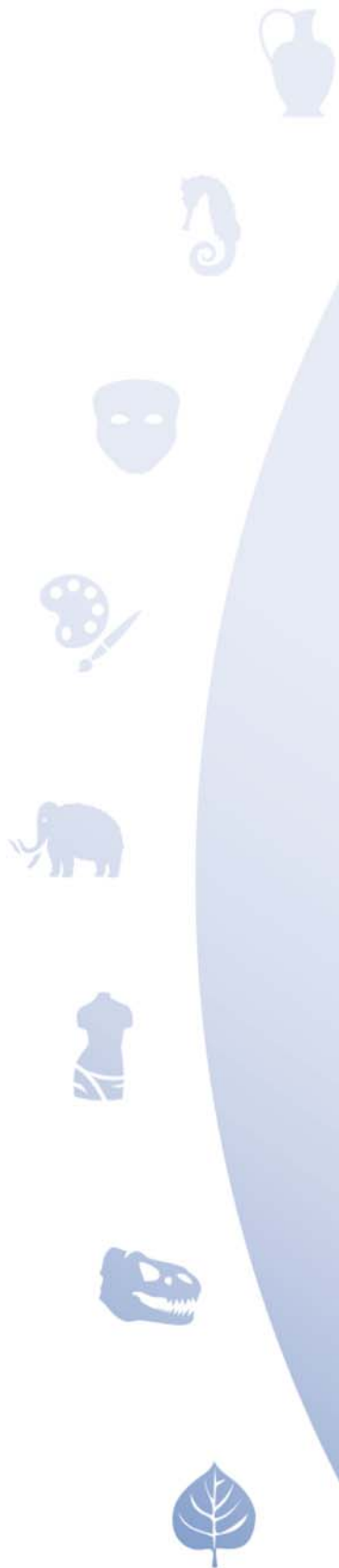
Hierarchy View tab • 1, 3, 8, 11

T

The primary selection • 3, 12

V

View menu options • 9



EMu Release Notes

ADO Reports

EMu 5.0

Document Version 1

EMu
Museum
Management
System

AXIELL
ARCHIVES LIBRARIES MUSEUMS



alm.axiell.com

© 2016 All rights reserved

Contents

SECTION 1	ADO Reports	1
	Note	1
SECTION 2	Crystal Reports	3
	How to create a Crystal ADO Report	3
	How to modify a Crystal Report to use ADO instead of ODBC	12
SECTION 3	Microsoft Excel	23
	How to create an Excel Report using the ADO RecordSet	23
	How to create an Excel Report with nested tables using the ADO RecordSet	35
SECTION 4	Registry entries	41
	Index	43

SECTION 1

ADO Reports

Report generation and performance have been improved with EMu 5.0 and it is now possible to report directly to an Open Database Connectivity (ODBC) data source and to an ActiveX Data Objects (ADO) RecordSet object, bypassing the ODBC filtering process.

The new report options are:

- Crystal Reports: report directly in ODBC format, bypassing the ODBC filtering process.
- Crystal ADO: report using ADO RecordSets for Crystal (which are accessible via Crystal's ADO connector).
- Microsoft ADO: report using ADO RecordSets for Microsoft products.



Crystal and Microsoft reports (Excel, Power Point and Word) which currently connect to an ODBC data source can be modified to use an ADO RecordSet.

It remains possible to create reports by connecting directly to an ODBC data source.

Note

This document assumes familiarity with Report creation in EMu. Full details about Report Creation are available in the EMu Help: **Working with EMu records>Reports.**

SECTION 2

Crystal Reports

Creating a Crystal Report using the new ADO RecordSet is similar to creating a Crystal report with a direct ODBC connection. The main differences are in selecting the data source. This document describes the differences.



How to create a Crystal ADO Report

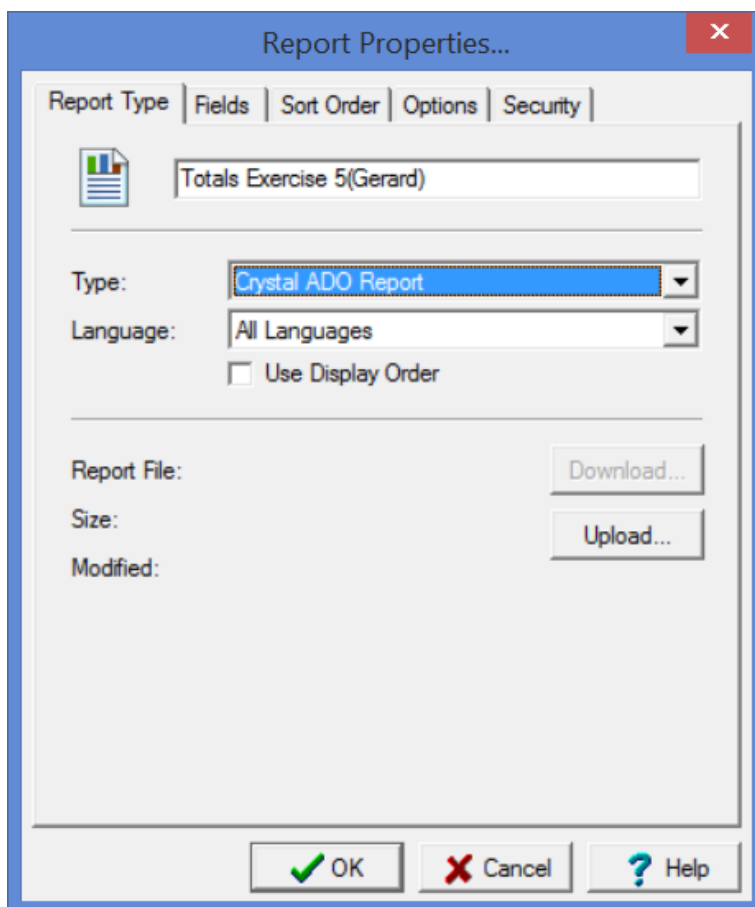
In EMu:

1. Search for or otherwise list a group of records on which to report.

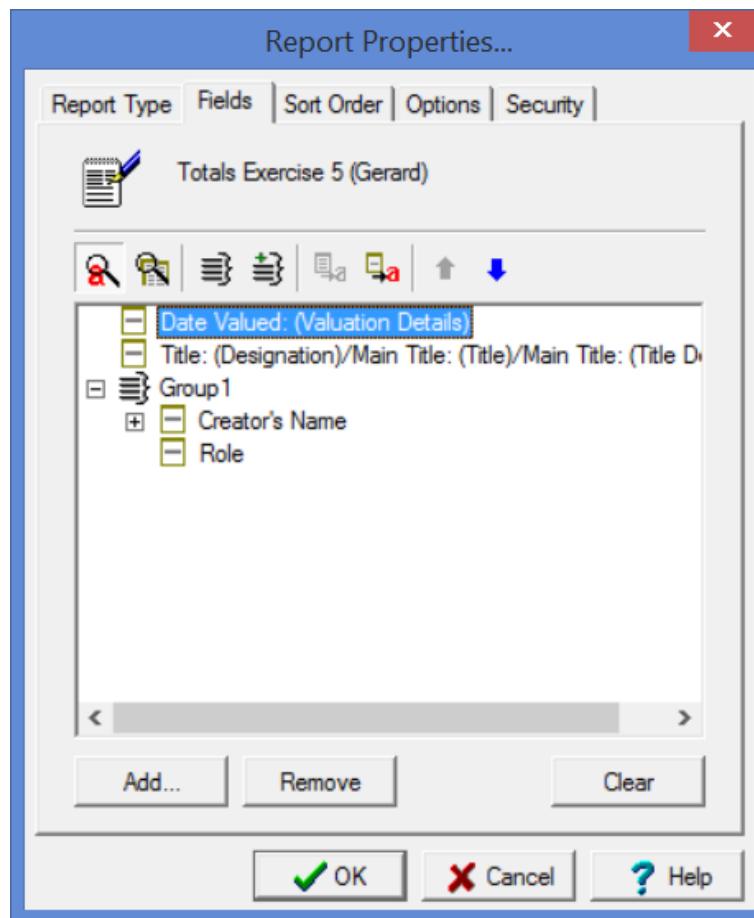


When designing a Crystal ADO report the records in your initial record set must have a value in each field to be included in the report. If not, the field name will not appear in the list of available columns. Once the report is defined, it does not matter if a record does not have values in every field included in the report.

2. Click **Reports**  in the Tool bar to display the Reports box.
3. Click  **New...** in the Reports box.
The Report Properties box displays.
4. Enter a descriptive name for the Report in the top text field.
5. Select Crystal ADO Report from the *Type* drop list:



6. On the Fields tab, add the fields to be included in the report.
In this example the fields selected are:




Note that a group was created using the **Create Group**  button.

7. Make changes on the other tabs as required.

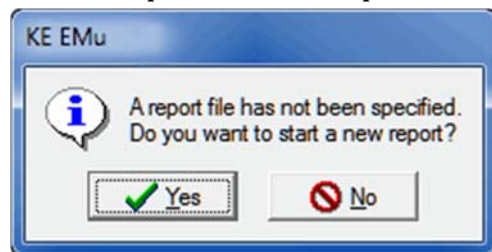
See the EMu Help for details about setting a sort order, sort options, and security.

8. Click .

The new report is added to the Reports box.

9. In the Reports box, select the new report and click  to run the report for the first time.

A message will display indicating that your report does not exist on the server. This is to be expected as the report has not yet been built in Crystal Reports:



10. Click .

An xml file is generated and saved with the data from your record set. The location of this file can vary, but typically it can be found in:

C:\Users\[*your username*]\AppData\Local\KESoftware\Reports\e[*module name*]

For example, a report run in the Parties module, will save the xmldata file to:

C:\Users\[*your username*]\AppData\Local\KESoftware\Reports\eparties

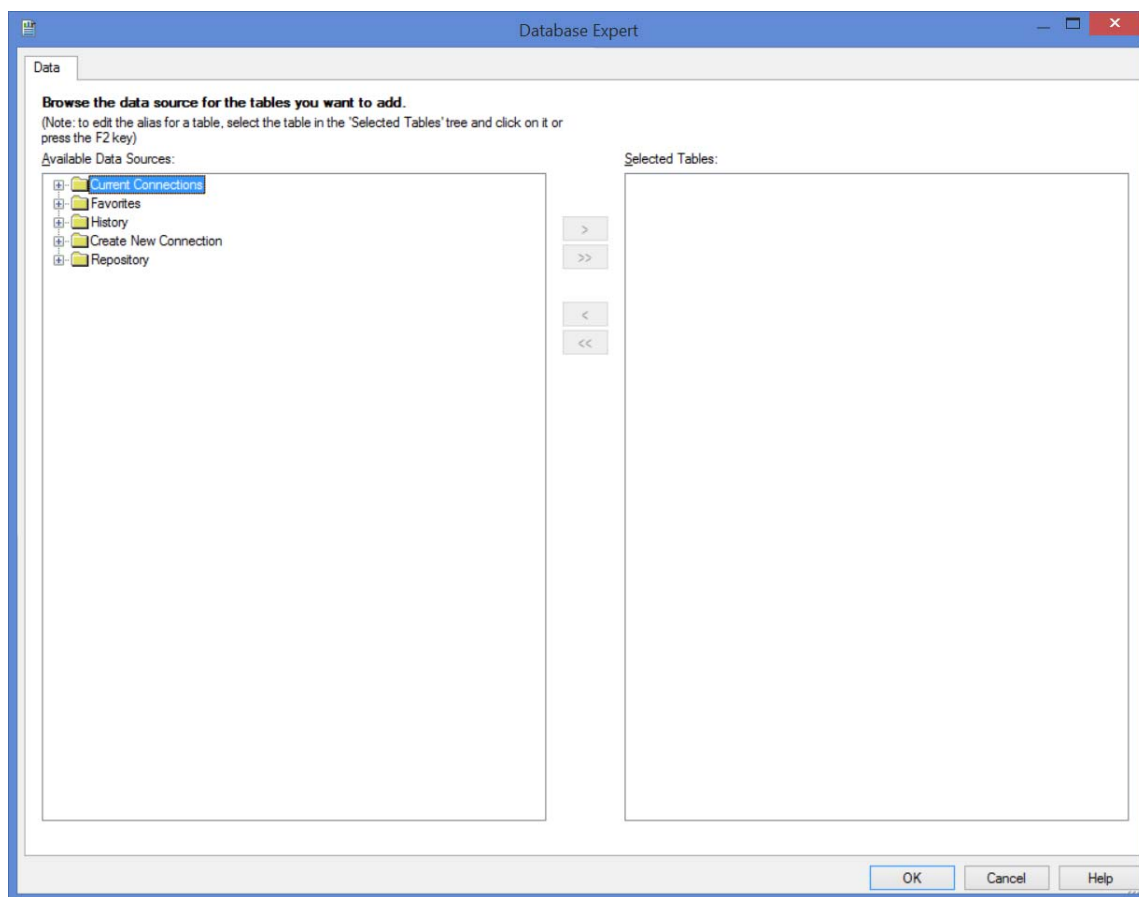
The Crystal Reports Designer application will open.

11. On the Start Page of the Crystal Reports Designer, select **Blank Report** under the New Reports heading

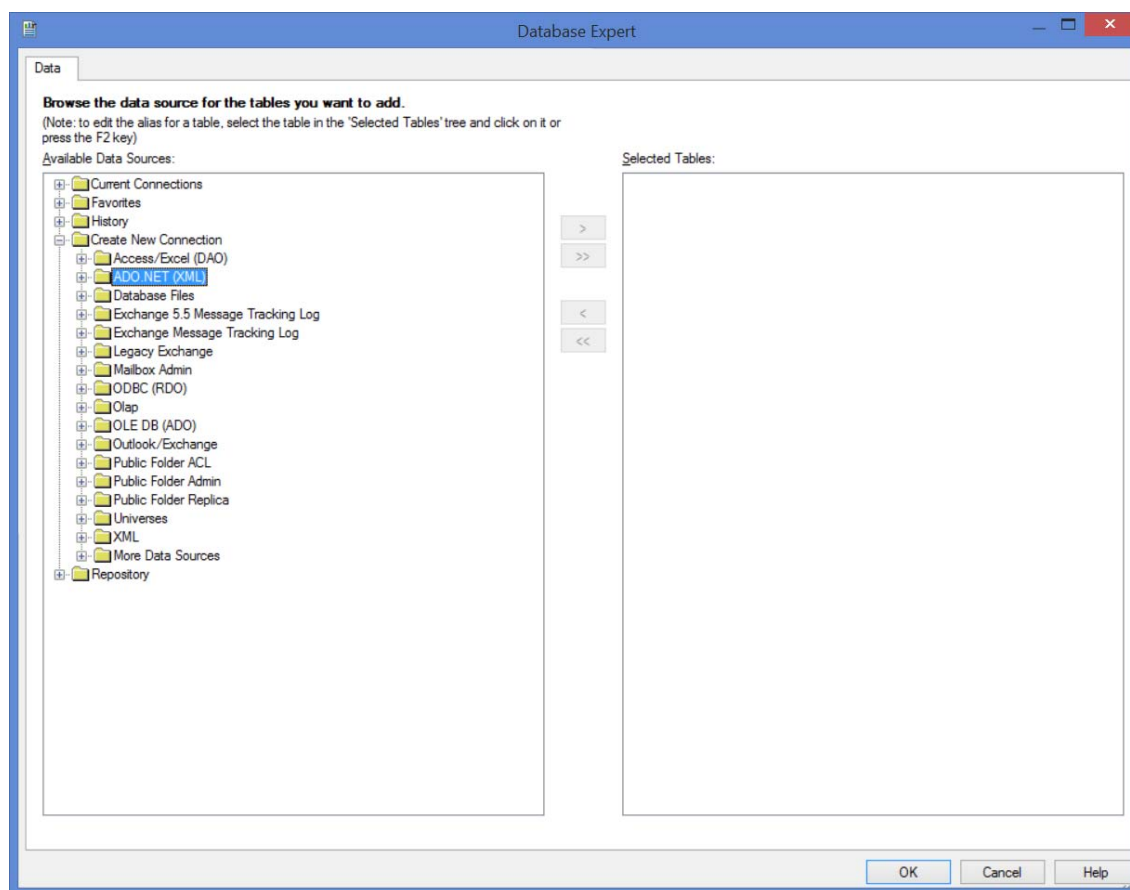
-OR-

Select **File>New>Blank Report** in the Menu bar.

