# EMu Documentation Scheduled Operations

Document Version 1.1

EMu version 4.3





kesoftware.com ©2014 KE Software All rights reserved

# Contents

SECTION	1	Overview	1
SECTION	2	How to schedule an operation	3
		The Operation tab	3
		Delete Operation: the Delete tab	6
		Image Import Operation: the Image Import tab	8
		Merge Operation: the Merge tab	10
		Examples	12
		Scenario 1	12
		Scenario 2	14
SECTION	3	Viewing Operation Results	17
		View Result Files	17
		Save all Result Files	18
		Save a Result File	18
SECTION	4	How to create an additional type of Scheduled	
		Operation	19
		Storage of Scheduled Operations scripts	20
		Invoking a scheduled operation	22
		Accessing information from a Scheduled Operations record	23
		An example operation	25
		Useful functions that may be called from within an operation	30
		OpenLogFile	31
		FileLog	31
		GetStartPosition	32
		AddToProcessed	32
		GetAttachmentFields	33
SECTION	5	emuoperations	35
		Using emuoperations	35
		Configuring emuoperations	36
		Index	37

#### $S \in C T I O N 1$

## **Overview**

Operations

2

In order to use the Scheduled Operations facility, a user must have (or be a member of a group that has) Table Access to the Operations module (eoperations) and the daInsert operations permission.

The Scheduled Operations facility introduced with EMu 4.3 enables the scheduling of operations to be run immediately or at a specified date and time. Operations are scheduled in the Scheduled Operations module, which is accessed by selecting

Scheduled	Operations (1) -	Display					in the second	) X	
File Edit S	elect View To	ols Results	Tabs Winde	ow Help					
	380	21 🖉 🎔		14 4 3	н		6		?
Active - Imag	je Import, Compl	ete (emu), .	lohn Smith's di	gital assets (	fonation			2	9
Operation	01								
Name:	John Smith's dig	tal assets dona	tion	5	53				
Туре:	Image Import			Modu	le:			7	
Execution Commence:	<ul> <li>∩ At A Specifier</li> <li>immediately</li> </ul>	l Time P	lun Date:			Job State	us: CR CC	un omplete	
Completion N	otification							course)	
Notify:	1 Active - Syste	em Administrato	or; Administrator					<u> </u>	
Result Files	1								
1 /home/ke	/emu/nhmlive/logs	/operations/Im	age Import/2014	-03-07/Image Ir	nport-2-2014	03-07.txt		- e	ŝ
2 /home/ke	/emu/nhmlive/logs	/operations/Im	age Import/2014	-03-07/1mage lr	nport-results-2	2014-03-07.c	3V		
Operation	Image Import	Notes	Security	Audit	Admir				-
Display Reco	ord 2 of 13					emu	Admin	20088	10

in the Command Centre:

With the Scheduled Operations facility it is possible to define:

- The type of operation to run
- The module to apply the operation to
- A time to commence the operation
- People to notify when the operation is complete

A scheduled operation is defined and stored as a record in the Scheduled Operations module.

When a scheduled operation is run, any files created during the operation are listed in the *Result Files* table on the Operation tab. Result files can be viewed and saved.

Audit logs are produced for all scheduled operations, allowing suitably authorised users to search / view the results of all operations performed by all users.

EMu 4.3 supports three types of scheduled operation:

- Merge Records
- Delete Records
- Image Import

System Administrators may define additional types of Scheduled Operation as required. See How to create an additional type of Scheduled Operation (page 19) for details.



#### SECTION 2

## How to schedule an operation

Scheduling an operation is similar to creating any other record.

## **The Operation tab**

1. Select Operations in the Command Centre to display the Scheduled Operations module:

ile Edit S	elect View Too	ols Results Tab	Window H	lelp				_
0 🖬 🔘	000			-C > 31			•	•
Operation								
Type:	Delete		•	Module:			_	7
Execution Commence:	<ul> <li>At A Specified</li> <li>⊂ Immediately</li> </ul>	Time Run Dat Run Tim	e:		i -	Job Status:	œ R ⊂ C	un omplete
Completion No Notify:	tification							ল <b>ল</b>
Sec. 7 /	Γ							
Result Files								
-								
					and the second se			

- 2. Enter a descriptive name for the operation in the *Name: (Operation)* field.
- 3. Select the type of operation to be performed from the *Type: (Operation)* drop list. By default, there are three types of operation to choose from:
  - Delete (page 6)
     Delete a series of IRNs from a module.
  - Image Import (page 8) Import images from a directory into the Multimedia module.
  - Merge (page 10) Merge one or more records with a Target record in a module.





