

# EMu Documentation

## Release Notes: EMu 4.0.03

Document Version 1

EMu Version 4.0





# Contents

Here you will find collected together the Release Notes for EMu 4.0.03, alongside all documents referenced in the notes. These release notes and documents are also available on the [KE 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 4.0.03	5
EMu Multimedia Web Service	29
Encrypted Connections	58
Exhibition Objects Module	89
Multi-Group Support	112
OAI New IMu Webservices	139
ReadOnly Modes	172



# Release Notes: EMu 4.0.03

Release Date: April 2011

## Requirements

- For Windows 2000, XP, 2003, [Microsoft Windows Services for UNIX](#) (version 3.5)
- [Texpress 8.3.001](#) or later
- [TexAPI 6.0.002](#) or later
- [Perl 5.8](#) or later

## Download

[Download a PDF document](#) containing these release notes and associated documents.

Please see the [latest version of the EMu Help](#) for the most up to date version of this documentation.

## New Features

- **IMu Web Framework:** The IMu web framework provides a series of tools for distributing data held within EMu via the Intranet/Internet. IMu includes:
  - A structure for developing web pages to access your collection. The structure includes:
    - Theme-able pages.
    - MVC (Model-View-Controller) setup (XML > XSLT > HTML).
    - Collection List Management.
    - Search and display results from multiple modules.
    - Customisable interface via XSLT.
  - Extensible application server allowing new functionality to be "plugged in" via handlers.
  - Libraries to develop web applications for the following languages:
    - Java
    - .Net
    - PHP
    - Perl

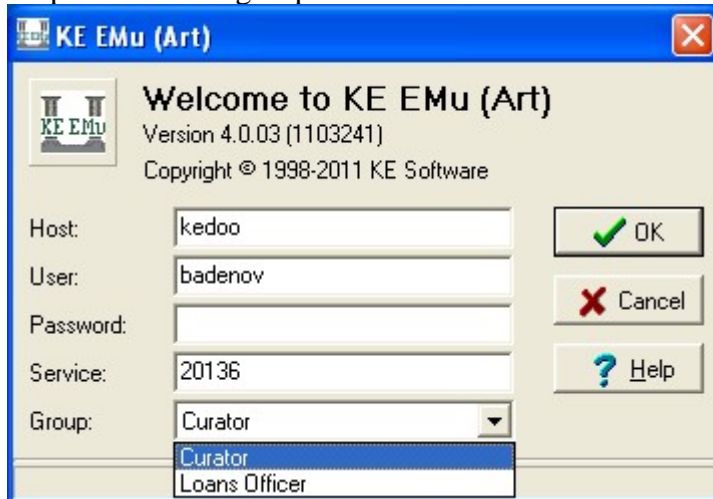
A complete description of the IMu web framework can be found in the [IMu Web Framework](#) documentation.

- **Multiple Group Support:** Multiple Group support allows a user to be a member of more than one group. The EMu Registry entry that defines the group into which a user is placed has been extended to allow a semicolon separated list of groups to be specified. For example:

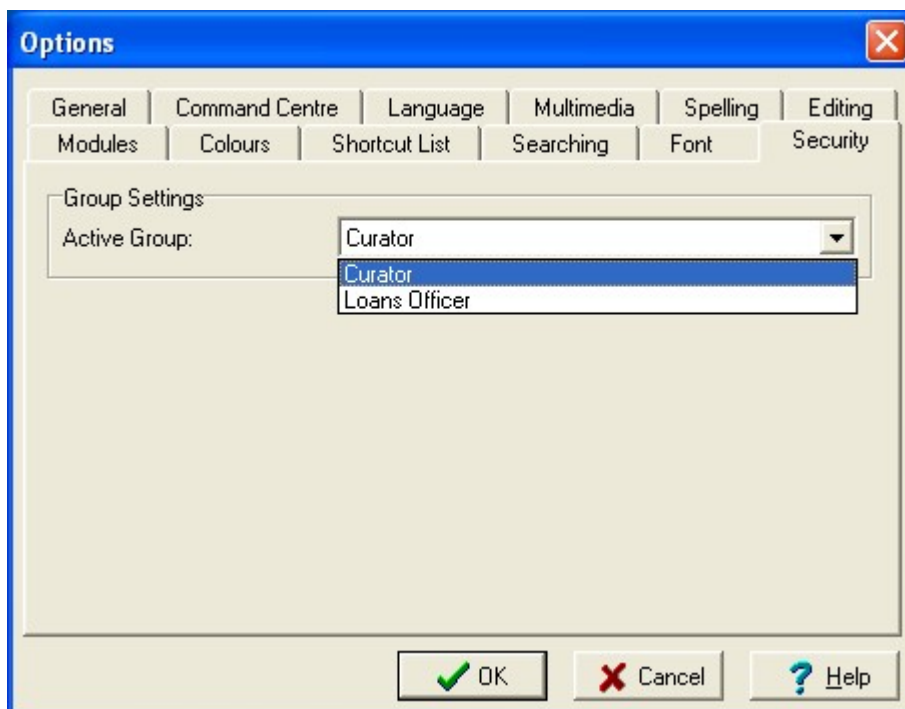
```
User|badenov|Group|Loans Officer;Curator
```

The above entry indicates user badenov is a member of two groups, Loans Officer and Curator. If a user is a member of multiple groups, the EMu Login screen provides a

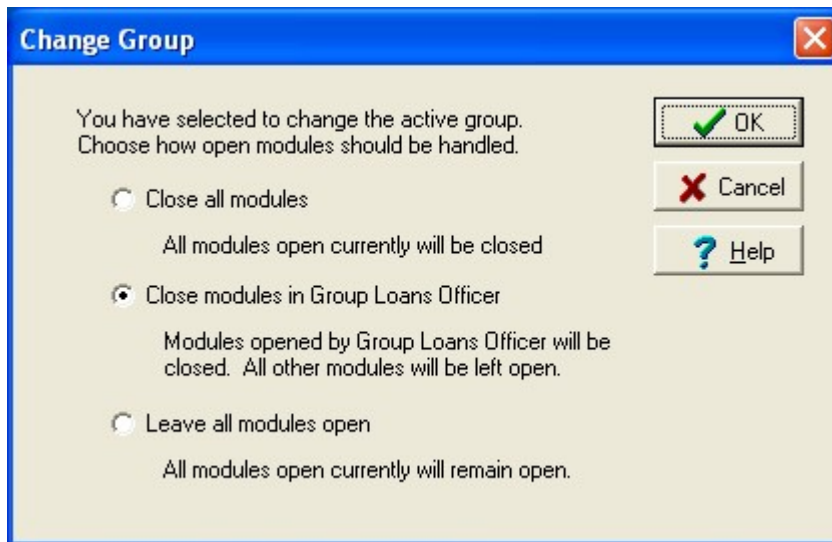
drop-list with the groups that the user is a member of:



The group selected becomes the *active* group and is used to determine permissions until another group is selected. The Security tab in the Options dialogue allows another group to be chosen:



When the active group is changed, a dialogue displays allowing the user to indicate how opened modules should be handled:



Any module left open will remain in its existing group. Hence it is possible to have two (or more) modules open, each in a different group. The permissions used for a given module are dictated by the group in which the module exists. The module's group is displayed in the status bar at the bottom of the module window.

A complete description of the support for multiple groups can be found in the [Multi-group Support](#) documentation.

- **Encrypted Connections:** Encrypted connections support allows all data transmissions between the EMu client and server to be encrypted. As part of the login process, X509 based certificates are exchanged allowing EMu clients to ensure they are connecting to the correct EMu server. TLS v1.0 (Transport Layer Security) is used to provide the encryption. System administrators may choose the encryption cipher used, allowing different levels of compression and security to be configured. The EMu server may be configured only to accept encrypted connections, otherwise encryption is optional.

Encrypted connections are also supported for:

- Java connections (TexJDBC)
- C/C++ connections (TexAPI)
- perl connections (texapi.pm)

A complete description of the support for encrypted connections can be found in the [Encrypted Connections](#) documentation.

- **Read-only Modes:** A new Registry entry allows System Administrators to make part or all of EMu read-only. The format of the new entry at a system wide level is:

```
System|Setting|ReadOnly|value
Group|Default|Setting|ReadOnly|value
Group|group|Setting|ReadOnly|value
User|user|Setting|ReadOnly|value
```

At a table level the format is:

Group	Default	Table	Default	ReadOnly	value
Group	Default	Table	table	ReadOnly	value
Group	group	Table	Default	ReadOnly	value
Group	group	Table	table	ReadOnly	value
User	user	Table	Default	ReadOnly	value
User	user	Table	table	ReadOnly	value

where *value* is either:

- o true - read-only functionality is enabled.
- o false - read-only functionality is disabled.

A complete description of the support for read-only modes can be found in the [Read-only Modes](#) documentation.

- **Exhibition Objects Module:** The EMu Art client has had a new module added. The Exhibition Objects module sits between the Catalogue and Events modules storing information about an object at a particular exhibition. When an object is added to an event, an Exhibition Objects record is created automatically. The Exhibition Objects record may then be used to record the usage of the object throughout the exhibition:

A complete description of the Exhibition Objects module can be found in the [Exhibition Objects](#) documentation.

- **Field Help Display:** A new option allows the field level help for a given field to be displayed below a module's Status bar. The help display area may be re-sized to suit:



The screenshot shows a software window titled "Parties (1) - Display". It has a menu bar with File, Edit, Select, View, Tools, Tabs, Multimedia, Window, and Help. Below the menu is a toolbar with various icons. The main area displays information for "Old Tote Co." with a reference number "105590".

Fields include:

- Party Type: Organisation
- Source of Information: (empty)
- Organisation Details:
  - Organisation: Old Tote Co.
  - Department: (empty)
  - Branch: (empty)
  - Section: (empty)
  - Position: (empty)
- Name Details:
  - Other Names: \*
  - Acronym: (empty)

At the bottom, there are tabs for Organisation, Address, Roles, Associations, History, Synonymy, Notes, and Mult. Below the tabs, it says "Display Party 1 of 2526" and "emu Admin 20136". A help text at the very bottom states: "The name of an organisation - can be a company, a government department, a non-profit organisation, a university, etc."

The Show Field Help in Module Window option is located on the General tab in the Options dialogue:

The screenshot shows the "Options" dialog box with the "General" tab selected. The "Display" section has two checked options: "Save Last Position" and "Show Field Help in Module Window". The "Record Updates" section has "Refresh changed records" checked, with a sub-option "Only refresh if less than: 0 matches". The "Registry" section has "Cache Registry Lookups" checked, with a "Clear Cache" button. The "System" section has "Cache System Resources" unchecked. At the bottom are "OK", "Cancel", and "Help" buttons.

- **Schedule Internal Movements:** The Movements module has been extended to allow internal movements to be scheduled as though they were external movements. This is often

the case when an internal movement requires objects to be shifted between off-site storage facilities:

**Movements (1) - Display**

File Edit Select View Tools Tabs Multimedia Window Help

[12 - Internal] 1

Movement Number: 12

Movement Direction: ☐ Incoming ☐ Outgoing ☒ Internal

Shipment Method: Car

Other Number:

Location Details:

New Location: [On Loan] On Loan

Requested By: Old Tote Co.

Released By: Old Tote Co.

Accepted By: Nimrod Theatre Co.

Movement Notes: Sent on loan

Movement Details:

Date Moved: 31/03/2011 Time Moved: 09:00

Internal Dates Handling Objects Finance Tasks Conservation N

Display Movement 1 of 4 emu Admin 20136

When the *Date Moved* is completed the system will ask if the attached objects should be relocated:

**KE EMu (Art)**

The internal movement is complete.  
Do you want all objects to be relocated to the new location?

Yes No

An affirmative response will relocate all objects to the new location.

- **Internal Pest Management Modules:** Two new modules have been added to the EMu Art client. These modules provide pest management support:
  - *Traps* - records information about the types and locations of traps used by an institution to capture pests:

**Traps (1) - Display**

File Edit Select View Tools Tabs Multimedia Window Help

Ichthyology, 1, Mouse Trap, [NM.MC.B] National Museum - Main Campus - Basement 1

Trap Details

Dept: Ichthyology Trap Number: 1 Type: Mouse Trap

Location: [NM.MC.B] National Museum - Main Campus - Basement

Placement Details

Placement Details: On floor near entry door.

Height (m): Proximity to Restaurant (m): Proximity to Eating Area (m):

Proximity to Window (m): Proximity to Internal Door (m): Proximity to External Door (m):

Trap Events

	Trap Events	Pest	Non-p...	Total
1	16/4/2011	0	1	1
2	14/4/2011	0	2	2
3	12/4/2011	0	1	1
4	10/4/2011	0	1	1
5	8/4/2011	0	12	12

Trap Tasks Notes Multimedia Security Audit Admin

Display Trap 1 of 1 emu Admin 20136

- *Trap Events* - records information about what was found in a trap at a given date and time:

**Trap Events (1) - Display**

File Edit Select View Tools Tabs Multimedia Window Help

10/4/2011 3

Trap

Ichthyology, 1, Mouse Trap, [NM.MC.B] National Museum - Main Campus - Basement

Trap Details

Dept: Ichthyology Trap Number: 1 Type: Mouse Trap

Location: [NM.MC.B] National Museum - Main Campus - Basement

Check Details

Date Checked: 10/04/2011 Checked By: Missing: ☐ Yes ☒ No

Pest Information

	Common Name	Species	Eco Type	Life Stage	Count	Notes
1	Mice			Adult	1	
*						

Total Pest: 0 Total Non-pest: 1 Total Count: 1

Trap Event Notes Multimedia Security Audit Admin

Display Trap Event 4 of 5 emu Admin 20136

- A comprehensive set of reports provide graphical data about the prevalence of pests within your institution.

- **Multimedia Web Service:** The Multimedia Web Service is an IMu service providing HTTP based access to the EMu Multimedia Repository. The following functions are provided:
  - *Resource Request* - ask for a multimedia resource (image, video, document, etc.). For image based resources a height, width and/or format may be specified, allowing images to be generated "on-the-fly".
  - *Metadata Request* - return any metadata associated with the resource. The metadata is an XML marked-up version of the data in the Multimedia record. For image based resources the following additional metadata formats are available:
    - EXIF
    - IPTC
    - Dublin Core
  - *Resource Discovery* - search the Multimedia Repository for resources with a given file name.
  - *Resource Listing* - list all resources stored in the Multimedia Repository.
  - *Resource Insertion* - add a new resource to the Multimedia Repository.
  - *Metadata Update* - update the metadata (fields in the Multimedia record) for a given resource.
  - *Resource Replacement* - replace the resource file (image, document, etc.) associated with a Multimedia Resource record.
  - *Resource Deletion* - remove a resource from the Multimedia Repository permanently.

A complete description of the Multimedia Web Service can be found in the [Multimedia Web Service](#) documentation.

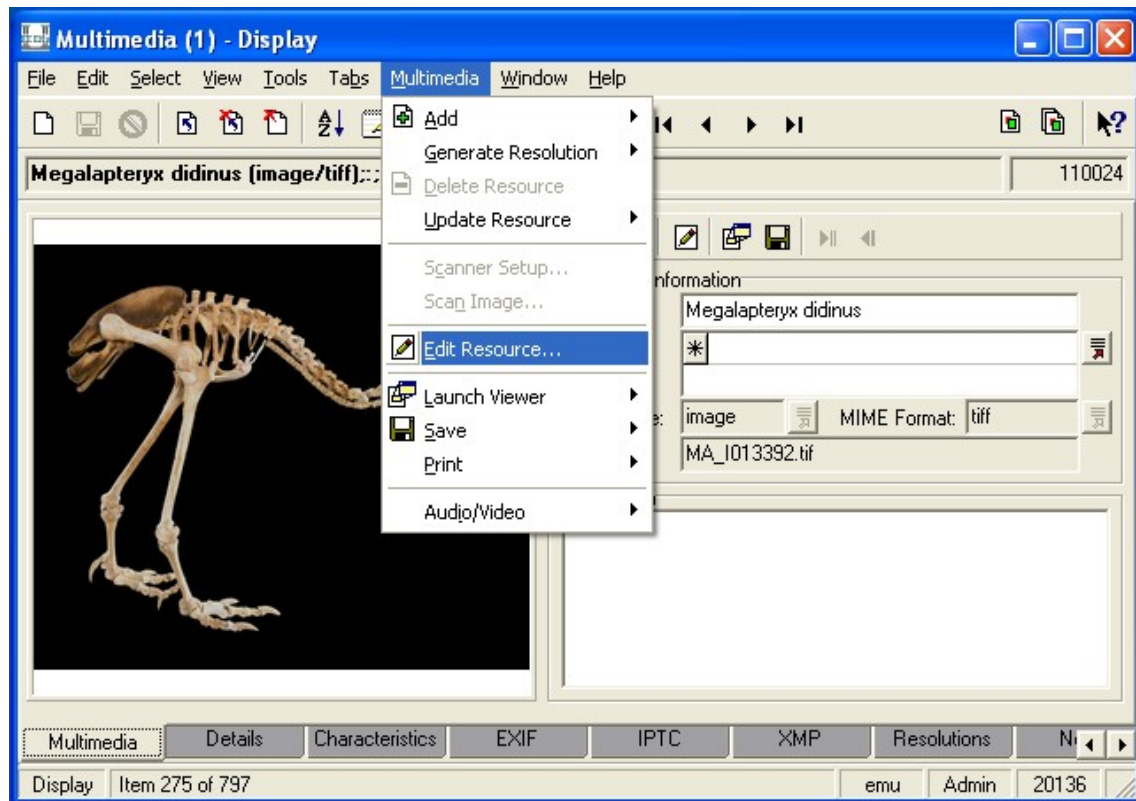
- **OAI Web Service:** The OAI (Open Archives Initiative) Web Service is an HTTP IMu service implementing the OAI 2.0 protocol. The service implements the six verbs required of all OAI repositories, namely:
  - *Identify* - request a summary of the service.
  - *ListMetadataFormats* - provide a list of the metadata formats the repository supports (e.g. Dublin Core, etc.).
  - *ListSets* - provide a list of data sets available for harvesting.
  - *ListIdentifiers* - provide a list of unique record identifiers (for EMu this consists of the institution acronym combined with the module name and record IRN).
  - *ListRecords* - retrieve a list of records in a given data set.
  - *GetRecord* - retrieve an individual record.

A complete description of the OAI Web Service can be found in the [OAI Web Service](#) documentation.

## Improvements

- **Edit Resource command:** A new command has been added to the Multimedia Repository allowing a user to invoke an external editor to modify the multimedia resource (image, document, etc.). The Edit Resource command downloads the resource from the EMu server, invokes an editor and monitors the file for any changes. It also looks for any other files created with the same name as the resource file, but with a different extension (file type). When EMu receives focus the user is asked whether the modified file should be re-imported into the Multimedia Repository. An affirmative response saves the resource on the EMu

server and generates any resolutions, etc. The new command streamlines the process of modifying multimedia resources:

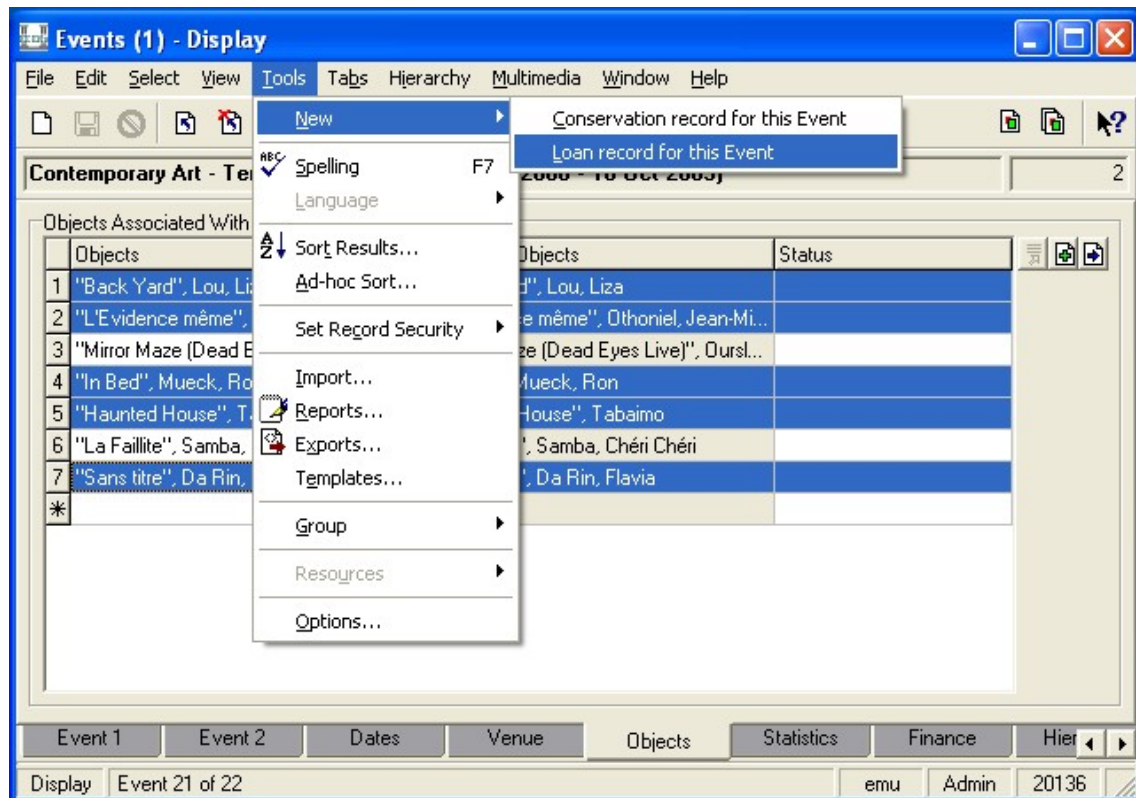


- **XMP and IPTC upgraded:** The Multimedia Repository has been upgraded to support the latest versions of the IPTC and XMP metadata standards. The standards implemented are:
  - IPTC Core Version 1.1
  - IPTC Extension Version 1.1
  - XMP July, 2010
- **Login details on a per client basis:** The Host/User/Service/Group login values are remembered on a per client basis. If an institution runs multiple clients, the details displayed in the login dialogue are the values from the last time that particular client was run, rather than the values for the last time any client was invoked.
- **Change password utility uses SSH:** The administration task allowing a user to change their password now uses SSH (secure shell) to effect the change. Previously TELNET was used, requiring the TELNET service to be enabled on the server for the utility to work. By using SSH, the TELNET service no longer needs to be enabled, rather the SSH service is required. Most sites enable SSH to allow secure access to the EMu server.
- **ICS support for notifications:** The EMu notification system, used to generate email notifications and HTML pages for upcoming activities (loans, events, task, etc.), now builds an iCalendar (.isc) file containing upcoming dates. The iCalendar file may be imported into a large number of products, including:
  - Google Calendar
  - Apple iCal
  - IBM Lotus Notes
  - Microsoft Outlook



Once imported, the upcoming dates, along with a detailed description, are incorporated into the user's calendar.

- **Reverse scheduling of tasks:** The Task tab functionality has been extended to allow the completion and start dates of each task to be computed by specifying the overall completion date. The new functionality allows a user to set the date on which all tasks should be completed and have EMu calculate the start and completion dates for each task.
- **Creation of attached records:** In order to streamline the creation of related records, a set of record creation commands has been added to the **Tools>New** menu:



The selection of a record creation command will start up a new instance of the selected module in New mode with a link to the original record in place. Any objects attached to the originating record that are selected are added to the record created. If objects are not selected, all objects are added to the new record.

For example, while in the Events module the user may select from the list of objects in the event and select **Tools>New>Loan records for this Event**. The command will start a new instance of the Loans module in New mode. The Associated Event field will be linked back to the Events record and the *Objects* field will contain the selected objects from the Events record. If objects are not selected, then all objects are attached.