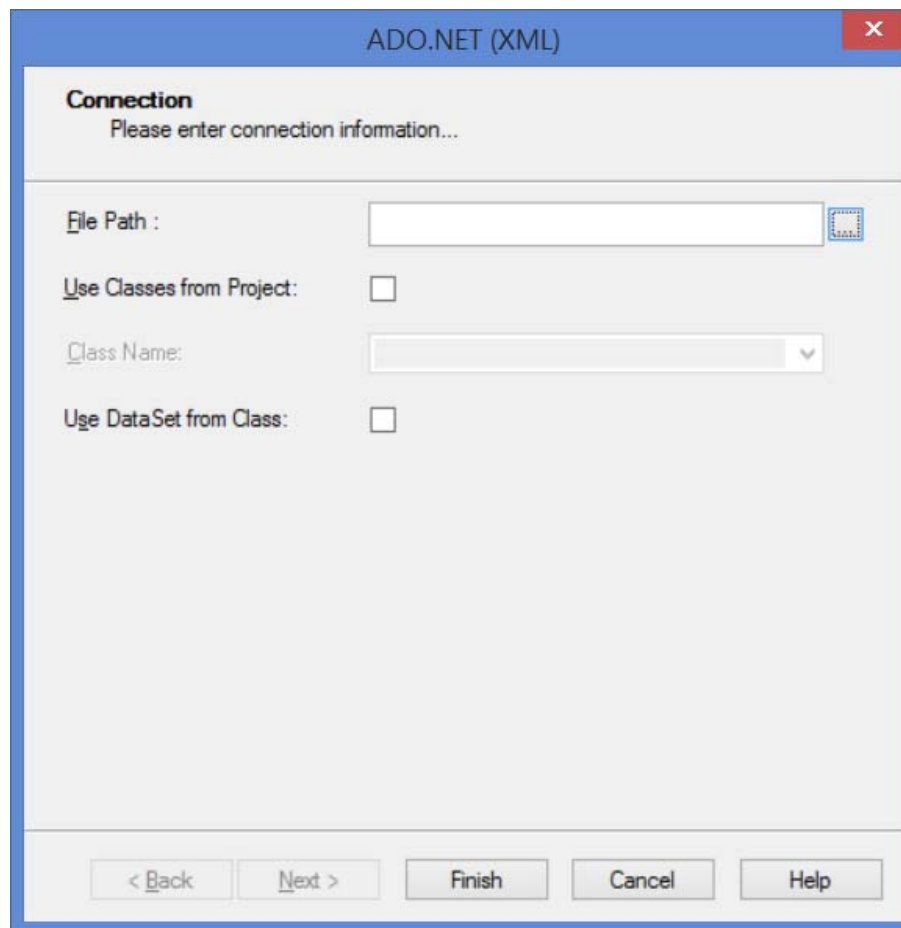
The Database Expert box displays:




12. Double-click **Create New Connection** and click  beside **ADO.NET (XML)**:

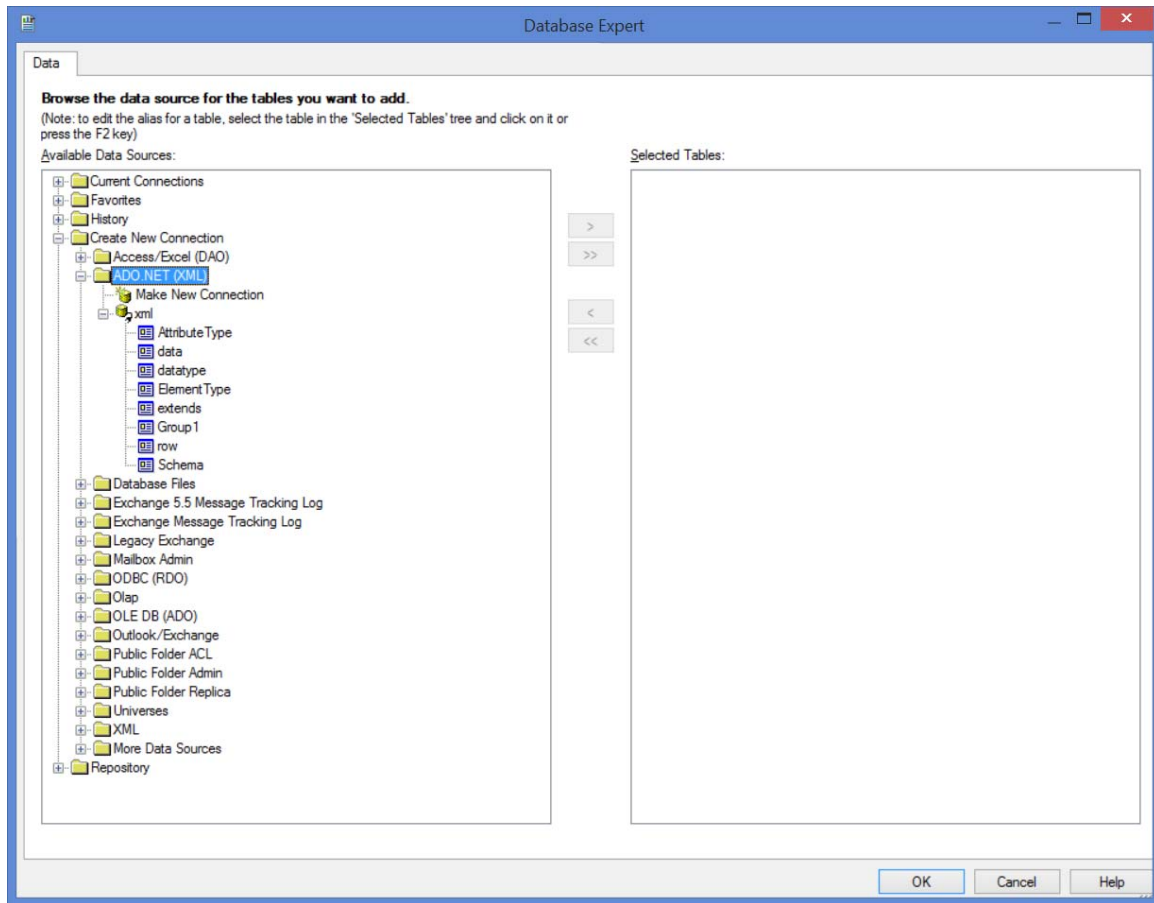


The following screen will display:



The image shows a Windows-style dialog box titled "ADO.NET (XML)". Inside the dialog, there is a section titled "Connection" with the instruction "Please enter connection information...". Below this, there are four fields: "File Path :" with a text box and a browse button (represented by a folder icon); "Use Classes from Project:" with an unchecked checkbox; "Class Name:" with a dropdown menu; and "Use DataSet from Class:" with an unchecked checkbox. At the bottom of the dialog, there are five buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".

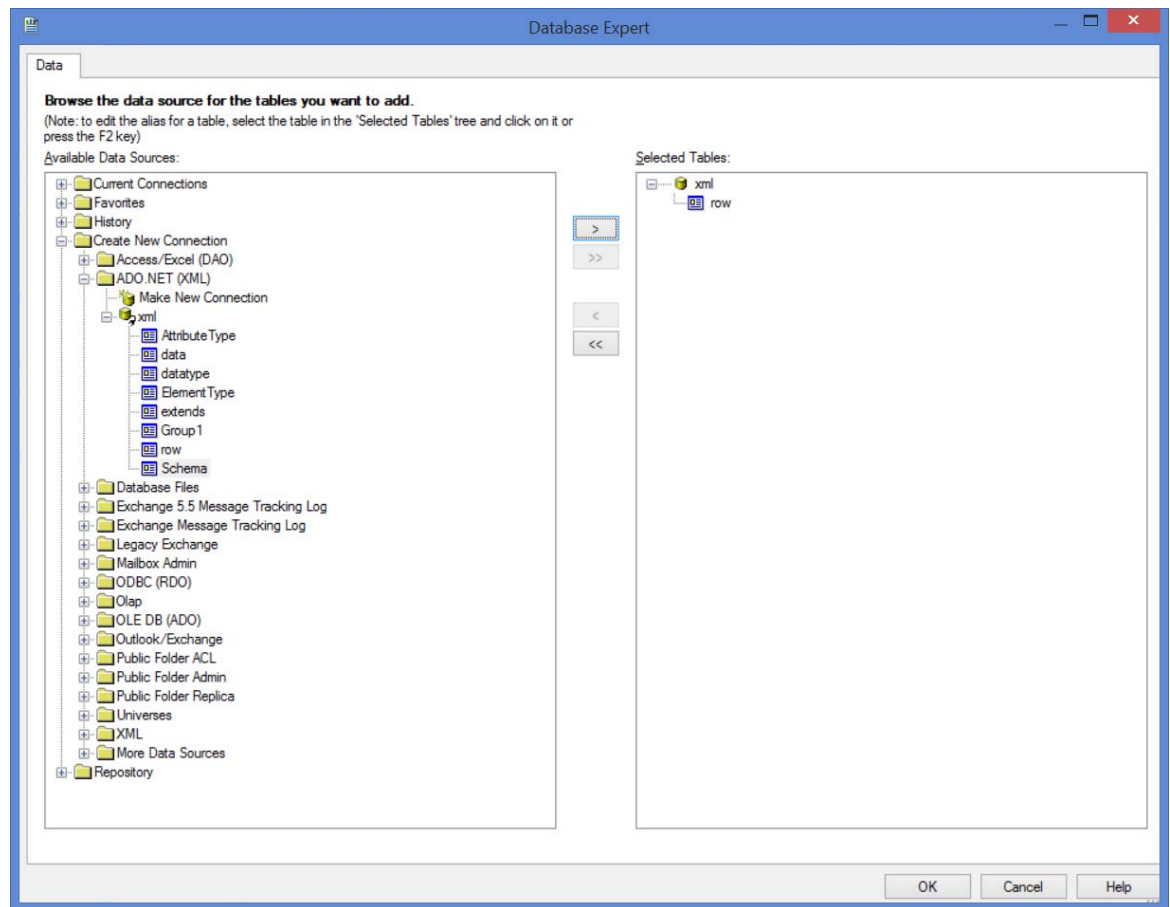
13. Click the button beside the *File Path* field to locate and select the `xmldata.xml` file created when the report was first run (Step 9).
See Step 10 for details of the location of the `xmldata.xml` file.
14. Click  to return to the Database Expert:



Group 1 contains values from fields that we grouped in the EMu report in this example (see Step 6). These fields are tables of values (they can hold more than one value). This data needs to be added to our report using a sub-report (see the EMu Help for details).

Field values from the Catalog are contained in the table called `row`.

15. Select **row** and add it to the *Selected Tables* pane:



16. Click .

The Crystal Report Designer displays, ready for you to design your Crystal report. See the EMu Help for details of designing a Crystal Report.

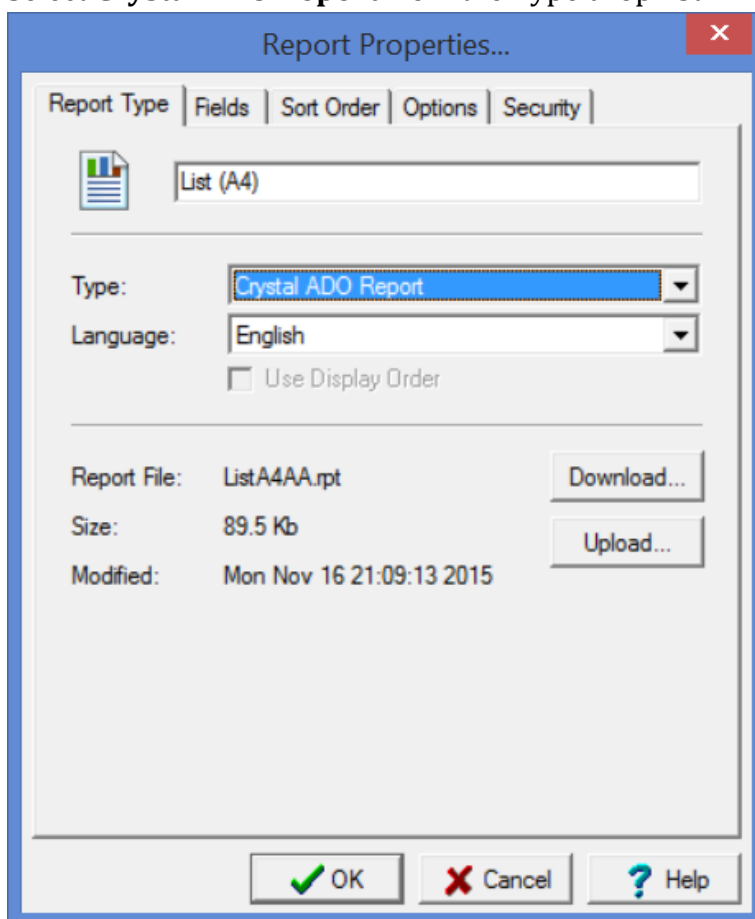


It is important not to move the `xmldata.xml` file as this will cause problems when sharing the report with other users.

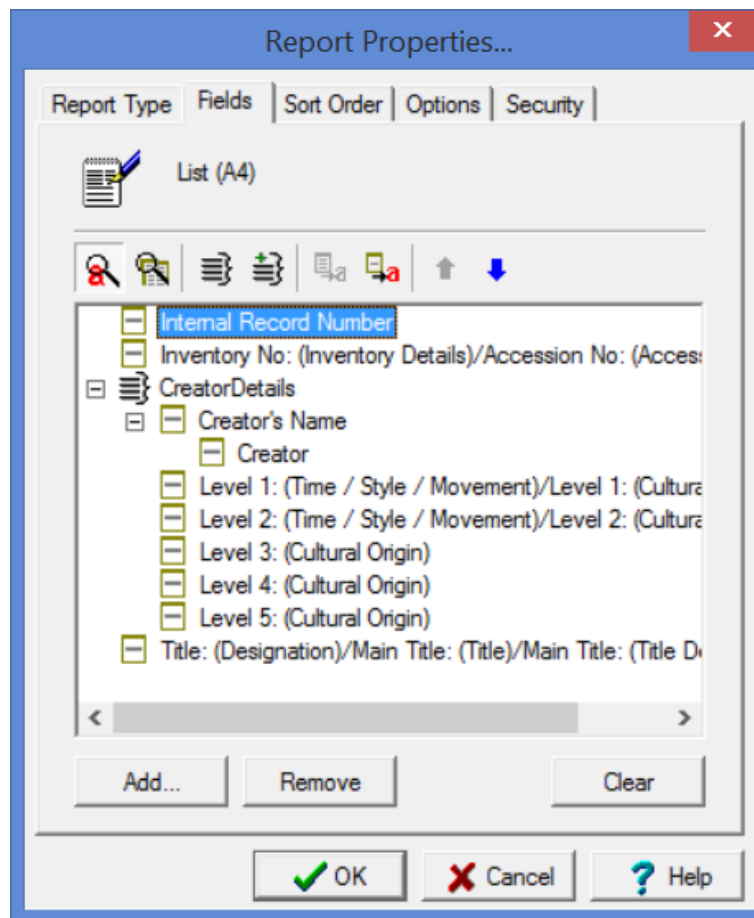
How to modify a Crystal Report to use ADO instead of ODBC

To modify a Crystal Report to use ADO rather than ODBC:

1. Open the Report Properties dialog for the report.
This example uses the default report `List (A4)`.
2. Select **Crystal ADO Report** from the Type drop list:



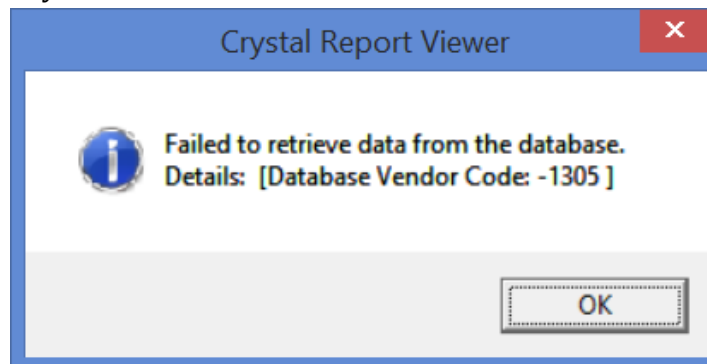
The fields for this report are:



Two tables are generated in this report.

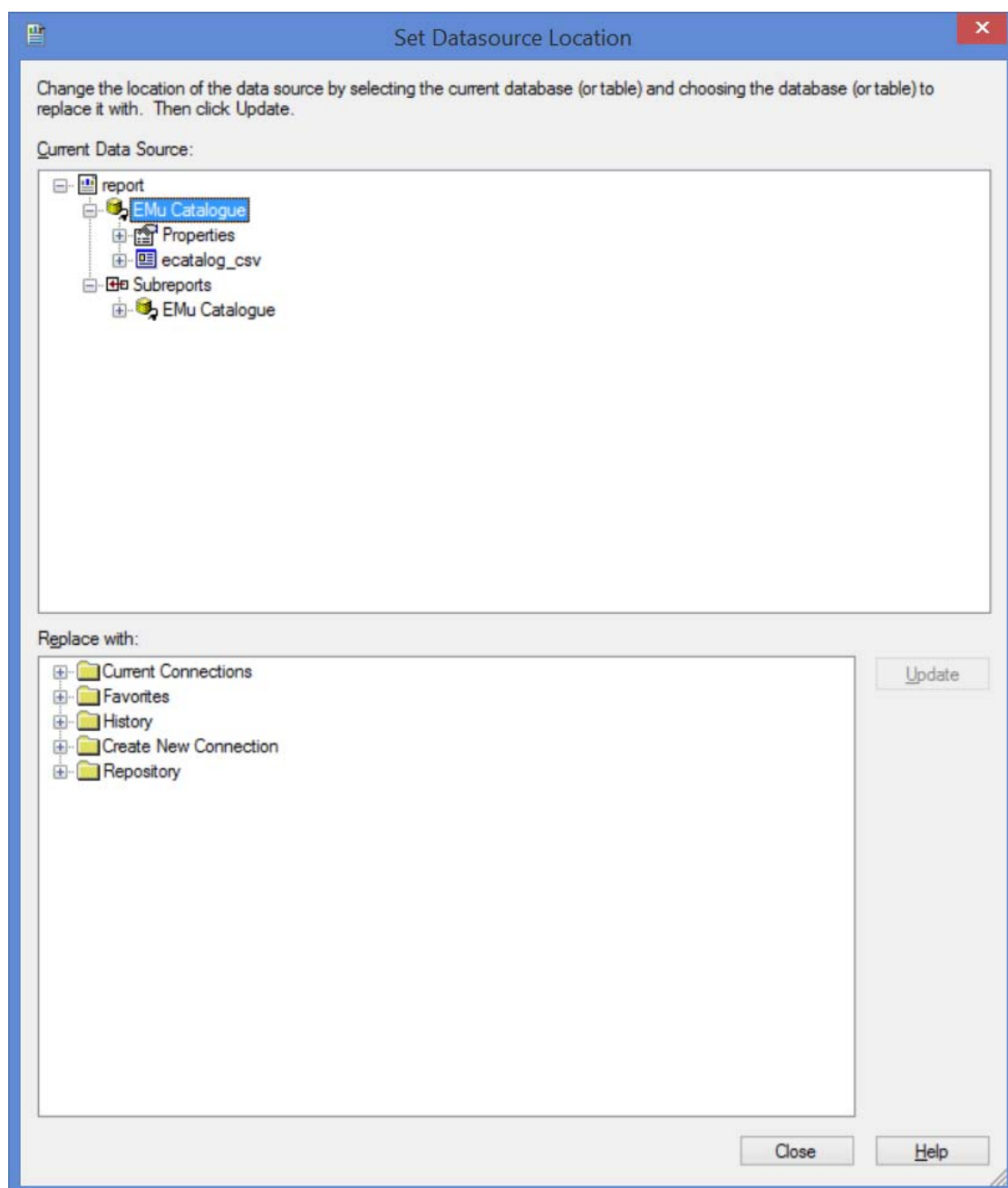
3. Click  and run the report.