System Administrators may define additional operations as required. See How to create an additional type of Scheduled Operation (page 19) for details.

4. In the *Module: (Operation)* field, select the module in which the operation is to be performed.



When scheduling an Image Import it is not necessary to specify a module as emultimedia (the Multimedia module) is implicit to the operation (images are imported into the emultimedia table).

- 5. In the *Execution* group of fields specify the time that the operation will be executed. There are two options:
  - At A Specified Time

With this option selected it is possible to specify a *Run Date* and *Run Time* for the operation to commence its processing. This allows operations to be run outside of normal business hours or at the weekend.

A date and time specified here is the **earliest** that the operation will be run. The actual time at which an operation is run will depend on when the emuoperations script is scheduled to run (page 22): emuoperations is the script used to execute an operation that has been scheduled in a record in the Schedule Operations module (page 35). When emuoperations is run, it looks for any operations that were scheduled to run prior to the current date and time and commences them. Thus, if emuoperations is scheduled to run once per day, it will commence any operation scheduled to run in the previous 24 hours (in theory an operation could have been scheduled to run 23 hours and 59 minutes earlier). If emuoperations is to be run once per day, it probably makes sense therefore to schedule operations close to the time at which emuoperations is run. Alternatively, emuoperations can be run at various times throughout the day.

Immediately

With this option the operation will commence as soon as the record is saved.



Scheduled	Operations (1) - I	New							- x
File Edit S	elect View To	ols Results Ž↓ 🗗 🌮	Tabs W	indow H	telp ◀ ▶	ы		•	G ×
Operation Name: Type:	Delete records fr	om the Catalog.	e	•	Module:	ecatalogu	e		
Execution Commence:	<ul> <li>At A Specified</li> <li>C Immediately</li> </ul>	l Time Ru Ru	n Date: n Time:	05/07/20	14		Job State	JS: OFF CO	lun Complete
Completion N Notify:	*								<b>B</b>
Result Files									- 6
Operation	Delete	Notes	Security	ł	udt	Admin			
New Reco	ord 3 of 11						emu	Admin	20088

6. In *Notify: (Completion Notification)* attach the Parties record for anyone who is to be notified by email when the scheduled operation has completed.



Email notifications will only be received by parties added to the *Notify: (Completion Notification)* table if their Parties record includes a valid email address in the *Email: (Internet Details)* field.

*Job Status: (Execution)* indicates that the operation is waiting to be run, or that it has been run and is complete. Note that if an operation terminates unexpectedly, the status will remain as *Run* until the operation is restarted and it completes.



#### **Delete Operation: the Delete tab**

When **Delete** is selected from the *Type: (Operation)* drop list on the Operation tab, the Delete tab displays:

👷 Scheduled Operations (1) - New			
File Edit Select View Tools Results Tabs Window Help			
🗅 🖬 🕲 🖄 🖄 🐉 🌽 🖤 📾 📾 н к к 🕨 м		<b></b>	
			5
Module         Progress           ecatalogue         x	No. Processed	:	
Records To Delete			Intel
Records (IRN) To Summary Data			90
75			-
Operation Delete Notes Security Audit Admin	1		
New Record 3 of 11	emu	Admin	20088 //

- The *Module* field will list the module from which records will be deleted if a module was specified (Step 4) on the Operation tab (page 3).
   If a module was not selected on the Operation tab, specify in the *Module* field which module the records are to be deleted from.
- 2. In the *Records To Delete* table add the records that are to be deleted from the module specified in the *Module* field.