The command exists for the following modules:

- Events -> Loans
- Events -> Conservation
- Loans -> Movements
- Loans -> Conservation

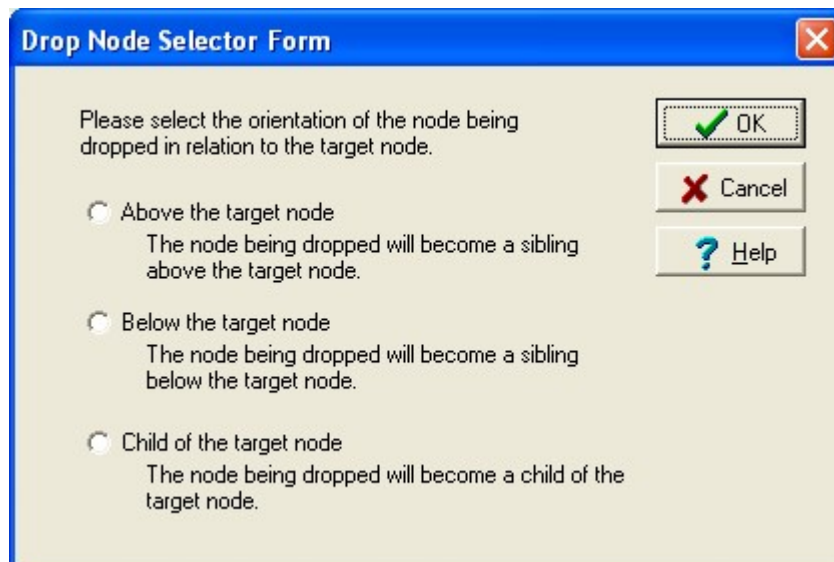
- Movements -> Conservation
  - Traps -> Trap Events
- **Link from Conservation to Movements:** A link has been added from the Conservation module to the Movements module to allow associated movements to be recorded. If the reason for the movement is transporting the object to another location for conservation work, the associated movement allows the movement details to be recorded against the conservation record:

The screenshot shows a software window titled "Conservation (1) - Display". It features a menu bar (File, Edit, Select, View, Tools, Tabs, Multimedia, Window, Help) and a toolbar with various icons. The main content area is divided into several sections:

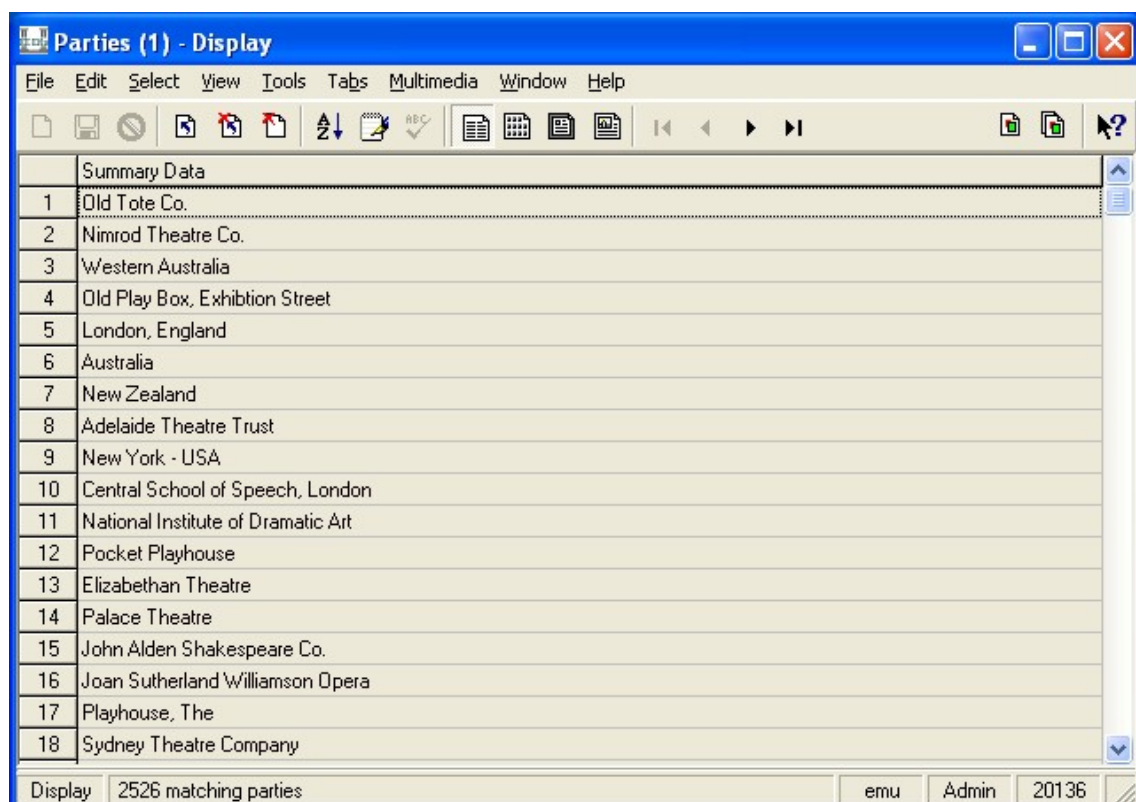
- Record Header:** Shows "C 231" in a box, with a page number "44" to the right.
- Form Fields:**
  - Conservation Number: C 231
  - Record Type: Repair (dropdown)
  - Record Status: Completed (dropdown)
  - Record Date: 12/10/2006
- Objects Conserved:** A table with two columns: "Object" and "Note".
 

Object	Note
1 "Charles, 9th Lord Cathcart" (1981.36). Reynolds, Joshua	
*	
- Record Information:**
  - Conservators: \*
  - Associated Event: Joshua Reynolds: The Creation of Celebrity - Touring Exhibition (12 Feb 2005 - 01 May 2006)
  - Associated Loan: returned - [L2004.9a - Outgoing] Tiziana Giuberti - Gallery of Modern and Contemporary Art
  - Associated Movement: [Outgoing] Tiziana Giuberti - Gallery of Modern and Contemporary Art of the City of Ferrara.
  - Other Reason:
- Navigation and Footer:**
  - Buttons: Objects, Description, Condition, Proposal, Treatment, Analysis, Accounting, Record.
  - Status bar: Display, Record 6 of 7, emu, Admin, 20136.

- **Modifications to emuperiodic:** The `emuperiodic` server-side command is used to run tasks on a regular basis. In particular, it is used to generate the raw data for the Statistics module. The command has been extended to allow:
  - Individual scripts to be specified, restricting the scripts executed.
  - Execution of local scripts only, via the `-l` option.
- **Drag and Drop on Archive Tab:** The Archive tab in the Catalogue module displays the relationship between the current record and all other records in the same archive. It is now possible to relocate the current record by dragging it to another position in the archive tree. When the record is dropped, the user may select how the current record should be positioned relative to the record on which it was dropped:



- **Read-only records in List View:** The display of read-only records in List View has been changed. Read-only records are now displayed with a grey background and black text, rather than the harder to read, grey text on a white background:



- **emusecurity run automatically when required:** A new Audit Trail handler has been added that monitors the EMu Registry for changes where a user's settings have been modified. Where settings have been changed, **emusecurity** is executed to update the security settings for all EMu databases. The following events are handled:
  - Addition of a new user.
  - Deletion of an existing user.
  - Changing the group(s) of an existing user.



- Altering database security via the `Security` Registry entry.

There should be no need to run `emusecurity` manually.

- **Image Quality available for JPEG 2000:** When generating JPEG 2000 based images, the Image Quality may now be defined. The Image Quality determines the level of sampling used when creating the image. The value is between 1 and 100, where 100 implies full sampling, that is no lose of image quality, and 1 represents the least amount of sampling, resulting in a much smaller file size, but with image degradation.



## Issues Resolved

### Issue

The functions used to convert between imperial and metric units are, in many cases, built into specific clients. Each function is essentially the same with minor differences. The minor differences cause some conversions to compute an incorrect precision for the calculated value.

When a conversion is applied between different units (e.g. feet to metres), the value entered by the user is flagged as the "Original" value. The flagging of the original value requires a number being placed in an *Orig* column indicating which field had the data entered. In some instances the *Orig* column may be shown as a text field when used in reports, rather than as an integer field.

The *Narrower Terms* and *Used For* fields in the Thesaurus module do not have attachment buttons displayed, even though they are attachment fields. The buttons are missing from Query and Detail modes.

If you enter a term into a Thesaurus controlled field that has the Thesaurus Validation Registry entry set to none, and the term entered appears in the Thesaurus, but not by itself (e.g. the term `sports` was entered, but only `Sports Car` exists in the Thesaurus), the term entered is cleared if you do not select a term. The current functionality makes it impossible to save a record with the original term as it is always cleared.

If a multi-term query is performed on an attachment field, where the field searched in the attached module has `Also Search` Registry entries associated with it, the query generated may not be the most efficient one possible.

For systems running in UTF-8 mode, the Audit Trail mechanism will drop audit records containing invalid UTF-8 characters. As the audit records are not posted to the Audit Trail table a "gap" may appear in the audit trail. The issue only

### Resolution

All conversion functions have been merged and checked to ensure the correct precision is implied for the calculated value.

All *Orig* fields have been updated to be integer based.

The attachment buttons have been enabled for *Narrower Terms* and *Used For*, in both Detail and Query modes.

The term entered is no longer cleared if a term is not selected from the Thesaurus where the Validation Registry entry is set to none.

The query generated for multi-term attachment queries where the attached column has `Also Search` Registry entries is now efficient.

The Audit Trail mechanism now replaces invalid UTF-8 characters with the invalid UTF-8 character (0xFFFD). The substitution allows the audit record to be processed without

## Issue

occurs where invalid data is stored in a record (normally as a result of bad data when importing into EMu).

The **View>Attachments** command, used to show all records attached to the current record, displays modules the user does not have permission to use. While it is useful to see whether any records in the module attach to the current record, the module cannot be invoked without displaying a permission denied error.

If a Taxonomy record is re-parented to a name with a higher rank, the intervening ranks are not discarded from the Hierarchy tab.

An **Access Violation** message may be displayed when visiting the Hierarchy tab in the Narratives module. The message is displayed only if a hierarchy of narratives exists.

When importing records into a Catalogue that uses the location SPLIT facility (used in parent records to indicate the child records are not all at the same location), spurious Internal Movements records may be created, even though none of the child records has been relocated. The spurious Internal Movements records note the location as unchanged.

A **Data validation failed on LatCentroidLongitude0** message may be displayed when calculating centroid values from decimal numbers. If converting the decimal value to degrees/minutes/seconds causes the seconds value to be close to 60, a rounding error may cause a value of 60 to be used. As this is not a legal value the error message is displayed.

The mode displayed in the Module Title bar may be repeated when switching between display languages (e.g. English to Arabic).

The **emulutsupdate** server-side script does not update Lookup List files in

## Resolution

error.

The **View>Attachments** command has been modified to show only those modules the user has permission to access. A check box has been added to *Show all modules*, allowing the old behaviour to be selected.

Re-parenting now causes the intervening ranks to be removed from the Hierarchy tab.

The message no longer appears when displaying the Hierarchy tab where a hierarchy of narratives exists.

Spurious Internal Movements records are no longer created when importing records into a Catalogue that uses the SPLIT location facility.

When converting decimal values to degrees/minutes/seconds where the second value may result in 60, the minute value is incremented and the seconds reset to zero.

The mode is no longer repeated when the display language is switched.

Local versions of Lookup List file are now updated by the

## Issue

`local/luts/default` if it exists. Files in the `luts/default` directory are always updated. The issue means that localised versions of Lookup Lists may not be updated correctly by EMu upgrade scripts.

A number of the background loads used to synchronise data require the **fifo** load to be running when they are started. The `emuload` command does not check to ensure the **fifo** load is running before starting other background loads.

If the Thesaurus module is opened in Browse mode, a term is selected in the tree, and then Details mode is chosen, the term record displayed may be read-only, even if the user has edit privileges.

If the Thesaurus module is maximised or enlarged in Browse mode and the window splitter is moved to the right to increase the size of the terms tree, after which the module is returned to normal size, the fields on the Browse tab may disappear off the right hand edge of the page.

If a second alternate term is added to a Thesaurus record, where the record was saved after the first alternate term was entered, an error message may display indicating non unique terms are present.

If the Refresh tree option is selected in the Thesaurus module and the current record has *Use* and *Use For* terms, then the *Use* and *Use For* terms are repeated at the top of the Browse tree. The terms are repeated each time the Refresh option is invoked.

An Access Violation message may be displayed when switching the display language in the Thesaurus module where the Browse tree is empty.

If a Narratives record may appear in two branches of a narratives hierarchy, the child narratives may not be shown for both branches of the hierarchy.

## Resolution

`emulutsupdate` server-side command.

`emuload` checks the **fifo** load is running before starting any other background load. If the **fifo** load is not running, it is started, then any other loads listed are invoked.

The term record is now displayed with the correct permissions, rather than being read-only.

The fields on the Browse tab are displayed in the correct position when the Thesaurus module is re-sized.

The Thesaurus record now saves without error when a second alternate term is entered.

The *Use* and *Use For* terms are no longer repeated when the Refresh option is invoked.

The message no longer appears when switching languages in the Thesaurus module with an empty Browse tree.

The child Narratives records are now displayed regardless of how many branches the parent narrative appears in.

## Issue

## Resolution

The field prompts on the Permits query tab in the Rights module may not display correctly. The prompts are set to display using a right-to-left orientation, rather than the standard left-to-right setting.

The prompts have been changed to use a left-to-right orientation by default.

Data in reference fields, that is fields displaying data from another module, may not contain the correct values after certain operations are performed. The affected operations are:

Data in reference fields now displays the correct values after the listed operations are performed.

- Discard Record
- Replace (Global Edit)
- Export Records
- Merge Record
- Sort
- Delete Record

The value in a read-only combo-box may be cleared using the `backspace` key the first time data is viewed. The `backspace` key will not clear the value when viewing subsequent records.

The `backspace` key is disabled when a combo-box is read-only, regardless of when the data is viewed.

Duplicate attached records may be created when using the Import facility to create attached Collection Events or Sites records where the data contains lat/long values. The issue is caused by the failure of the lat/long conversion code when importing records.

The correct lat/long conversion now occurs, removing the creation of duplicate Collection Events or Sites records with the same data.

Random Access Violation messages may display indicating an error has occurred in `mshtml.dll`. The error message may even appear when a user is not at their machine.

The random error message is no longer displayed.

When performing a spell check on a RichEdit control that has multiple lines of text, the highlighting of the words spelt incorrectly may be out by a few characters.

Words spelt incorrectly are now highlighted correctly.

## Issue

## Resolution

The View Attachments button next to a read-only grid may not be enabled when a single row is displayed in the grid. The button is only disabled for grids displaying data from another module (a reverse reference column).

The View Attachment button is now enabled when a single value is in a read-only grid.

The default value for the *Gender* field in the Parties module is set to Female. Ideally the default values should be Unknown.

The default value for the *Gender* field is now Unknown.

When changing the colours for various attributes (e.g. mandatory fields, etc.) via the Options dialogue box, the new colours selected will not take effect in modules already open. New instances will display the correct colours.

Existing modules now update their colours correctly when the Options dialogue box is closed.

When switching to the Details view of a record on a tab that contains an HTML control, the input focus is set to the HTML control, rather than the first control on tab.

The input focus is set to the first control on a tab when switching from Details view, even if the tab contains an HTML control.

*A List of Values failure: fail to get values* message may be displayed when generating a Crystal Report that contains dynamic parameters.

The error message is no longer displayed for Crystal Reports containing dynamic parameters.

The Audit Trail facility does not flush logging information when the processing of a record has been completed. The lack of flushing means the log file contents may lag behind the processing of records.

The Audit Trail facility now flushes logging information after each records processed.

If a taxonomic name appears in the Names History tab of other records it is difficult, if not impossible, to delete the name. Since the original name points to the preferred name, and the preferred name contains a history of previous names, a loop is formed. When a record is deleted it is not allowed to be referenced by any other record. A loop makes it impossible to delete either record, as it is referenced by the other record.

When deleting a taxonomic name, the Names History tab is now checked to ensure the original record is not referenced. This allows the record to be deleted.

The Taxonomy module has NULL indexing enabled on many of its fields.

All NULL indexing has been disabled in the Taxonomy module.

**Issue**

Unfortunately, this imposes a very large indexing overhead and as the indexing is enabled in the base module it cannot be disabled via the Registry.

The Taxonomy module does not generate the correct scientific name for autonyms.

The auto filling of values for the upper levels of hierarchies may be slow where the controls that form the hierarchy appear on a number of tabs.

**Resolution**

System Administrators may enable it on selected fields via the EMu Registry. The indexing overhead for the Taxonomy module has been reduced dramatically.

The author is now inserted at the appropriate place in the sequence in the scientific name for autonyms.

The speed of auto filling of values in hierarchies has been improved dramatically.



## Upgrade Notes

The upgrade from EMu Version 4.0.02 to EMu 4.0.03 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. Upgrading comprises the following steps:

- [Stopping EMu services](#)
- [Installing Texpress](#)
- [Upgrading TexAPI](#)
- [Upgrading EMu Server](#)
- [Starting EMu services](#)
- [Upgrading EMu Client](#)

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 each **data** file is empty and no **data.t** files exist.  
If this is not the case, 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-4-0-03`

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

### Installing Texpress

Installing Texpress 8.3 is only required for the first client upgraded to EMu 4.0.03. Once Texpress 8.3 has been installed, this section may be skipped for subsequent upgrades.

1. Enter `cd ~emu`
2. Enter `mkdir -p texpress/8.3.001/install`
3. Enter `cd texpress/8.3.001/install`
4. Obtain the appropriate Texpress version for your Unix machine via the *Texpress* hyperlink at the top of the page.  
Save the release in `~emu/texpress/8.3.001/install`, calling it **texpress.sh**.
5. Enter `sh texpress.sh`  
The Texpress release will be extracted.
6. Enter `. ./profile`
7. Enter `bin/texinstall ~emu/texpress/8.3.001`  
The Texpress installation script will commence.
8. Enter `cd ~emu/texpress/8.3.001`
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/texpress`
14. Enter `ln -s 8.3.001 8.3`

#### Upgrading KE TexAPI

Installing TexAPI is only required for the first client upgraded to EMu 4.0.03. Once TexAPI has been installed, this section may be skipped for subsequent upgrades.

1. Enter `cd ~emu/texpress`
2. Enter `mkdir 6.0.002`
3. Obtain the appropriate TexAPI version for your Unix machine via the *TexAPI* hyperlink at the top of the page.  
Save the release in `~emu/texpress`, calling it **texapi.sh**.
4. Enter `sh texapi.sh -i ~emu/texpress/6.0.002` (expand the `~emu`).
5. Enter `\rm -f texapi`
6. Enter `ln -s 6.0.002 texapi`
7. Enter `\rm -f texapi.sh`

#### Upgrading EMu Server

1. Enter `cd ~emu/clientname`
2. For Unicode based clients only.  
Enter `vi .profile-local`  
Change `langcode=utf8` to `langcode=utf-8` and save the change.
3. Enter `mkdir install`
4. Enter `cd install`
5. Obtain the appropriate EMu server version bundle via the *EMu Server* hyperlink at the top of the page.  
Save the release bundle file in `~emu/clientname/install` calling it **emu.sh**.

6. Enter `sh emu.sh`  
The EMu release will be extracted.
7. Enter `. ../.profile`
8. Enter `bin/emuinstall clientname`  
The EMu installation script will commence.
9. Enter `cd ~emu/clientname`
10. Enter `cp .profile.parent ../.profile`
11. Enter `. ../.profile`
12. Enter `client clientname`
13. Enter `emubldinstall`
14. Enter `emubldlinks`
15. Removal of the temporary directory (and its contents) is recommended:  
Enter `\rm -fr install`
16. Enter `emureindex`
17. Enter `upgrade-4-0-03`  
The client will now be upgraded to EMu 4.0.03. If you are upgrading from a version prior to EMu 4.0.02, you must run the upgrade scripts for all versions after the old version before running the EMu 4.0.03 upgrade.

#### Starting EMu services

1. Enter `emuload start`
2. Enter `emuweb start`

#### Record Session

The recording of the upgrade session may now be terminated.

1. Enter `exit`

The session output is available in **/tmp/output-4-0-03**.

#### Upgrading EMu Client

When upgrading to EMu 4.0.03 the Windows client should be upgraded on each individual machine. The client upgrade installs new DLLs on each individual machine in order for all reports with dynamic parameters to function correctly. To upgrade the EMu Client follow the [Installing EMu Client](#) notes.

#### Perl Packages

If the administration utility to change a user's password is used, the perl `Net::SSH::Expect` package must be installed. If the password utility is not used, this step may be skipped. To install the package:

1. Log in as **root**
2. Enter `perl -MCPAN -e shell`
3. Enter `install Net::SSH::Expect`
4. Enter `quit`

# EMu Documentation

# Multimedia Web Service

Document Version 1.1

Version 1.1





# Contents

Introduction	3
Resource Request	4
Metadata Request	7
Resource Discovery	10
Resource Listing	11
Resource Insertion	13
Metadata Update	15
Resource Replacement	17
Resource Deletion	19
Appendix 1: Authentication/Security	20
Appendix 2: Supported Image Formats	21
Appendix 3: Native Format	26



## Introduction

This document provides a detailed description the facilities available in the EMu Multimedia Web Service. The service accepts HTTP GET or PUT requests and responds with standard HTTP responses. The following types requests are accepted:

- Resource Request
- Metadata Request
- Resource Discovery
- Resource Listing
- Resource Insertion
- Metadata Update
- Resource Replacement
- Resource Deletion

Each of these request types is described in the following sections.



## Resource Request

A resource is requested by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of “resource”.

### Parameters for all Resource requests

Parameter Name	Mandatory	Default Value	Description
<b>request</b>	Yes	n/a	To request a multimedia resource the value of the request parameter must be “resource”.
<b>irn</b>	No	n/a	Integer containing the EMu internal record number of the multimedia resource. A request must contain an <b>irn</b> or <b>identifier</b> , or both.
<b>identifier</b>	No	n/a	Text containing a 3 <sup>rd</sup> Party unique identifier for the resource. A request must contain an <b>irn</b> or <b>identifier</b> , or both.
<b>encoding</b>	No	raw	The encoding of the resource to be returned. Acceptable values are “raw” (the default) or “base64”.
<b>checksum</b>	No	none	If set to a value other than “none” (which is the default) a checksum of the resource will be added to the response generated. Acceptable values are “none” or “crc32”.

### Additional parameters for image-specific Resource requests

Parameter	Mandatory	Default Value	Description
<b>width</b>	No	Width of the master image.	Pixel width of generated image. Note that if width is not specified but height is, then width will scale to maintain aspect ratio.
<b>height</b>	No	Height of the master image.	Pixel height of generated image. Note that if height is not specified but width is, then height will scale to maintain aspect ratio.
<b>format</b>	No	n/a	The image format. See Appendix 2 for a list of supported formats. If no format is specified the resource will be in the same format as the resource it is generated from.
<b>aspect</b>	No	yes	If set to “yes” (or “y”), aspect ratio will be conserved in the generated image. If set to “no” (or “n”) aspect ratio will not necessarily be preserved.

	<b>rendermaster</b>	No	no (or n)	If set to “yes” (or “y”) the generated image will be rendered from the master image in EMu. If set to “no” (or “n”) the generated image will be rendered from smallest generated resolution in EMu that is larger than this request’s resolution. For lossy formats it is better to be generating an image from the master image.
--	---------------------	----	-----------	---

**Example 1:**

<http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234>

This is the simplest form of this request. It retrieves the multimedia resource associated with the MMR record with an IRN of 1234. The resource is returned in raw (binary) format.

**Example 2:**

<http://server/optionalpath/multimedia/entry.php?request=resource&identifier=abcd&encoding=base64>

This request retrieves the multimedia resource associated with the MMR record which has “abcd” in its identifier (MulIdentifier) field. This request fails if more than one record matches this identifier. The resource returned is base64 encoded.

**Example 3:**

[http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&th=30](http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&width=30)

This request retrieves the multimedia resource associated with the MMR record with an IRN of 1234. An image with a width of 30 pixels is generated. Aspect ratio of the image is preserved. The returned image is generated from the smallest resolution of the master image that is larger than the image being generated.

**Example 4:**

[http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&th=30&height=30&format=tif&aspect=no](http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&width=30&height=30&format=tif&aspect=no)

This request retrieves the multimedia resource associated with the MMR record with an IRN of 1234. An image with a width and height of 30 pixels is generated. Aspect ratio of the image is not preserved. The generated image is a TIFF, and is generated from the smallest resolution of the master image in EMu that is larger than the image being generated.

**Example 5:**

[http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&th=30&height=30&format=tif&rendermaster=yes](http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&width=30&height=30&format=tif&rendermaster=yes)

---

This request retrieves the multimedia resource associated with the Multimedia record with an IRN of 1234. An image with a width and height of 30 pixels is generated. Aspect ratio of the image is preserved. The generated image is a TIFF, and is generated from the master image in EMu.

## Response

### Success

An HTTP 200 OK response containing the requested multimedia. The content is encoded using the requested encoding, if any. Otherwise the content will be in raw (binary) form. The details of the HTTP response headers will depend on the resource being delivered. As an example the headers for a response containing a JPEG image include:

```
Content-type: image/jpeg
Content-Length: xxx
Content-Transfer-Encoding: base64
Content-Crc32: deadbeef

...contents of image...
```

### Failure

HTTP 403 Forbidden if the access is not granted to the resource.

HTTP 404 Not Found if the resource cannot be located.

---

## Metadata Request

Metadata for a resource is requested by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of “metadata”.

### Parameters for all Metadata requests

Parameter Name	Mandatory	Default Value	Description
<b>request</b>	Yes	n/a	To request the metadata of a multimedia resource the value of the request parameter must be “metadata”.
<b>irn</b>	No	n/a	Integer containing the EMu internal record number of the multimedia resource. A request must contain an <b>irn</b> or <b>identifier</b> , or both.
<b>identifier</b>	No	n/a	Text containing a 3 <sup>rd</sup> Party unique identifier for the resource. A request must contain an <b>irn</b> or <b>identifier</b> , or both.

### Additional parameters for image-specific Metadata requests

Parameter	Mandatory	Default Value	Description
<b>format</b>	No	all	The <b>format</b> parameter specifies what metadata to return. See below for supported metadata formats.

The requested metadata is returned in Adobe's Extensible Metadata Platform (XMP) format (<http://www.adobe.com/products/xmp/>).

The “format” parameter will accept the following values:

- **EXIF**

Requests with this value will return the full set of EXIF 2.2 elements embedded in the XMP document. See <http://www.exif.org/Exif2-2.PDF> for more details. If the resource does not contain EXIF metadata an error response will be returned.

- **IPTC**

Requests with this value will return the full set of IPTC 4.1 elements embedded in the XML document. See <http://www.iptc.org/IPTC4XMP/> for more details. If the resource does not contain IPTC metadata an error

---

response will be returned.

- **DC**

Requests with this value will return the full set of Dublin Core fields recorded in the MMR record embedded in the XMP document.

- **native**

Requests with this value will include a complete description of the data in the MMR record embedded in the XML document. The values returned are not limited to metadata embedded within the digital resource itself. The format of the XML is based on the standard RDF document structure.

- **all**

Requests with this value will return all known metadata formats: EXIF, IPTC, DC and native in the one XMP document. This is the default.

### Example 1:

<http://server/optionalpath/multimedia/entry.php?request=metadata&irn=1234>

This is the simplest form of this request. It retrieves all known metadata formats for the multimedia resource associated with IRN 1234 in the MMR.