Crystal will create the ADO record set and the following error will display:



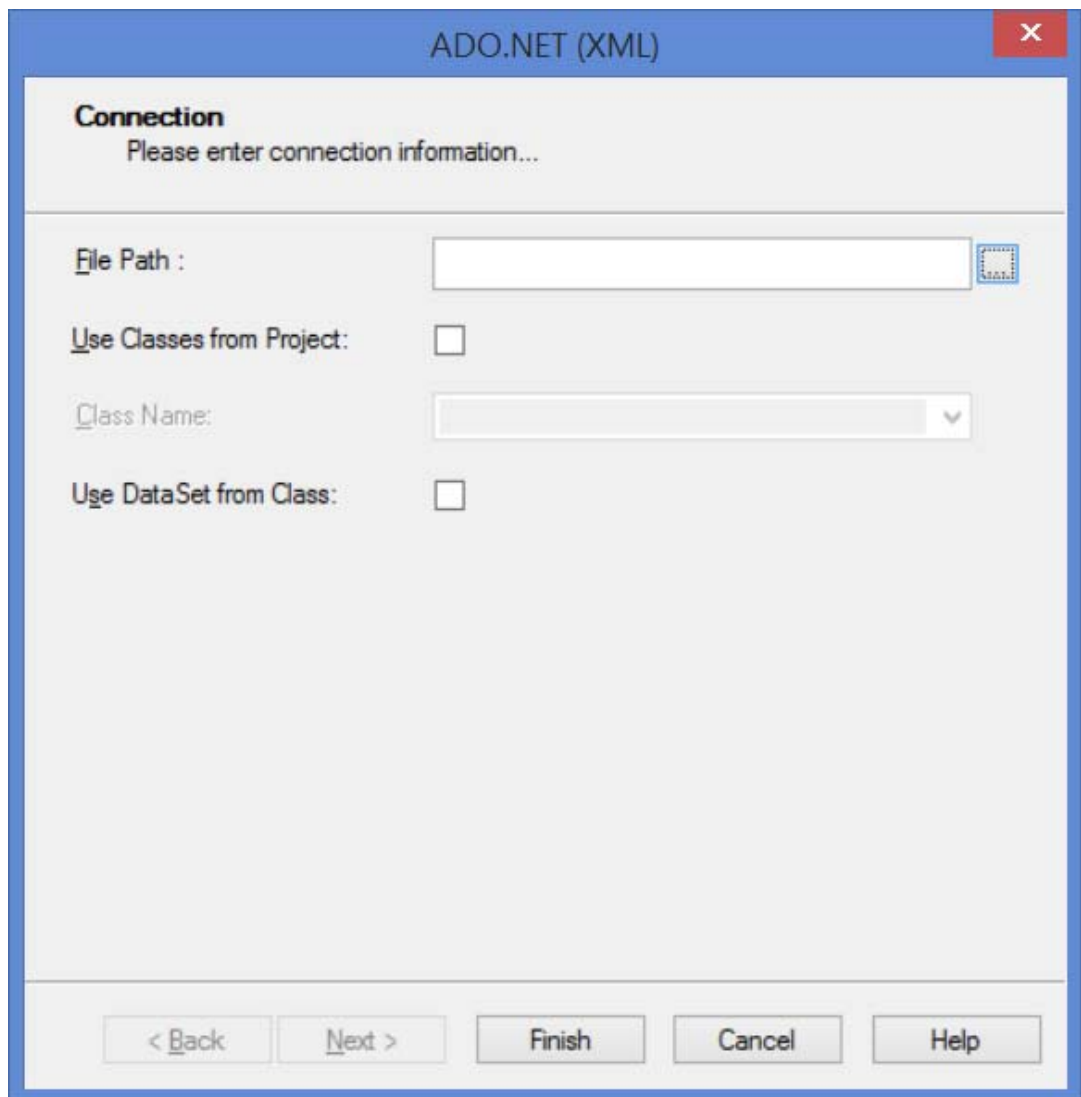
4. Open the Crystal report in the Crystal Report Designer and select the **Database>Set Datasource Location** menu option.

The Set Datasource Location dialog will display:



5. Select **Create New Connection** in the *Replace with* pane and click  beside **ADO.NET (XML)**.

The following screen will display:



The image shows a Windows dialog box titled "ADO.NET (XML)". Inside the dialog, there is a section titled "Connection" with the instruction "Please enter connection information...". Below this, there are four fields: "File Path :" with a text box and a browse button (represented by a folder icon); "Use Classes from Project:" with an unchecked checkbox; "Class Name:" with a text box and a dropdown arrow; and "Use DataSet from Class:" with an unchecked checkbox. At the bottom of the dialog, there are five buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".

6. Click the button beside the *File Path* field to locate and select the `xmldata.xml` file created when the report was run.

The location of this file can vary, but typically it can be found in:

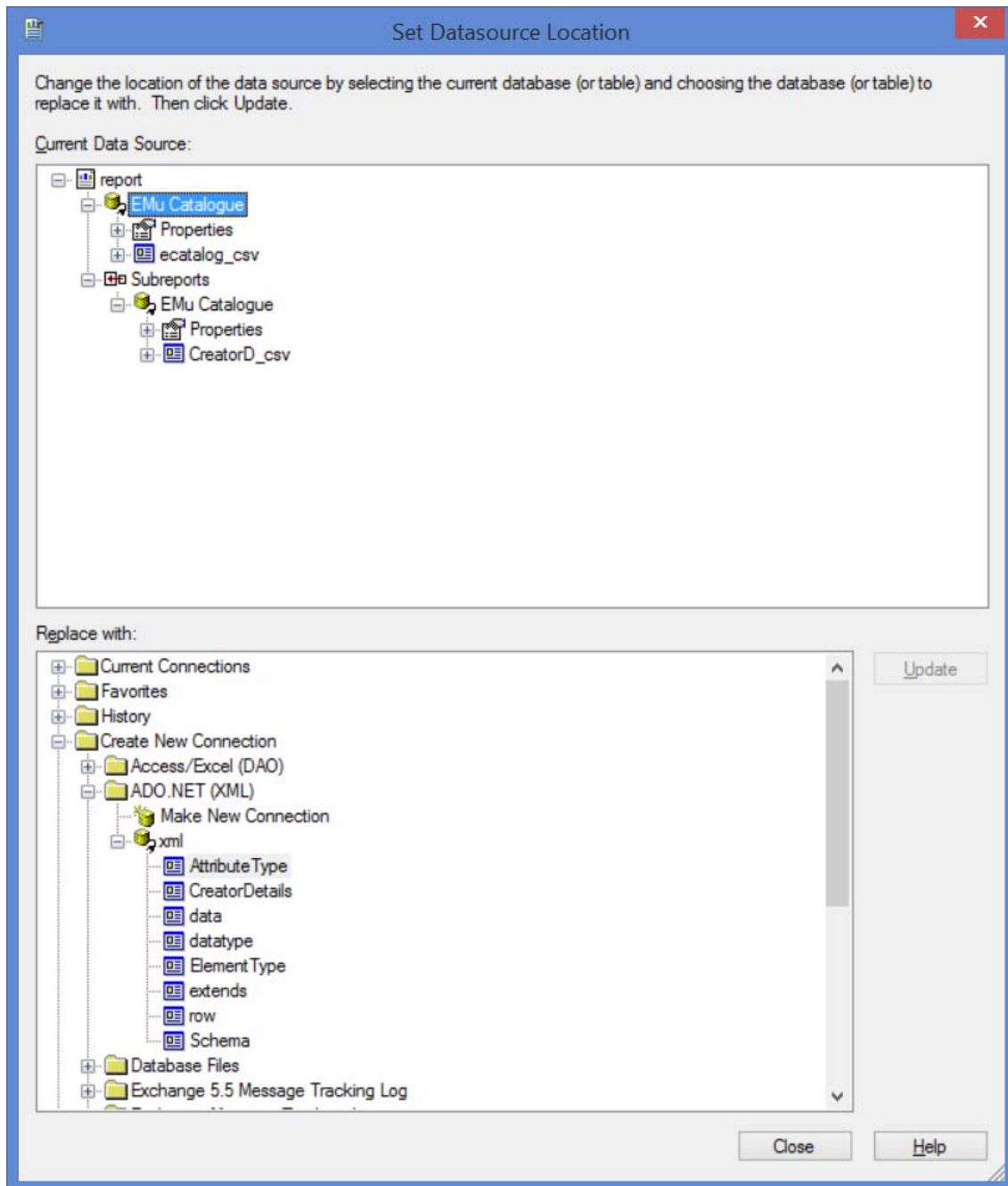
`C:\Users\[your username]\AppData\Local\KESoftware\Reports\e[module name]`

For example, a report run in the Parties module, will save the `xmldata` file to:

`C:\Users\[your username]\AppData\Local\KESoftware\Reports\eparties`

7. Click .

You are returned to the Set Datasource Location dialog:



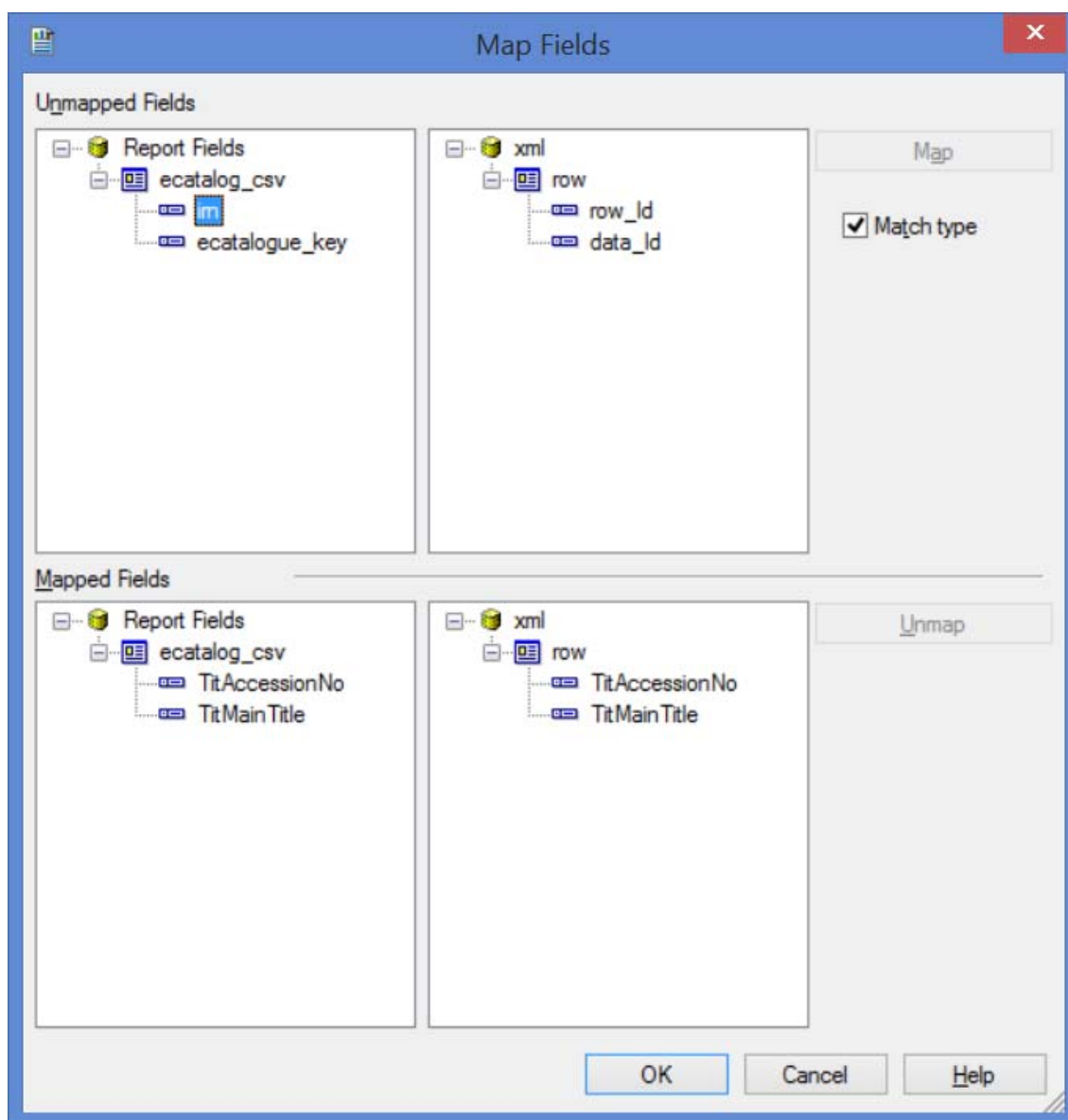
Next it is necessary to map fields from the old ODBC data source to the new ADO RecordSet.

In this example there are two tables to map and one sub-report.

8. To map the old ODBC Catalog fields to the new Catalog table, click **ecatalogue_csv** in the *Current Data Source* pane and then click the **row** table in the *Replace with* pane.

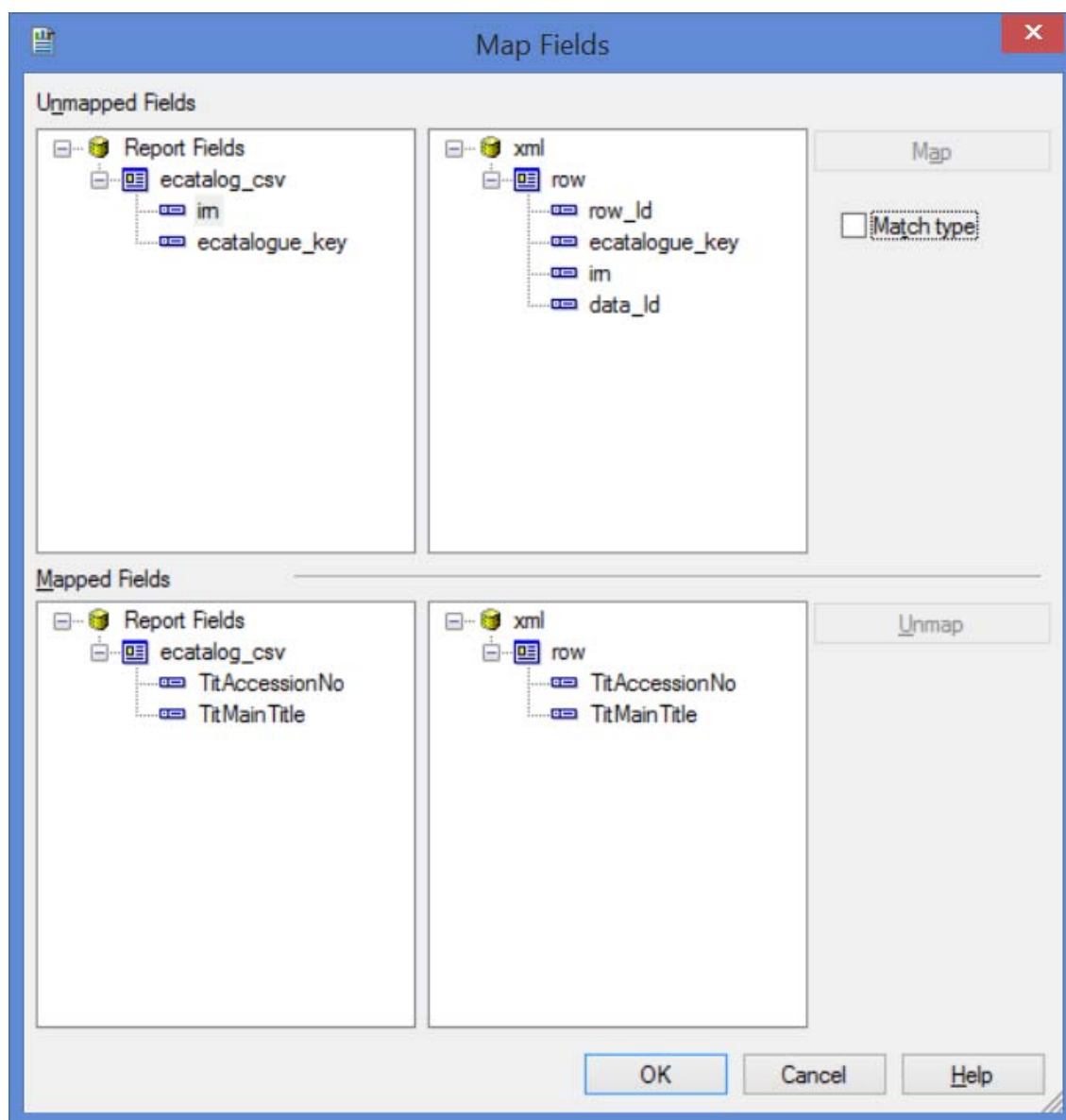
The Update button will be enabled.

9. Click the **Update** button and the Map Fields dialog will display:



Fields with the same name will be mapped automatically.