Records can be added through the attachment or drag and drop process:

- 2.1. Click 🖻 beside the *Records To Delete* table to open the module specified in the *Module* field.
- 2.2. Search the module for the record or records to delete and click Attach
  Current Record or Attach Selected Records in the Tool bar to add the record(s) to the *Records To Delete* table in the Scheduled Operations module.
  -OR-
- 2.3. Open the module specified in the *Module* field and search for the record or records to be deleted.
- 2.4. Select the record or records in List View and drag and drop them to the *Records To Delete* table in the Scheduled Operations module.
- 3. Save the record:



		DIAL CY NG D					B	D N
1				I4 4 >	<b>)</b>		0	
ctiv	re - Delete, ecat	alogue, Run 0:10 05/07/	2014 (emu	, Catalogue r	ecords to b	e deleted		
Mod	tule		Progress	7755				_
eca	talogue	7	No. To Pro	cess: 11	No	Processed	0	
Rec	ords To Delete							-
	Records (IRN) T	Summary Data					5.0	- 00
1	37340	69, HMS Serpent 7, 29/02/	1868, 14 12	N; 51 35 30 E; I				
2	13703	BM.1957,525(5), /08/1946,	53 40 N; 7	0 E; NA, Collec	ted by Dr J.D	.H.Wiseman		
3	30746	32, 14/06/1914, 12 31 S; 1	66 6 E; P					
4	24769	BM.1981,0156, 19, HMS H	ecla, 13/06/	1981, 13 26 18 1	N; 16 33 54 V	V: NA, Dietz	La F	
5	37340	69, HMS Serpent ?, 29/02/	1868, 14 12	N; 51 35 30 E; I	h.			
6	13703	BM.1957,525(5), /08/1946,	53 40 N; 7	0 E; NA, Collec	ted by Dr J.D	H.Wiseman		
7	30746	32, 14/06/1914, 12 31 S; 1	66 6 E; P					
8	24769	BM. 1981,0156, 19, HMS H	ecla, 13/06/	1981, 13 26 18	N; 16 33 54 V	V: NA, Dietz	La F	
9	21462	BM.1976,0417, 113, HMS H	lecla, 06/04	/1974, 57 28 20	N: 5 58 15 V	V; NA, Shipel	k grab	
10	32841	571, CS Britannia, 15/08/19	01, 5 34 45	S; 167 21 15 W	P. Sounding			
11	4621970	Botany, Flowering Plants, B!	0000				. 0	
Op	eration Dele	te Notes S	ecurty	Audt	Admin			
100.00	and a second second		COLD COLD					



#### **Image Import Operation: the Image Import tab**

When **Image Import** is selected from the *Type: (Operation)* drop list on the Operation tab, the Image Import tab displays:

🔜 Scheduled Operations (1) - New			×
File Edit Select View Tools Results Tabs Window Help			
□ 🖬 🕲 🖻 🎦 🛃 🛹 🌮 📾 📾 🔤 н к к → м	1	6	2
	Г		7
Directory Path To Image Files			-
I Image Import Identifier			
	_	_	
Operation Image Import Notes Security Audit Admin			_
New Record 3 of 13 emu Admin		20088	

1. In *Directory Path To Image Files*, enter the pathway to the image files to be imported.

The path may be a full path: /home/emu/..

or a relative path:

~/../.. or ../..

- 2. If required, enter an identifier in the *Image Import Identifier* field. The value entered here will be stored in the *Import Identifier* field on the Admin tab of all Multimedia records created through this scheduled import.
- 3. Save the record:



Scheduled Operations (1) - Display				×
File Edit Select View Tools Results Tabs Window Help				
		6	0	<b>N</b> ?
Active - Image Import, Run 0:10 05/07/2014 (emu), Images to be imported				7
Directory Path To Image Files				-
~/images/John Smith				
Image Import Identifier				
John Smith Donation				-
				-
	_		_	
Operation Image Import Notes Security Audit Admin				
Display Record 3 of 12	emu	Admin	20088	1



#### Merge Operation: the Merge tab

When **Merge** is selected from the *Type: (Operation)* drop list on the Operation tab, the Merge tab displays:

🔜 Scheduled Operations (1) - New	- 6	x c
File Edit Select View Tools Results Tabs Window Help		
D 🖬 🕲 🖸 🖄 Ži 🛃 🐼 🌮 💷 🕮 🔤 I4 4 → >I	•	R
		1
Module Progress No. To Processed:		
Target Record		00
Records To Be Merged With Target Records (IRN) To Summary Data		00
Operation Merge Notes Security Audit Admin		
New Record 1 of 1 emu Ad	min	20088

The *Module* field will list the module in which records will be merged if a module was specified (Step 4) on the Operation tab (page 3).
 If a module was not selected on the Operation tab, specify in the *Module* field in which module the merge will take place.

2. In the *Target Record* field add the record that will be the target of the merge (i.e. the record with which one or more records will be merged).