### Example 2:

<http://server/optionalpath/multimedia/entry.php?request=metadata&irn=1234&format=EXIF>

This request retrieves the EXIF metadata for the multimedia resource associated with IRN 1234 in the MMR. If the resource does not contain EXIF metadata the server will respond with an HTTP 400 Bad Request error

## Response

### Success

An HTTP 200 OK response containing the requested metadata. For example:

```
Content-type: text/xml
Content-Length: xxx
```

```
<?xml version="1.0"?>
<?xpacket begin=" " id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description
      rdf:about=" "
      xmlns:dc="http://purl.org/dc/elements/1.1"
      xmlns:tiff="http://ns.adobe.com/tiff/1.0/"
      xmlns:exif="http://ns.adobe.com/exif/1.0/"
      xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
      xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
      xmlns:native="http://kesoftware.com/emu/xmlns/emultimedia/"
    >

      <!-- Dublin Core -->
```

---

```

<dc:format>image/jpeg</dc:format>
<dc:identifier>picture.jpg</dc:identifier>
...

<!-- EXIF -->
<exif:ThumbnailOffset>4852</exif:ThumbnailOffset>
<exif:XResolution>72</exif:XResolution>
<exif:ExposureTime>1/400</exif:ExposureTime>
<exif:Model>E-20,E-20N,E-20P</exif:Model>
...

<!-- IPTC -->
<Iptc4xmpCore:DateCreated>2006:12:18</Iptc4xmpCore:DateCreated>
<Iptc4xmpCore:City>Broome</Iptc4xmpCore:City>
<Iptc4xmpCore:Province-State>WA</Iptc4xmpCore:Province-State>
...

<!-- Native -->
<native:AdmDateInserted>19/08/2009</native:AdmDateInserted>
<native:AdmDateModified>19/08/2009</native:AdmDateModified>
<native:AdmTimeInserted>10:44</native:AdmTimeInserted>
<native:AdmTimeModified>10:44</native:AdmTimeModified>
<native:ChaFileSize>1331398</native:ChaFileSize>
<native:ChaImageColorDepth>8</native:ChaImageColorDepth>
<native:ChaImageHeight>1920</native:ChaImageHeight>
<native:ChaImageResolution>144</native:ChaImageResolution>
<native:ChaImageWidth>2560</native:ChaImageWidth>
<native:ChaMd5Sum>7f648024f39541e439a666ccd3de4493</native:ChaMd5Sum>
<native:DocBitsPerPixel_tab>
  <rdf:seq>
    <rdf:li>8</rdf:li>
    <rdf:li>16</rdf:li>
  </rdf:seq>
</native:DocBitsPerPixel_tab>
<native:DocColourSpace_tab>
  <rdf:seq>
    <rdf:li>RGB</rdf:li>
    <rdf:li>RGB</rdf:li>
  </rdf:seq>
</native:DocColourSpace_tab>
<native:DocCompression_tab>
  <rdf:seq>
    <rdf:li>JPEG</rdf:li>
    <rdf:li>None</rdf:li>
  </rdf:seq>
</native:DocCompression_tab>
...

</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

## Failure

HTTP 404 Not Found if the record cannot be located.

HTTP 403 Forbidden if the access is not granted to the record.

HTTP 400 Bad Request if the resource does not include the requested metadata

---

## Resource Discovery

Resources can be discovered by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of “search”.

### Parameters for Resource Discovery requests

Parameter Name	Mandatory	Default Value	Description
<b>request</b>	Yes	n/a	To request a list of matching resources the value of the request parameter must be “search”.
<b>filename</b>	No	n/a	Filename to match. This may include “right-truncation” wildcards such as “123-456*”.

### Example 1:

[http://server/optionalpath/multimedia/entry.php?request=search&filename=123-456\\*](http://server/optionalpath/multimedia/entry.php?request=search&filename=123-456*)

This request searches for resources which have a filename that starts with “123-456”.

## Response

### Success

An HTTP 200 OK response containing an XML document with a list of the EMu IRNs of the records that match. For example:

```
Content-type: text/xml
Content-Length: xxx

<?xml version="1.0" ?>
<response matches="6">
  <irn>1235</irn>
  <irn>4567</irn>
  ...
</response>
```

The IRNs returned can be used to identify the resources in subsequent requests.

If no matches are found a success response will still be sent but the *matches* attribute in the *response* tag will contain the value 0.

---

## Resource Listing

A list of the entire set of resources in the MMR in the repository can be retrieved by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of “list”.

### Parameters for Resource Listing requests

Parameter Name	Mandatory	Default Value	Description
<b>request</b>	Yes	n/a	To request a list of matching resources the value of the request parameter must be “list”.
<b>type</b>	No	“all”	The type of resources to list. Acceptable values are “external”, “shared”, or “all” (the default). Requesting “all” returns the full list of all “external” and “shared” resources. No “internal” resources are listed.

#### Example 1:

<http://server/optionalpath/multimedia/entry.php?request=list>

This is the simplest form of this request. It lists all resources in the MMR which are flagged as “external” or “shared”.

#### Example 2:

<http://server/optionalpath/multimedia/entry.php?request=list&type=external>

This request lists all resources in the MMR which are flagged as “external”.

## Response

### Success

Asn HTTP 200 OK response containing an XML document with a list of the EMu IRNs of the records that match. For example:

```
Content-type: text/xml
Content-Length: xxx

<?xml version="1.0"?>
<resources matches="2">
  <resource type="shared">
    <irn>42</irn>
    <identifier>picture.tif</identifier>
  </resource>
  <resource type="external">
```



---

```
        <irn>86</irn>
      <identifier>football.jpg</identifier>
    </resource>      ...
  </resources>
```

The IRNs returned can be used to identify the resources in subsequent requests.

If no resources can be listed a success response will still be sent but the *matches* attribute in the *resources* tag will contain the value 0.

---

## Resource Insertion

New resources can be added to the MMR. A single multimedia resource is uploaded using an HTTP POST request with a `Content-type: multipart/form-data` header.

The first part of multi-part request specifies the request type. It must contain the value “insert”.

The second part of the request contains the asset type. This can be “external” or “shared”. This part is optional and the asset type will default to “external” if not supplied.

The third part contains a 3<sup>rd</sup> party identifier. This part is also optional.

The fourth part contains a checksum. The data supplied is formatted as

```
<checksum-type>;<checksum-value>
```

The `<checksum-type>` identifies the checksum algorithm used. The only acceptable value is “crc32”. The `<checksum-value>` is the actual value of the checksum, represented as a hexadecimal string. This part is optional. If this part is supplied the file contents will be checked against the checksum. If the two do not agree an error response will be generated.

The final part of the request contains the multimedia resource itself. The content must be submitted in either raw (binary) or base64 encoded format.

### Example 1.

A request to insert an “external” asset comprising an image file named “image.jpg” with a 3<sup>rd</sup> party identifier of “abcd” would contain:

```
Content-type: multipart/form-data, boundary=AaB03x
Content-Length: xxx

--AaB03x
content-disposition: form-data; name="request"

insert
--AaB03x
content-disposition: form-data; name="type"

external
--AaB03x
content-disposition: form-data; name="identifier"

abcd
--AaB03x
content-disposition: form-data; name="checksum"

crc32;deadbeef
--AaB03x
Content-disposition: form-data; filename="image.jpg"
Content-type: image/jpeg
Content-Transfer-Encoding: base64
```

---

```
...contents of image.jpg...  
--AaB03x--
```

## Response

### Success

The response is an HTTP 200 OK response containing an XML document with the EMu IRN of the record inserted. For example:

```
Content-type: text/xml  
Content-Length: xxx  
  
<?xml version="1.0"?>  
<response>  
  <irn>1235</irn>  
</response>
```

The IRN returned can be used to identify the resources in all subsequent requests. If a 3<sup>rd</sup> party identifier is supplied it can also be used to identify the resource.

### Failure

HTTP 400 Bad request if the checksum fails

HTTP 403 Forbidden response if access is not granted to insert the resource, or if the format of the resource is excluded within the EMu Registry explicitly.

---

## Metadata Update

The metadata associated with a resource can be updated. The metadata for a single multimedia resource is updated using an HTTP POST request with a `Content-type: multipart/form-data` header.

The first part of multi-part request specifies the request type. It must contain the value “update”.

The second part of multi-part request identifies the resource to be updated. This can be either an MMR IRN or a 3<sup>rd</sup> party identifier. This part is mandatory.

The third part contains the native-format XML specifying which fields in the MMR should be modified. Note that this updates the fields in the MMR record. It does not alter any technical metadata stored in the physical resource file. Details of the native format are given in Appendix 3.

### Example 1.

To add a Title and list of Creators to the MMR record with identifier “abcd” the following request would be transmitted:

```
Content-type: multipart/form-data, boundary=AaB03x
Content-Length: xxx

--AaB03x
content-disposition: form-data; name="request"

update
--AaB03x
content-disposition: form-data; name="identifier"

abcd
--AaB03x
Content-disposition: form-data; name="metadata"
Content-type: text/xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
  <tuple>
    <atom name="MulTitle">An image of a house.</atom>
    <table name="MulCreator_tab">
      <tuple>
        <atom>John Smith</atom>
      </tuple>
      <tuple>
        <atom>Bill Wilson</atom>
      </tuple>
    </table>
  </tuple>
</table>
--AaB03x--
```

---

## Response

### Success

An HTTP 200 OK message.

### Failure

An HTTP 403 Forbidden response if the access is not granted to update the resource, or the format of the metadata supplied is invalid.

---

## Resource Replacement

The file associated with a record in the MMR can be replaced by submitting an HTTP POST request with a `Content-type: multipart/form-data` header.

The first part of multi-part request specifies the request type. It must contain the value “replace”.

The second part of multi-part request identifies the resource to be modified. This must contain the IRN of the resource. This part is mandatory.

The remaining parts are identical to those for the Resource Insertion request.

The third part contains a 3<sup>rd</sup> party identifier. This part is optional.

The fourth part contains a checksum. The data supplied is formatted as

```
<checksum-type>;<checksum-value>
```

The `<checksum-type>` identifies the checksum algorithm used. The only acceptable value is “crc32”. The `<checksum-value>` is the actual value of the checksum, represented as a hexadecimal string. This part is optional. If this part is supplied the file contents will be checked against the checksum. If the two do not agree an error response will be generated.

The final part of the request contains the multimedia resource itself. The content must be submitted in either raw (binary) or base64 encoded format.

When the resource is replaced all the technical metadata in the MMR record is updated. This includes any EXIT, IPTC or XMP data extracted from the new resource file. New resolutions are also generated and the resolutions table in the MMR record is updated. No other information in the MMR record (with the exception of record modification date and time stamps) is updated.

### Example 1.

A request to replace the resource file associated with the MMR record with an IRN of 1234 with an asset comprising an image file named “football.jpg” with a 3<sup>rd</sup> party identifier of “abcd”:

```
Content-type: multipart/form-data, boundary=AaB03x
Content-Length: xxx

--AaB03x
content-disposition: form-data; name="request"

replace
--AaB03x
content-disposition: form-data; name="irn"

1234
--AaB03x
content-disposition: form-data; name="identifier"

abcd
--AaB03x
content-disposition: form-data; name="checksum"
```

```
crc32;deadbeef
--AaB03x
Content-disposition: form-data; filename="football.jpg"
Content-type: image/jpg
Content-Transfer-Encoding: base64

...contents of image.jpg...
--AaB03x--
```

## Response

### Success

An HTTP 200 OK message.

### Failure

An HTTP 403 Forbidden response if the access is not granted to update the resource, or the format of the metadata supplied is invalid.

---

## Resource Deletion

Resources can be removed from the MMR by sending an HTTP POST or GET request to the Multimedia Web Service with a request type of “delete”.

### Parameters for Resource Deletion requests

Parameter Name	Mandatory	Default Value	Description
<b>request</b>	Yes	n/a	To remove a specific multimedia resource from the repository the parameter must be “delete”.
<b>irn</b>	No	n/a	Integer containing the EMu internal record number of the multimedia resource. A request must contain an <b>irn</b> or <b>identifier</b> , or both.
<b>identifier</b>	No	n/a	Text containing a 3 <sup>rd</sup> Party unique identifier for the resource. A request must contain an <b>irn</b> or <b>identifier</b> , or both.

The requested resource will be removed from the MMR if permitted. Note that the physical resource may not be removed at the time of deletion. Instead the record in the MMR may be flagged for deletion and removed at a later time.

As of version 1.1 a request to delete a “shared” resource will fail. This is to avoid problems where other EMu modules have attached to this resource.

### Example 1:

<http://server/optionalpath/multimedia/entry.php?request=delete&irn=1234>

The request deletes the resources associated with IRN 1234.

### Success

An HTTP 200 OK message.

### Failure

An HTTP 403 Forbidden response if the access is not granted to update the resource, or the format of the metadata supplied is invalid.



## Appendix 1: Authentication/Security

Access to the Multimedia Web Service is controlled by the web server. If the web server used is Apache this would be handled using standard Basic Authentication configured by an .htaccess file.

For more secure transmission of HTTP requests the web server can be configured to accept HTTP requests across an SSL connection only.

Within the Multimedia Web Service, processes are run as a non-privileged EMu user, and access to records is controlled internally using Texpress's record level security.

---

## Appendix 2: Supported Image Formats

Image formats supported for requests.

ART	PFS: 1st Publisher
AVI	Microsoft Audio/Visual Interleaved
AVS	AVS X image
BMP	Microsoft Windows bitmap
CGM	Computer Graphics Metafile
CIN	Kodak Cineon Image Format
CMYK	Raw cyan, magenta, yellow, and black samples
CMYKA	Raw cyan, magenta, yellow, black, and alpha samples
CR2	Canon Digital Camera Raw Image Format
CRW	Canon Digital Camera Raw Image Format
CUR	Microsoft Cursor Icon
CUT	DR Halo
DCM	Digital Imaging and Communications in Medicine (DICOM) image
DCR	Kodak Digital Camera Raw Image File
DCX	ZSoft IBM PC multi-page Paintbrush image
DIB	Microsoft Windows Device Independent Bitmap
DJVU	
DNG	Digital Negative
DOT	Graph Visualization
DPX	SMPTE Digital Moving Picture Exchange 2.0 (SMPTE 268M-2003)
EMF	Microsoft Enhanced Metafile (32-bit)
EPDF	Encapsulated Portable Document Format

EPI	Adobe Encapsulated PostScript Interchange format
EPS	Adobe Encapsulated PostScript
EPS2	Adobe Level II Encapsulated PostScript
EPS3	Adobe Level III Encapsulated PostScript
EPSF	Adobe Encapsulated PostScript
EPSI	Adobe Encapsulated PostScript Interchange format
EPT	TIFF preview
EXR	High dynamic-range (HDR) file format developed by Industrial Light & Magic
FAX	Group 3 TIFF
FIG	FIG graphics format
FITS	Flexible Image Transport System
FPX	FlashPix Format
GIF	CompuServe Graphics Interchange Format
GPLT	Gnuplot plot files
GRAY	Raw gray samples
HPGL	HP-GL plotter language
HTML	Hypertext Markup Language with a client-side image map
ICO	Microsoft icon
INFO	Format and characteristics of the image
JBIG	Joint Bi-level Image experts Group file interchange format
JNG	Multiple-image Network Graphics
JP2	JPEG-2000 JP2 File Format Syntax
JPC	JPEG-2000 Code Stream Syntax
JPEG	Joint Photographic Experts Group JFIF format
MAN	Unix reference manual pages

MAT	MATLAB image format
MIFF	Magick image file format
MONO	Bi-level bitmap in least-significant-byte first order
MNG	Multiple-image Network Graphics
MPEG	Motion Picture Experts Group file interchange format (version 1)
M2V	Motion Picture Experts Group file interchange format (version 2)
MPC	Magick Persistent Cache image file format
MRW	Sony (Minolta) Raw Image File
MSL	Magick Scripting Language
MTV	MTV Raytracing image format
MVG	Magick Vector Graphics.
NEF	Nikon Digital SLR Camera Raw Image File
ORF	Olympus Digital Camera Raw Image File
OTB	On-the-air Bitmap
P7	Xv's Visual Schnauzer thumbnail format
PALM	Palm pixmap
PAM	Common 2-dimensional bitmap format
PBM	Portable bitmap format (black and white)
PCD	Photo CD
PCDS	Photo CD
PCL	HP Page Control Language
PCX	ZSoft IBM PC Paintbrush file
PDB	Palm Database ImageViewer Format
PDF	Portable Document Format
PEF	Pentax Electronic File

PFA	Postscript Type 1 font (ASCII)
PFB	Postscript Type 1 font (binary)
PFM	Portable float map format
PGM	Portable graymap format (gray scale)
PICON	Personal Icon
PICT	Apple Macintosh QuickDraw/PICT file
PIX	Alias/Wavefront RLE image format
PNG	Portable Network Graphics
PNM	Portable anymap
PPM	Portable pixmap format (color)
PS	Adobe PostScript file
PS2	Adobe Level II PostScript file
PS3	Adobe Level III PostScript file
PSD	Adobe Photoshop bitmap file
PTIF	TIFF
PWP	Seattle File Works multi-image file
RAD	Radiance image file
RAF	Fuji CCD-RAW Graphic File
RGB	Raw red, green, and blue samples
RGBA	Raw red, green, blue, and alpha samples
RLA	Alias/Wavefront image file
RLE	Utah Run length encoded image file
SCT	Scitex Continuous Tone Picture
SFW	Seattle File Works image
SGI	Irix RGB image
SHTML	Hypertext Markup Language client-side image map

SUN	SUN Rasterfile
SVG	Scalable Vector Graphics
TGA	Truevision Targa image
TIFF	Tagged Image File Format
TIM	PSX TIM file
TTF	TrueType font file
TXT	Raw text file
UIL	X-Motif UIL table
UYVY	Interleaved YUV raw image
VICAR	VICAR rasterfile format
VIFF	Khoros Visualization Image File Format
WBMP	Wireless bitmap
WMF	Windows Metafile
WPG	Word Perfect Graphics File
X	display or import an image to or from an X11 server
XBM	X Windows system bitmap, black and white only
XCF	GIMP image
XPM	X Windows system pixmap
XWD	X Windows system window dump
X3F	Sigma Camera RAW Picture File
YCbCr	Raw Y, Cb, and Cr samples
YCbCrA	Raw Y, Cb, Cr, and alpha samples
YUV	CCIR 601 4:1:1

## Appendix 3: Native Format

The “native” request format can be used to update all or part of the database record associated with the multimedia asset.

The format used is the native export format used by EMu’s underlying database system Texpress for data export.

Documents submitted in “native” format are structured as follows:

1. A standard XML header
2. A top-level “table” tag.
3. An inner-level “tuple” tag.
4. Repeated “column-level” tags within the “tuple” tag. The column-level tags have the following format:
  - a. Single-valued columns are represented by simple text enclosed in an “atom” tag.
  - b. Multi-valued columns are enclosed in a “table” tag. The “table” tag itself includes multiple “tuple” tags, just as the outer-level data does. This is a recursive structure.

Each column-level tag includes the name of the column as an attribute.

An example of the “native” format is shown below.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<table>
  <tuple>
    <atom name="MulTitle">Footballers I Have Known</atom>
    <atom name="DetCoverage">AFL</atom>
    <atom name="DetRights">Museum Victoria (2009)</atom>
    <atom name="SecAssetType">shared</atom>
  </tuple>
</table>
```

The following table describes each of the columns in the MMR database:

Column	Description
ChaAudience_tab	The intended audience for the multimedia resource.
ChaAudioBitsPerSample	The precision of the digital audio sample is determined by the number of bits per sample, typically 8 or 16 bits.
ChaAudioNumberOfChannels	The number of simultaneous sounds possible in the audio file.
ChaAudioSamplesPerSec	The number of samples of a sound that are taken per second to

	represent the event digitally.
ChaFileSize	The size of the multimedia resource as a stored file, in bytes.
ChaImageColorDepth	The number of colours in the image, e.g. 2 (B&W).
ChaImageHeight	The height of the image in mm, inches, etc.
ChaImageResolution	The number of pixels in the image, e.g. 640 x 480.
ChaImageWidth	The width of the image in mm, inches, etc.
ChaMediaForm	The format of the multimedia resource, e.g. JPG, MPG, etc.
ChaRepository_tab	The storage facility that houses the multimedia resource.
ChaVideoFilmLength	The length of the video in minutes and seconds.
DetContributor_tab	Lists other parties who have contributed to the creation of the multimedia resource.
DetCoverage	The extent or scope of the content of the resource.
DetDate0	
DetLanguage_tab	The primary language used in the title or content of the multimedia resource.
DetPublisher	The publisher of the multimedia resource, i.e. the party responsible for making the resource publicly available.
DetRelation_tab	A reference to a related resource.
DetResourceType	The type of multimedia resource, e.g. image, sound, etc.
DetRights	Rights associated with the multimedia resource, e.g. copyright, access privileges, etc.



DetSource	The person or organisation who provided the multimedia resource.
DetSubject_tab	The subject matter of the multimedia resource, as chosen from prescribed standards.
MulCreator_tab	The party responsible for creating the multimedia item.
MulDescription	An account of the content of the resource, for instance a description of an image, an abstract or table of contents of a text document. (Dublin Core Metadata standard)
MulDocumentType	
MulIdentifier	The file name and extension for the multimedia resource.
MulMimeFormat	The format of the multimedia resource, e.g. JPG, MPG, MS Word document, etc. (MIME is an abbreviation for Multipurpose Internet Mail Extensions)
MulMimeType	The type of multimedia resource, e.g. image, video, document, etc. (MIME is an abbreviation for Multipurpose Internet Mail Extensions)
MulTitle	The name by which the multimedia item is known.
NotNotes	Text notes that are not already contained in a defined field.
SecAssetType	The type of asset. This will contain the value “external”, “shared” or “internal”.
SecRecordStatus	Status of the record. The default value is blank, which indicates an active record. A common alternative is 'Retired'.



## EMu Documentation

# Encrypted Connections

Document Version 1

**EMu Version 4.0.03**





# Contents

<b>SECTION 1</b>	<b>Encrypted Connections</b>	<b>1</b>
	How it works	2
	Requirements	4
	EMu server setup	5
	Important!	5
	EMu client setup	6
	TexJDBC setup	7
<b>SECTION 2</b>	<b>Generating certificates</b>	<b>9</b>
	Self signed	10
	Step 1: Generate a private key	10
	Step 2: Generate public digital certificate	10
	Step 3: Installing the files	12
	Root signed	13
	Step 1: Generate a private key	13
	Step 2: Generate a certificate signing request	13
	Step 3: Installing the files	14
	Chain signed	15
	Step 1: Generate a self signed CA certificate	16
	Step 2: Generate a private key	16
	Step 3: Generate a certificate signing request	16
	Step 4: Sign the certificate with your CA certificate	17
	Step 5: Creating the certificate chain	17
	Step 6: Installing the files	18
<b>SECTION 3</b>	<b>Configuring ciphers</b>	<b>19</b>
	Server	20
	TexAPI/texapi.pm	20
	TexJDBC	21
<b>SECTION 4</b>	<b>Common Errors</b>	<b>23</b>
	Client does not support SSL	24
	Server certificates not installed	24
	Server/Client protocol error	24
	Cannot verify certificate	25
	Bad host name	26
	<b>Index</b>	<b>27</b>



## SECTION 1

# Encrypted Connections

When using EMu over public networks it may be desirable to encrypt all data transferred between the EMu client and server. EMu 4.0.03 has been extended to support an encrypted connection between the client and server programs. The encrypted connection uses TLS v1.0 (Transport Layer Security) for the transmission of data, ensuring data integrity and security. The use of data encryption is optional and is not required for internal networks where the risk of unauthorised access to data is minimal. The EMu server may also be configured to accept connections only from clients who request data encryption. This provides system administrators with the ability to enforce data encryption, or not, as required.

---

## How it works

The TLS v1.0 protocol uses Public Key Infrastructure (PKI). A key is a sequence of bytes, normally 40, 56, 64, 128 or 265 bits in length, which is used by a cipher (an encryption algorithm) to encrypt data. Using the same cipher with different keys will produce different output. Hence, the key is used to "lock" the encrypted data. In order to unlock the data, that is decrypt it, the right key is required. With public/private key infrastructure two keys are generated, a public key and a private key. Data encrypted with the public key requires the private key in order to be decrypted and data encrypted with the private key requires the public key in order to be decrypted. In other words the keys are symmetrical.

The private key must be kept safe to ensure data privacy. If someone has both the private and public keys, they can decrypt the data and so compromise data security. The public key may be made available to anyone without compromising security as the private key is required to decrypt data encrypted using the public key. The public key is wrapped in a digital certificate, which consists of:

- **Public Key**
- **Subject** - details about the owner of the certificate.
- **Serial Number** - unique number used to identify the certificate.
- **Issuer** - details about who verified and issued the certificate.
- **Valid Dates** - start and end dates for which the certificate is valid.
- **Key Usage** - purpose(s) for which the public key may be used.
- **Thumbprint** - a check-sum to ensure the certificate has not been modified.

In order for a digital certificate to be valid it is necessary to verify the details of the Issuer. A small number of companies are allowed to issue valid and verifiable certificates. When you want a public digital certificate you approach one of these companies and they verify your details before issuing your public digital certificate. They *sign* your certificate with their own private key, making them the Issuer. In order to read your digital certificate you need the Issuer's public key. The key is embedded in their digital certificate which is available freely. These Issuer public digital certificates are known as Certificate Authorities (CA). In order to verify your certificate it is necessary to determine who the Issuer is and locate their CA certificate. Using the public key in their certificate your certificate can now be decrypted and the check-sum verified to ensure it has not been altered.

The private key is stored on the EMu server. Login access to the EMu server is generally restricted to user `emu`, hence the key is not available for general access. The public digital certificate is also stored on the EMu server. All CA certificates are stored with the EMu client.

When a connection is initiated the EMu server sends its public digital certificate to the EMu client. The client uses the CA certificates stored locally to verify that the certificate is valid. The server's public digital certificate contains the full host name of the EMu server machine. The EMu client checks the host name against the machine to ensure it has not connected to a rogue server.

Once the EMu client has verified the server's public digital certificate it sends a random number to the server encrypted using the public key in the server's digital certificate. As the server is the only machine with the private key it can decrypt the random number. The number is used as a key to a cipher (an encryption algorithm). The cipher uses the key to encrypt all data between the client and server. As the client and server are the only two machines which know the encryption key, data security and integrity is guaranteed.

The complete steps required to establish an encrypted connection are:

- The EMu client connects to the EMu server requesting a secure connection. The client provides a list of ciphers it supports.
- The EMu server selects the strongest cipher it supports from the client's list and notifies the client.
- The EMu server sends its public digital certificate to the client. The certificate contains the server's host name, the Issuer used to create the certificate and the server's public encryption key.
- The EMu client looks up the CAs on its machine and verifies that the server's certificate is valid.
- The EMu client generates a random number and encrypts it with the server's public encryption key. The random number is sent to the server.
- The EMu server decrypts the random number using its private encryption key (known only by the server).
- The random number is used as a key for the selected cipher. All data transferred is now encrypted using the agreed cipher.