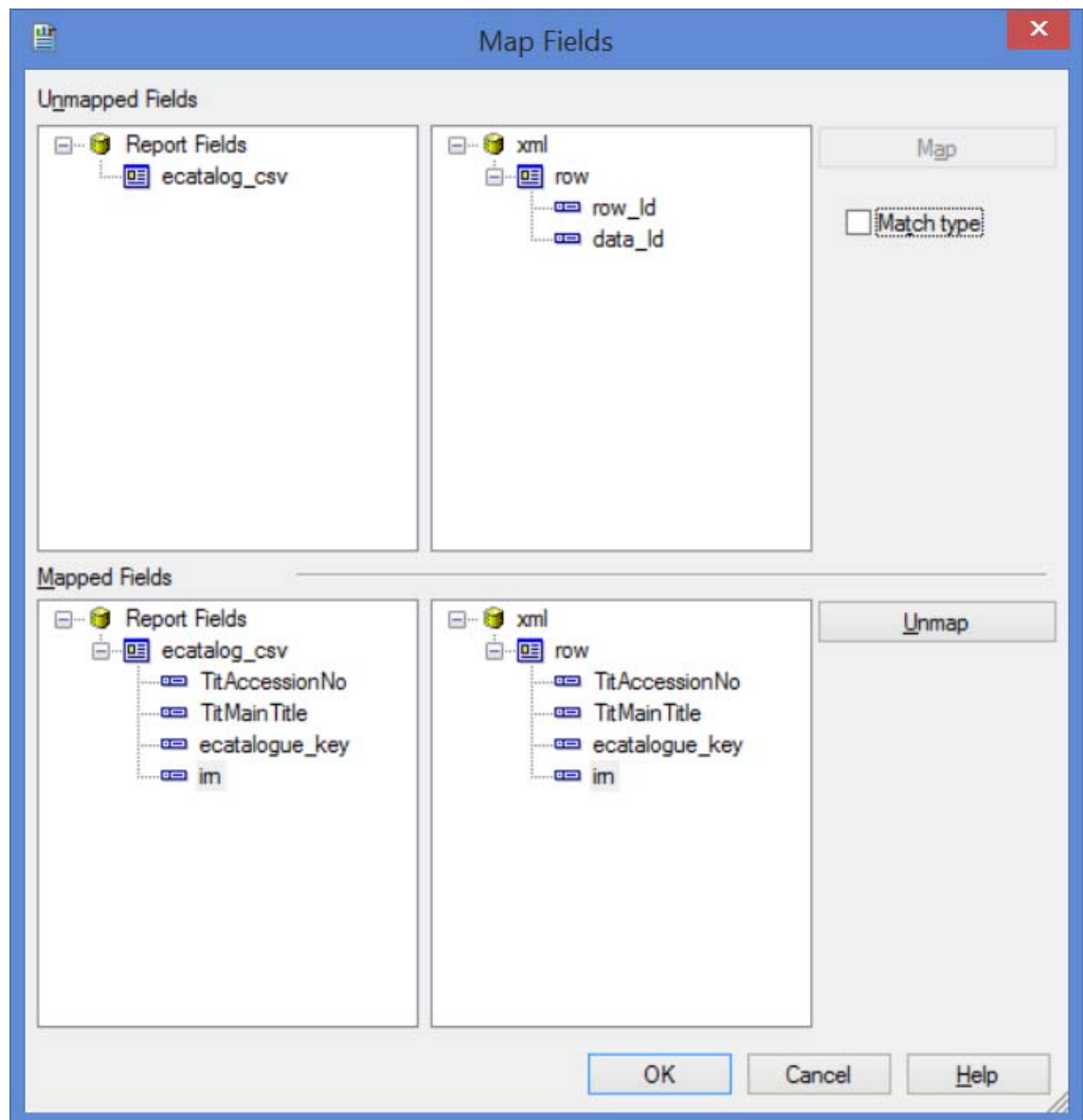
10. Uncheck the **Match type** check box to reveal more fields in the *Unmapped Fields* pane:

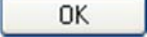


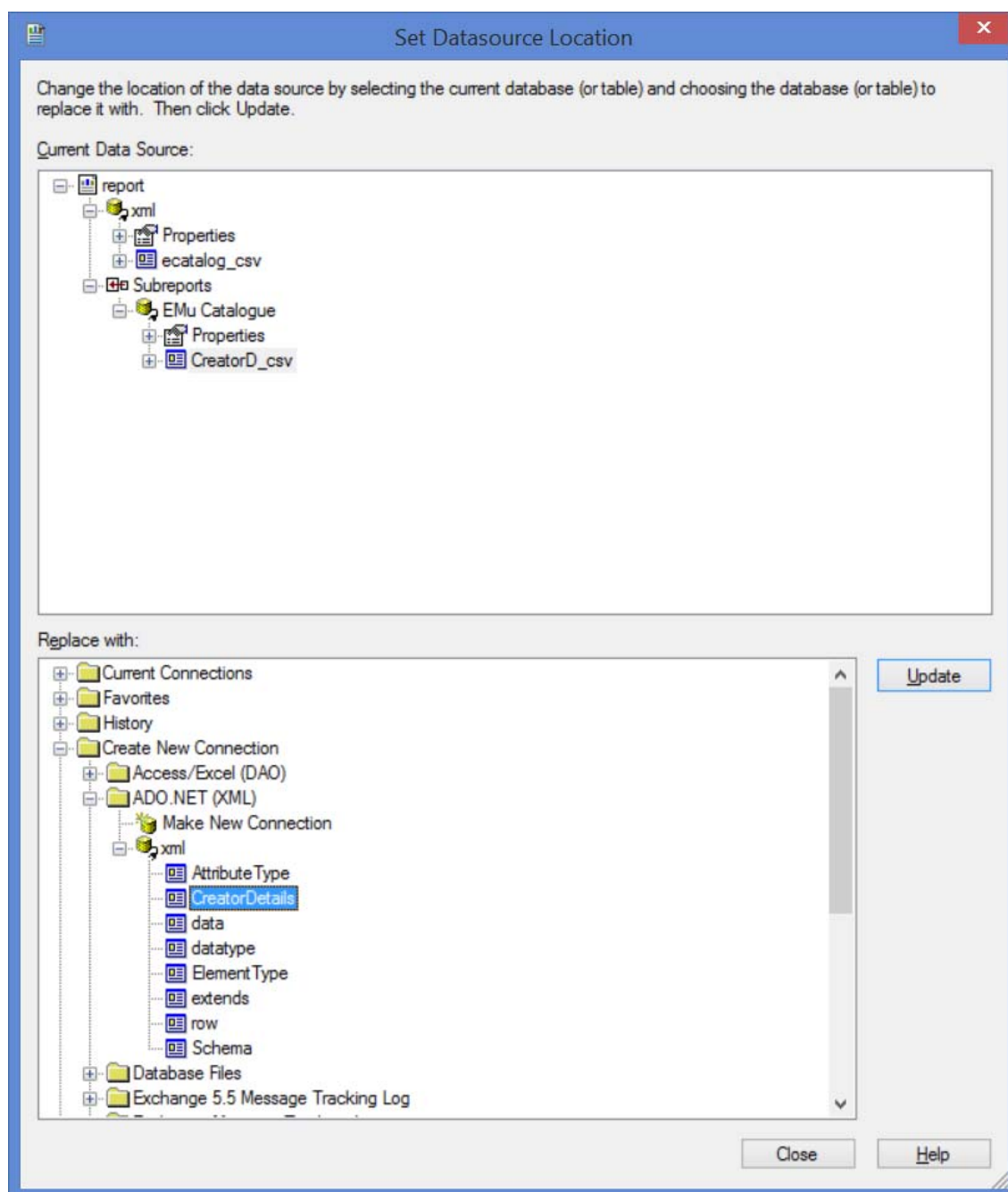
11. Complete mapping fields in the *Unmapped Fields* pane.

In this example we map `ecatalogue_key` to `ecatalogue_key` and `irn` to `irn` by selecting both fields to map and clicking the **Map** button.

Once mapped, fields will be moved to the *Mapped Fields* pane:



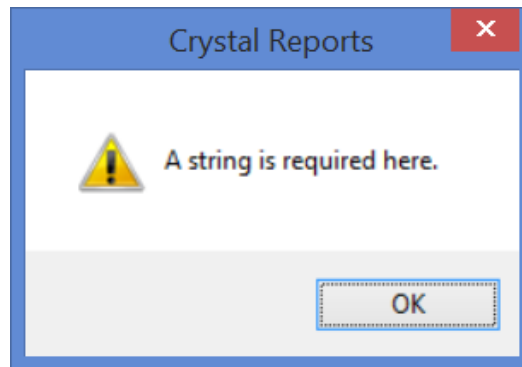
12. Click  when all fields are mapped.
You are returned to the Set Datasource Location dialog.
13. Repeat the mapping process for all fields (in this example, mapping fields in the CreatorD_csv table to the ADO table CreatorDetails):




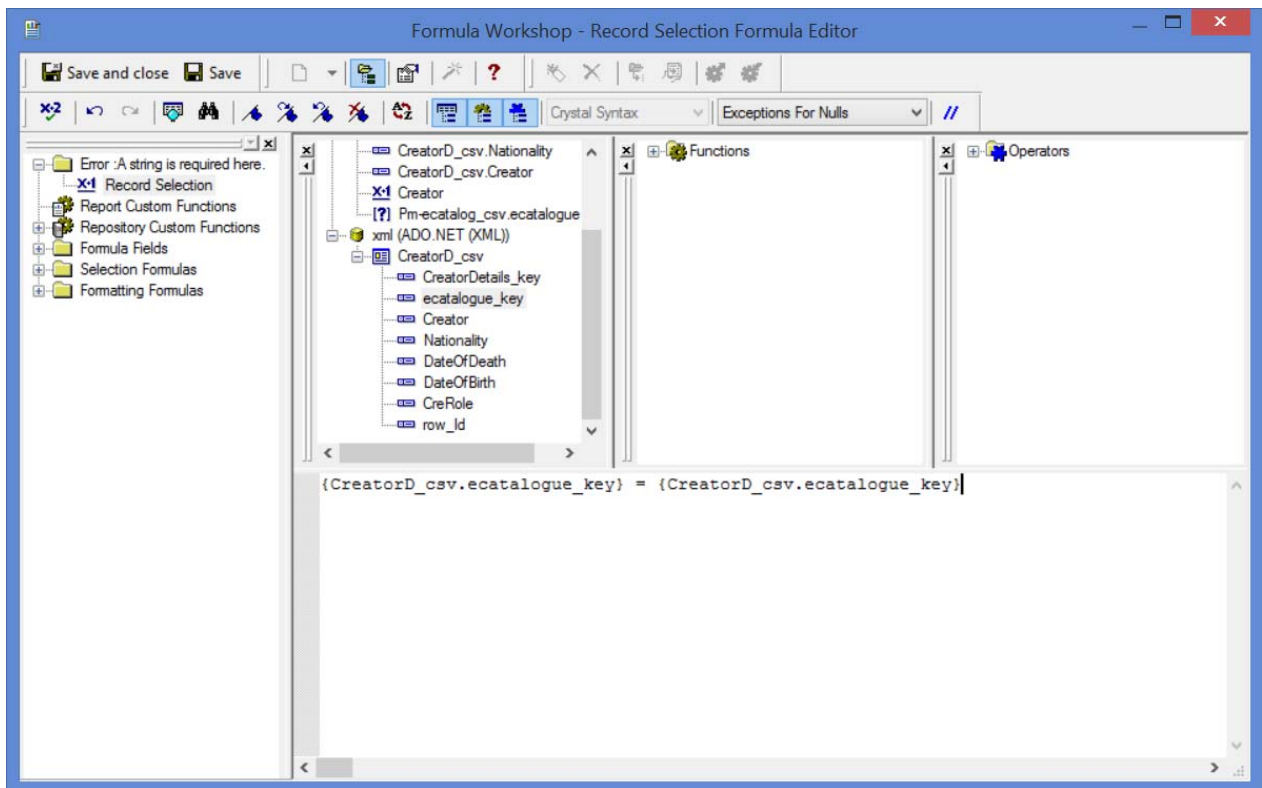
14. Once all fields have been remapped in all tables click **Close**.

You are returned to the Crystal design window.

If you refresh report data at this stage and you have a sub-report object, you will probably receive an error regarding sub-report links, e.g.:



Click  to open the Record Selection Formula Editor. Change the link key field used by the old ODBC table to the link key field referenced by the ADO RecordSet:



The report should now work correctly.

SECTION 3

Microsoft Excel



The following examples demonstrate how to create a basic Excel report using VBA. Please note that it is not the intention of this document to teach VBA.

Excel 2013 was used to create these reports.



How to create an Excel Report using the ADO RecordSet

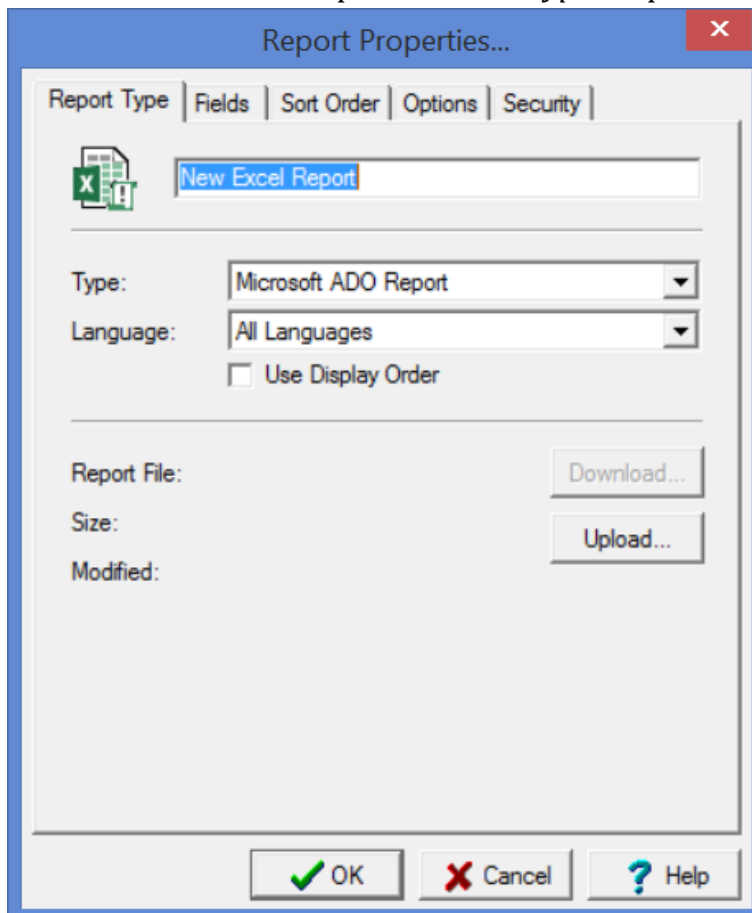
With ODBC data sources there is an option in Excel to open a connection without writing Visual Basic code. This is not the case when making a connection to an ADO record set and it is necessary to write VB code.

Step 1: Create a new report in EMu

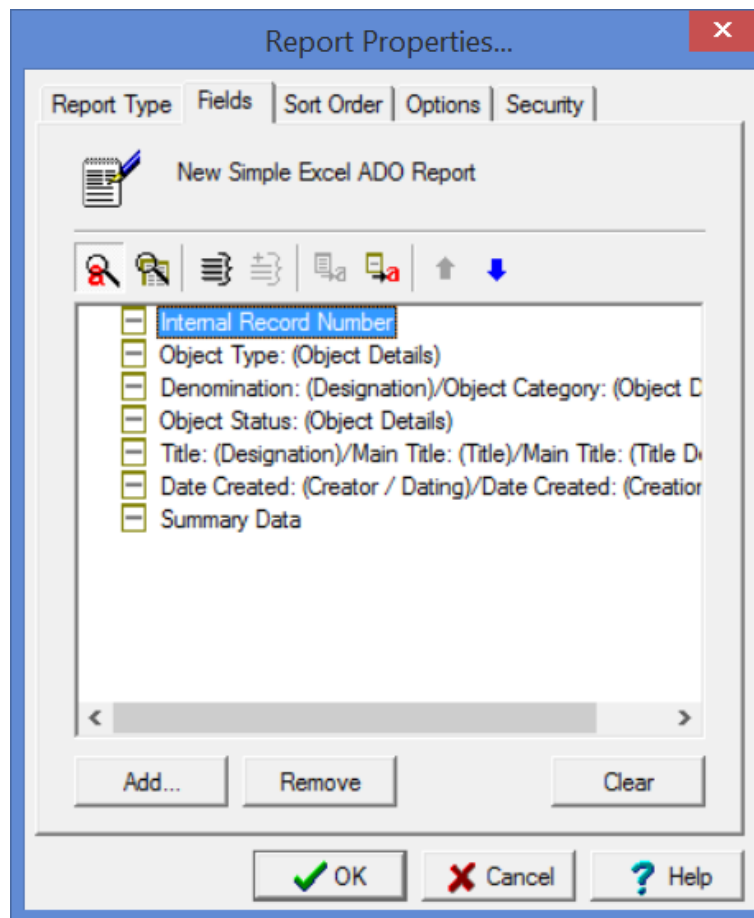
This first example is a simple report on single value fields from the Catalog module. The VBA code provided in this example will automatically populate headings and row data for each column selected.