Records can be added through the attachment or drag and drop process:

- 2.1. Click 🖻 beside the *Target Record* field to open the module specified in the *Module* field.
- 2.2. Search the module for the Target Record and click Attach CurrentRecord in the Tool bar to add the record to the *Target Record* field in the Scheduled Operations module.

-0R-

- 2.3. Open the module specified in the *Module* field and search for the Target Record.
- 2.4. Drag and drop the Target Record to the *Target Record* field in the Scheduled Operations module. There are various ways to do this:
  - In List View click the record to drag and drop it on the *Target Record* field in the Scheduled Operations module.
  - Select the record in List View and drag the Drag Current Record button in the Tool bar to the *Target Record* field in the



Scheduled Operations module.

- Display the record in Details View and drag the Drag Current Record button in the Tool bar to the *Target Record* field in the Scheduled Operations module.
- 3. In the *Records To Be Merged With Target* table add the records that are to be merged with the Target Record Records can be added through the attachment or drag and drop process described earlier (page 6).
- 4. Save the record:

🔛 s	cheduled Operation	ons (1) - Display					eran el	9	×
File	Edit Select V	iew Tools Results	Tabs Windo	w Help					
۵		1 1 21 🖓 🎔		H 4 >	H			6	<b>h</b> ?
Acti	ive - Merge, ecat	alogue, Run 0:10 05	5/07/2014 (emu	), Catalogue (	record merge				9
Mo	adule atalogue		Progress No. To Pr	ocess: 5	No	Processe	d: 0		-1
Та [37	rget Record 7340	69, HMS Serperit 7, 2	29/02/1868, 14 12	N; 51 35 30 E	1				•
Re	cords To Be Merged	With Target							
	Records (IRN) To .	Summary Data						6	•
1	13703	BM. 1957, 525(5), /08/	/1946, 53 40 N; 7	10 E; NA, Colle	cted by Dr J.D.	H.Wisema	an		
2	30746	32, 14/06/1914, 12 3	1 S; 166 6 E; P						
3	24769	BM.1981,0156, 19, H	IMS Hecla, 13/06/	1981, 13 26 18	N; 16 33 54 W	NA, Diet	z La Fond	77	
4	21462	BM.1976,0417, 113,	HMS Hecla, 06/04	/1974, 57 28 2	0 N: 5 58 15 W	NA, Ship	oek grab		
5	32841	571, CS Britannia, 15	/08/1901, 5 34 45	S; 167 21 15 W	P. Sounding				
*									
0	peration Mer	ge Notes	Security	Audt	Admin				
Disp	Nay Record 3 of 1	3				emu	Admin	20088	3

## **Examples**

#### **Scenario 1**

A record clean up project is under way. As part of the clean up we wish to merge five variations of John Smith's Parties record into one. As users are still entering records, we need to wait until 1 July before we can run the Merge.

#### **Solution**

1. Add a Scheduled Operations record with a *Type* of Merge for eparties, scheduled to run at 12:10 AM on 1 July:

Operation         Name:       John Smith merge         Type:       Merge <ul> <li>Module:</li> <li>eparties</li> </ul> Execution <ul> <li>At A Specified Time</li> <li>Run Date:</li> <li>01/07/2014</li> <li>Job Status:</li> <li>Ru</li> <li>Commence:</li> <li>At A Specified Time</li> <li>Run Date:</li> <li>00:10</li> <li>Configuration</li> </ul> Completion         Notification             Notify: <ul> <li>Active - System Administrator: Administrator</li> <li>Result Files</li> </ul>	rge, eparti	es, Run, John Si	mith merge						<u> </u>	
Execution Commence:  At A Specified Time Run Date: 01/07/2014 Job Status:  Commediately Run Time: 00:10 Completion Notification Notify:  1 Active - System Administrator: Administrator Result Files	peration ame: ype:	John Smith merge Merge			•	Module:	eparties			7
Completion Notification Notify: 1 Active - System Administrator: Administrator Result Ries	xecution commence:	<ul> <li>At A Specified</li> <li>○ Immediately</li> </ul>	Time R R	un Date: un Time:	01/07/20	14		Job Status:	(∓ Rur (∩ Con	n nplete
Result Ries	ompletion No lotify:	fication           1         Active - System           *         *	m Administrato	r, Administrato	ŕ					<u> 1</u>
	lesult Files									6