EMu allows public digital certificates to be:

- **Self signed**  
A certificate that is verified by itself, that is the Issuer certificate is the same as the certificate itself. Self signed certificates allow institutions to generate their own digital certificates without the need to have them authenticated by an outside authority. In order for the certificate to be verified the self signed certificate must exist on both the client and server machines, the client version being the CA certificate.
- **Root signed**  
A certificate verified by one of a select set of "root" certificates. A root certificate is distributed as part of the public key infrastructure and can be installed on client machines to provide certificate verification.
- **Chain signed**  
A certificate is verified by its Issuer certificate. The issuer certificate itself is verified by its issuer certificate and so on until either a root or self signed certificate is found.

EMu allows both the client and server machines to define a list of ciphers they will support. When a connection is created the strongest (that is hardest to break) cipher supported by both the client and server is selected. System administrators may restrict the ciphers available on the server, forcing the client to use very strong encryption only (e.g. 256 bit ciphers).



---

## Requirements

Support for encrypted connections between the EMu client and server requires the following software versions:

- Texpress 8.2.009 or later
- EMu 4.0.03 or later
- TexAPI 6.0.02 or later
- TexJDBC 0.9.6 or later

If any software is earlier than the version listed above, EMu will drop back to using unencrypted connections. In order to use encrypted connections your System Administrator must create the required keys (public/private) and public digital certificate and install them on the EMu server. The CA certificates may also need to be installed on the EMu client.

---

## EMu server setup

The default installation of the EMu server does not have encrypted connections enabled. The following files need to exist to enable encrypted connections:

- `server.key`            EMu server's private key
- `server.crt`            EMu server's public digital certificate
- `ciphers`                list of ciphers the server will support

These files must be placed in a directory called `certs` under the `Texpress etc` directory. For a standard EMu installation this corresponds to:

```
$EMUHOME/texpress/8.2/etc/certs.
```

If the files are not found, EMu will not attempt to use encrypted connections.

See *Generating certificates* (page 9) for details on how to create `server.key` and `server.crt`.

See *Configuring ciphers* (page 19) for details about configuring `ciphers`.

### Important!

The permissions on the `server.key` file should restrict access to read-only by user `root`. All other permissions should be disabled, that is the file owner should be `root` and the permissions should be `r-----`. If these permissions are not set, it is possible that someone may access the file and so compromise the integrity of the system!

As described in *Requirements* (page 4), the EMu server will drop back to an unencrypted connection if versions earlier than 4.0.03 of the EMu client are used. If you want to enforce encrypted connections the `-s` option should be added to the `texserver` command configured in `inetd/xinetd/svcs`.

For example, the following `inetd` entry will accept encrypted connections only:

```
emuclient stream tcp nowait root /home/emu/client/bin/emurun  
emurun texserver -aemu -i -L -t60 -s
```

---

## EMu client setup

The EMu client needs to verify the EMu server's public digital certificate for an encrypted connection to be established. In order to verify the certificate the client must be able to locate a valid CA (Certificate Authority) certificate for the Issuer of the server's certificate. In the case of a certificate chain this must be the Issuer of the first or "root" certificate. There are two locations used to hold CA certificates:

1. The first is on the EMu server and is used by programs using either TexAPI or `texapi.pm` (a perl based interface to TexAPI). These programs include web services and IMu.
2. The second is on the EMu client's machine and is used by the Windows client.

CA certificates stored on the EMu server must be placed in the `$EMUPATH/etc/certs` directory. The certificates should be stored in files with a `.crt` extension. CA bundles are also supported. A bundle is simply the concatenation of a number of certificates into one file. An optional `ciphers` file may also exist in the certificates directory. If it exists, it should list the ciphers the EMu client is willing to support.

See *Configuring ciphers* (page 19) for details about configuring ciphers.

CA certificates required by the EMu Windows client should be stored in the `certs` directory under the location where the EMu executable (`emu.exe`) is installed. As with CA certificates stored on the EMu server, the files must have a `.crt` extension and CA bundles are supported. A `ciphers` file may also be supplied defining the ciphers the client is willing to use.

## TexJDBC setup

As with the EMu client, TexJDBC needs to be able to verify the EMu server's public digital certificate. The required CA certificates must be stored in an accessible Java Key Store (JKS). The system key store is located at `$JAVA_HOME/jre/lib/security/cacerts`. A key store may have a password associated with it. The password allows the integrity of the stored certificates to be checked when they are accessed. The password is not required to access the key store. The location of the key store used may be altered by setting the following system properties:

`javax.net.ssl.trustStore`

The location of the Java Key Store file containing the CA certificates to use for verifying the server's certificate.

`javax.net.ssl.trustStorePassword`

The password to use to check the integrity of the Java Key Store.

For example, if you want to use a key store located at `/home/emu/etc/certs/cacerts` with a password of `emustore`, you could invoke java using:

```
java -Djavax.net.ssl.trustStore=/home/emu/etc/certs/cacerts -
Djavax.net.ssl.trustStorePassword=emustore -jar application.jar
```

The location and password of the key store may also be specified using the `trustStore` and `trustStorePassword` connection properties:

```
Properties props = new Properties();
props.setProperty("trustStore", "/home/emu/etc/certs/cacerts");
props.setProperty("trustStorePassword", "emustore");
...
Connection conn =
DriverManager.getConnection("jdbc:texpress:socket", props);
```

The `keytool` command should be used to import a CA certificate into a java key store:

```
keytool -importcert -alias alias -file certfile -storetype JKS -
keystore keystore
```

where:

<i>alias</i>	is an arbitrary unique name used to define the certificate within the key store
<i>certfile</i>	is the file containing the CA certificate
<i>keystore</i>	is the location of the key store file into which the certificate is imported

If *keystore* does not exist, a new key store is created. You will be prompted for the password if the key store already exists, otherwise you will be asked to set the password for the key store created.

To list the certificates in a key store use:

```
keytool -list -v -keystore keystore
```

where *keystore* is the location of the key store file.

## SECTION 2

# Generating certificates

In this section we will look at how to generate a private/public key pair and how the public digital signature is created for the public key. As mentioned earlier EMu provides support for the following public certificates:

- **Self signed**
- **Root signed**
- **Chain signed**

Each of these types will be examined and appropriate commands provided for OpenSSL (via the `openssl` command).

---

## Self signed

A self signed certificate is one where the certificate Issuer is the same as the certificate Subject. In other words the certificate is used to verify itself. In order for the certificate to be trusted the certificate must be included with the client CA certificates. Self signed certificates are used when you only need one certificate (for example if you only have one EMu server and all clients connect to that server). As the certificate is self signed it has not been verified by an external agency and so should be used for internal use only. The steps required to generate the required files are:

### Step 1: Generate a private key

Create your private key. The key is a 1024 bit RSA key stored in PEM (Privacy Enhanced Mail, a Base64 encoding of the key) format. It is readable as ASCII text:

```
openssl genrsa -out server.key 1024
Generating RSA private key, 1024 bit long modulus
..+++++
...+++++
e is 65537 (0x10001)
```

The file `server.key` now contains your private key. Remember to keep it safe!

### Step 2: Generate public digital certificate

Using the private key generated in the first step the public digital certificate is generated. A number of questions will be asked as part of the creation process. It is important that the Common Name (CN) is set to the full host name of your EMu server machine (e.g. `emu.institution.org`). Support for wild card host names is provided by replacing any leading component of the name with an asterisk (e.g. `*.institution.org`, or `*.org`):

```
openssl req -new -x509 -key server.key -out server.crt -days 1095
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AU
State or Province Name (full name) [Some-State]:Victoria
Locality Name (eg, city) []:Melbourne
Organization Name (eg, company) [Internet Widgits Pty Ltd]:KE
Software Pty Ltd
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:*.mel.kesoftware.com
Email Address []:info@mel.kesoftware.com
```

The resulting public digital certificate will be stored in PEM format in `server.crt`.

You can view the contents of the public certificate using:

### **openssl x509 -text -in server.crt**

Certificate:

Data:

```

    Version: 3 (0x2)
    Serial Number:
        f5:02:b4:7d:c3:5b:ad:a7
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=AU, ST=Victoria, L=Melbourne, O=KE Software Pty
Ltd, CN=*.mel.kesoftware.com/emailAddress=info@mel.kesoftware.com
    Validity
        Not Before: Nov 19 11:47:46 2010 GMT
        Not After : Nov 18 11:47:46 2013 GMT
    Subject: C=AU, ST=Victoria, L=Melbourne, O=KE Software Pty
Ltd, CN=*.mel.kesoftware.com/emailAddress=info@mel.kesoftware.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
            Modulus (1024 bit):
                00:c4:c9:0f:04:8f:cd:98:5f:d9:c6:3b:00:54:b2:
                88:07:9b:06:4c:ea:f2:41:74:a3:68:7d:16:2a:de:
                cf:bb:cf:73:d5:97:f2:d8:4e:38:b1:7d:a8:94:48:
                5b:4a:fd:92:3b:45:8c:1b:ce:85:e5:18:2e:c1:60:
                db:4d:09:32:46:72:b4:a3:f1:f8:ab:96:4a:db:a5:
                4c:32:6d:83:ee:f9:02:4e:8f:f1:8b:ba:b4:62:b6:
                29:00:97:fb:3b:06:73:a2:56:5f:04:2c:79:3e:2e:
                f8:1b:eb:f5:8b:a6:cf:6b:56:bd:74:16:cb:53:a6:
                91:dd:ec:af:7a:77:40:b0:e5
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Subject Key Identifier:

F6:72:9C:A4:91:C2:E2:51:70:26:05:FE:06:C3:E4:E9:4F:AF:A0:D5
        X509v3 Authority Key Identifier:

keyid:F6:72:9C:A4:91:C2:E2:51:70:26:05:FE:06:C3:E4:E9:4F:AF:A0:D5
        DirName:/C=AU/ST=Victoria/L=Melbourne/O=KE
Software Pty
Ltd/CN=*.mel.kesoftware.com/emailAddress=info@mel.kesoftware.com
        serial:F5:02:B4:7D:C3:5B:AD:A7

    X509v3 Basic Constraints:
        CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
        4c:89:a2:57:d2:3b:3a:11:70:63:41:56:4e:b6:36:8e:28:c5:
        29:d7:7d:22:86:c4:43:90:4f:74:d1:31:32:7f:39:d8:f3:20:
        80:05:53:99:cd:17:28:b8:16:3b:a3:9a:84:ae:2c:08:f5:b0:
        11:6a:d5:ba:42:81:9d:e7:36:8f:39:9d:b4:15:13:52:23:fc:
        37:f6:5c:88:39:f9:9b:d1:e0:06:82:3f:e2:56:a3:f3:83:55:
        4d:8b:7c:69:a3:bc:fb:3a:66:18:f2:07:67:bc:39:54:28:c3:
        eb:3e:5c:d9:89:d8:ea:c7:d2:c4:fe:87:ee:24:e0:ce:c0:4f:
        d1:e7
-----BEGIN CERTIFICATE-----
MIIDtTCCAx6gAwIBAgIJAPUctH3DW62nMA0GCSqGSIb3DQEBBQUAMIGZMQswCQYD
VQQGEwJBVTERMA8GA1UECBMlVmljdG9yaWExEjAQBgNVBACTCU1lbGJvdXJuZTEc

```



MB0GA1UEChMTS0UgU29mdHdhcmUgUHR5IEx0ZDEedMBsGA1UEAxQUKi5tZWwua2Vz  
b2Z0d2FyZS5jb20xJjAkBgkqhkiG9w0BCQEWf2luZm9AbWVsLmtlc29mdHdhcmUu  
Y29tMB4XDTEwMTEwOTExNDc0Nl0xDTExMTEwOTExNDc0Nl0wGzKxCzAJBgNVBAYT  
AkFVMREwDwYDVQQIEWhaWwN0b3JpYTESMBA1UEBxMjTWVsYm91cm51MRwwGgYD  
VQQKEzNLRSBtb2Z0d2FyZSBQdHkgTHRkMR0wGwYDVQQQDFBqLm11bC5rZXNvZnR3  
YXJlLmNvbTEuMCQGCsGSIb3DQEJARYXaW5mb0BtZWwua2Vzb2Z0d2FyZS5jb20w  
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMTJDwSPzZh2cY7AFSYiAebBkzq  
8kF0o2h9Firez7vPc9WX8th0OLF9qJRIW0r9kjtFjBvOheUYLSfg200JmKZytKPX  
+KuWStu1TDJtg+75Ak6P8Yu6tGK2KQCX+zsGc6JWXwQseT4u+Bvr9Yumz2tWvXQW  
yl0mkd3sr3p3QLDLAgMBAAGjggEBMIH+MB0GA1UdDgQWBbT2cPykKcLiUXAmBf4G  
w+TpT6+glTCBzgYDVR0jBIHGMiHDgBT2cPykKcLiUXAmBf4Gw+TpT6+glAGBn6SB  
nDCBmTELMaKGA1UEBhMCQVUxETAPBgNVBAgTCTFpY3RvcmlhMRiWEAYDVQQHEwln  
ZWxib3VybmUxHDAaBgNVBAoTE0tFIFNvZnR3YXJlIFB0eSBMdGQxHTAbBgNVBAMU  
FCoubWVsLmtlc29mdHdhcmUuY29tMSYwJAYJKoZIhvcNAQkBFhdpbmZvQG11bC5r  
ZXNvZnR3YXJlLmNvbYIJApuCTH3DW62nMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcN  
AQEFBQADgYEATImiV9I7OhFwY0FWTrY2jiJfKdd9IobEQ5BPdNexMn852PMggAVT  
mc0XKLgW06OahK4sCPWwEWrvukKBnec2jzmdtBUTUiP8N/ZciDn5m9HgBoI/41aj  
84NVTYt8aa0+ZpmGPIHz7w5VCjD6z5c2YnY6sfSxP6H7iTgzsBP0ec=  
-----END CERTIFICATE-----

### Step 3: Installing the files

We now have the two files we require:

- `server.key` - private key (must be kept safe)
- `server.crt` - self signed public digital certificate

On the EMu server these two files should be placed in the directory `$TEXHOME/etc/certs` where `$TEXHOME` contains the location where Texpres is installed. Suitable permissions should be set on the private key file:

```
mv server.key server.crt $TEXHOME/etc/certs
chmod 644 $TEXHOME/etc/certs/server.crt
su root
Password:
chown root $TEXHOME/etc/certs/server.key
chmod 400 $TEXHOME/etc/certs/server.key
exit
```

Next, the public certificate should be stored on the EMu server for use by API based programs (TexAPI and texql.pm):

```
cp $TEXHOME/etc/certs/server.crt $EMUPATH/etc/certs
chmod 644 $EMUPATH/etc/certs/server.crt
```

Finally, on EMu Windows client machines the `server.crt` file must be placed in a directory called `certs` in the same location as the EMu executable (`emu.exe`).

Now that all the required files are in the right place it is possible to connect using encrypted connections. As mentioned earlier, the `-s` option for `texserver` (page 5) may be used to enforce secure connections.

## Root signed

A root signed certificate is a public digital certificate created and verified by an external entity. You forward a certificate request to the external entity and they return the signed public digital certificate. Root entities distribute their CA certificates (really just a special form of self signed certificate) for all to use, allowing any certificate signed by them to be verified. Root signed certificates are used when you need a verifiable certificate for external use.

The steps required to generate the required files are:

### Step 1: Generate a private key

Create your private key. The key is a 1024 bit RSA key stored in PEM (Privacy Enhanced Mail, a Base64 encoding of the key) format. It is readable as ASCII text:

```
openssl genrsa -out server.key 1024
Generating RSA private key, 1024 bit long modulus
..++++++
...++++++
e is 65537 (0x10001)
```

The file `server.key` now contains your private key.

### Step 2: Generate a certificate signing request

We use the private key generated in the first step to create a certificate signing request (CSR). The file generated will contain the Subject information without an Issuer being assigned, that is a certificate that has not yet been signed. The resulting file, `server.csr` is then sent to an external entity for signing (e.g. Verisign).

```
openssl req -new -key server.key -out server.csr
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AU
State or Province Name (full name) [Some-State]:Victoria
Locality Name (eg, city) []:Melbourne
Organization Name (eg, company) [Internet Widgits Pty Ltd]:KE
Software Pty Ltd
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:*.mel.kesoftware.com
Email Address []:info@mel.kesoftware.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Once the external entity has verified the Subject information in the request they will generate a public digital certificate and return it to you. You should save the certificate in a file called `server.crt`.

You can view the contents of the certificate signing request using:

```
openssl req -in server.csr -noout -text
```

```
Certificate Request:
```

```
  Data:
```

```
    Version: 0 (0x0)
```

```
    Subject: C=AU, ST=Victoria, L=Melbourne, O=KE Software Pty  
Ltd, CN=*.mel.kesoftware.com/emailAddress=info@mel.kesoftware.com
```

```
    Subject Public Key Info:
```

```
      Public Key Algorithm: rsaEncryption
```

```
      RSA Public Key: (1024 bit)
```

```
        Modulus (1024 bit):
```

```
          00:c4:c9:0f:04:8f:cd:98:5f:d9:c6:3b:00:54:b2:  
          88:07:9b:06:4c:ea:f2:41:74:a3:68:7d:16:2a:de:  
          cf:bb:cf:73:d5:97:f2:d8:4e:38:b1:7d:a8:94:48:  
          5b:4a:fd:92:3b:45:8c:1b:ce:85:e5:18:2e:c1:60:  
          db:4d:09:32:46:72:b4:a3:f1:f8:ab:96:4a:db:a5:  
          4c:32:6d:83:ee:f9:02:4e:8f:f1:8b:ba:b4:62:b6:  
          29:00:97:fb:3b:06:73:a2:56:5f:04:2c:79:3e:2e:  
          f8:1b:eb:f5:8b:a6:cf:6b:56:bd:74:16:cb:53:a6:  
          91:dd:ec:af:7a:77:40:b0:e5
```

```
        Exponent: 65537 (0x10001)
```

```
    Attributes:
```

```
      a0:00
```

```
    Signature Algorithm: sha1WithRSAEncryption
```

```
      ae:e0:68:b8:fe:56:53:5e:f4:f4:e0:8d:19:2c:62:ee:ee:83:  
      01:d2:8d:55:d0:2d:18:b8:18:0a:f2:5b:c4:a5:da:75:fd:ca:  
      87:69:cd:3f:2e:7c:9e:a2:c2:b7:b1:4a:bd:85:2e:24:84:8d:  
      cc:81:64:9d:0c:a4:ad:c4:c5:54:4d:cf:22:dc:08:51:3f:ed:  
      6d:45:d6:91:e3:a6:c0:7e:2e:f0:0f:9e:be:70:ef:6a:f8:2c:  
      93:59:8d:90:ca:23:c4:07:f9:ae:2c:09:03:fd:cf:43:d6:b7:  
      8c:2e:48:96:28:98:5c:c3:e8:66:55:b3:4a:8d:bb:c8:d0:bb:  
      c8:41
```

### Step 3: Installing the files

The process for installing the two files `server.key` (private key) and `server.crt` (public certificate) is exactly the same as for a self signed certificate (page 12).

---

## Chain signed

A chain signed certificate is a certificate that is not self signed and is not root signed. In order for the certificate to be verified, the Issuer of the certificate is verified, then the Issuer of the Issuer certificate is verified and so on until either a self signed or root signed certificate is encountered. If the top certificate is root signed, then the chain signed certificate has the same level of verification as if the certificate had been root signed directly. If the top certificate is self signed, then the level of verification is the same as for any other self signed certificate. Chain signed certificates are used where you will be generating multiple certificates and you only want to distribute one CA certificate to verify them all. The only CA required is the top level self signed or root signed certificate.

The steps below outline how to produce a self signed CA certificate that can then be used to sign all other certificates generated. If you require a root signed CA certificate, you need to generate a certificate signing request (as per the previous section) and have the external entity generate the CA certificate.

## Step 1: Generate a self signed CA certificate

The first step creates a self signed CA certificate. The CA certificate is the "root" certificate used to sign (and hence verify) all other certificates we generate. The public digital certificate of the CA certificate needs to be installed on client machines. We only need to generate the CA certificate once.

```
echo "01" > ca.srl
openssl req -new -x509 -nodes -extensions v3_ca -keyout ca.key -out ca.crt -
days 365
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AU
State or Province Name (full name) [Some-State]:Victoria
Locality Name (eg, city) []:Melbourne
Organization Name (eg, company) [Internet Widgits Pty Ltd]:KE
Software Pty Ltd
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:KE Software CA certificate
Email Address []:info@mel.kesoftware.com
```

## Step 2: Generate a private key

Once we have the CA certificate we can generate a new certificate. The first step is to generate the private key:

```
openssl genrsa -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.++++++
..++++++
e is 65537 (0x10001)
```

## Step 3: Generate a certificate signing request

We use the private key generated in the previous step to create a certificate signing request (CSR). The file generated will contain the Subject information without an Issuer being assigned, that is a certificate that has not been signed. Make sure Common Name is set to the host name of your EMu server.

**openssl req -new -key server.key -out server.csr**

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:AU

State or Province Name (full name) [Some-State]:Victoria

Locality Name (eg, city) []:Melbourne

Organization Name (eg, company) [Internet Widgits Pty Ltd]:KE Software Pty Ltd

Organizational Unit Name (eg, section) []:

Common Name (eg, YOUR name) []:\*.mel.kesoftware.com

Email Address []:info@mel.kesoftware.com

Please enter the following 'extra' attributes to be sent with your certificate request

A challenge password []:

An optional company name []:

**Step 4: Sign the certificate with your CA certificate**

Using our CA certificate we sign the CSR to produce our public digital certificate:

**openssl x509 -CA ca.crt -CAkey ca.key -CAserial ca.srl -req -in server.csr -out server.crt -days 365**

Signature ok

subject=/C=AU/ST=Victoria/L=Melbourne/O=KE Software Pty

Ltd/CN=\*.mel.kesoftware.com/emailAddress=info@mel.kesoftware.com

Getting CA Private Key

**Step 5: Creating the certificate chain**

A certificate chain is simply the concatenation of all the signing public certificates from the certificate just generated to the root certificate (there may be any number of intermediate certificates). As we only have one CA in the chain in this example we do not need to concatenate the certificates, however if more than one CA has been used all certificates in the chain must be placed in one file. While not required for this example we will concatenate the certificates anyway (it does not hurt):

**cat server.crt ca.crt > chain.crt**

## Step 6: Installing the files

First we install the private key and chain certificates on the EMu server:

```
mv server.key $TEXHOME/etc/certs/server.key
mv chain.crt $TEXHOME/etc/certs/server.crt
chmod 644 $TEXHOME/etc/certs/server.crt
su root
Password:
chown root $TEXHOME/etc/certs/server.key
chmod 400 $TEXHOME/etc/certs/server.key
exit
```

Next, the public CA certificate should be stored on the EMu server for use by API based programs (TexAPI and texql.pm):

```
cp ca.crt $EMUPATH/etc/certs
chmod 644 $EMUPATH/etc/certs/ca.crt
```

Finally, on EMu Windows client machines the `ca.crt` file must be placed in a directory called `certs` in the same location as the EMu executable (`emu.exe`).

Now that the CA certificate is installed on the EMu clients there is no need to add any further files when generating new certificates signed by the same CA certificate.

## SECTION 3

# Configuring ciphers

When an encrypted connection is formed the client and server negotiate to determine the highest level of encryption on which they can both agree. A list of ciphers may be set for the server and/or client allowing System Administrators to enforce a certain level of encryption. As the strongest cipher is chosen as part of the connection negotiation it is not necessary to restrict the list of ciphers used in most cases. The ciphers to use can be set at three levels:

- Server
- TexAPI/texapi.pm
- TexJDBC



---

## Server

The list of ciphers the server will support is found in a file called `ciphers`. The file is located in the `$TEXHOME/etc/certs` directory. The format of the file is a colon separated list of cipher names. For details on what ciphers are supported and the exact format of the setting see the Ciphers section of the OpenSSL documentation.

For example, to enforce the use of MD5 ciphers the following cipher file could be used:

```
#
# The allowable ciphers for use between the client and server are
# defined by the last line in this file. See ciphers(1) in
# OpenSSL
# (http://www.openssl.org/docs/apps/ciphers.html) for the format
# of
# statement detailing the ciphers to use.
#
MD5
```

---

## TexAPI/texapi.pm

To set the ciphers supported by the Windows client and client side programs using `texapi.pm` the `TEXCIPHERS` environment variable should be used. For example, to enforce the use of MD5 ciphers the following setting could be used:

```
TEXCIPHERS="MD5"
export TEXCIPHERS
```

It is also possible to set the ciphers when using TexAPI directly. The `Ciphers` member of the `TEXSESSINFO` structure may be used:

```
TEXSESSINFO info;
TEXSESSION session;

...
info.Ciphers = "MD5";
...
TexSessConnect(&info, &session);
...
```

For `texapi.pm` the `Ciphers` key is used:

```
my $session = ke::texapi->new(
{
    ...
    Ciphers => 'MD5',
    ...
});
```

---

## TexJDBC

When using TexJDBC the `ciphers` connection property may be set to restrict the ciphers used for a connection:

```
Properties props = new Properties();

props.setProperty("ciphers", "MD5");
...