In EMu:

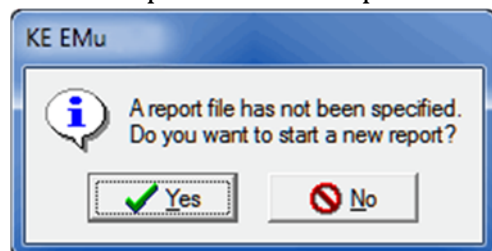
1. Search for or otherwise list a group of records on which to report.
2. Click **Reports**  in the Tool bar to display the Reports box.
3. Click  in the Reports box.
The Report Properties box displays.
4. Enter a descriptive name for the Report in the top text field.
5. Select Microsoft ADO Report from the *Type* drop list:

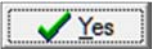


6. On the **Fields** tab, add the fields to be included in the report.
Fields selected in this example are:



7. Make changes on the other tabs as required.
See the EMu Help for details about setting a sort order, sort options, and security.
8. Click .
The new report is added to the Reports dialog box.
9. Select the new report and click  to run the report for the first time.
A message will display indicating that your report does not exist on the server. This is to be expected as the report has not yet been built in Excel:



10. Click .
An xml file is generated and saved with the data from your record set. The location of this file can vary, but typically it can be found in:
C:\Users\[your username]\AppData\Local\KESoftware\Reports\e[module

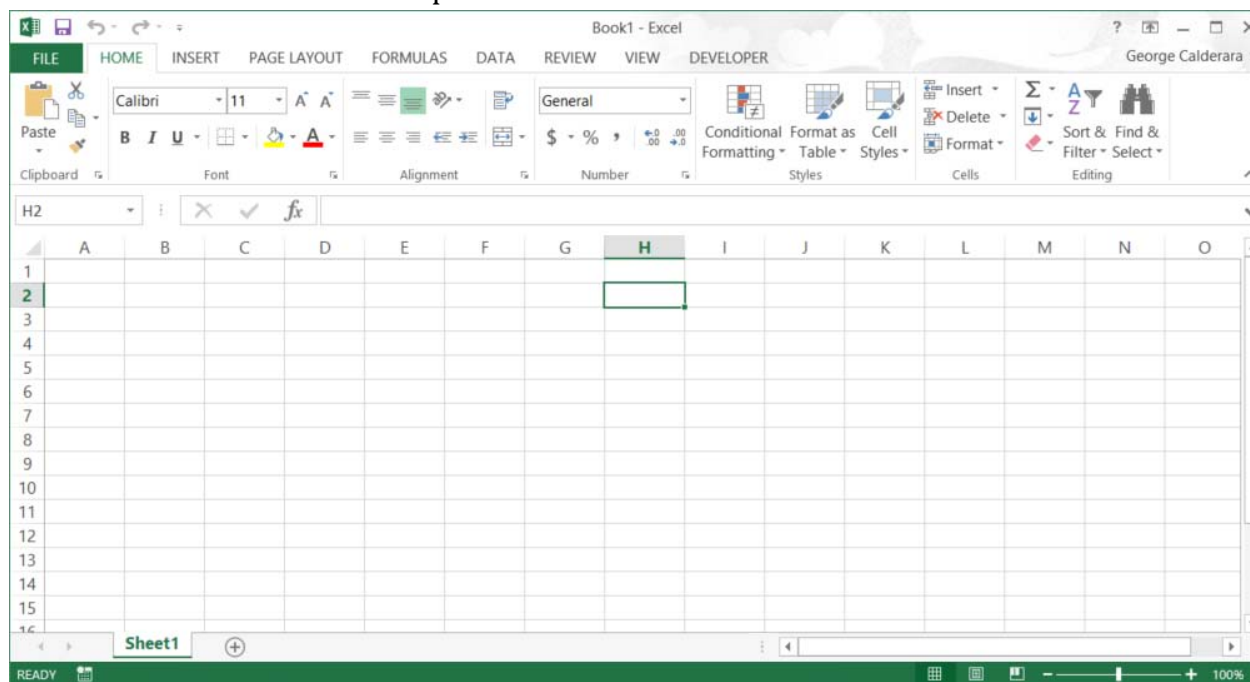
name]

For example, a report run in the Parties module, will save the xmldata file to:

C:\Users\[*your*

username]\AppData\Local\KESoftware\Reports\eparties

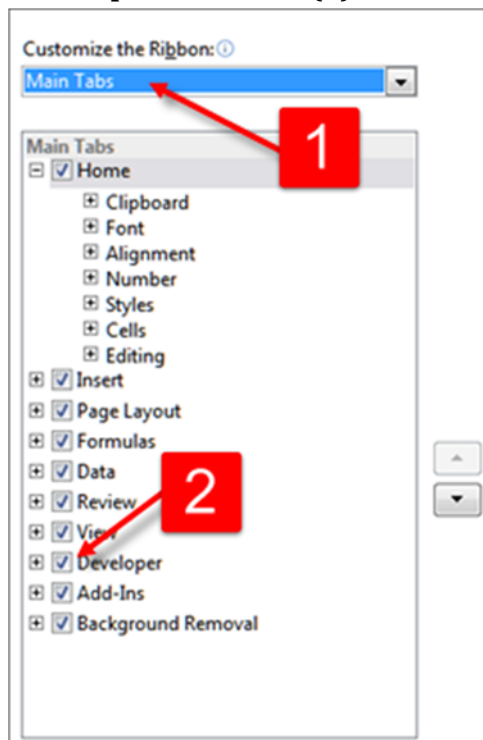
Microsoft Excel will open with a blank worksheet as follows:




Ensure that Excel is setup correctly

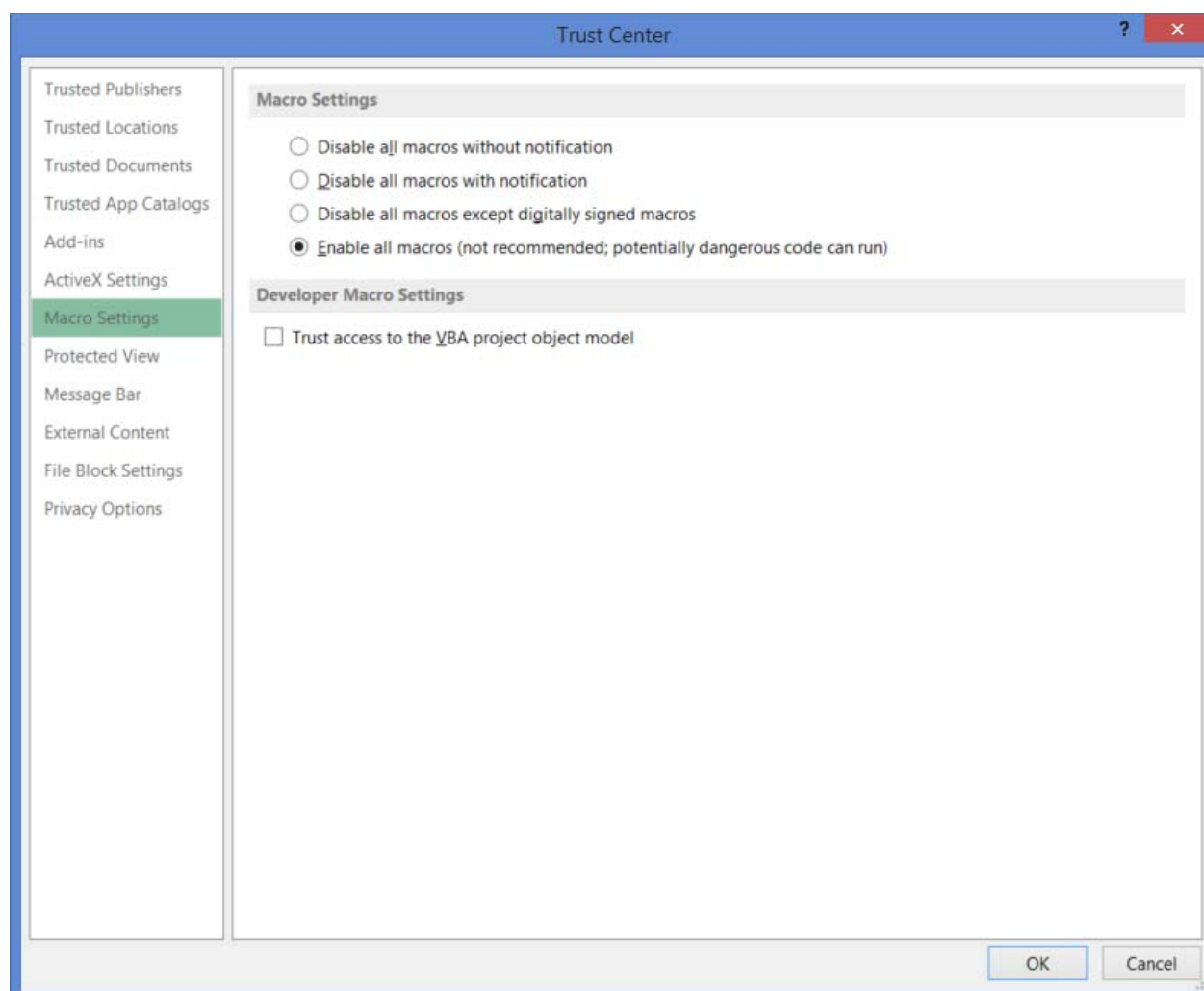
If the Developer tab does not display in the Ribbon:

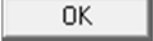
1. Click **File>Options>Customize Ribbon**.
2. With **Main Tabs** selected from the *Customize the Ribbon* drop list (1), select the **Developer** check box (2):




In order to run the macros that we will create with our reports, we need to ensure that the Security level in Excel is appropriate:

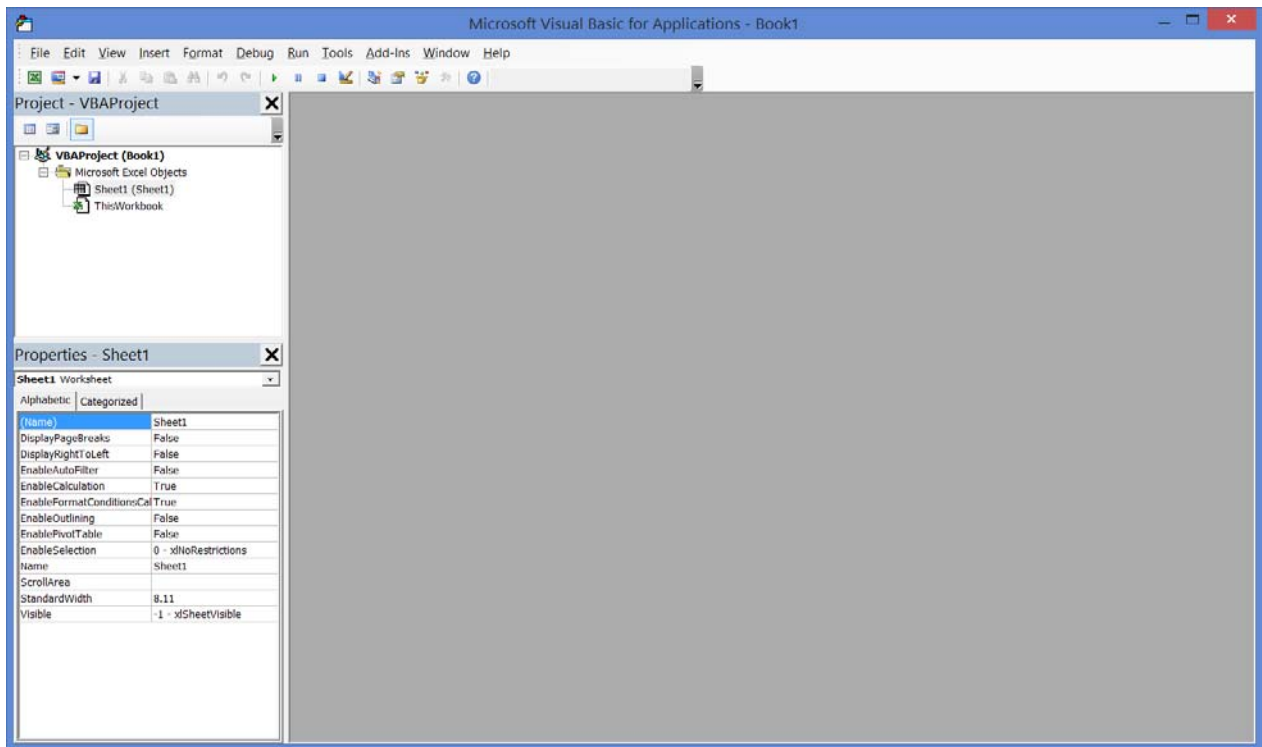
1. On the Developer tab, click  Macro Security
2. Enable all macros:



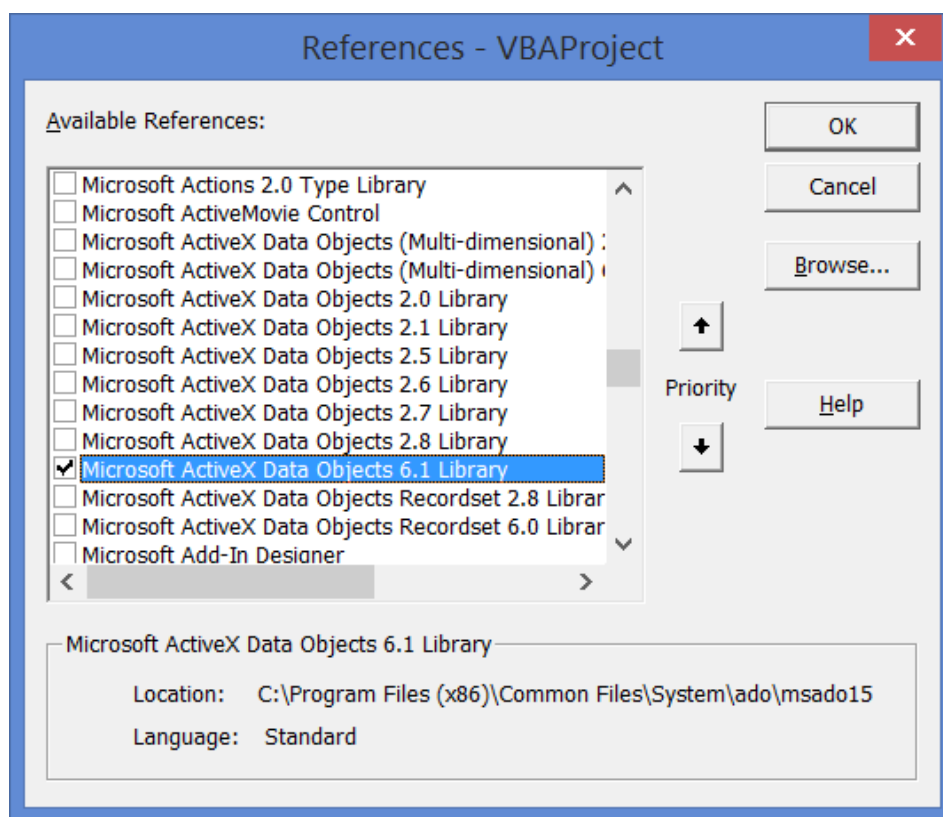
3. Click  to close the Trust Center.



4. On the Developer tab, click .
The following screen displays:



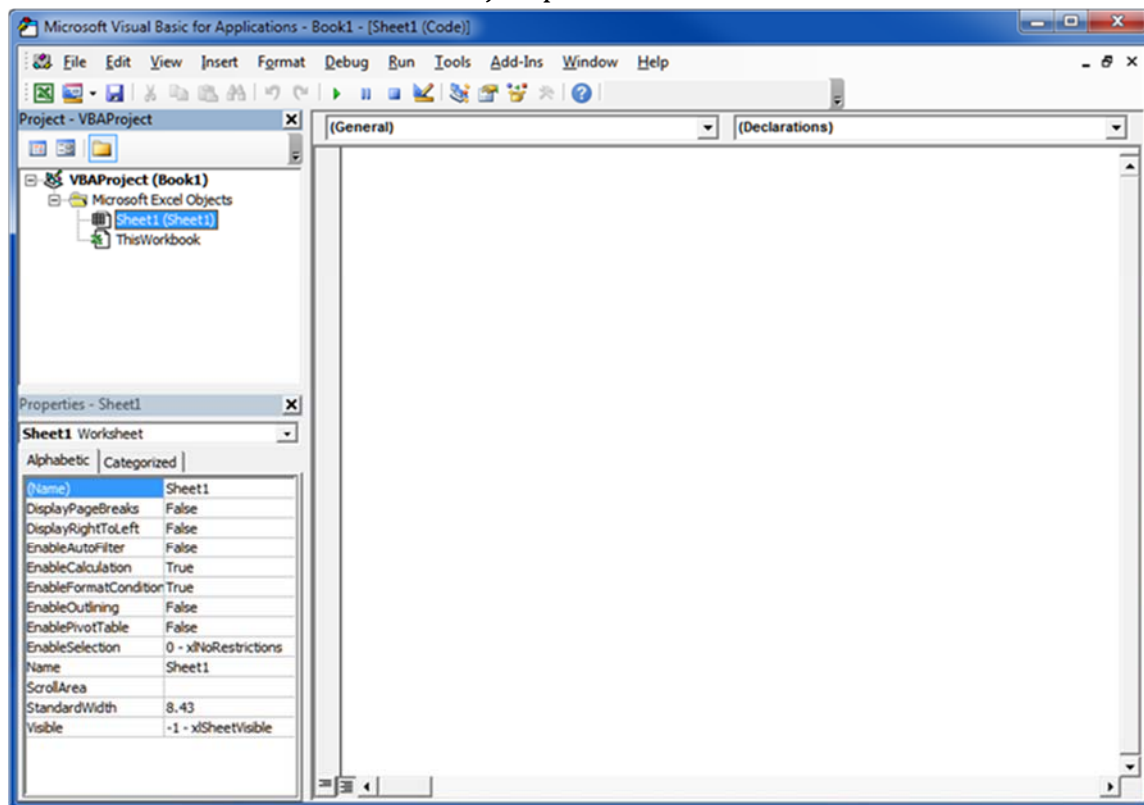
5. Ensure that the Microsoft ActiveX Data Objects Library is available:
 - 5.1. Select **Tools>References** in the Menu bar
In the References – VBAPROJECT dialog that displays, make sure that the following checkbox is checked:



5.2. Click .

Step 2: Design the report in Excel

1. Double-click **Sheet1** in the VBAProject pane:



2. Copy and paste the following VB code:

```
Sub OpenAdoFile()
    Dim RecordSet As ADODB.RecordSet
    Dim Worksheet As Excel.Worksheet
    Dim h As Long
    Dim col As Long
    Dim datarow As Long
    Dim source As String

    ' Get the persisted record set
    source = Environ("LocalAppData") & "\KESoftware\
Reports\ecatalogue\xml\data.xml"
    Set RecordSet = New ADODB.RecordSet
    RecordSet.Open source, "Provider=MSPersist"

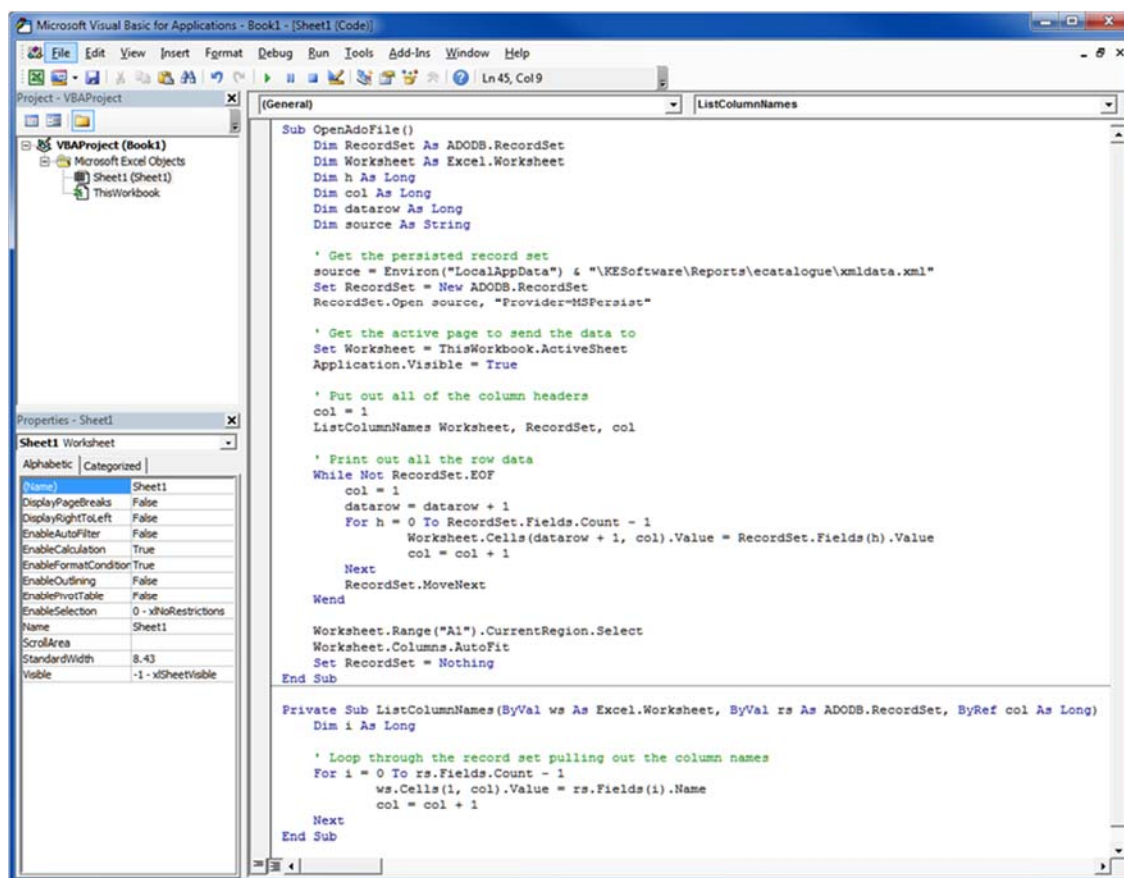
    ' Get the active page to send the data to
    Set Worksheet = ThisWorkbook.ActiveSheet
    Application.Visible = True
```

```
' Put out all of the column headers
col = 1
ListColumnNames Worksheet, RecordSet, col

' Print out all the row data
While Not RecordSet.EOF
    col = 1
    datarow = datarow + 1
    For h = 0 To RecordSet.Fields.count - 1
        Worksheet.Cells(datarow + 1, col).Value =
RecordSet.Fields(h).Value
        col = col + 1
    Next
    RecordSet.MoveNext
Wend

Worksheet.Range("A1").CurrentRegion.Select
Worksheet.Columns.AutoFit
Set RecordSet = Nothing
End Sub

Private Sub ListColumnNames(ByVal ws As Excel.Worksheet, ByVal rs
As ADODB.RecordSet, ByRef col As Long)
    Dim i As Long
    ' Loop through the record set pulling out the column names
    For i = 0 To rs.Fields.count - 1
        ws.Cells(1, col).Value = rs.Fields(i).Name
        col = col + 1
    Next
End Sub
```