- 2. Identify one of the five John Smith Parties records as the Target Record and attach it to the *Target Record* field on the Merge tab of the Scheduled Operation record.
- 3. Add the remaining four Parties records for John Smith to the *Records To Be Merged With Target* table:



Merge, eparties, Run, John Smith merge       Progress         Module       Progress         eparties       Image: Record         69683       Active - Mr John P. Smith; Smith         Records To Be Merged With Target       Image: Record (IRN) To         Records (IRN) To       Summary Data         1       69687         2       69686         Active - JP Smith; Smith
Module     Progress       eparties     No. To Processe:       Target Record       69683       Active - Mr John P. Smith; Smith       Records To Be Merged With Target       Records (IRN) To Summary Data       1       69687       Active - JP Smith; Smith       2       69686
Target Record         Active - Mr John P. Smith; Smith         Image: Second Sec
Records To Be Merged With Target           Records (IRN) To Summary Data           1         69687           2         69686           Active - JP Smith; Smith           2         69686
1         69687         Active - JP Smith; Smith           2         69686         Active - John Smith; Smith
3 69685 Active - J P. Smith: Smith 4 69684 Active - John P. Smith: Smith
*



#### **Scenario 2**

A large number of digital assets have been donated to your institution. Rather than load them individually, you would like to have them loaded automatically commencing immediately.

#### **Solution**

1. Add a Scheduled Operations record with a *Type* of Image Import to commence loading the digital assets immediately:

1 1 1 1 1 1 1		A.I	1 1-20 1020 1020	in the second second			DA IPA	1.0
		24 🕑 💙		14 4 1	H			
								- 3
Operation	53	-						
Name:	John Smith's dig	tal assets dona	stion					
Type:	Image Import			Module:				7
Completion N Notify:	Immediately  Iotification	R em Administrato	iun Time:			C	Complet	e
Result Files	1						_1	P
								Land I

When scheduling an Image Import it is not necessary to specify a module as emultimedia (the Multimedia module) is implicit to the operation (images are imported into the emultimedia table).

2. On the Image Import tab specify the directory where the digital assets are stored and an identifier for the created records:



Scheduled Operations (1) - New				x
File Edit Select View Tools Results Tabs Window Help				
		6	1	<b>k?</b>
				3
Directory Path To Image Files				
[∼/images/John Smith				
Image Import Identifier				
John Smith Donation				
				-
Onemation (Landa Mater Convolut Auda Admin				_
uperation image import notes Security Audit Admin				
New Record 1 of 1	emu	Admin	20088	1

When this record is saved the digital asset import will commence without the need for any further action from the user, who will be able to continue with their other work.

#### SECTION 3

# **Viewing Operation Results**

Scheduled operations are run automatically by EMu. For each operation executed a Results File is created and added to the *Result Files* table on the Operation tab of the Scheduled Operations record. The files are stored on the EMu server:

ctive - Ima	ge Import, Compl	ete (emu), J	ohn Smith's di	gital assets do	nation			
Operation Name:	John Smith's dig	tal assets dona	tion					_
Type:	Image Import			Module	¢ [			-
Execution Commence:	At A Specified     Immediately	l Time Ri Ri	un Date:			Job Status:	C RunC Complete	te
Completion N Notify:	Active - Syste	m Administrato	r, Administrator					
Result Files	1						1	-
1 /home/ka	e/emu/nhmlive/logs e/emu/nhmlive/logs	/operations/lm /operations/lm	age import/2014- age import/2014-	03-07/Image Imp 03-07/Image Imp	ort-2-2014-03 ort-results-20	14-03-07.csv		

## **View Result Files**

Select Results>Launch Viewer>[Result File] in the Menu bar.
 -OR-

Select the row in the *Result Files* table with the file to be viewed & click The application / viewer associated with the file extension is invoked to display the file.



## **Save all Result Files**

1. Select beside the *Result Files* table -OR-Select **Results>Save>All** in the Menu bar.

The Browse for Folder dialogue displays.