Connection conn =
DriverManager.getConnection("jdbc:texpress:socket", props);
```



## SECTION 4

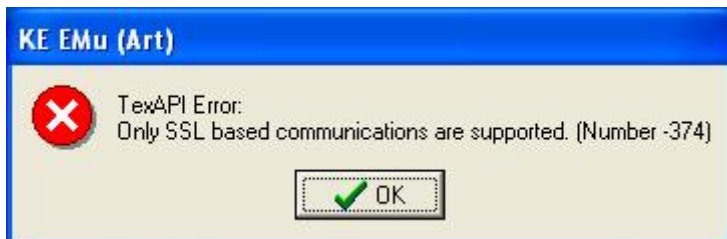
# Common Errors

When configuring the use of encrypted connections a number of common errors may occur. In this section a description of these errors is given, along with possible solutions.

---

## Client does not support SSL

The System Administrator may configure the EMu server to accept encrypted connections only. The `-s` option for `texserver` will force the server to only accept connections where encryption is enabled. If the EMu client is a version prior to 4.0.03, then encryption is not supported and the following error message is displayed:



The solution is to upgrade the EMu client to version 4.0.03 or greater.

---

## Server certificates not installed

If the System Administrator has turned on the `-s` option for `texserver`, thus forcing encrypted connections, and the server side certificates (`server.crt` and `server.key`) are not installed, the following error message is displayed:

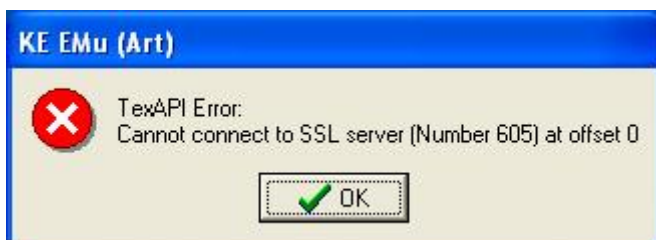


The solution is to generate the private key (`server.key`) and public digital certificate (`server.crt`) and place them in the correct location (`$TEXHOME/etc/certs`).

---

## Server/Client protocol error

The initiation of an encrypted connection involves a handshake between the client and server programs. If an error occurs as part of the handshake, the following message is displayed:



There are many reasons why a protocol error may occur. The most common are:

- The private key file `$TEXHOME/etc/certs/server.key` cannot be read. The error implies the contents of the private key file are corrupted.
- The public digital certificate file `$TEXHOME/etc/certs/server.crt` cannot be read. The error implies the contents of the public certificate file are corrupted.
- A private key/public certificate mismatch. The public digital key `server.crt` was not generated using the private key found in `server.key`. Either the private key or public certificate is incorrect.
- An acceptable cipher cannot be found. The client and server cannot agree on a cipher to use for the connection encryption. The server ciphers file should be altered to match a client cipher or vice versa.

Server side debugging may be required to determine the exact cause of the error. The Texpress debug flags `s15,16` will output the reason for the protocol error. For details on how to set Texpress debug flags please contact KE Software support.

---

## Cannot verify certificate

In order for the client to verify the server's certificate the client must have a copy of the server's top level public CA certificate. If the top level certificate is not installed on the client, the following error is displayed:



The solution is to install the top level CA certificate on the client. The certificate should be placed in a directory called `certs` located in the same place as the EMu client executable.

---

## Bad host name

The *Common Name* field of the server's public digital certificate must contain the host name of the EMu server. Wild cards are supported using the asterisk character. If the *Common Name* field of the server's certificate does not match the host name to which the client is connected, the following error is displayed:



The solution is to fix up your DNS so that the host name of the EMu server matches the host name stored in the server's certificate. If this is not possible, a new server certificate should be generated with the correct host name.

# Index

## B

Bad host name • 28

## C

Cannot verify certificate • 27

Chain signed • 17

Client does not support SSL • 26

Common Errors • 25

Configuring ciphers • 6, 7, 21

## E

EMu client setup • 7

EMu server setup • 6, 14

Encrypted Connections • 1

## G

Generating certificates • 6, 11

## H

How it works • 2

## I

Important! • 6

## R

Requirements • 5, 6

Root signed • 15

## S

Self signed • 12

Server • 22

Server certificates not installed • 26

Server/Client protocol error • 26

Step 1

Generate a private key • 12, 15

Generate a self signed CA certificate •  
18

Step 2

Generate a certificate signing request •  
15

Generate a private key • 18

Generate public digital certificate • 12

Step 3

Generate a certificate signing request •  
18

Installing the files • 14, 16

Step 4

Sign the certificate with your CA  
certificate • 19

Step 5

Creating the certificate chain • 19

Step 6

Installing the files • 19

## T

TexAPI/texapi.pm • 22

TexJDBC • 23

TexJDBC setup • 8





## EMu Documentation

# Exhibition Objects Module

Document Version 1.1

**EMu Version 4.0**





# Contents

<b>SECTION 1</b>	<b>Introduction</b>	<b>1</b>
	Purpose	1
	Scope	1
	Version	1
<b>SECTION 2</b>	<b>Exhibition Objects module</b>	<b>3</b>
	Exhibition Objects tabs	4
	Links to other modules	15
	Events module	15
	Catalogue module	16
	<b>Index</b>	<b>19</b>



## SECTION 1

# Introduction

---

## Purpose

This document describes the Exhibition Objects module, which is designed for Art based EMu clients. The module holds details about a particular object at a particular event, and includes:

- Exhibition information, e.g. the location of the object within the exhibition.
  - Maintenance information, e.g. does the object need to be replaced or does it have any special requirements while the exhibition is underway.
- 

## Scope

The document provides a set of screenshots of each tab in the Exhibition Objects module and a description of the fields on each tab.

---

## Version

Version Number	Updated By	Date Updated
1.0	Bernard Marshall	24 November 2010
1.1	Bernard Marshall	23 March 2011



## SECTION 2

# Exhibition Objects module

The Exhibition Objects module sits between the Events and Catalogue modules and holds details about an object that is associated with an event.

When an object is attached to an Events record, an Exhibition Objects record is automatically created. Removing an object from an event deletes the associated Exhibition Objects record.

In the Events record, each object has a *Status* which indicates whether the object is *Proposed*, *Selected* or *Discarded* from the event. This value is mirrored in the Exhibition Objects record associated with the object.



Institutions may set up their own *Status* values as required.

The Exhibition Objects module can be accessed from:

- The EMu Command Centre
- The Events module (on the Objects tab)
- The Catalogue module (on the Exhibit Objects tab)

## Exhibition Objects tabs

### Tabs Description

#### Status

**Exhibition Objects (1) - Display**

File Edit Select View Tools Tabs Multimedia Window Help

Portrait Section: "Charles, 9th Lord Cathcart" [1981.36], Reynolds, Joshua; Selected 134

Accession Number: 1981.36 Event Catalogue Number: 126 Status: Selected Day One: ☒ Yes ☐ No

Details:

Event: Zwei Jahrhunderte Englische Malerei (1979 - 1980)

Object: "Charles, 9th Lord Cathcart" [1981.36], Reynolds, Joshua

Current Location: [NM.MC.0.SER.W.2] National Museum - Main Campus - Ground Floor - Special Exhibitions Room - V

Credit Line: Supported by the Victoria and Albert Museum Purchase Grant Fund and the National Heritage Memorial Fund.

Notes:

Status History

Status	Modified By	Date Modified
1 Selected	KE EMu Administrator	12/02/2011
2 Proposed	KE EMu Administrator	23/03/2011

Status Section Instructions Maintenance Condition Tasks Notes Mu


Display: Exhibition Object 1 of 1 emu Admin 20136

An Exhibition Objects record holds details about an object (extracted from a Catalogue record) participating in or proposed for an event (extracted from an Events record). The Status tab holds key information extracted from these two modules.

The fields are:

Fields	Description
<i>Accession Number</i>	The Accession Number of the object. The number is extracted from the object's Catalogue record and is read-only.
<i>Event Catalogue Number</i>	A number assigned to the object for the Event. The number generally appears in the event booklet and is used to track the object while part of the exhibition.
<i>Status</i>	Indicates the status of the object for use in the exhibition. Possible values are: <ul style="list-style-type: none"> <li>Proposed - under consideration for inclusion in the exhibition.</li> <li>Selected - part of the exhibition.</li> <li>Discarded - no longer considered for the exhibition.</li> </ul> Institutions are able to specify their own statuses as required.



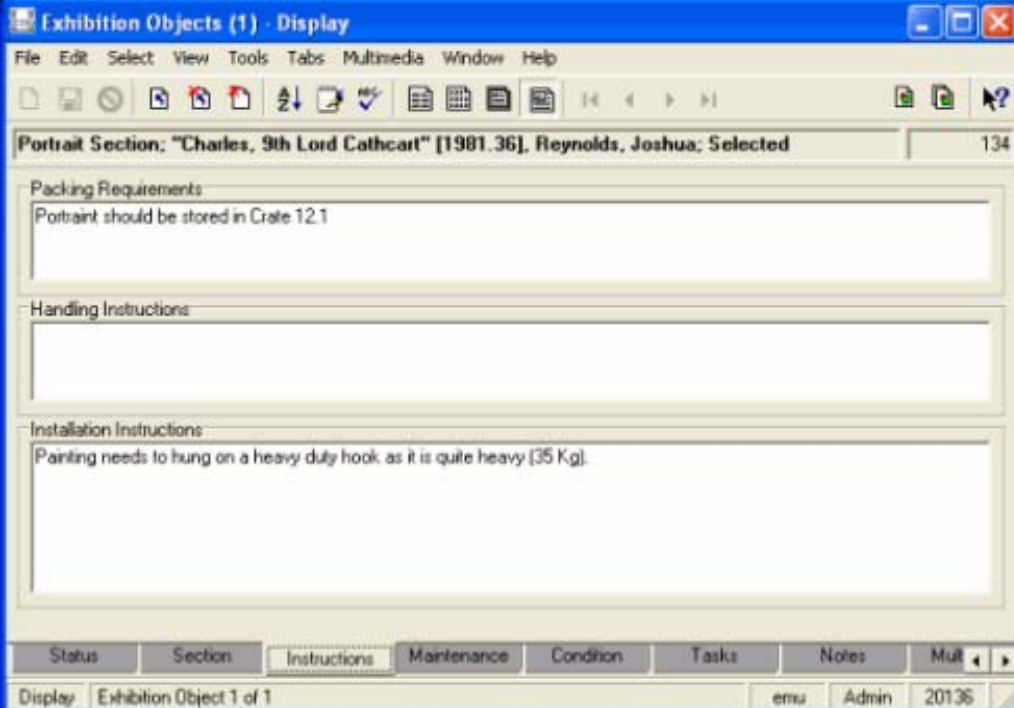
Tabs	Description
<i>Day One</i>	Indicates whether the object is part of the exhibition from the first day. This field is used to assemble a packing list for all objects required when the exhibition commences.
<i>Event: (Details)</i>	<p>The event (recorded in the Events module) for which the object is to be considered.</p> <p>This is a read-only field as the association between the object and the event is established and maintained in the Events record.</p>
<i>Object: (Details)</i>	<p>The object (recorded in the Catalogue module).</p> <p>This is a read-only field. Although selecting the <b>View Attachments</b>  button will display the object's Catalogue record, the association between the object and event is established and maintained in the Events record (that is, an object record is attached to an Events record).</p>
<i>Current Location: (Details)</i>	<p>The current location of the object.</p> <p>This is a read-only field with the value extracted from the object's Catalogue record.</p>
<i>Credit Line: (Details)</i>	The credit line to be used for the object throughout the exhibition. This is a read-only field with the value extracted from the object's Catalogue record.
<i>Notes: (Details)</i>	Any miscellaneous notes about the status of the object. The notes may include a credit line specific to the exhibition, or the reason why the object was discarded from the exhibition, etc.
<i>Status History: (Details)</i>	A table containing an audit trail of changes made to the <i>Status</i> . The table is maintained automatically and cannot be modified.

Tabs	Description
------	-------------

## Section

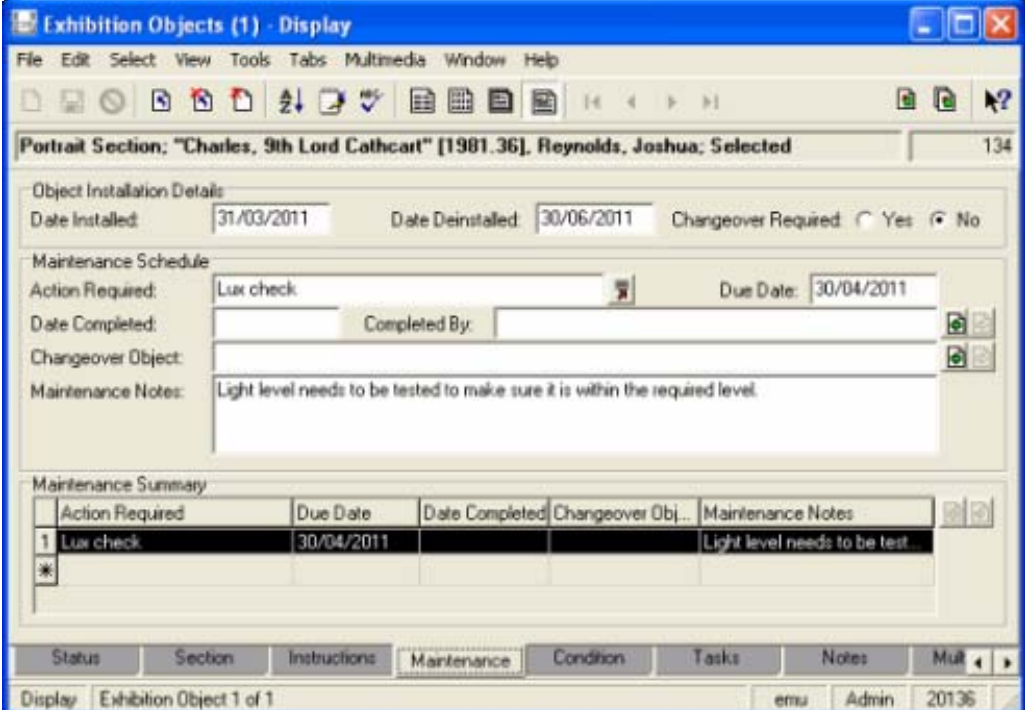
Holds information about where the object is to be located in the exhibition. An exhibition may consist of a number of sections (e.g. 1950s section, Interactive section, etc.). The fields are:

Fields	Description
<i>Section Title</i>	The name of the section in the exhibition in which the object will be located (e.g. 1950s).
<i>Section Number</i>	Many institutions allocate a section number to a section. The number generally refers to a location on the exhibition floor plan.
<i>Case/Wall</i>	The exhibition case or the wall where the object is to be located. If the object is free-standing, a value like <code>Free-standing</code> can be used.
<i>Notes</i>	Any notes about the location of the object within the exhibition. For free-standing objects this may contain a verbose description of its exact location.

Tabs	Description
Instructions	

Holds information about the packing, handling and set-up of the object for the exhibition. The fields are:

Fields	Description
<i>Packing Requirements</i>	Any special packing requirements for transporting the object to and from the exhibition.
<i>Handling Instructions</i>	Any instructions to be observed when handling the object. For example, the object may require chemical free gloves to be worn when lifted.
<i>Installation Instructions</i>	Instructions detailing how the object is to be installed in the exhibition. For example, the object may need to be mounted on a piece of felt.

Tabs	Description
Maintenance	 <p>The screenshot shows a software window titled 'Exhibition Objects (1) - Display'. It has a menu bar (File, Edit, Select, View, Tools, Tabs, Multimedia, Window, Help) and a toolbar. The main content area is divided into sections: 'Portrait Section: "Charles, 9th Lord Cathcart" [1981.36]. Reynolds, Joshua; Selected' with a count of 134; 'Object Installation Details' with fields for 'Date Installed: 31/03/2011', 'Date Deinstalled: 30/06/2011', and 'Changeover Required: No'; 'Maintenance Schedule' with 'Action Required: Lux check', 'Due Date: 30/04/2011', and 'Maintenance Notes: Light level needs to be tested to make sure it is within the required level.'; and a 'Maintenance Summary' table with one row: '1 Lux check 30/04/2011 Light level needs to be test...'. At the bottom, there are tabs for 'Status', 'Section', 'Instructions', 'Maintenance' (selected), 'Condition', 'Tasks', 'Notes', and 'Mult'. The status bar shows 'Display: Exhibition Object 1 of 1' and 'emu Admin 20136'.</p>

Records any actions required while the object is part of the exhibition. Also used to indicate whether the object needs to be:

- Removed from the exhibition before the exhibition is over (the *Date Deinstalled* will be prior to the end date of the exhibition and *Changeover Required* is set to No).
- OR-
- Removed from the exhibition and replaced with another object (the *Date Deinstalled* will be prior to the end date of the exhibition and *Changeover Required* is set to Yes).

The fields are:

Fields	Description
<i>Date Installed:</i> (Object Installation Details)	The date the object is installed in the exhibition (that is, the date it is added to the exhibition).
<i>Date Deinstalled:</i> (Object Installation Details)	The date the object is removed from the exhibition.

Tabs	Description
<i>Changeover Required: (Object Installation Details)</i>	A <b>Yes</b> value indicates that the object is to be removed from the exhibition before the exhibition concludes and replaced with another object.
<i>Action Required: (Maintenance Schedule)</i>	The maintenance activity required for the object (e.g. Cleaning).
<i>Due Date: (Maintenance Schedule)</i>	The date by which the maintenance activity should have taken place.
<i>Date Completed: (Maintenance Schedule)</i>	The actual date on which the maintenance activity was completed.
<i>Completed By: (Maintenance Schedule)</i>	The person who performed the maintenance activity.
<i>Changeover Object: (Maintenance Schedule)</i>	A link to the replacement object if the object is to be replaced before the exhibition concludes.
<i>Maintenance Notes: (Maintenance Schedule)</i>	Any notes specific to the maintenance of the object (e.g. When the object is cleaned a static free cloth must be used).
<i>Maintenance Summary</i>	A list of all the maintenance activities required for the object over the life of the exhibition.

Tabs	Description
------	-------------

Condition

Holds the condition history of the object and details of any conservation work undertaken. All values are extracted from the object's Catalogue and Conservation records and cannot be modified. The fields are:

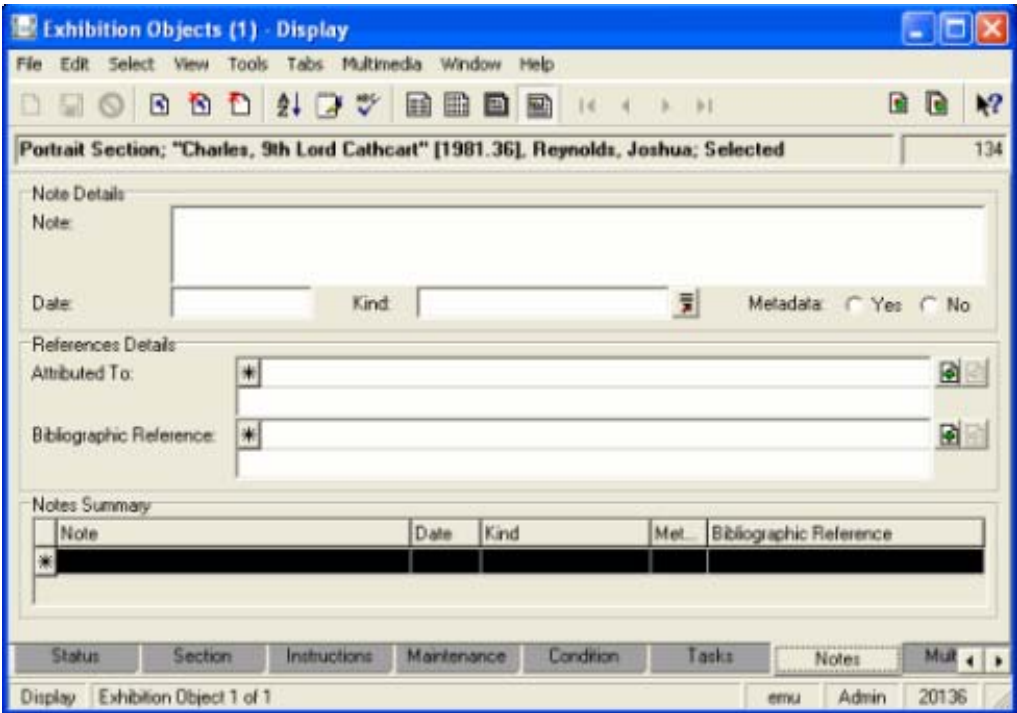
Fields	Description
<i>Status:</i> (Condition Check)	The condition status of the object. Typical values are: <ul style="list-style-type: none"> <li>• Excellent</li> <li>• Good</li> <li>• Fail</li> <li>• Poor</li> </ul>
<i>Date:</i> (Condition Check)	The date on which a condition check was last performed on the object.
<i>Checked By:</i> (Condition Check)	The person who performed the last condition check.
<i>Details:</i> (Condition Check)	Any notes concerning the condition of the object (e.g. Deteriorated and cannot be repaired).
<i>Condition Check History</i>	A table displaying details of all condition checks performed on the object. Select a row in the table to display its details in the <i>Condition Check</i> group of fields above.
<i>Conservation History</i>	A list of all conservation work carried out on the object. The table links to the Conservation module, allowing complete details of the work carried out to be viewed.

Tabs	Description
Tasks	<p>Standard EMu Tasks tab. Records tasks to be carried out on the object for this exhibition. For example, conservation work may be required, photographic pictures may need to be taken, etc.</p>



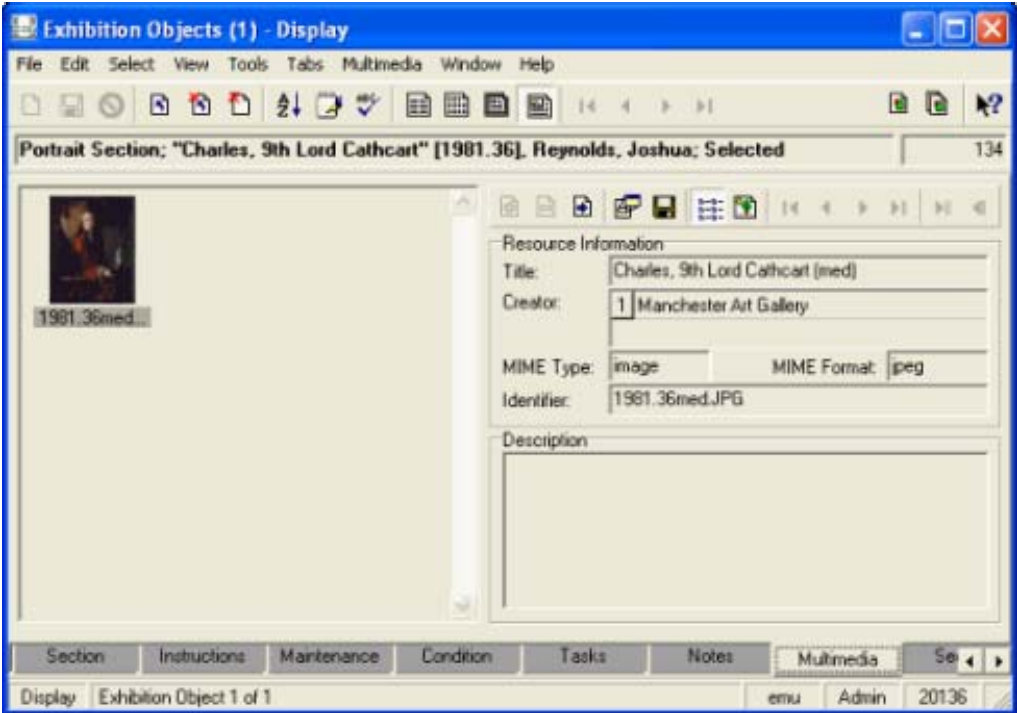
Tabs	Description
------	-------------

Notes



Standard EMu Notes tab. Used for general purpose attributed notes. When a note is entered, not only is the text recorded but also who the note is attributed to and any associated bibliographic references.

Multimedia

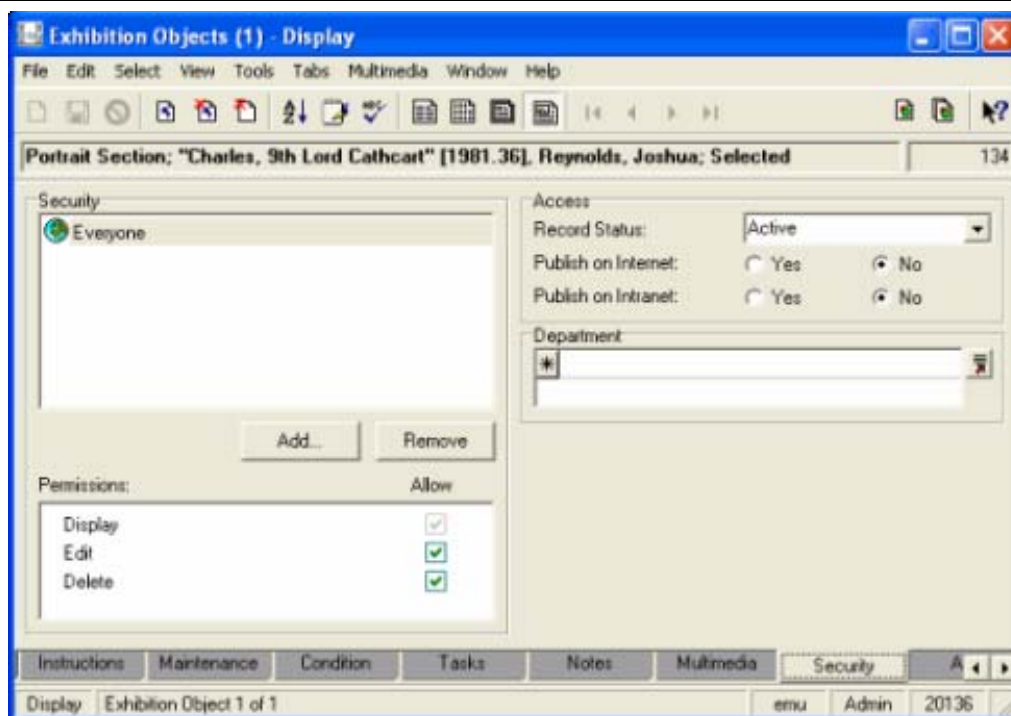


Standard EMu Multimedia tab. Displays details of the first multimedia item in the object record. As the multimedia and details are extracted from the object record, the values cannot be changed.



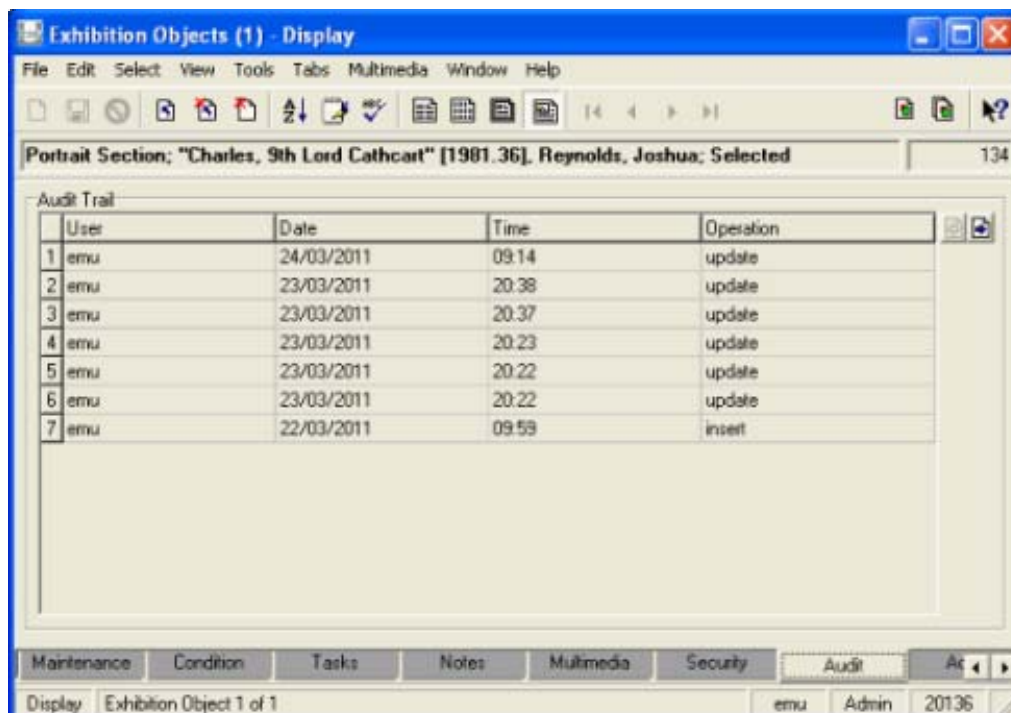
Tabs	Description
------	-------------

## Security



Standard EMu Security tab. Identifies who is permitted to modify the record and whether the record should be available for access via the Internet/Intranet. The *Record Status* determines whether the record is still *Active* or *Discarded*.

## Audit



Tabs	Description
	<p>Standard EMu Audit tab. Holds an audit trail of all changes made to the record. The table includes details about:</p> <ul style="list-style-type: none"><li>• Who made the change</li><li>• When the change was made</li><li>• What operation was performed</li></ul> <p>Complete details of the changes may be found in the Audit module.</p>

#### Admin

Exhibition Objects (1) - Display

File Edit Select View Tools Tabs Multimedia Window Help

Portrait Section: "Charles, 9th Lord Cathcart" [1981.36]. Reynolds, Joshua; Selected 134

Legacy Data

Insertion Details

Inserted By: KE EMu Administrator

Insertion Date: 22/03/2011

Insertion Time: 09:59

Modification Details

Modified By: KE EMu Administrator

Modification Date: 24/03/2011

Modification Time: 09:14

Import Details

Import Identifier:

System Identifier:

Condition Tasks Notes Multimedia Security Audit Admin

Display: Exhibition Object 1 of 1 emu Admin 20136

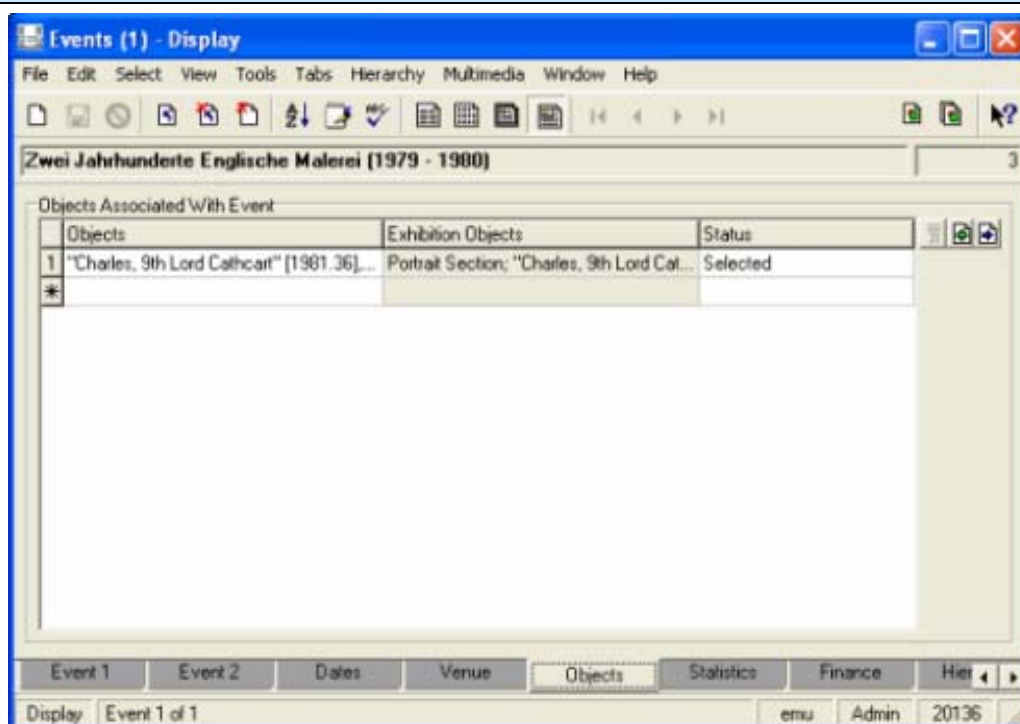
Standard EMu Admin tab. Holds details about the last time the record was modified and when it was created. If the record was built from pre-existing data, the legacy data will also be displayed.

## Links to other modules

### Events module

Tab	Description
-----	-------------

Objects



The Objects tab in the Events module lists all objects in an exhibition event. The list also includes objects which were considered for the exhibition but which were discarded.

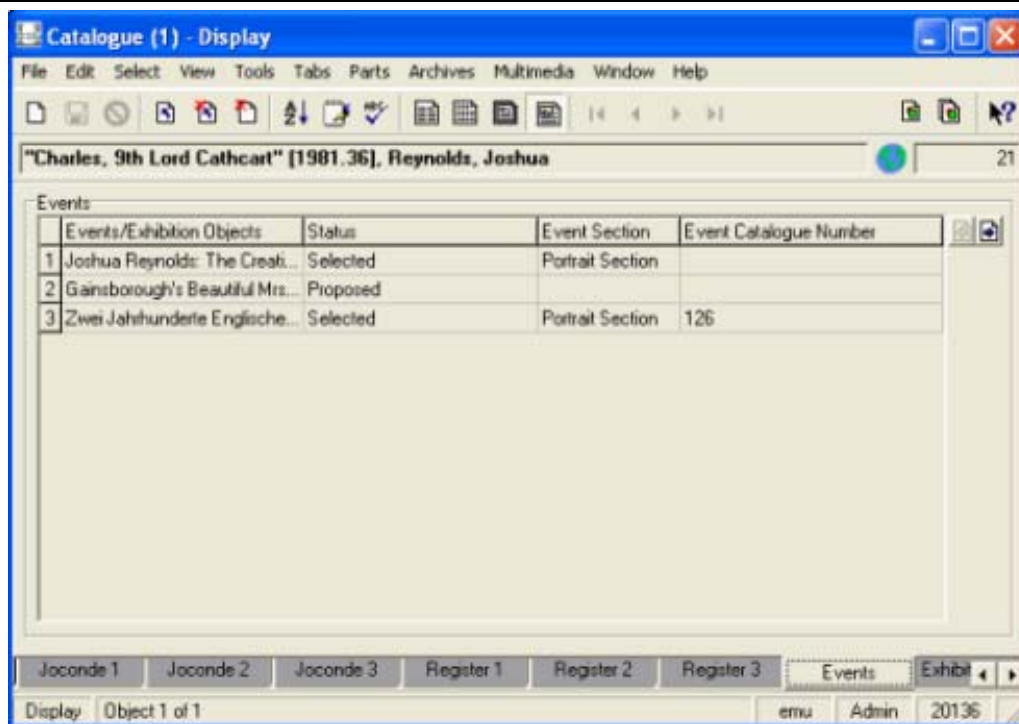
The fields are:

Fields	Description
<i>Objects</i>	The Catalogue object considered for display as part of the exhibition. Links to the object's record in the Catalogue module.
<i>Exhibition Objects</i>	Links to the Exhibition Objects record for this object and exhibition. Exhibition Objects records are created / updated / removed by EMu so the value cannot be modified.
<i>Status</i>	A copy of the <i>Status</i> field from the associated Exhibition Objects record. The <i>Status</i> may be modified here (in the Events module) or in the Exhibition Objects module and the change will be updated in both locations.

## Catalogue module

Tab	Description
-----	-------------

Events



Lists all the events with which the object has been associated. The list also includes events for which the object was considered but discarded.

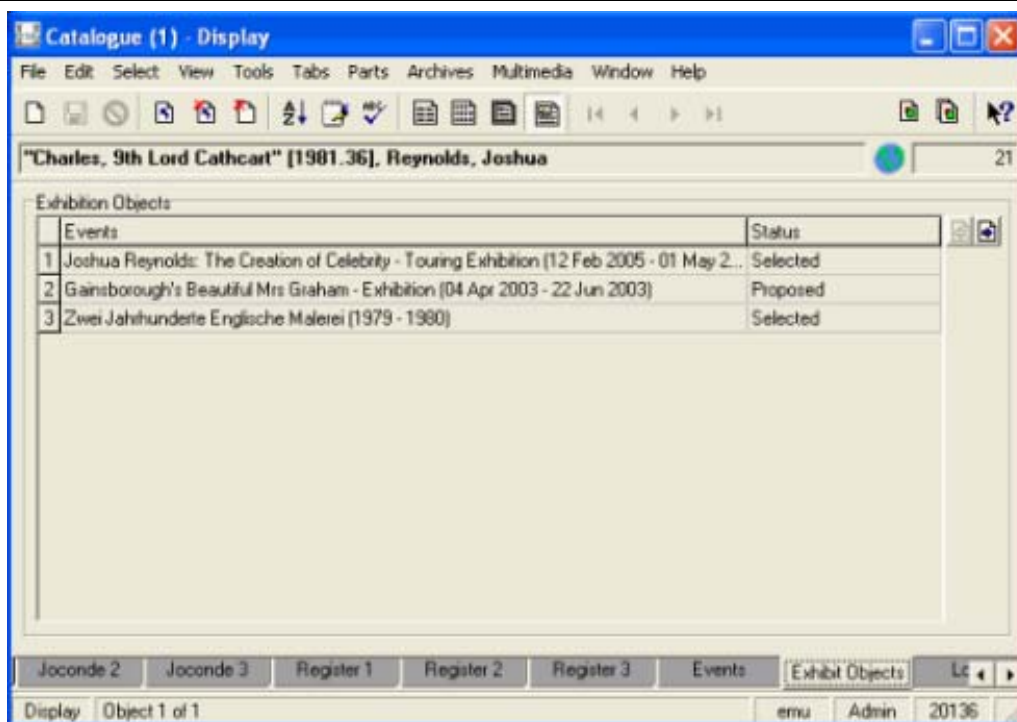
Selecting the **View Attachments**  button will open the Events module.

The fields are:


Field	Description
<i>Events / Exhibition Objects</i>	The name of the event in which the object was displayed or for which it was considered.
<i>Status</i>	Whether the object was part of the exhibition or whether it was only considered.
<i>Event Section</i>	The section of the exhibition in which the object was located.
<i>Event Catalogue Number</i>	The number allocated to the object within the exhibition. The number generally appears in the exhibition hand book.

Tab	Description
-----	-------------

Exhibit  
Objects



Holds details of all the events with which the object has been associated. The list also includes events for which the object was considered but discarded.

Selecting the **View Attachments**  button will open the Exhibition Objects module.