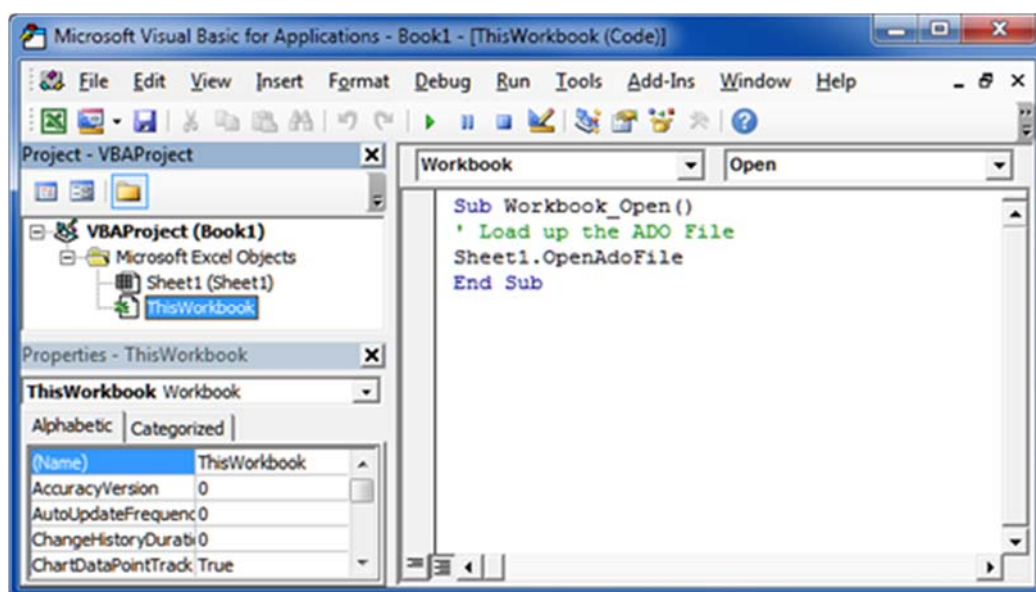


3. Double-click **ThisWorkbook** in the VBAProject pane and copy and paste the following code:

```

Sub Workbook_Open()
    ' Load up the ADO File
    Sheet1.OpenAdoFile
End Sub

```



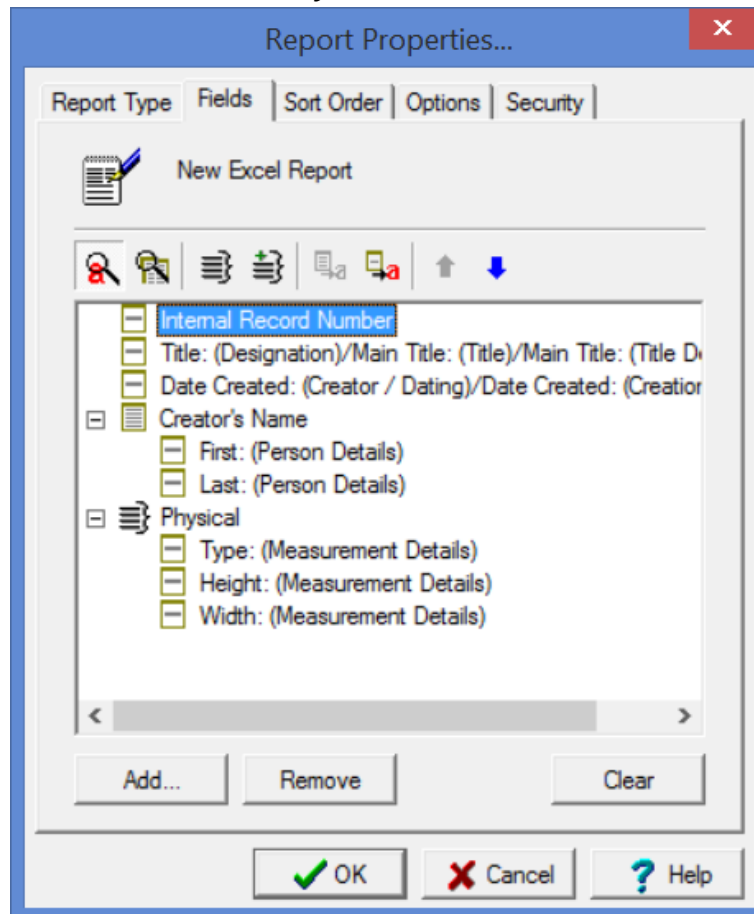
4. Save the report and upload it to your EMu report (page 24) on the Report Type tab of the Report Properties box.

When the report is run in EMu, an Excel report is generated:

Object Type	Category	Status	Title	Date Created
Building Structure	Accessioned	Old Parliament House, Canberra, Australia	1927	"Old Parliament House, Canberra, Australia"
Building Structure	Accessioned	Exhibitions - Old Parliament House, Canberra		"Exhibitions - Old Parliament House, Canberra"
Building Structure	Accessioned	King's Hall - Old Parliament House, Canberra		"King's Hall - Old Parliament House, Canberra"
Building Structure	Accessioned	The Cabinet Room - Old Parliament House, Canberra		"The Cabinet Room - Old Parliament House, Canberra"
Building Structure	Accessioned	The House of Representatives - Old Parliament House, Canberra		"The House of Representatives - Old Parliament House, Canberra"
Building Structure	Accessioned	The Parliamentary Library - Old Parliament House, Canberra		"The Parliamentary Library - Old Parliament House, Canberra"
Building Structure	Accessioned	The Prime Minister's Office - Old Parliament House, Canberra		"The Prime Minister's Office - Old Parliament House, Canberra"
Building Structure	Accessioned	The Senate Chamber - Old Parliament House, Canberra		"The Senate Chamber - Old Parliament House, Canberra"
Musical Instrument	Accessioned	Cello 'Marquis de Corberon' by Antonio Stradivari, Cremona	1726	"Cello 'Marquis de Corberon' by Antonio Stradivari, Cremona"
Musical Instrument	Accessioned	Harp-lute by Edward Light, with two French lyre-guitars, early 19th century		"Harp-lute by Edward Light, with two French lyre-guitars, early 19th century"
Musical Instrument	Accessioned	Viola 'Archinto' by Antonio Stradivari, Cremona	1696	"Viola 'Archinto' by Antonio Stradivari, Cremona"
Technology	Accessioned	A set of standard grain weights with gilt brass and platinum		"A set of standard grain weights with gilt brass and platinum"

How to create an Excel Report with nested tables using the ADO RecordSet

1. Repeat Step1: Create a new report in EMu (page 24).
For this example, the following fields were selected. Note the two nested tables - *Creator's Name* and *Physical*:



2. In Excel, click on the Developers tab.
3. Double-click **Sheet1** in the VBAProject pane:
4. Copy and paste the following VB code:


```
Sub Read_XML_Data()  
  
    Dim rst As ADODB.Recordset  
    Dim Worksheet As Excel.Worksheet  
    Dim i As Long  
    Dim j As Long  
    Dim source As String  
    Dim datarow As Long  
    Dim saverow As Long  
    Dim lastrow As Long  
    Dim col As Long  
  
    ' These next declaration is a little odd. Its needed in cases  
    where the entire value  
    ' of a nested table is blank. In these cases it is necessary  
    to force a number of columns to be skipped when printing  
    ' out field values. Oddly, as long as a nested table has at  
    least one value, then there is no issue.  
    ' There is only a need to declare one variable for each nested  
    table.  
    ' In this example there are only two nested tables so two  
    declarations are needed  
    ' The value assigned to each variable will depend on the  
    number of fields in that nested table.  
    ' In this example the first nested table is the  
    CreCreatorRef_tab, which has two fields, i.e. NamFirst and  
    NamLast  
    ' and the second nested table, i.e Physical, has 3 fields,  
    i.e. PhyType, PhyHeight and PhyWidth  
  
    Dim firstnestedtable As Long  
    Dim secondnestedtable As Long  
    Dim nestedtablecount As Long  
  
    firstnestedtable = 2  
    secondnestedtable = 3  
    nestedtablecount = 1  
  
    ' Get the persisted record set  
    source = Environ("LocalAppData") &  
    "&KESoftware\Reports\ecatalogue\xml\data.xml"  
    Set rst = New ADODB.Recordset  
    rst.Open source, "Provider=MSPersist"
```

```

' Get the active page to send the data to
Set Worksheet = ThisWorkbook.ActiveSheet
Application.Visible = True

'Add column labels
Worksheet.Cells(1, 1).Select
ActiveCell.EntireRow.Insert
Worksheet.Cells(1, 1).Value = "Record No"
Worksheet.Cells(1, 2).Value = "IRN No"
Worksheet.Cells(1, 3).Value = "Title"
Worksheet.Cells(1, 4).Value = "Date Created"
Worksheet.Cells(1, 5).Value = "Creator First"
Worksheet.Cells(1, 6).Value = "Creator Last"
Worksheet.Cells(1, 7).Value = "Physical Type"
Worksheet.Cells(1, 8).Value = "Physical Length"
Worksheet.Cells(1, 9).Value = "Physical Width"

col = 1
' Start printing data from Row 3
datarow = 3
lastrow = datarow
While Not rst.EOF
    col = 1

    If datarow < lastrow Then
        datarow = lastrow
    End If

    For j = 0 To rst.Fields.Count - 1
        If rst.Fields(j).Type = adChapter Then
            If rst.Fields(j).Value.BOF Then
                Worksheet.Cells(datarow, col).Value = ""
                If nestedtablecount = 1 Then
                    col = col + firstnestedtable
                    nestedtablecount = nestedtablecount + 1
                ElseIf nestedtablecount = 2 Then
                    col = col + secondnestedtable
                    nestedtablecount = nestedtablecount + 1
                End If
            Else
                If rst.Fields(j).Value.EOF Then
                    Worksheet.Cells(datarow, col).Value = ""
                    If nestedtablecount = 1 Then

```

```
col = col + firstnestedtable
nestedtablecount = nestedtablecount
+ 1
ElseIf nestedtablecount = 2 Then
col = col + secondnestedtable
nestedtablecount = nestedtablecount
+ 1
End If
Else
saverow = datarow
ListNestedValues Worksheet,
rst.Fields(j).Value, col, datarow, lastrow, saverow,
nestedtablecount
End If
End If
Else
If IsNull(rst.Fields(j).Value) Then
Worksheet.Cells(datarow, col).Value = ""
Else
Worksheet.Cells(datarow, col).Value =
rst.Fields(j).Value
End If
col = col + 1
End If
Next
rst.MoveNext
datarow = datarow + 1
nestedtablecount = 1
Wend

'Closing the recordset.
rst.Close

'Release object from memory.

Worksheet.Range("A1").CurrentRegion.Select
Worksheet.Columns.AutoFit
Set rst = Nothing

End Sub

Private Sub ListNestedValues(ByVal ws As Excel.Worksheet, ByVal
rs As ADODB.Recordset, ByRef col As Long, ByRef datarow As Long,
ByRef lastrow As Long, ByRef saverow As Long, ByRef
```

```

nestedtablecount As Long)
    Dim i As Long
    Dim j As Long
    Dim startrow As Long

    ' Loop through a nested table pulling out the row values
    j = 0
    startrow = saverow
    While Not rs.EOF
        max = 1
        j = col
        For i = 0 To rs.Fields.Count - 1
            ' Don't print key values
            If rs.Fields(i).Name <> "ecatalogue_key" And
rs.Fields(i).Name <> "CreCreatorRef_key" And rs.Fields(i).Name <>
"Physical_key" _
            Then
                If IsNull(rs.Fields(i).Value) Then
                    ws.Cells(startrow + 1, j).Value = ""
                    j = j + 1
                Else
                    If rs.Fields(i).Type = adChapter Then
                        ListNestedValues ws, rs.Fields(i).Value,
j, datarow, lastrow, saverow, nestedtablecount
                        datarow = startrow
                    Else
                        ws.Cells(startrow, j).Value =
rs.Fields(i).Value
                        j = j + 1
                    End If
                End If
            End If
        Next
        rs.MoveNext
        startrow = startrow + 1
    Wend

    If (j > 0) Then
        col = j
    End If

    If startrow > lastrow Then
        lastrow = startrow
    End If

```

```
nestedtablecount = nestedtablecount + 1
```

```
End Sub
```

- Double-click **ThisWorkbook** in the VBAProject pane and copy and paste the following code:

```
Sub Workbook_Open()
```

```
' Load up the ADO File
```

```
Sheet1.Read_XML_Data
```

```
End Sub
```

- Save the report and upload it to your EMu report (page 24) on the Report Type tab of the Report Properties box.

When the report is run in EMu, an Excel report is generated:

Record No	IRN No	Title	Date Created	Creator First	Creator Last	Physical Type	Physical Length	Physical Width
1	1000133	Gladioli gown worn by Dame Edna Everage in Tears Before Bedtime, Australian tour, 1985 and	1985					
2	1000127	Arrungu Dreaming at Ulyitjirki, 1984	1983					
3	1000134	Gold hotpants worn by Kylie Minogue - "Spinning Around" video from the album Light Years, 2000	2000					
4	1000128	Bizet's Carmen in the Bullring, 1985	1985	John	Olsen			
6	1000080	Stained glass window from Glenferrie house, Malvern	1872					
7	1000057	Limpet - underside						
8	14	A Young Gentleman (or A Portrait of James Wolfe, Later General Wolfe)	c1760-65	Thomas	Gainsborough	Canvas	76.5	63.5
9	15	John Sidney, 6th Earl of Leicester	1728	Joseph	Highmore	Canvas	76.2	63.5
10	1000061	Riftia Plume						
11	1000053	Cirrate Octopus						
12	113456	Gussey Galah puppet	1967-73	Axel	Axelrad			135
13	107939	The Maestro's Company	1984				440	
14	58	Painting Two by Gerard	17/02/2011	Gerard	Wood			
15	57	Painting One by Gerard	17/02/2011	Gerard	Wood			
16	27	40 A View of St. Peter's Place and Manner in which the Manchester Reform Meeting was disposed	1819					
17	52000	Artworkers calendar, 1984: August	1984	Colin	Russell			
18	41	The British Butcher Supplying John Bull with a substitute for Bread	1795					
19	37	1000194 Boronia pinnata						
20	38	1000148 Women's open robe	1760					
21	39	103 Painting Two by Train6	29/05/2013	Train	Six			
22	40	102 Painting One by Train6	29/05/2013	Train	Six			
23	41	1000054 Frullania pycnantha epiphyll		Wilhelm	Focke	Frame	1	2
24				Larry	Foster			
25	42	100766 Painting of Kenneth Laird	1963	John	Kandor	Frame	300	600
26				Wilhelm	Focke			
27				Jochen	Heinrichs			
28				Gregorio	Dauphin			
29				Christopher	Schlüter			
30	43	103000 Iffley Mill, Oxford	1917	Sydney	Long	plate-mark Frame	15.8	100

SECTION 4

Registry entries

The Type Registry entry indicates which export type to use for each report request.
The format of this Registry entry is ;

System|Setting|Reports|Type|Crystal CSV|*value*

value is 0 or 1:

- 0 Generates data in the existing format.
- 1 Generates data in the new Crystal ODBC format.



If this entry is not present, a *value* of 0 is assumed.

System|Setting|Reports|Type|Crystal ADO|*value*

value is 0 or 2:

- 0 Generates data in the existing format.
- 2 Generates data in the new Crystal ADO record set.



If this entry is not present, a *value* of 0 is assumed.

System|Setting|Reports|Type|Microsoft ADO|*value*

where:

value is 0 or 3:

- 0 Generates data in the existing format.
- 3 Generates data in the new Microsoft ADO format.



If this entry is not present, a *value* of 0 is assumed.

Index

A

ADO Reports • 1

C

Crystal Reports • 3

E

Ensure that Excel is setup correctly • 27

H

How to create a Crystal ADO Report • 3

How to create an Excel Report using the ADO RecordSet •
23

How to create an Excel Report with nested tables using the
ADO RecordSet • 35

How to modify a Crystal Report to use ADO instead of
ODBC • 12

M

Microsoft Excel • 23

N

Note • 1

R

Registry entries • 41

S

Step 1

Create a new report in EMu • 24, 34, 35, 40

Step 2

Design the report in Excel • 31



EMu Documentation

Unicode in EMu 5.0

Document Version 1

EMu 5.0

EMu
Museum
Management
System



KE Software

An AXIELL Group Company
emu.axiell.com

© 2015 All rights reserved

Contents

SECTION 1	Unicode	1
	Overview	1
	Code Points	3
	Inputting Unicode Characters	6
	Graphemes	10
	Index Terms	11
SECTION 2	Searching	15
	Transformations	17
	Regular Expressions	18
	Anchors	19
	Proximity	20
	Conditionals	22
SECTION 3	Auto-phrasing	23
SECTION 4	Collation	25
SECTION 5	Lookup Lists	27
	Index	29

SECTION 1

Unicode

Overview

EMu 5.0 sees implementation of support for the Unicode 8.0 (<http://www.unicode.org/versions/Unicode8.0.0/>) standard. While earlier versions of EMu allowed Unicode characters to be stored and retrieved, the system did not interpret the characters entered, leading to very limited searching functionality. In order to retrieve a Unicode character it was necessary to enter the search term in exactly the same case (upper or lower) along with the same diacritics. For example, a search for the name `Frederic` would not match `Frédéric` as the `e` acute character was not interpreted as an `e` character with a diacritic associated with it.

EMu 5.0 supports case folding and base character mapping:

- Case folding is similar to converting a character to its lower case equivalent except that it handles some special cases. The purpose of case folding is to make searching case insensitive. One special case is that the German lower case sharp `s` character (`ß`) is generally written in upper case as `SS`. So `Großen` would be converted to `GROSSEN` in upper case. When searching we would like to enter either of the previous terms and find all case variations. In order to do this the `ß` character needs to be folded to `ss` for searching purposes.
- The base version of a character is its most basic representation after all diacritics and marks have been removed. For example the base character of `é` is `e`.

The combination of case folding and base characters provides the basic mechanisms required to provide flexible searching over the full range of Unicode characters.

All data stored in EMu 5.0 is encoded in UTF-8 format. UTF-8 is a compact way of representing Unicode characters, particularly ASCII characters. The World Wide Web has adopted UTF-8 as the character encoding format to be used in web documents. EMu 5.0 enforces the use of UTF-8 by not allowing any invalid byte sequences to be stored in the system. The change has implications for data imports as all data imported **must** be encoded in UTF-8. In earlier versions of EMu, systems may have been configured to allow ISO-8859-1 (latin1) as the standard input format. ISO-8859-1 encoding is no longer supported.

Searching in EMu 5.0 has been extended to include punctuation characters. It is now possible to search for punctuation either as individual characters (`?`) or as part of a more complex string (`fred@global.com`). In EMu 4.3 and earlier certain punctuation characters have a special meaning when used in a search. For example a search for `fre*`

will find all words beginning with the letters `fre`. The introduction of punctuation searching in EMu 5.0 means that these *special* characters need to be "escaped" to have their special meaning applied. Escaping a character involves preceding the character with a backslash (`\`). Thus, an EMu 4.3 search for `fre*` becomes `fre*` in EMu 5.0.

In the following sections we explore what changes have been implemented and how they impact usage of EMu 5.0.

Code Points

The basic unit of information in Unicode is known as a code point. A code point is simply a number between zero and $10FFFF_{16}$ that represents a single entity. Code points are generally represented as hexadecimal numbers, that is base 16. An entity may be a:

Entity	Description
<i>graphic</i>	A letter, mark, number, punctuation, symbol or space, e.g. the letter a.
<i>format</i>	Controls the formatting of text, e.g. soft hyphen (–) for breaking a word over lines.
<i>control</i>	A control character, e.g. the tab character (^I).
<i>private-use</i>	Not defined in the Unicode 8.0 standard but used by other non-Unicode scripts, e.g. unused cp 1252 character, 91_{16} .
<i>surrogate</i>	Used to select supplementary planes in UTF-16. Characters in the range D800-DFFF ₁₆ .
<i>non-character</i>	Permanently reserved for internal use. Characters in the range FFFE-FFFF ₁₆ and FDD0-FDEF ₁₆ .
<i>reserved</i>	All unassigned code points, that is code points that are not one of the above.

The table below lists some code points along with their representation, label and category:

Code point (hex)	Representation	Label	Category
E9	é	Latin small letter e with acute	graphic (letter - lower case)
600	﹁	Arabic number sign	format (other)
D6A1	호	Hangul syllable hoeng	graphic (letter - other)
B4	’	Acute accent	graphic (symbol - modifier)
F900	𠀎	Chinese, Japanese, Korean (cjk) compatibility ideograph	graphic (letter - other)

A piece of text is logically just a sequence of code points, where each code point represents a part of the text. For example, the piece of text:

豈 ↔ how?

consists of the following code points:

Code point (hex)	Representation	Label
F900	豈	Chinese, Japanese, Korean (cjk) compatibility ideograph
20		Space
2194	↔	Left right arrow
20		Space
68	h	Latin small letter h
6F	o	Latin small letter o
77	w	Latin small letter w
3F	?	Question mark

The code point sequence defines the text itself. There are a number of different ways that the code point sequence can be saved on a computer. One method, called UTF-32, represents each code point as a 32 bit (4 byte) quantity. Such a scheme uses a large amount of storage space as most text uses the Latin alphabet (ASCII), which can be represented in a single byte.

Another encoding is UTF-8. This allows ASCII characters to be stored as a single byte (code points 00-7F), with multiple bytes used for higher code points. UTF-8 is very efficient space wise where the text consists of mainly ASCII characters, and the World Wide Web has adopted it as the preferred encoding method for Unicode code points. EMu 5.0 also uses UTF-8 as the encoding method. Below, we show a string encoded in UTF-32 with a space between each code point:

豈 ↔ how?

```
0000F900 00000020 00002194 00000020 00000068 0000006F 00000077
0000003F
```

And the same string encoded in UTF-8:

```
EFA480 20 E28694 20 68 6F 77 3F
```

As you can see the UTF-8 encoding saves considerable space.

Prior to EMu 5.0 either UTF-8 or ISO-8859-1 could be configured as the encoding used by EMu. EMu 5.0 drops support for ISO-8859-1 and only supports UTF-8 encoded characters. The change means that moving to EMu 5.0 requires all data to be converted from ISO-8859-1 to UTF-8 before the system may be used. The upgrade process performs this important function.



EMu 5.0 will not allow non UTF-8 sequences to be input. If an illegal character is encountered, an error message is displayed. The enforcement of UTF-8 encoding means that all data entered into EMu, either by direct entry or by importing, must be in UTF-8 format. Data encoded in ISO-8859-1 cannot be loaded. If you receive import data from a third party source, ensure that it is in UTF-8 format otherwise errors will be generated for all non-ASCII characters. An ISO-8859-1 encoded data file can be converted to UTF-8 using the UNIX `iconv` utility.

Inputting Unicode Characters

Now that we understand that text is made up of a sequence of Unicode code points it is worth considering how these characters can be entered into EMu.

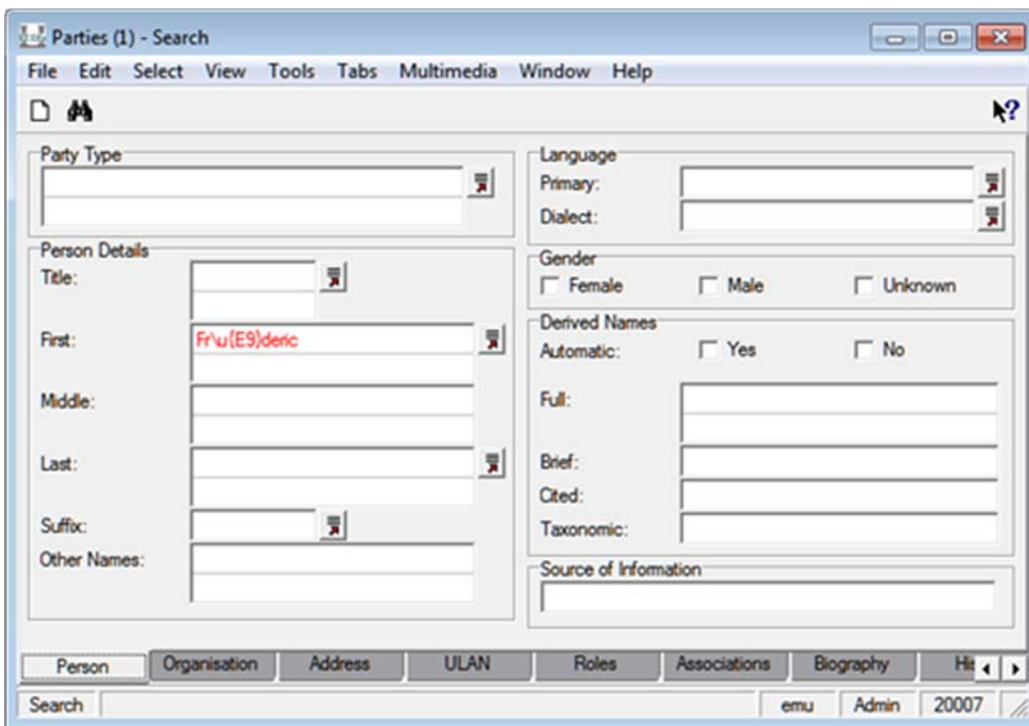
EMu supports two mechanisms:

- Escaped code point
- Raw characters

Escaped code point

The escaped code point mechanism allows an escape sequence to be placed in a text string to represent a Unicode code point. When the string is sent to the EMu server, the escape sequence is converted into a Unicode code point encoded in UTF-8.

For example, if the text `Fr\u{E9}deric` was input while creating or modifying a record, the data saved would be `Frédéric`. The format of the escape sequence is `\u{x}` where `x` is the code point in hexadecimal of the Unicode character required. The escape sequence may also be used when entering search terms:



The escape sequence may also be used in `texql` statements whenever a string constant is required. For example, the query statement:

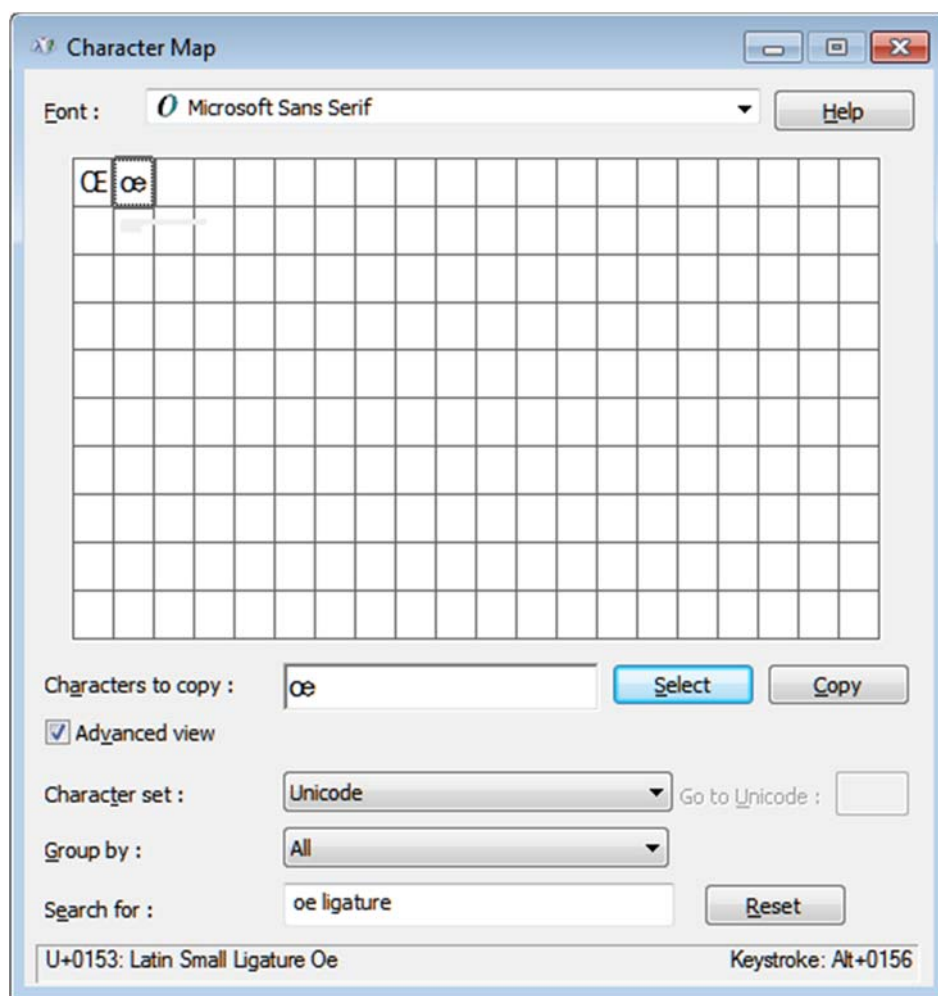
```
select NamFirst
from eparties
where NamFirst contains 'Fr\u{E9}deric'
```

will find all Parties records where the First Name is `Frédéric` (and variations where diacritics are ignored). The escape sequence format may also be used for data imported into EMu via the Import facility.

Raw characters

The raw character method involves pasting Unicode characters into the required EMu field. There are a number of ways of adding Unicode characters to the Windows clipboard. One way is to use the Windows Character Map application. This can be found on a Window PC by selecting search on the Windows Start menu (or pressing the Windows Logo key and the letter `s` at the same time) and searching for `charmap`.

The Windows Character Map application allows you to select a character and copy it to the clipboard. By selecting **Advance view**, it is possible to search for a character by name. For example to find the `oe` ligature character (`œ`), enter `oe ligature` in the *Search for:* field and press **Search**. A grid of all matching characters is displayed:



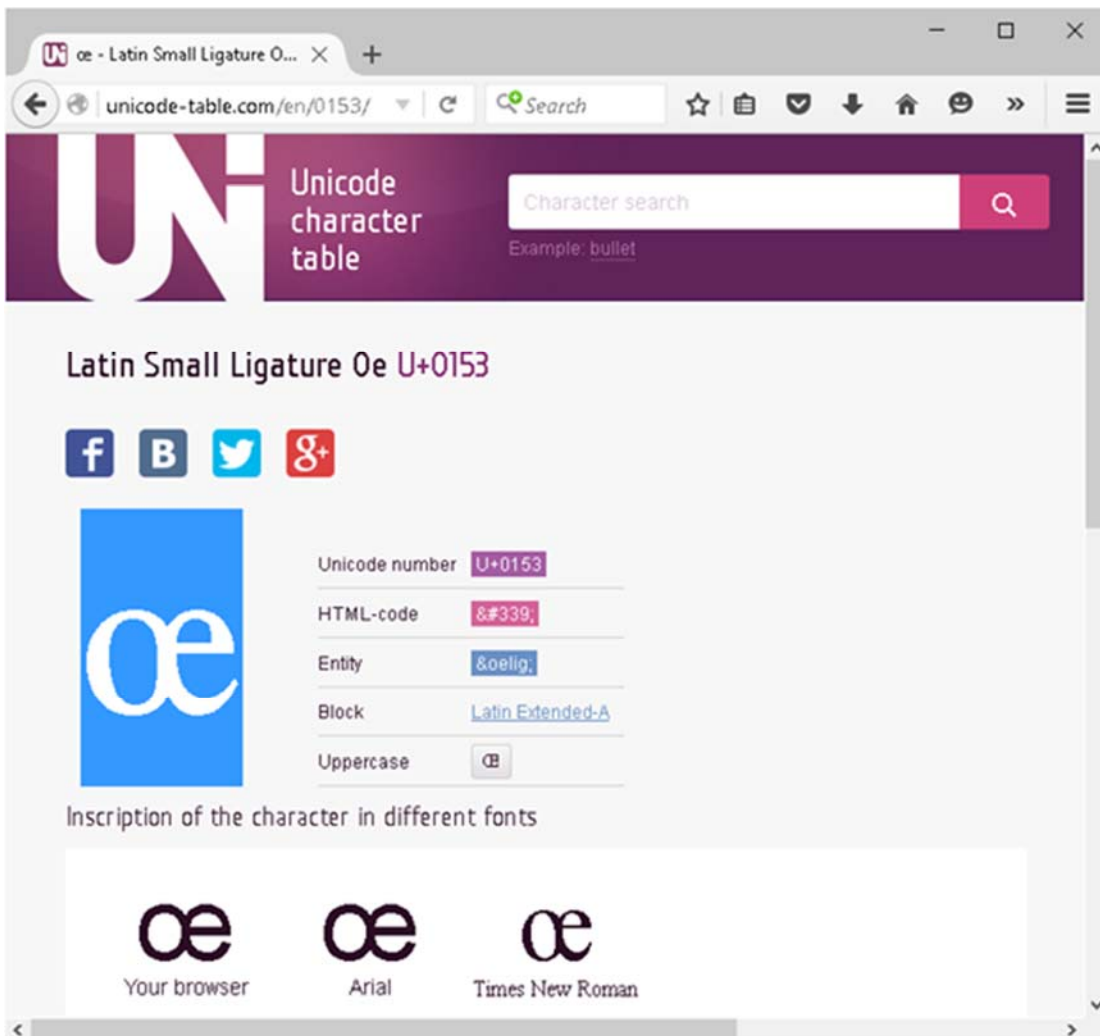
Double-click the required character, then press **Copy** to place it on the Windows clipboard. The character can then be pasted into EMu.

Alternative methods

Another way to add a Unicode character to the Windows clipboard is to use a website that allows Unicode characters to be searched for and displayed. Two useful sites are:

- graphemica.com (<http://graphemica.com/>)
- unicode-table.com (<http://unicode-table.com/>)

With both of these sites it is possible to search for a character by name or code point (in hex), e.g.:



Highlight the character on the page and copy it to the clipboard. The character can then be pasted into the required EMu field.

Both of these websites display the code point for the character. In the picture above, the code point for œ is hex 153. If you wanted to use the escaped code point method, the escape sequence to use would be:

```
\u{153}
```

If you need to enter certain Unicode characters on a regular basis, you could create a WordPad (or Word) document that contains the characters. When you need a character, simply copy the character from the document and paste it into EMu, without the need to search for the character.

Graphemes

It is important to understand that what we think of as a character, that is a basic unit of writing, may not be represented by a single Unicode code point. Instead, that basic unit may be made up of multiple Unicode code points.