The fields are:

Field	Description
<i>Events</i>	The name of the event in which the object was displayed or for which it was considered.
<i>Status</i>	Whether the object was part of the exhibition or whether it was only considered.



# Index

## C

Catalogue module • 19

## E

Events module • 18

Exhibition Objects module • 5

Exhibition Objects tabs • 6

## I

Introduction • 1

## L

Links to other modules • 18

## P

Purpose • 1

## S

Scope • 2

## V

Version • 3



## EMu Documentation

# Multi-group Support

Document Version 1

**EMu Version 4.0.0.3**







# Contents

<b>SECTION 1</b>	<b>Overview</b>	<b>1</b>
<b>SECTION 2</b>	<b>Specifying a user's groups</b>	<b>3</b>
	How to set an existing user's groups	4
	How to set a new user's groups	6
<b>SECTION 3</b>	<b>Using multiple groups</b>	<b>7</b>
	Active group and module group	8
	Logging in to a group	9
	Switching groups via a module	10
	Switching groups via the Command Centre	12
<b>SECTION 4</b>	<b>Registry cache</b>	<b>13</b>
	Flushing the Registry cache	14
	Enable / disable the Registry cache	15
<b>SECTION 5</b>	<b>Module caching</b>	<b>17</b>
<b>SECTION 6</b>	<b>Record Level Security</b>	<b>19</b>
<b>SECTION 7</b>	<b>Security profile extensions</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



## SECTION 1

# Overview

Users are given access to EMu by assigning them to a group. Each group has a set of permissions associated with it, and the user inherits the permissions of the group to which they have been assigned. Individual user based permissions may then be defined to override group based settings as required. The use of group based permissions means that it's not necessary to specify all the permissions on a per user basis, and only the difference between the group permissions and any user specific permissions need to be defined for any one user.

Groups are generally based around real life roles, with each group reflecting the permissions required by all users who undertake the role. For example:

- A *Curatorial* group might allow users to create new Catalogue records, but not to register new loans.
- A *Loans Officer* group would have permission to create loans records, but not to create Catalogue records, and only to update the *Condition Check* and *Locations* fields for existing Catalogue records.

Assigning a user to the *Curatorial* group, casts that user in the curatorial role.

In general, the group system works well in EMu, except when a user performs more than one role within the institution. For example, a curator (group *Curatorial*) may also manage the loans for a small part of the collection (and so requires *Loans Officer* permissions). Until now, EMu provided two solutions to this dilemma:

1. Create two usernames, assigning one to group *Curatorial* and the other to group *Loans Officer*. The user must then use the correct username when logging into EMu to perform the required role.

Moving from one role to the other (e.g. a new loan arrives while the user is logged in as a curator), requires the user to log out of EMu and back in using the *Loans Officer* username. This could become tedious, and it requires that the user remembers two user names and two passwords.

2. The second solution requires a new group to be created which is a merge of the permissions of the *Curatorial* and *Loans Officer* groups.

The problem with this solution is that the combined privileges present a view of the world that is neither curatorial nor loan specific: it is possible to alter any field in a Catalogue record while raising a new loan. In other words, a new hybrid role is created (curator-loans office).

In most cases the two original roles are sufficient and all that is required is a mechanism to switch between the two groups without having to log out and back in as another user.

EMu 4.0.03 introduces multi-group support, which allows a single user to be registered in more than one group:

- A user who is a member of more than one group can select the group name to use from the Login dialogue box when logging into EMu.
- At any time it is possible to switch to another group without logging out and back in again.
- When switching groups the user decides whether opened modules should remain open or whether they should be closed.
- Any new modules opened will use the group permissions assigned to the group that the user switched to.

By not closing open modules it is possible to have modules in different groups open at the same time. EMu ensures the correct group permissions are observed based on the group associated with the module.

With multi-group support, a user is able to log in using one group, switch to another group while leaving existing modules open, perform operations in the new group and then switch back to the previous group, all without having to close any modules. A key feature of multi-group support is that it allows users to change roles without losing their current work position and then to return to that position at a later time.



Texpress 8.2.009 or later must be installed to provide the back-end security facilities for multi-group support.

## SECTION 2

# Specifying a user's groups

The EMu Group Registry entry defines which group a user is assigned to. It is consulted whenever a user logs in to EMu. The format of the Registry entry is:

User | *username* | Group | *groupname*

where *groupname* specifies the group to use when determining permissions.

For example:

User | badenov | Group | Curatorial

specifies that user badenov is a member of the Curatorial group and has all the permissions assigned to that group.

In order to provide multi-group support, the Group Registry entry has been extended to allow a semi-colon separated list of groups to be specified. The order of the groups is not important.

For example:

User | badenov | Group | Curatorial;Loans Officer

specifies that user badenov is a member of both the Curatorial and Loans Officer groups.



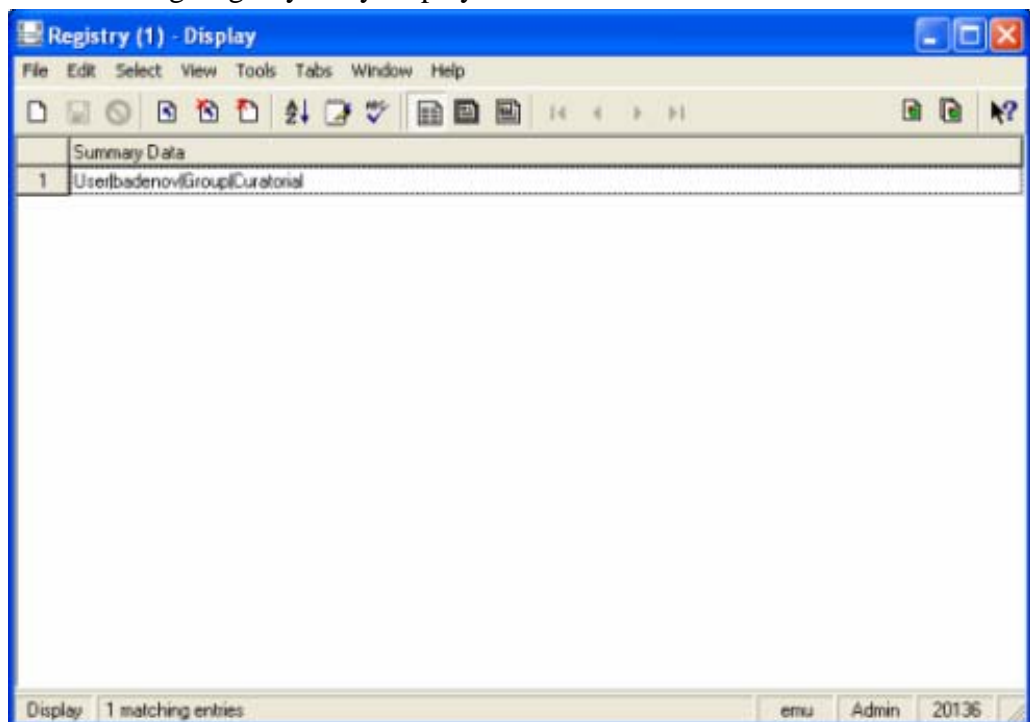
Changes to the list of groups that a user has access to only come into effect when the user logs into EMu following any change to their Group Registry entry. If they are logged in when the change is made, they will need to log out and back in again in order to have access to the updated list of groups.

---

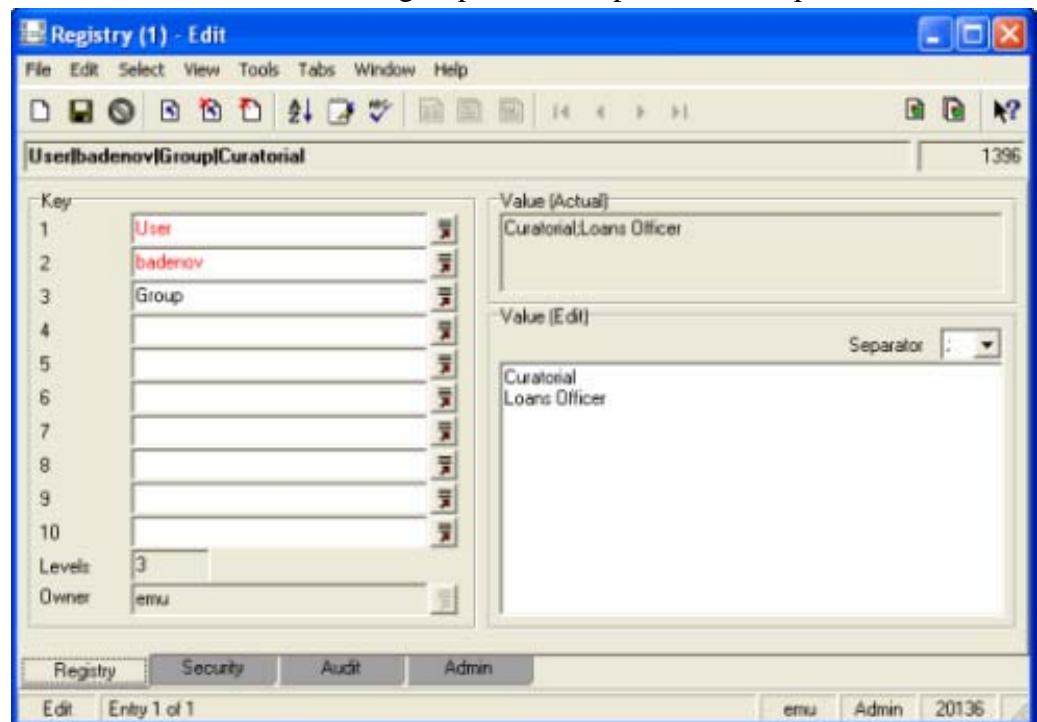
## How to set an existing user's groups

1. Log in to EMu as a System Administrator.
2. In the Registry module, search for the user's Group Registry entry:
  - i. Enter `User` into the *Key 1* field.
  - ii. Enter the user's username into *Key 2*.
  - iii. Enter `Group` into *Key 3*.

The matching Registry entry displays:



3. In the *Value (Edit)* field on the Registry tab, enter the name of all groups that the user is a member of. Each group should be placed on a separate line:



4. Save the record.



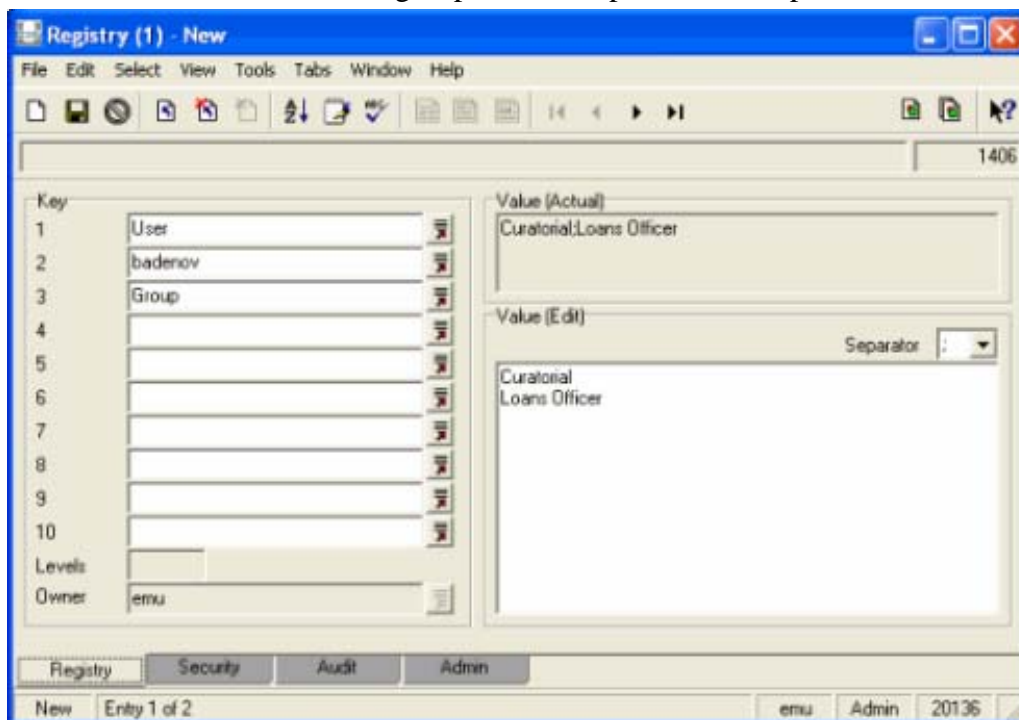
As of EMu 4.0.03, security profiles are generated automatically after saving any user registration based Registry entries and it is no longer necessary to run **Tools>Generate Record Security** after changing user settings.



---

## How to set a new user's groups

1. Log in as a System Administrator.
2. Add a new record in the Registry module:
  - i. Enter `User` into the *Key 1* field.
  - ii. Enter the user's username into *Key 2*.
  - iii. Enter `Group` into *Key 3*.
3. In the *Value (Edit)* field on the Registry tab, enter the name of all groups that the user is a member of. Each group should be placed on a separate line:



4. Save the record.



As of EMu 4.0.03, security profiles are generated automatically after saving any user registration based Registry entries and it is no longer necessary to run **Tools>Generate Record Security** after changing user settings.

## SECTION 3

# Using multiple groups

When a user has been registered to use multiple groups, the EMu Login dialogue box includes a drop list with the names of all available groups for the user.



The list of groups that displays is determined by the combination of *username* + *service*. A user can have the same *username* on multiple services, and in each service may be a member of different groups.

The list is displayed only if the *username* supports more than one group.

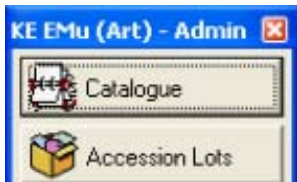


A user must log in to EMu after changes have been made to their group membership before the (updated) list of groups will appear at login. Once a user has successfully logged in to EMu, the system Options may be used to switch to another group. See *Switching groups via a module* (page 10) for details.

---

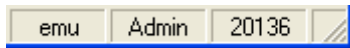
## Active group and module group

EMu keeps track of the group selected when a user logs in to the system. This group is known as the active group and it is displayed in the title bar of the Command Centre. In this example, the active group is group Admin:



Modules started by clicking a button in the Command Centre belong to the active group. The active group is also highlighted in the Login dialogue box the next time the user logs in to EMu.

Modules are also associated with a group. When a module is invoked from the Command Centre its group is that of the Command Centre, namely the active group. If a module is created from within another module, by selecting **Window>New>module**, the module is associated with the same group as the module from which it was created. A module's group is displayed in the Status bar at the bottom right of the window. In this example the module's group is group Admin:



A module cannot change group. Once it is created and associated with a group, it will remain in that group.

It is possible to change the active group however. By doing so it is possible to create modules in different groups, thus allowing users to have multiple roles within the one EMu session.

---

## Logging in to a group

1. Start the EMu application.


The Login dialogue box displays:



2. Enter your username into the *User* field.

If you are registered for more than one group with the current Service and you have logged in successfully previously, a Group drop list will be added to the Login dialogue box:

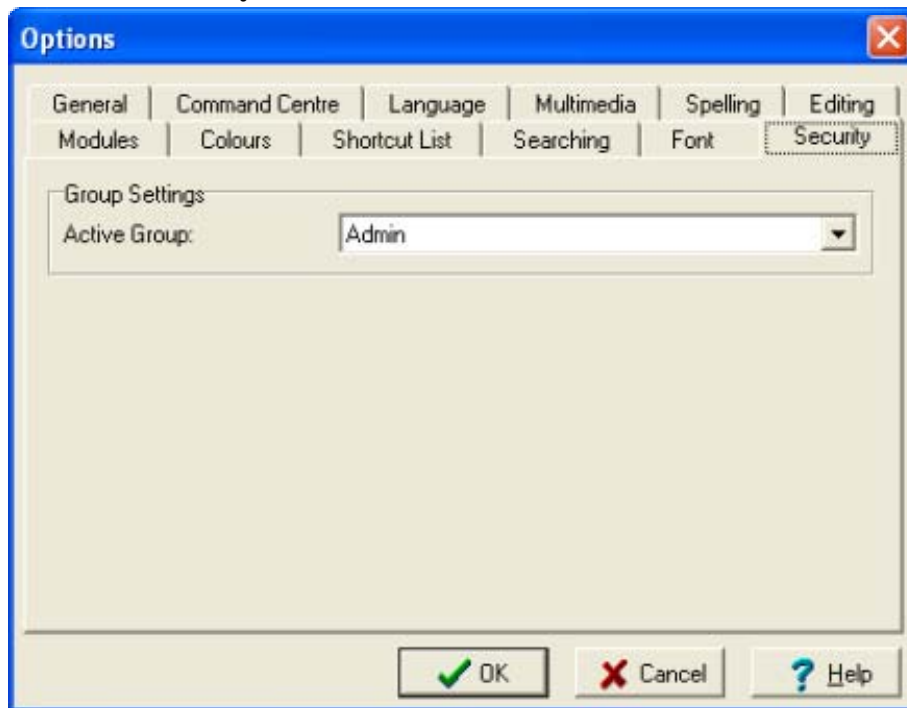


3. Select a group from the Group drop list.
4. Complete the rest of the log in details and select .


---

## Switching groups via a module

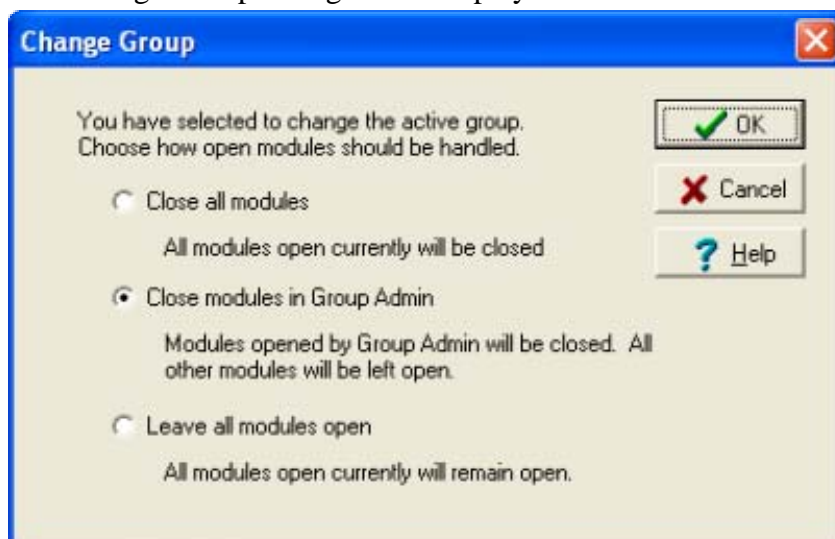
1. Select **Tools>Options** from the Menu bar of any open module.  
The Options dialogue box displays.
2. Select the **Security** tab:



The current active group is displayed in the *Active Group* drop list.

3. Select a group from the *Active Group* drop list.  
The list only contains those groups the user is registered to use.
4. Select .

The Change Group dialogue box displays.



5. Select how any open modules should be handled when the active group is changed:

- **Close all modules**

All open modules will be closed. Only the Command Centre will remain open.

- **Close all modules in Group *group***

All modules open in the active group (*group*) will be closed.

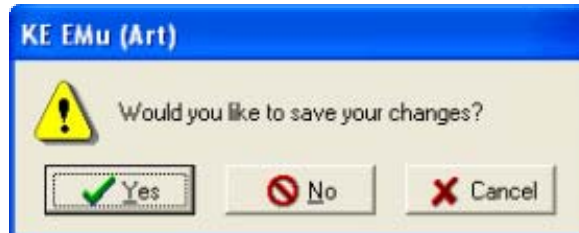
This option is useful for closing all modules created since the last time the group was switched. For example, if you were asked a question that required switching groups to respond, this option will close the module(s) created to answer the question. All other open modules are left untouched.


- **Leave all modules open**

All modules currently open will remain open. Each module will remain in its current state and group. Once the switch is complete the modules may be used as required. Each open module will continue to provide functionality consistent with the module's group.

6. Select .

The active group will be switched. If a module scheduled for closure contains unsaved data, a message will display prompting the user to save the data before the module is terminated:



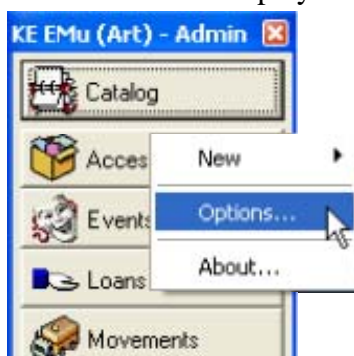
Selecting  will abort the switching process.

---

## Switching groups via the Command Centre

1. Right click the Command Centre.

A context menu displays:



2. Select **Options**.  
The Options dialogue displays
3. Continue from Step 2 above (page 10).

## SECTION 4

# Registry cache

The permissions of all groups and users in the system are specified in the EMu Registry and this information is stored on the EMu server. The Registry must be consulted each time the EMu client is required to determine whether a user is permitted to perform a given operation. As EMu provides a sophisticated security module the Registry may be accessed heavily for certain operations (starting a new module for example). In order to reduce the time required to query the Registry for permissions, the EMu client now contains a Registry cache. The cache is a mechanism that remembers what permissions have been asked for and the associated response. If the same question is asked again, the answer can be given without the need to access the EMu server. The Registry cache reduces traffic to the EMu server and provides a speed improvement in the EMu client.

The one disadvantage of using a cache is that changes made to the EMu Registry by other users will not be picked up until either the cache is cleared (flushed) or you log out and log back in again. For example, if you are using EMu and you have been producing reports from the Parties module, and another user adds a new report to the Parties module, you will not see the report listed until you log out and log back in again, or until you flush the cache.

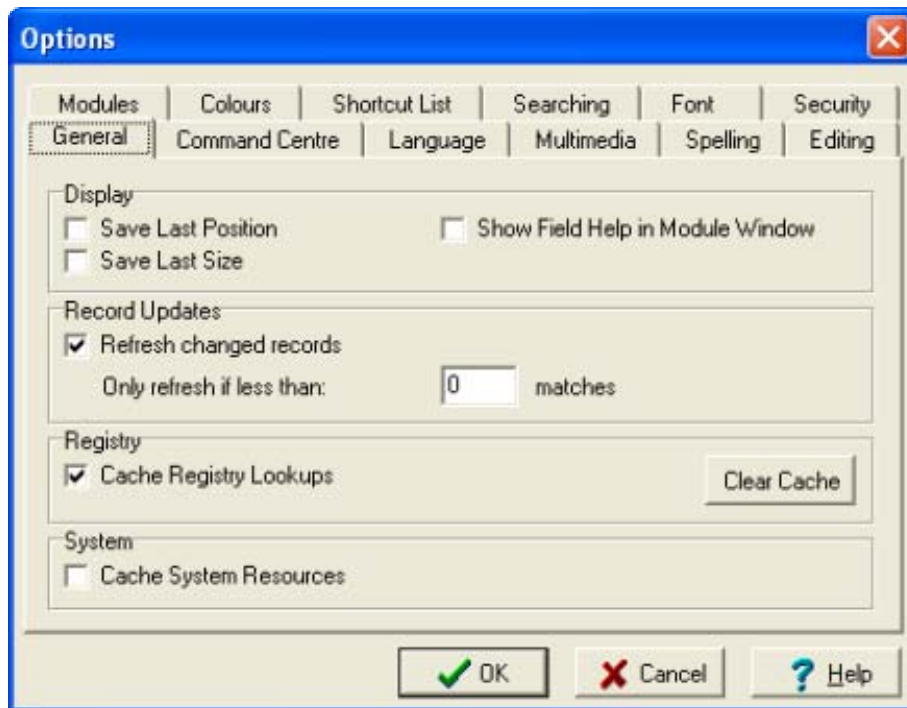
In practice this tends not to be a serious issue and the speed gains more than outweigh any inconvenience. However, it may occasionally be necessary to clear the Registry cache:

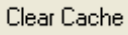



---

## Flushing the Registry cache

1. Select **Tools>Options** from the Menu bar of any open module.  
The Options dialogue box displays.
2. Select the **General** tab.



3. Select .
  4. Select .
- A message displays once the cache is empty.

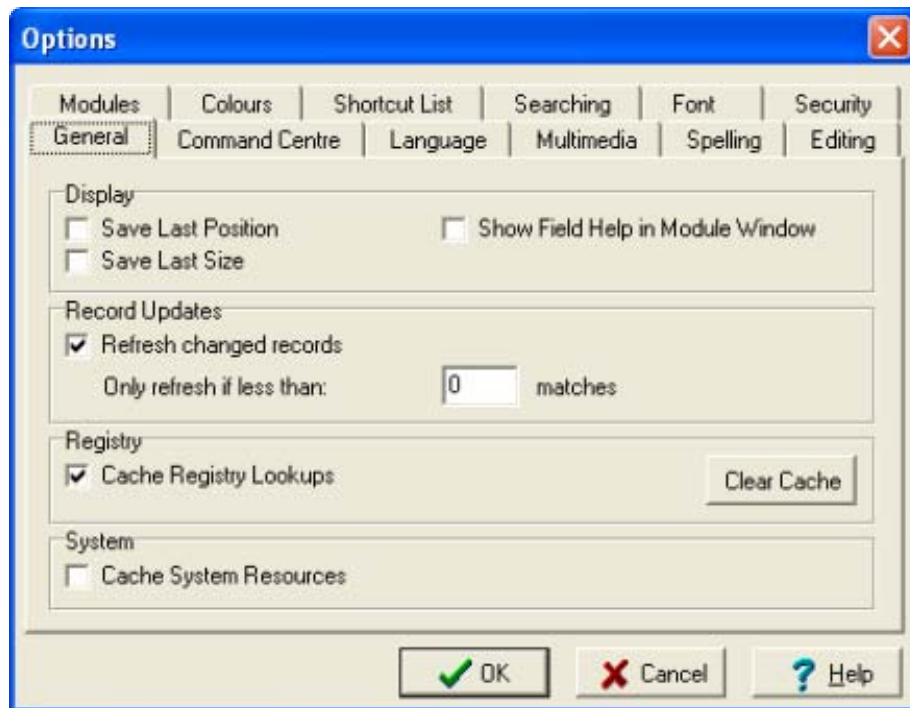


Flushing the cache removes all Registry responses stored in the EMu client. Next time a Registry question is raised, the EMu client will contact the server for a response and the response will be added to the cache again. Until the cache becomes populated again a decrease in client performance may be experienced.

---

## Enable / disable the Registry cache

1. Select **Tools>Options** from the Menu bar in any open module.  
The Options dialogue box displays.
2. Select the **General** tab:



3. Tick / un-tick the **Cache Registry Lookups** checkbox to enable / disable the Registry cache.
4. Select .

If the Registry cache is disabled, it is flushed silently as its entries can no longer be used.

Disabling the Registry cache will cause the EMu client to run significantly slower as all Registry queries must be answered by the server. The only useful purpose achieved by disabling the cache is to test the effect changing Registry entries has on a module, without the need to flush the Registry cache. In day to day use, the cache should be enabled.

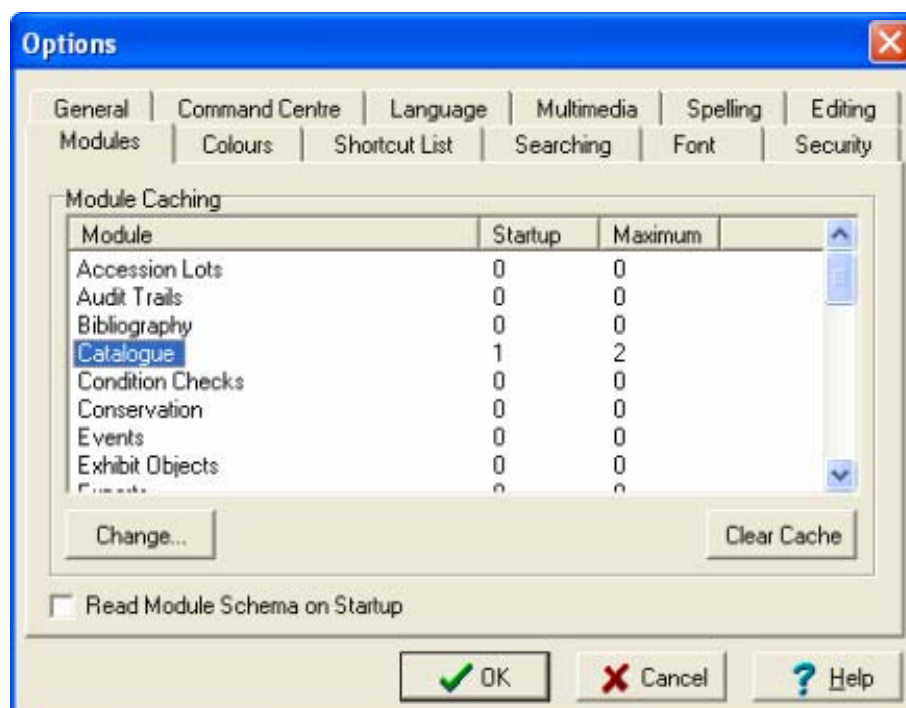


## SECTION 5

## Module caching

Allowing users to switch between groups has implications for how EMu performs module caching.

Module caching is a mechanism that allows modules to be hidden, rather than destroyed, when they are closed. If a new instance of the module is required, the hidden module may be used. Module caching provides significant speed improvements in situations in which modules are opened and closed on a regular basis. Module caching is configured via the Modules tab in the Options dialogue box:



In the example above, one instance of the Catalogue module is created when EMu is invoked. When a Catalogue module is closed, a maximum of two instances are maintained in the module cache.

With the introduction of support for multiple groups, the number of modules to cache now applies on a per group basis. Thus for the previous example, up to two instances of the Catalogue module will be cached per group.

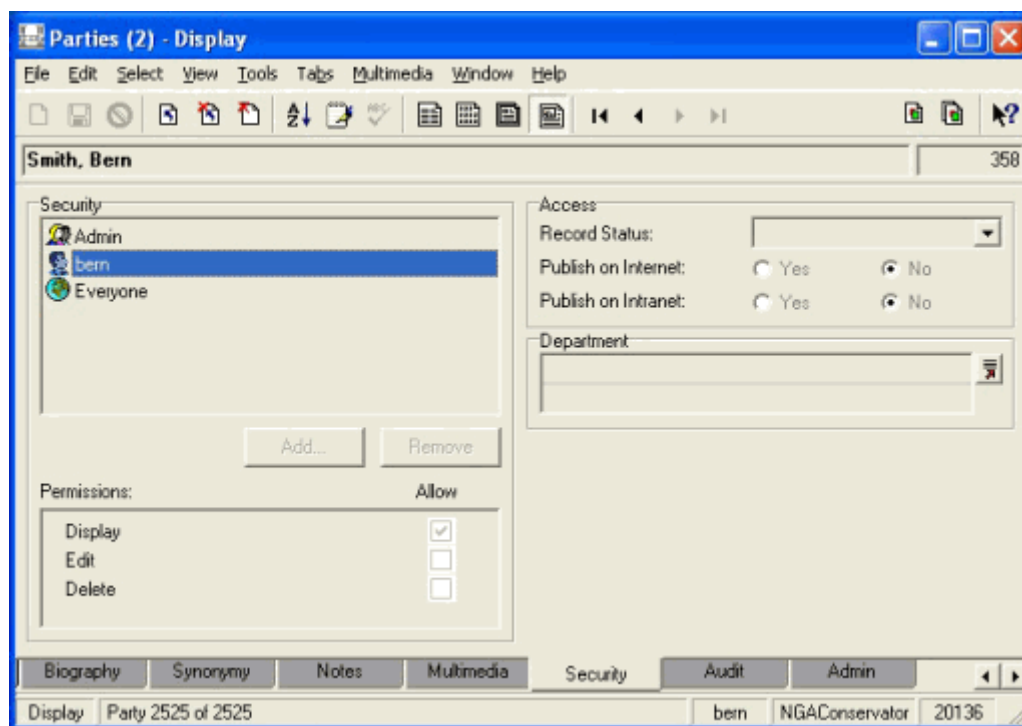
When switching groups, the Change Groups dialogue box allows the user to indicate not only how open modules should be handled, but also how cached modules are dealt with. For example, if you choose to *Close all modules*, then all open modules are closed and all cached modules are flushed.



## SECTION 6

## Record Level Security

Enabling users to be a member of multiple groups has implications for how record level security permissions are displayed in the EMu client.



Name selected in the <i>Security</i> field:	Permissions displayed
The person currently logged in to EMu	<p>The permissions displayed are the group permissions that apply to the module.</p> <p>A module joins a group in one of two ways:</p> <ul style="list-style-type: none"><li>• If the module was created by clicking a button in the Command Centre, the module's group is the one that the user was logged in as when the module was opened.</li><li>• If the module was created by selecting <b>Window&gt;New&gt;module</b> from a module Menu bar, the module's group is the same as the module from which it was created.</li></ul> <p>The module's group is shown in the module's Status bar at the bottom right of the window (<i>NGAConservator</i> in the example above).</p> <p>For example, if a user is in two groups, Curatorial and Loans Officer, and the module was opened from the Command Centre when the user was logged in as a member of the Curatorial group, then the permissions displayed are those of the Curatorial group.</p> <p>This is as expected as the permissions reflect what operations (edit, delete) can be performed on the displayed record.</p> <p>The permissions displayed are a merging of all the user's group permissions.</p> <p>For example, if a user is in two groups, Curatorial and Loans Officer, where group Curatorial has permission to delete the record, but group Loans Officer does not, then the permissions displayed will indicate that the user has delete permission.</p> <p>This is as expected as the user does have permission to delete the record, provided they log in or switch to group Curatorial.</p>
Anyone else	<p>In the example above, user <i>bern</i> is logged in to EMu and the Parties module is in group <i>NGAConservator</i> (indicated in the Status bar). The record level permissions indicate that <i>bern</i> does not have permission to edit or delete the record, even though <i>bern</i> is also in group <i>Admin</i>, which does have edit permission.</p>

## SECTION 7

# Security profile extensions

The EMu server security profiles have been extended to provide support for multiple groups per user. The profiles are maintained in XML format in a file named `security` in the database directory. Two new attributes have been added to the `<user>` tag to provide support for multiple groups:

- `level`  
The `level` attribute defines a label for the user profile. By changing the value of the `level` label for a given user, a different set of security settings is enabled. The group name is used as the label value for EMu databases. To switch between groups the EMu client changes the `level` value to match the group of the module with focus, that is, the module with which the user is interacting.
- `default`  
A "yes" value indicates this set of security settings should be used if the client has not set a `level` value. When the EMu client first connects, a `level` has not been set as the EMu Registry has not yet been consulted (a chicken and egg problem). Once the Registry can verify the log in group, the `level` is set to the supplied value.

A user security profile is created for each group that a user is registered to use (via the `User|username|Group` Registry entry). If user `badenov` has the following Registry entry:

`User|badenov|Group|Curatorial;Loans Officer`

the following XML security segments are generated:

```
<user name="badenov" level="Curatorial" default="yes">
...
</user>
<user name="badenov" level="Loans Officer">
...
</user>
```

The security profiles are built by the `emusecurity` command. This server side command consults the EMu Registry and builds suitable security profiles for all modules, for all users, for each group a user is in. The command is invoked automatically whenever a `User|username|Group` Registry entry is created, modified or deleted. `emusecurity` sets the `default` attribute to "yes" for the first group listed for each user.

The security level is set via the `secllevel` database option. The value of the option is the security level to use. If the option has not been set or the value is empty, the security profile with the `default="yes"` attribute specified is used. For example, to load data into the parties module using group Curatorial the following commands could be used:



```
epartiesopts=seclevel=Curatorial  
export epartiesopts  
texload ....
```

When using TexAPI, the `seclevel` is set via the `TexOptionSet()` call. For example, to change the security level to use group Curatorial for all EMu tables, the following call could be used:

```
TexOptionSet(session, NULL, "seclevel", "Curatorial");
```

For perl based scripts, the `OptionSet()` call is used to alter the security level. For example, to change the security level to use group Curatorial for all EMu tables, the following call could be used:

```
$session->OptionSet("", "seclevel", "Curatorial");
```

The `seclevel` option may be set on a per database basis or a system wide basis.

# Index

## A

Active group and module group • 8

## E

Enable / disable the Registry cache • 15

## F

Flushing the Registry cache • 14

## H

How to set a new user's groups • 6

How to set an existing user's groups • 4

## L

Logging in to a group • 9

## M

Module caching • 17

## O

Overview • 1

## R

Record Level Security • 19

Registry cache • 13

## S

Security profile extensions • 21

Specifying a user's groups • 3

Switching groups via a module • 7, 10, 12

Switching groups via the Command Centre  
• 12

## U

Using multiple groups • 7



## IMu Documentation

# OAI: New IMu Webservice

Document Version 1

**EMu Version 4.0**





# Contents

<b>SECTION 1</b>	<b>What is OAI?</b>	<b>1</b>
	Where to find general information on OAI	2
	Some OAI Terminology and Practice	3
<b>SECTION 2</b>	<b>OAI - a Brief Tour</b>	<b>5</b>
	Interface: the Six verbs	5
	Extra parameters	7
	XML	8
	Flow control	9
<b>SECTION 3</b>	<b>Setting up a KE IMu OAI web service</b>	<b>11</b>
	Configuration	12
	How localisation is done	13
	Where Configuration files are located	14
	File Names	15
	OAI Service	15
	Metadata formats	16
	Configuration File Structure	17
	Simple settings	18
	A list of items	19
	Associative array of items	20
	OAI Service Configuration	21
	Some important parameters	22
	Example	23
	Metadata Configuration	24
	Some important parameters	25
	How field mappings are specified	26
	Special field mappings	27
	Example of a metadata configuration file	28
	<b>Index</b>	<b>29</b>



## SECTION 1

# What is OAI?

OAI is a web service that returns XML data in response to arguments passed to it. Depending on the passed arguments it is able to return information about the service and its capabilities, as well as record data.

The XML record data can be wrapped in various metadata formats. As a minimum the system will use Dublin Core, however a data source (a repository) may also implement other formats and can wrap the data on request in an alternative format.

OAI has limited query flexibility (confined to data modification dates or to querying preconfigured sets of records) as it is primarily meant for data harvesting rather than generalised querying.

EMu via the IMu Web framework can act as an OAI repository, providing an OAI web service to OAI harvesters.

---

## Where to find general information on OAI

- <http://www.openarchives.org>
- <http://www.openarchives.org/documents/FAQ.html>
- <http://www.openarchives.org/OAI/openarchivesprotocol.html>
- [http://en.wikipedia.org/wiki/Open\\_Archives\\_Initiative\\_Protocol\\_for\\_Metadata\\_Harvesting](http://en.wikipedia.org/wiki/Open_Archives_Initiative_Protocol_for_Metadata_Harvesting)



---

## Some OAI Terminology and Practice

- Data sources are called **repositories**.
- Systems that access repositories are called **harvesters**.
- OAI is designed as a harvesting protocol rather than a generalised search protocol.
- Typically data is harvested by date range (i.e. give me what has changed between these dates).
- Despite not having arbitrary search criteria, a repository can provide data sets which are in effect a pre-defined grouping of records. For example, a repository may define sets such as *The James Cook Images* or *The Fred Nerk Collection*. Harvesters can ask for records from just a specific set.
- OAI allows the use of arbitrary metadata formats rather than a single fixed one. However, as a minimum requirement it must always be able to provide Dublin Core metadata.
- OAI should have functionality for deleted records (<http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm#DeletedRecords>). However this might simply involve the repository declaring when asked that 'I don't maintain info about deleted records'.



## SECTION 2

# OAI - a Brief Tour

---

## Interface: the Six verbs

Harvesters have six methods (or verbs) to use when communicating with a repository. Each verb, included in a URL request sent to the repository, returns an XML response. Some verbs require or allow additional parameters:

Verb	Description
Identify	Requests a summary of the service. For example: <code>http://caltechcstr.library.caltech.edu/perl/oai2?verb=Identify</code>
ListMetadataFormats	Requests a description of which metadata formats the repository can provide records in. Must at least return Dublin Core but can optionally return others. For example: <code>http://kamome.lib.ynu.ac.jp/dspace-oai/request?verb=ListMetadataFormats</code> In this case the repository provides Dublin Core and something called <code>junii2</code> (which includes link to schema).
ListSets	Requests a list of sets (i.e. predefined queries). For example: <code>http://genesis2.jpl.nasa.gov/perl/oai2?verb=ListSets</code> In this case the repository has five sets: <ul style="list-style-type: none"><li>• Status = Unpublished</li><li>• Status = Published</li><li>• Status = In Press</li><li>• Subject = Presentations</li><li>• Subject = Publications</li></ul>

Verb	Description
ListIdentifiers	<p>Requests a list of record identifiers.</p> <p>All records in a repository must have a unique identifier (for example, KE's typically uses a mixture of the institution name plus EMu module plus the record's IRN to make that identifier). This verb is asking the system to provide a list of the unique record identifiers.</p> <p>For example:</p> <pre>http://genesis2.jpl.nasa.gov/perl/oai2?verb=ListIdentifiers&amp;metadataPrefix=oai_dc</pre>
ListRecords	<p>Requests a list of records.</p> <p>For example:</p> <pre>http://genesis2.jpl.nasa.gov/perl/oai2?verb=ListRecords&amp;metadataPrefix=oai_dc&amp;set=7375626A656374733D67656E657369732D707562</pre> <p>In this case the list of records is filtered by set.</p>
GetRecord	<p>Requests an individual record.</p> <p>For example:</p> <pre>http://genesis2.jpl.nasa.gov/perl/oai2?verb=GetRecord&amp;metadataPrefix=oai_dc&amp;identifier=oai%3AGenericEPrints%3A200701001</pre>

## Extra parameters

Some verbs require or allow extra parameters. Extra parameters are:

Parameter	Description
from	Optional for ListIdentifiers, ListRecords. For example: from=1998-01-15 This restricts the results to records modified on or after the given date.
until	Optional for ListIdentifiers, ListRecords. For example: until=2008-01-21 This restricts the results to records modified on or before the given date.
set	Optional for ListIdentifiers, ListRecords. For example: set=Cook_Collection Records can be defined as belonging to various sets (defined by the institution). This parameter restricts the results to records that belong to a particular set.
metadataPrefix	Required for GetRecord, ListIdentifiers or ListRecords.
identifier	Required for GetRecord.
resumptionToken	For flow control (page 9).

## XML

Returned record XML is wrapped in some minimal OAI XML and the record in its requested metadata is inserted in a `metadata` section.

For example, an XML response that uses Dublin Core Metadata might look something like:

```
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
    http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2011-01-27T23:01:20Z</responseDate>
  <request verb="GetRecord" metadataPrefix="oai_dc"
    identifier="oai:jair.org:806">http://jair.fetch.com/oai2.php</req
uest>
  <GetRecord>
  <record>
  <header>
    <identifier>oai:jair.org:</identifier>
    <timestamp>2001-11-01T00:00:00Z</timestamp>
  </header>
  <!-- METADATA SECTION -->
  <metadata>
    <oai_dc:dc
      xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
        http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
      <dc:title>Infinite-Horizon Policy-Gradient
Estimation</dc:title>
      <dc:description>Gradient-based approaches to direct policy
search in
reinforcement learning have received much
recent attention as a means
to solve problems of partial observability
and to avoid some of the
problems associated with policy degradation
in value-function methods.
</dc:description>
      <dc:publisher>Journal Of Artificial Intelligence
Research</dc:publisher>
      <dc:contributor>P. L. Bartlett</dc:contributor>
      <dc:contributor>J. Baxter</dc:contributor>
      <dc:date>2001-11-01 00:00:00</dc:date>
      <dc:type>Text</dc:type>
      <dc:format>application/pdf</dc:format>
      <dc:identifier>doi:10.1613/jair.806</dc:identifier>
      <dc:language>en</dc:language>
      <dc:rights>Copyright held by the AI Access
Foundation</dc:rights>
    </oai_dc:dc>
  </metadata>
  </record>
  </GetRecord>
</OAI-PMH>
```

## Flow control

Responses that return lists of data (`ListIdentifiers`, `ListRecords`, `ListSets`) may be returned from the repository as an incomplete list because there is too much data to return in one hit. XML returned will be valid but will not have all records. In this case the repository will return a resumption token with the first block of records. The harvester sends another request with the resumption token to the repository. For example:

```
http://AN.OAI.ORG/OAI-  
script?verb=ListIdentifiers&resumptionToken=xxx45abtttyz
```

The response to a passed resumption request may also be incomplete, in which case a new resumption token will be received. This process is repeated as many times as required to return the complete set of data.





## SECTION 3

# Setting up a KE IMu OAI web service

The IMu system can provide an OAI repository for an EMu installation. For the most part setting up a basic IMu/OAI service is a matter of extracting the IMu components and configuring them as required. If the required service uses Dublin Core (or other metadata formats that KE has already implemented for other customers), there should be no real need for any coding to be done.

---

## Configuration

The OAI service consists of two types of components:

1. The service itself.
2. Any metadata formats provided by the service (of which there must be one but may be many).

Configuration of the service and of each metadata format is done separately.

## How localisation is done

There are three levels of localisation:

Level	Description
<code>common</code>	KE provides default configuration settings for the OAI service and basic configuration settings for the mandatory <code>oai_dc</code> metadata format. These settings are common to all customers. Typically they comprise a minimum set of field mappings, XML namespaces, etc.
<code>client</code>	<p>Each customer will also have their own customised configurations for their service and metadata requirements. KE sets these up and distributes them with the IMu code for the customer.</p> <p>These override the <code>common</code> settings. Typically they include things like the customer's name, modules, etc.</p> <div data-bbox="718 992 775 1046" data-label="Image"> </div> <p>These client specific settings will be replaced by the KE version of the customer's setup whenever IMu is upgraded by KE. If you edit files here, make sure the changes are sent to KE to prevent them being overwritten during an upgrade.</p>
<code>local</code>	<p>A site specific configuration overrides settings supplied by KE at the <code>common</code> and <code>client</code> levels. Settings here will not be overwritten by a KE upgrade and are solely for the customer to manage and change at will.</p> <p>Typically they include things such as server details, set definitions that may be changed or added to in an ad-hoc way, admin email addresses, etc.</p>

Configuration settings can be placed at any of the three levels. Settings made at the `local` level will override any set at the `client` level and the `client` level settings will override those at the `common` level.

---

## Where Configuration files are located

Configuration files can be located in three places:

Level	Location
Default settings ( <code>common</code> )	<code>~imu/ws/oai/common</code>
Customer specific settings ( <code>client</code> )	<code>~imu/ws/oai/client</code>
Site specific settings ( <code>local</code> )	<code>~imu/ws/oai/local</code>

Any settings stored in `local` override those stored in `client` and `client` settings override `common` settings.

---

## File Names

### OAI Service

The OAI service configuration is contained in files named `config.xml` placed in any (or all) of the following directories:

`~imu/ws/oai/common`

`~imu/ws/oai/client`

`~imu/ws/oai/local`

## Metadata formats

Any metadata settings (e.g. metadata field mappings to EMu fields, etc.) are placed in files typically named after the metadata format.

For example:

- `oai_dc.xml` has Dublin Core field mappings.
- `oai_pa.xml` has Picture Australia field mappings.

These can be placed in any (or all) of the following directories:

`~imu/ws/oai/common`

`~imu/ws/oai/client`

`~imu/ws/oai/local`

---

## Configuration File Structure

The configuration files are XML and can represent various settings, including:

- Simple setting=value type items
- Lists of values (e.g. a list of field mappings)
- Associative arrays

## Simple settings

Simple settings are just name=value and are represented as:

```
<earliestDatestamp>1970-01-01</earliestDatestamp>
```

This translates as the earliestDatestamp is 1970-01-01



## A list of items

A simple list of items is represented as:

```
<EMuModules type="array">
  <item>ecatalogue</item>
  <item>eparties</item>
  <item>emultimedia</item>
  <item>enarratives</item>
</EMuModules>
```

This indicates that four EMu modules can participate in the service and they are:

- ecatalogue
- eparties
- emultimedia
- enarratives

The parameter is of type "array" and has one or more items in it.

## Associative array of items

Where items in a list need to be named to distinguish them from each other, an attribute called `key` is used.

For example:

```
<metadataHandlers type="array">
  <item key="oai_dc" type="array">
    <item key="description">default Dublin Core Handler</item>
    <item key="configuration">oai_dc.xml</item>
  </item>
  <item key="oai_pa" type="array">
    <item key="description">default Picture Australia
Handler</item>
    <item key="configuration">oai_pa.xml</item>
  </item>
  <item key="umbl" type="array">
    <item key="description">UMBL mixed DC and EAD
Handler</item>
    <item key="configuration">umbl.xml</item>
  </item>
</metadataHandlers>
```

This XML indicates that there is a set of three `metadatahandlers`, with the names `oai_dc`, `oai_pa` and `umbl`. Each of these handlers has two properties, `description` and `configuration`.

---


## OAI Service Configuration

The root element of the main service OAI configuration XML should be named `EMuOaiConfig`. As it contains a list of values it has an attribute to say it contains a number of items (an array):

For example:

```
<EMuOaiConfig type="array">  
  ...  
  ...  
</EMuOaiConfig>
```

## Some important parameters

Parameter	Description
repositoryName	Displayed on the identify response. For example: The Museum Of Nature's OAI Repository
host	The host of the EMu IMu Server.
port	The port the EMu IMu server is listening on.
identifierScheme	Used to build unique resource identifiers.
identifierNamespace	Used to build unique resource identifiers.
identifierLocalPrefix	Used to build unique resource identifiers.
adminEmail	Email of the person who administers the service.
granularity	Can you search to nearest day or nearest second?
earliestDatestamp	What is the earliest 'from value' that can be requested?
biteSize	The maximum number of records to be returned before a resumption token gets sent and the response completed. Used to throttle the amount of data sent in each transaction.
metadataHandlers (array)	A list of metadata types the service will provide. <div> Each will require their own configuration file.</div>
knownSets type (array)	A list of sets (and the criteria used to define them) that the service provides.
xslStylesheet	To allow human friendly display of the OAI responses you can specify an XSL stylesheet that will display the response in a styled way when the response is returned to a web browser.
EMuModules type (array)	A list of EMu modules that the service will allow direct access to as records.
deletedRecord	Should be set to no. The IMu OAI system does not support deleted record functionality at the moment but this parameter is provided as such support may be added in the future.

## Example

```
<EMuOaiConfig type='array'>
  <repositoryName>Museum of Quirky Stuff OAI
Repository</repositoryName>
  <identifierScheme>oai</identifierScheme>
  <identifierNamespace>quirkymuseum.edu</identifierNamespace>
  <identifierLocalPrefix>EMu.eQuirkyStuff</identifierLocalPrefix>

  <adminEmail>fred@quirkymuseum.edu</adminEmail>
  <granularity>day</granularity>
  <earliestDatestamp>1970-01-01</earliestDatestamp>
  <deletedRecord>no</deletedRecord>
  <biteSize>50</biteSize>

  <metadataHandlers type='array'>
    <item key='oai_dc' type='array'>
      <item key="description">default Dublin Core Handler</item>
      <item key="configuration">oai_dc.xml</item>
    </item>
  </metadataHandlers>

  <knownSets type='array'>
    <item key="multimedia" type='array'>
      <item key="name">multimedia</item>
      <item key="description">All Multimedia Records</item>
      <item key="EMuModule">emultimedia</item>
    </item>
  </knownSets>

  <xslStylesheet>./common/oai2.xsl</xslStylesheet>

  <EMuModules type='array'>
    <item>ecatalogue</item>
    <item>eparties</item>
    <item>emultimedia</item>
    <item>enarratives</item>
  </EMuModules>
</EMuOaiConfig>
```

---

## Metadata Configuration

Each metadata format provided by the service requires a separate configuration file.

The root element of the metadata configuration XML should be named `EMuOaiMetadataHandler`. As it contains a list of values it has an attribute to say it contains a number of items (an array).

For example:

```
<EMuOaiMetadataHandler type="array">
  ...
  ...
</EMuOaiMetadataHandler>
```

## Some important parameters

Parameter	Description
<code>namespaces (array)</code>	A list of <code>namespaces</code> that will be declared in the responses.
<code>holdingElement</code>	The root element of any metadata.
<code>holdingElementNamespaces</code>	<code>namespace</code> declarations that need to be declared in the metadata root element.
<code>fieldMappings (array)</code>	A list of mappings that define how the metadata fields are to be assembled.

## How field mappings are specified

Each field mapping consists of:

- The metadata field name  
-AND-
- A list of EMu modules and fields that contain that metadata.

A default mapping can be made which is used if no specific Module>EMu Field>Metadata Field is defined.

For example:

```
<item key="dc:title" type="array">
  <map key="ecatalogue">EADUnitTitle</map>
  <map key="emultimedia">MulTitle</map>
  <map key="default">SummaryData</map>
</item>
```

This specifies that the `dc:title` metadata field comes from the EMu field `EADUnitTitle` for Catalogue records, from the `MulTitle` field for Multimedia records and from `SummaryData` for records of any other type.



## Special field mappings

In addition to field data coming directly from EMu fields, it is also possible to specify that returned values are either *hard-coded* to always return a fixed value or *templated* to return a mixture of hard-coded text plus data from various EMu fields.

For example, a hard-coded mapping:

```
<item key="dodgy" type="array">
  <map key="default" modifier="static">THIS METADATA FORMAT IS AN
  EXAMPLE - configure as required!</map>
</item>
```

This specifies that the "dodgy" metadata field will always be returned for records from every module and will always have the value THIS METADATA FORMAT IS AN EXAMPLE - configure as required!

A templated mapping:

```
<item key="multimediaLink" type="array">
  <map key="ecatalogue"
  modifier="dynamic">http://someserver?irn=[MulMultiMediaRef_tab.ir
  n]&thumb=no</map>
</item>
```

This specifies that the multimedia metadata field will be returned for Catalogue records only and will be made up of a mixture of hard-coded text plus the value of the EMu field:

"MulMultiMediaRef\_tab->emultimedia->irn" substituted into it

So for actual records it might appear as:

```
http://someserver?irn=123&thumb=no
...
http://someserver?irn=9856&thumb=no
...
http://someserver?irn=325002&thumb=no
etc
```

For example, using the earlier mapping:

```
<item key="multimediaLink" type="array">
  <map key="ecatalogue"
  modifier="dynamic">http://someserver?irn=[MulMultiMediaRef_tab.irn]&a
  mp;thumb=no</map>
</item>
```

and assuming that the Catalogue record being harvested has a value of 98765 in its *MulMultiMediaRef\_tab* field, the OAI system would create a value for the *multimediaLink* field of:

http://someserver?irn=98765&thumb=no

## Example of a metadata configuration file

```
<EMuOaiMetadataHandler type="array">
  <namespaces type="array">

    <namespace>http://www.openarchives.org/OAI/2.0/oai_dc/</namespace>

    <namespace>http://www.openarchives.org/OAI/2.0/oai_dc.xsd</namespa
    ce>

    <namespace>http://www.openarchives.org/OAI/2.0/oai_dc/</namespace>
    </namespaces>

    <holdingElement>oai_dc:dc</holdingElement>

    <holdingElementNamespaces>
      xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:dcterms="http://purl.org/dc/terms/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
        http://www.openarchives.org/OAI/2.0/oai_dc.xsd"
    </holdingElementNamespaces>

    <fieldMappings type="array">
      <item key="dc:title" type="array">
        <map key="default">SummaryData</map>
        <map key="ecatalogue">EADUnitTitle</map>
        <map key="emultimedia">MulTitle</map>
      </item>
      <item key="dc:creator" type="array">
        <map key="ecatalogue">CreCreatorRef_tab.SummaryData</map>
        <map key="emultimedia">MulCreator_tab</map>
      </item>
      <item key="dc:subject" type="array">
        <map key="ecatalogue">TitObjectCategory</map>
        <map key="emultimedia">DetSubject_tab</map>
      </item>
      <item key="dc:description" type="array">
        <map key="default">ExtendedData</map>
        <map key="ecatalogue">EADScopeAndContent</map>
        <map key="emultimedia">MulDescription</map>
      </item>
      <item key="dc:publisher" type="array">
        <map key="emultimedia">DetPublisher</map>
      </item>
      <item key="dc:contributor" type="array">
        <map key="emultimedia">DetContributor_tab</map>
      </item>
      ...
      ...
      ...
    </fieldMappings>
  </EMuOaiMetadataHandler>
```

# Index

## A

- A list of items • 12
- Associative array of items • 13

## C

- Configuration • 9
- Configuration File Structure • 12

## E

- Example • 16
- Example of a metadata configuration file • 19
- Extra parameters • 5

## F

- File Names • 11
- Flow control • 5, 7

## H

- How field mappings are specified • 17
- How localisation is done • 9

## I

- Interface  
the Six verbs • 3

## M

- Metadata Configuration • 17
- Metadata formats • 11

## O

- OAI - a Brief Tour • 3
- OAI Service • 11
- OAI Service Configuration • 14

## S

- Setting up a KE IMu OAI web service • 9
- Simple settings • 12
- Some important parameters • 14, 17
- Some OAI Terminology and Practice • 2
- Special field mappings • 18

## W

- What is OAI? • 1

- Where Configuration files are located • 10
- Where to find general information on OAI • 1

## X

- XML • 6



## EMu Documentation

# EMu Read-Only Modes

Document Version 1

**EMu Version 4.0.03**





# Contents

<b>SECTION 1</b>	<b>EMu Read Only Modes</b>	<b>1</b>
	Overview	1
	System ReadOnly Registry setting	2
	Table ReadOnly Registry Setting	4
	Combining system and table ReadOnly Registry entries	6
	Examples	7
	<b>Index</b>	<b>9</b>



## SECTION 1

# EMu read-only modes

---

## Overview

KE EMu 4.0.03 adds two new Registry entries to make all or part of the system read-only. These entries allow system administrators to "turn off" record insertions and updates while still allowing the system to be operational. Prior to EMu 4.0.03 it was possible to make the back-end tables read-only (and it still is), however the EMu client did not reflect the read-only nature of the table in the module interface: it was still possible to select the menu entry to insert a new record, but an error would then be displayed as the table was read-only. The `ReadOnly` Registry entries control the EMu client while still leaving the back-end tables operational. This means that the menu entry to insert a new record is now disabled if the table or system is marked as read-only.

The system wide read-only entry ensures that data is not modified in any modules, including the EMu Registry. Some uses for this entry include:

- Disabling data changes while the system is upgraded.
- Providing read-only access for certain users / groups (e.g. students).
- Allowing data loads to be checked before going live.

The table specific read-only entry allows individual tables, or all tables, to operate in a read-only state. If a table is read-only, data insertions and modifications are not permitted. The only exception is that changes to the EMu Registry are permitted even if the `eregistry` table is set to read-only. Possible uses for this entry include:

- Disabling data changes to a single table (as it may require maintenance).
- Providing read-only access to a table for certain users/groups (e.g. students).
- Allowing Registry based operations (e.g. create a new sort) while the data is read-only.

The addition of the two `ReadOnly` Registry entries provides system administrators with the ability to control access to data stored in EMu at a system wide and table specific basis.

The `ReadOnly` Registry entries work alongside existing EMu Registry entries. For example, if a table is designated as read/write by a `ReadOnly` Registry entry, a user still needs the `daInsert` operational privilege to be able to create records. The `ReadOnly` entries do not override existing Registry entries to provide more privileges than would be the case if the entry was not set.



---

## System ReadOnly Registry setting

The format of the system ReadOnly Registry setting is:

```
System|Setting|ReadOnly|value  
Group|Default|Setting|ReadOnly|value  
Group|group|Setting|ReadOnly|value  
User|user|Setting|ReadOnly|value
```

where:

*value* is either true (read-only functionality is enabled) or false (read-only functionality is not enabled).

The system ReadOnly entry provides the following functionality:

- Data in all modules cannot be updated or created.
- Operations involving updates to tables are disabled, namely:
  - Creating groups (uses egroups table).
  - Manipulating the records in groups (uses egroups table).
  - Creating batch exports (uses eschedule table).
  - Running batch exports (uses eexports table).
  - Create task templates (uses etemplates table).
  - Change help contents (uses efieldhelp table).
- Operations that update the Registry are disabled, namely the creation, modification or deletion of any resources:
  - List Settings
  - Default Values
  - Ditto Record
  - Record Recall
  - Record Template
  - Reports
  - Shortcut Settings
  - Sorts
- Operations that update the Registry when performed, but only to record the entry selected are not disabled, namely:
  - List Settings
  - Page View
  - Query Default Values
  - Reports
  - Shortcut Settings
  - Sorts
  - Ad-hoc Sorts

The system `ReadOnly` entry may be used to limit access on a group or user basis. For example, if users in group `Student` may not update any data, the following entry may be used:

```
Group|Student|Setting|ReadOnly|true
```

Entries may also be used in combination to achieve the desired effect. For example, if all users except those in group `Admin` should have read-only access, the following entries may be used:

```
System|Setting|ReadOnly|true  
Group|Admin|Setting|ReadOnly|false
```

## Table ReadOnly Registry setting

The format of the table ReadOnly Registry setting is:

```
Group|Default|Table|Default|ReadOnly|value
```

```
Group|Default|Table|table|ReadOnly|value
```

```
Group|group|Table|Default|ReadOnly|value
```

```
Group|group|Table|table|ReadOnly|value
```

```
User|user|Table|Default|ReadOnly|value
```

```
User|user|Table|table|ReadOnly|value
```

where:

*value* is either `true` (all operations that would result in record creation or updates in the specified table are disabled) or `false` (update operations are allowed, subject to other permissions, e.g. operational permissions).

The one exception to the rule is the Registry table (eregistry). If the Registry is made read-only, only explicit changes, that is changes made from the Registry module, are disabled. All implicit changes (e.g. adding a new sort definition) are still permitted. Hence the creation, modification or deletion of any resources is permitted, namely:

- List Settings
- Default Values
- Ditto Record
- Record Recall
- Record Template
- Reports
- Shortcut Settings
- Sorts

Using the first form of the table ReadOnly Registry entry:

```
Group|Default|Table|Default|ReadOnly|true
```

disables data creation and updates in all tables while still permitting most operational commands (e.g. report creation). When compared with the system ReadOnly setting, the table setting restricts only data modifications, rather than data and operational functionality.

User/group based variants of the table Registry entry may be used to restrict access to a select number of individuals. For example, if users in group `Student` are not allowed to create or modify records in the Catalogue module, the following Registry entry may be used:

```
Group|Student|Table|ecatalogue|ReadOnly|true
```

As with the system `ReadOnly` entry, table based entries may be combined to produce the desired effect. For example, if users in group `Student` were only allowed to update the `egroups`, `eschedule` and `eexports` tables (allowing them to create groups and batch exports), then the following Registry entries may be used:

Group	Student	Table	Default	ReadOnly	true
Group	Student	Table	egroups	ReadOnly	false
Group	Student	Table	eschedule	ReadOnly	false
Group	Student	Table	eexports	ReadOnly	false

---

## Combining system and table ReadOnly Registry entries

In general, it is not wise to mix system based and table based ReadOnly Registry entries. Consider the following settings:

```
Group | Student | Setting | ReadOnly | true
Group | Student | Table | eparties | ReadOnly | false
```

Which entry takes precedence? Can users in group `Student` write to the `eparties` table? The current implementation gives priority to the system entries over table based entries. So for the example above, users in group `Student` would not be able to write to the `Parties` module.

## Examples

### Example 1

Your EMu installation is to be upgraded to the next release of the software. While the upgrade is proceeding you would like users to be able to view data in the live system, but not make any changes as they will be lost when the upgraded system replaces the live system. The following Registry entry may be used:

System	Setting	ReadOnly	true
--------	---------	----------	------

The above setting will disable all EMu functions that would result in any data changes. Users may still produce reports, sort records, etc., however data changes are not allowed.

### Example 2

You have received an initial data load in EMu ready for checking. You do not want users to modify any data as they are only determining the integrity of the load. In order to make the reporting of issues easier, you would like users to be able to create groups so they can group records where the data does not look correct. The following Registry entries may be used:

Group	Default	Table	Default	ReadOnly	true
Group	Default	Table	egroups	ReadOnly	false

The disabling of the read-only status on egroups allows users to create and manipulate groups.

### Example 3

You have created a duplicate version of your live system for access via the web. You do not want users to be able to alter any data in the web copy, however all other functionality should be available. The following Registry entries may be used:

Group	Default	Table	Default	ReadOnly	true
Group	Default	Table	egroups	ReadOnly	false
Group	Default	Table	eschedule	ReadOnly	false
Group	Default	Table	eexports	ReadOnly	false
Group	Default	Table	etemplates	ReadOnly	false
Group	Default	Table	efieldhelp	ReadOnly	false

The enabling of read/write access to the egroups, eschedule, eexports, etemplates and efieldhelp tables will allow full command functionality on a read-only system. You may want to exclude some of these tables if you want to further limit functionality. For example, removing efieldhelp will disable the updating of field level help.

### Example 4

You are undertaking a clean up of the Lookup List table (eluts) so you would like to restrict updates to lookup lists to users in group `Admin`. The following Registry entries may be used:

```
Group|Default|Table|eluts|ReadOnly|true
Group|Admin|Table|eluts|ReadOnly|false
```

When the `eluts` table is read-only, users may not insert new values into the table. This means **all** lookup lists are treated as read-only. Users may find this annoying as it may make it difficult to save records.

### Example 5

You have just received an allocation of staff to add field level help for all modules in the system. You do not want the staff to alter any other data apart from field level help. You have created a group called `Volunteers` and placed the allocated staff in the group. The following Registry entries may be used:

```
Group|Volunteers|Table|Default|ReadOnly|true
Group|Volunteers|Table|efieldhelp|ReadOnly|false
```

In order for the volunteers to be able to create the field level help for all modules, you would also need to allocate the `daEditHelp` operational privilege for group `Volunteers`.

# Index

## C

Combining system and table ReadOnly  
Registry entries • 6

## E

EMu Read Only Modes • 1

Examples • 7

## O

Overview • 1

## S

System ReadOnly Registry setting • 2

## T

Table ReadOnly Registry Setting • 4