For example, "g" + *acute accent* (ǵ) is a *user-perceived character* as we think of it as a single character, however it is represented by two Unicode code points (67 301). A *user-perceived character*, which consists of one or more code points, is known as a grapheme. The use of graphemes is important for:

- collation (sorting);
- regular expressions;
- indexing; and
- counting character positions within text.

EMu 5.0 uses graphemes as the basic building block for text. Thus a text string is handled as a sequence of graphemes.

A grapheme consists of one or more base code points followed by zero or more zero width code points and zero or more non-spacing mark code points. In the case of "g" + *acute accent* (ǵ), the letter ǵ is the base code point (67) and the *acute accent* is a non-spacing mark code point (301). The table below shows some multiple code point graphemes:

Grapheme	Code points
각	1100 (ㄱ) Hangul choseong kiyeok (base code point) 1161 (ㅏ) Hangul jungseong a (base code point) 11A8 (ㅑ) Hangul jongseong kiyeok (base code point)
ǵ	64 (d) Latin small letter d (base code point) 325 (◌̣) combining ring below (non-spacing mark) 301 (´) combining acute accent (non-spacing mark)
á	61 (a) Latin small letter a (base code point) 301 (´) combining acute accent (non-spacing mark)

Some common multiple code point graphemes have been combined into a single code point. For example, the last entry in the table above, á, can also be represented by the single code point [E1](#). Hence we have two representations, or two graphemes, that represent the same character (á is this case).

Index Terms

An index term is the basic unit for searching. It is a sequence of one or more graphemes that can be found in a search but for which searching of sub-parts is not supported (except if regular expressions are used). EMu provides word based searching, so an index term corresponds to a word. You can search for a word, and records that contain that word will be matched. In languages that define a word as a sequence of letters separated by either spaces or punctuation, an index term corresponds to a word. In languages in which single (or sometimes multiple) letters make up a word, such as kanji, an index term corresponds to each individual letter. EMu 5.0 adds support for searching for punctuation, so each punctuation character is considered to be an index term.

Consider the following text:

香港 is Chinese for "Hong Kong" (香 = fragrant, 港 = harbour).

The index terms for the above text are:

Index Term

香

港

is

Chinese

for

"

Hong

Kong

"

(

香

=

fragrant

,

港

=

harbour

)

.

Each of the above terms can be used in a search and the query will be able to use the high speed indexes to locate the matching records. It is possible to use regular expression characters (e.g. `fra*` to find all words beginning with `fra`) to search for sub-parts of words, however the high speed indexes will not be used in this case (unless partial indexing is enabled).

Each index term is folded and converted to its base form. The folding process, as described in the overview section (page 1), removes case significance from the term. The conversion to its base form involves removing all "mark" code points from the term and then converting the remaining code points to their compatible form (as defined by the Unicode 8.0 standard). The compatible form for a code point is a mapping from the current code point to a base character that has the same meaning. For example the code point for subscript 5 (₅) has a compatible code point of 5.

The table below shows some more examples for compatibility:

Type	Compatibility Examples		
Font variants	ℋ	→	H
	ℍ	→	H
Positional variants	ℰ	→	ℰ
	℄	→	ℰ
	℅	→	ℰ
	℆	→	ℰ
Circled variants	①	→	1
Width variants	㍑	→	力
Rotated variants	↯	→	{
	↰	→	}
Superscripts / subscripts	i ⁹	→	i9
	i ₉	→	i9

Unfortunately, some of the compatibility mappings in the Unicode 8.0 standard are narrower than we might expect when searching text. For example the oe ligature (œ) does not map to the characters "oe". So the French word cœur ("heart") does not have an index term of `coeur`, but remains as `cœur`. When searching you need to enter `cœur` as the search term otherwise `cœur` will not be found.

In order to *correct* some of the compatibility mappings, EMu 5.0 provides a mapping file where a code point can be mapped to its compatible code point(s), hence "œ" can be mapped to "oe". The mapping file is located in the Texpress installation directory in the `etc/unicode/base.map` file.

A sample file is (as distributed currently):

```
#
# The following file is used to extend the Unicode NFKD mappings for
# characters not specified in the standard. The format of the file is
# a sequence of numbers as hex. Each number represents a single code
# point in UTF-32 format. The first code point is the code point to
# map
# and the second and subsequent code points are what it maps to.
#
00C6 0041 0045      # Latin capital letter AE -> A E
00E6 0061 0065      # Latin small letter ae -> a e
00D0 0044           # Latin capital letter Eth -> D
00F0 0064           # Latin small letter eth -> d
00D8 004F           # Latin capital letter O with stroke -> O
00F8 006F           # Latin small letter o with stroke -> o
00DE 0054 0068      # Latin capital letter Thorn -> Th
00FE 0074 0068      # Latin small letter thorn -> th
0110 0044           # Latin capital letter D with stroke -> D
0111 0064           # Latin small letter d with stroke -> d
0126 0048           # Latin capital letter H with stroke -> H
0127 0068           # Latin small letter h with stroke -> h
0131 0069           # Latin small letter dotless i -> i
0138 006B           # Latin small letter kra -> k
0141 004C           # Latin capital letter L with stroke -> L
0142 006C           # Latin small letter l with stroke -> l
014A 004E           # Latin capital letter Eng -> N
014B 006E           # Latin small letter eng -> n
0152 004F 0045      # Latin capital ligature OE -> O E
0153 006F 0065      # Latin small ligature oe -> o e
0166 0054           # Latin capital letter T with stroke -> T
0167 0074           # Latin small letter t with stroke -> t
```

Compatible mappings may be added to the file as required.



If the file is modified, a complete reindex of the system is required in order for the new mappings to be used to calculate the index terms.

If you consider the French phrase:

Sacré-Cœur est situé à Paris.

the index terms after folding and conversion to base form are:

Index Term

sacre

coeur

est

situe

a

paris

.

When a record is saved in EMu all index terms are folded and converted to their base form before indexing occurs. Similarly, when a search is performed, the query terms are folded and converted to their base form before the search commences. Hence a search for "coeur" or "Cœur" or even "COEUR" will still match the text in the French phrase above.

SECTION 2

Searching

Now that we understand what an index term is we can talk about searching. The incorporation of Unicode into EMu has resulted in the searching mechanism being extended to handle all code points that have a base representation. In essence this is all *graphic* (page 3) code points except for marks and spaces, namely:

- letters
- numbers
- punctuation
- symbols

The inclusion of punctuation as an index term means that punctuation may now be included in searches and the high speed indexes will be used to locate matches.

An issue arises in EMu versions prior to 5.0 where certain punctuation characters were used to adjust the type of searching performed. For example, in EMu 4.3 a search for @John would find all records containing words that sound like John (phonetic searching). Similarly a search for ^joh* would match records where the first word starts with the letters joh (case ignored). A search for =John would locate records containing John with case significance (that is an upper case J and lower case ohn). Since EMu 4.3 and earlier removed punctuation and symbols from searching (only letters and numbers were supported) there was no ambiguity about the punctuation associated with search terms (as in the previous examples). As EMu 5.0 allows symbols and punctuation to be searched for, some ambiguity can creep in. For example, what does searching for fred@global.com mean? In EMu 4.3 it would have meant finding:

- "fred"
- AND the phonetic of "global"
- AND "com"

However, in EMu 5.0 is the @ character to be treated as punctuation or does it mean the phonetic of the word "global"?

When searching for a word prior to EMu 5.0 you simply entered the word and performed the search. We have taken the same approach in EMu 5.0 with punctuation characters. In other words, when you have punctuation in a search, only records containing the punctuation are matched. Thus, in the previous example the @ character is treated as punctuation and so must appear in matching records.

How then do we indicate that the @ character means we want the phonetic version of the following word? We proceed the character with a special marker indicating the character is to take on its phonetic meaning. The marker character used is the backslash (\) character. The introduction of a marker character to alter the meaning of a character is not new in EMu. For example, \n can be used in strings to represent the newline character; similarly \u{ } is used to introduce the escape sequence for a Unicode code point.

EMu 5.0 has a simple rule to determine how to format a search:

All *graphic* (page 3) characters, except for spaces and marks, in a search are matched as the character. Where the special meaning of a character (e.g. @) is required, the character must be preceded by the backslash (\) escape character. The only exception to this rule is that the backslash character itself must be entered twice (\\) where the actual character is required.

The table below compares some searches in EMu 4.3 and their equivalent in EMu 5.0:

Find	EMu 4.3	EMu 5.0
Records containing Fred	fred	fred
Records where Fred is the only word in the field	^fred\$	\\^fred\\\$
Records that contain Fred phonetically	@fred	\\@fred
Records containing Fred with matching case	=Fred	\\=Fred
Records containing the phrase Sacré-Cœur	"sacré cœur"	\\"sacre-coeur\\"
Records where blue and sky are within five index terms of each other	(blue sky) <= 5 words	\\(blue sky\\) <= 5 words

In the following sections we will look at all available special search operators and show examples of their use in EMu 5.0. Each of the operators is displayed with its leading escape character, the backslash character.

Transformations

Transformations are an operator that is applied to a search term to alter its interpretation. The table below lists all valid transformations:

Transformation	Description
\~	Search for all variations of a word. For example, searching for \~elect will match elect, election, electing and elected, but not electricity (its base word is electric)
\&	Ignore the case (upper or lower) of the search term. This is the default transformation if one is not specified explicitly.
\@	Use phonetic or sounds like searching for the specified word.
\=	Perform the search using case significance for the following word.
\==	Perform the search not only matching the case but also matching any marks (diacritics).

A transformation is always applied to a word and is placed immediately before the word to which it applies. Some examples are:

Find	Search
Records containing all tenses of the word locate.	\~locate
Records where melbourne is all in lower case.	\=melbourne
Records with Sacré and Cœur exactly as specified, that is matching case and diacritics, but not necessarily next to each other.	\==Sacré \==Cœur
Records containing words similar to smythe phonetically.	\@smythe

Regular Expressions

Regular expressions provide a mechanism for searching for patterns in a word. With regular expressions, sub-parts of a word may be matched. In general the high speed indexes cannot be used with regular expression searches. The only exception is trailing regular expressions (that is a regular expression that has leading letters), where partial indexing has been enabled.

Regular expressions can be intermixed with the `\=` and `\==` transformations to enforce case and diacritic significance.

The table below lists all valid regular expressions:

Regular Expression	Description
<code>\?</code>	Matches any single grapheme.
<code>*</code>	Matches zero or more graphemes.
<code>\[range\]</code>	Matches only one of a sequence of graphemes specified in <i>range</i> . <i>range</i> may consist of individual graphemes or a beginning and end grapheme may be specified separated by a minus sign (e.g. <i>a-z</i>).
<code>\{range\}</code>	Matches one or more of a sequence of graphemes specified in <i>range</i> . <i>range</i> may consist of individual graphemes or a beginning and end grapheme may be specified separated by a minus sign (e.g. <i>0-9</i>).

Some examples are:

Find	Search
Records containing words starting with <i>abs</i> .	<code>abs*</code>
Records containing Arabic numbers.	<code>\{٠-٩\}</code>
Records with a three grapheme word.	<code>\?\?\?</code>
Records with <i>organisation</i> spelt with either an <i>s</i> or <i>z</i> .	<code>organi\[sz\]ation</code>
Records with at least one word containing a capital <i>S</i> .	<code>\=*S*</code>
Records containing either an upper case or lower case <i>é</i> .	<code>\==*\[éÉ\]*</code>

Anchors

Anchors are used to indicate that a search term should be located as either the first or last word in a piece of text. Anchors can be used in combination with all other types of search operators, namely transformations, regular expressions, phrases and proximity.

The table below lists all valid anchors:

Anchors	Description
\^	The search term following must be the first word in the text.
\\$	The search term following must be the last word in the text.

Some examples are:

Find	Search
Records that have text ending in a question mark.	?\\$
Records with text beginning with the word <i>the</i> .	\^the
Records where the text contains only the word <i>Unknown</i> .	\^Unknown\\$
Records with text where the first word starts with a lower case Latin letter.	\^\\=[a-z\\]*

Proximity

Proximity searching provides a mechanism for finding a list of words within a specified distance (either words, sentences or paragraphs). EMu supports two types of proximity searches:

- The first is phrase searches where the words must appear next to each other and in the order they are specified. The words in a phrase search may have transformations, regular expressions and anchors applied.
- The second is a regular proximity search. Proximity searches may include transformations, regular expressions, anchors and phrases.

The table below lists all valid proximity operators:

Proximity	Description
<code>\ "search terms"</code>	The <i>search terms</i> enclosed within the phrase operator (<code>\ "</code>) must appear next to each other and in the order they are specified.
<code>\ (search terms\) distance</code>	<p>The <i>search terms</i> may appear in any order unless otherwise specified. The <i>distance</i> between the terms indicates the range within which the search terms must appear. The syntax for <i>distance</i> is:</p> <p><code>[ordered] relop number type</code></p> <p>where:</p> <ul style="list-style-type: none"> • <i>relop</i> is one of the relational operators <code><</code>, <code><=</code>, <code>=</code>, <code>></code>, <code>>=</code> • <i>number</i> is the distance to use • <i>type</i> is one of words, sentences or paragraphs <p>The keyword <code>ordered</code> is optional, but if given, requires the search terms to be in the order specified.</p>

Some examples are:

Find	Search
Records where the phrase the black cat occurs.	<code>\ "the black cat"</code>
Records containing only the phrase Not Applicable.	<code>\ "\^Not Applicable\\$"</code>
Records where Fred occurs case significantly in the same sentence as the	<code>\ (\=Fred \@Smith\) ordered = 1 sentence</code>

Find

Search

phonetic of Smith where Fred appears first.

Records where the kanji character 豈 \ (豈 \"香 港\") <= 5 words appear within 5 characters of the phrase 香 港.

Conditionals

EMu provides support for one conditional operator, `NOT`. The `NOT` operator reverses the sense of the next search term. The `NOT` operator can be applied to any of the other search operators, that is transformations, regular expressions, anchors and proximity.

The table below lists the valid conditional operator:

Conditionals	Description
<code>\!</code>	The sense of the next search term is reserved.

Some examples are:

Find	Search
Records that do not contain the kanji 豈.	<code>\!豈</code>
Records that contain anything apart from the single word <code>Unknown</code> .	<code>\!\^Unknown\</code>
Records that do not contain the phrase <code>Not Applicable</code> .	<code>\!\ "Not Applicable"</code>
Records containing the phrase <code>Sacré Cœur</code> with case and diacritic significance but not <code>Paris</code> .	<code>\"\==Sacré \==Cœur" \!Paris</code>

SECTION 3

Auto-phrasing

Unicode graphemes are broken down into one of three categories for use in EMu 5.0. The categories are:

Category	Description
<i>combining</i>	A grapheme that is a simple letter or number. It is not a word in its own right but requires other characters to form words. Examples are the Latin, Arabic and Hebrew letters and numbers.
<i>single</i>	A single grapheme is used to represent a base word or meaning. Examples are Kanji and punctuation characters.
<i>break</i>	A character that delineates words, typically a space character.

Consider the following text:

香港 = "Hong Kong".

The graphemes along with categories are:

Grapheme	Category
香	<i>single</i>
港	<i>single</i>
	<i>break</i>
=	<i>single</i>
	<i>break</i>
"	<i>single</i>
H	<i>combining</i>
o	<i>combining</i>
n	<i>combining</i>
g	<i>combining</i>
	<i>break</i>
K	<i>combining</i>
o	<i>combining</i>

Grapheme	Category
n	<i>combining</i>
g	<i>combining</i>
"	<i>single</i>
.	<i>single</i>

EMu uses the category to determine what is an index term. Each *single* grapheme is treated as a separate index item, while *combining* graphemes are joined together to form a "word" up to a *break* or *single* category grapheme. A *break* grapheme is not an index term and is discarded.

In general, a phrase-based search must be performed where you want to find records where a list of index terms occur sequentially. For example, to find the two kanji characters 香港 (Hong Kong) next to each other, the query "\"香 港\"" may be used. Where a grapheme is part of the *single* category (like the two kanji characters), the system knows what the index term is and is able to treat them as a phrase provided a *break* character is not found. In fact EMu 5.0 treats a combination of *combining* and *single* graphemes as a phrase without the need for the phrase operator until a *break* grapheme is encountered. This process is known as auto-phrasing.

Auto-phrasing means that a query of 香港 is equivalent to "\"香 港\"" without the need to add the quotes or space. Another example is an email address such as fred@global.com. In this case the index terms fred, @, global, ., com must be located sequentially. Auto-phrasing effectively allows you to enter non-space separated terms and EMu will retrieve records where the terms are adjacent. If you do not want the terms to appear next to each other, for example if you want to find 香 (fragrant) 港 (harbour), then simply placing a space between the two kanji characters will disable auto-phrasing.

SECTION 4

Collation

Collation is the general term for the process of determining the sorting order of strings of characters. EMu 5.0 uses the Default Unicode Collation Element Table (DUCET), as defined in the Unicode 8.0 standard, to determine how text should be sorted. DUCET provides a locale independent mechanism for ordering values.

If you are interested in the ordering used by DUCET, please consult the Unicode Collation Charts (<http://unicode.org/charts/collation/>).

SECTION 5

Lookup Lists

The addition of support for searching on punctuation in EMu 5.0 has flowed through to other parts of the system. The most notable change is that punctuation is now significant in Lookup List values.

When comparing Lookup List entries prior to EMu 5.0, punctuation was removed before the entries were processed. Hence a Lookup List entry of `Smith (?)` was treated the same as an entry for `Smith`, so only one value (the first one entered in the system) would be stored. The problem is that these two entries are very different in meaning. The first implies a level of uncertainty with the name which is not present in the second.

EMu 5.0 retains punctuation when comparing Lookup List values, meaning that the two entries in our example are treated as separate and we end up with two entries in the Lookup List itself.

Index

A

Alternative methods • 9

Anchors • 21

Auto-phrasing • 25

C

Code Points • 3, 17, 18

Collation • 27

Conditionals • 24

E

Escaped code point • 6

G

Graphemes • 11

I

Index Terms • 12

Inputting Unicode Characters • 6

L

Lookup Lists • 29

P

Proximity • 22

R

Raw characters • 7

Regular Expressions • 20

S

Searching • 17

T

Transformations • 19

U

Unicode • 1, 13