- 2. Choose the directory into which all Result Files will be saved.
- 3. Select OK

## Save a Result File

1. Select **Results>Save>**[*Result File*] in the Menu bar:

Scheduled	Operations (1) -	Display					UC.N	0 X
File Edit S	elect View To	ools Results	] Tabs Wind	low Help				
	3 8 1	21 P Lau	nch Viewer	14 4 3	<b>b</b> 1		6	10 K?
Active - Ima	ge Import, Comp	lete venuy.	ANTI CHINTTO U	AJI			_	29
Operation Name:	John Smith's dig	ital assets don	ation	Image Imp	ort=2=2014=0 ort=results=2	014-03-	07.csv	
Type:	Image Import		hann	Module:		_		7
Commence: Completion N Notify:	At A Specifie     Immediately      Introduction     Active - Syst     *	d Time F F em Administrato	lun Date:		_	Job Stat	us: C	Run Complete
Result Files	s/emu/nhmlive/logi s/emu/nhmlive/logi	s/operations/In s/operations/In	age Import/2014 age Import/2014	1-03-07/image impo 1-03-07/image impo	nt-2-2014-03 nt-results-201	-07.bd  4-03-07/	CSV	- @ 
Operation	Image Import	Notes	Security	Audit	Admin			
Display Rec	ord 2 of 13					emu	Admin	20088

The Save As dialogue displays.

2. Choose the location to save the Result File and click



ł

Save...

#### SECTION 4

# How to create an additional type of Scheduled Operation

EMu provides three Scheduled Operations functions by default:

- Delete
- Image Import
- Merge

In this section we examine how System Administrators can create an additional type of Scheduled Operation.



## **Storage of Scheduled Operations scripts**

Each type of Scheduled Operation (e.g. Delete, Merge, etc.) is defined by a script which resides under the etc/operations or local/etc/operations directory on the EMu server.

When adding a script for an additional type of Scheduled Operation for your EMu system, place it under local/etc/operations to avoid the risk of having it overwritten during EMu upgrades.

The script includes the name of the operation which will be listed in the *Type:* (*Operation*) drop list on the Operation tab of the Scheduled Operations module.

When the emuoperations process runs it scans the etc/operations and local/etc/operations directories to locate the scripts for all types of Scheduled Operations (files that end with a .pl extension) and registers a name for each type of operation found. The following example registers the Delete Scheduled Operation:

```
sub
Register
{
    my $plugins = shift;
    #
    # We handle the "Delete" method.
    #
    $plugins->{"Delete"} = \&Delete;
}
```

When a new type of Scheduled Operation is added to EMu, a Lookup List entry needs to be added to the *Operation Type* Lookup List. For the above example a Lookup List record was added to the *Operation Type* Lookup List with a value of Delete:



Lookup Lists (1) - Display	
File Edit Select View Tools Tabs Multimedia Window Help	
	<b>ⓑ ∖</b> ?
Operation Type: Delete	1964313
Lookup List Name         Persistent         Hidden         Sort Order           Operation Type         Image: Construction Type         Image: Con	
Values	
1 Delete	
2	
3	
4	
5	
9	
10	
Lookup 1 Lookup 2 Lookup 3 Notes Mutimedia Security Audit	Adala
Display Record 3 of 3	20088



## **Invoking a scheduled operation**

When scheduling an operation in a record in the Scheduled Operations module, the operation can be scheduled to commence:

- At A Specified Time -OR-
- Immediately

If *Commence: (Execution)* is set to Immediately, the operation will be invoked as soon the Scheduled Operations record is saved. The operation will commence running on the EMu server and control returned to the user to continue with their work.

If *Commence: (Execution)* is set to At A Specified Time, the operation will be invoked by the emuoperations script on the EMu server at the appropriate time.

The execution of each pending operation consumes a licence in the same way that a user would consume a licence to complete the task. Similarly to users performing tasks, multiple operations can be run simultaneously up to the system licence limit.

The emuoperations script is designed to be run from cron with an entry similar to the following:

30 17 \* \* \* /home/ke/emu/client/bin/emurun emuoperations 2>&1 | /home/ke/emu/client/bin/emurun emulogger -t "KE EMu Operations" -z operations

The script is typically run once per day but can be configured to run any number of times during the day. When the emuoperations script runs it looks for any operations that were scheduled to run prior to the current date and time and commences them.

i

A date and time specified in a Scheduled Operations record is thus the **earliest** that the operation will be run. The actual time at which an operation is run will depend on when the emuoperations script is scheduled to run (page 22): emuoperations is the script used to execute an operation that has been scheduled in a record in the Schedule Operations module (page 35). When emuoperations is run, it looks for any operations that were scheduled to run prior to the current date and time and commences them. Thus, if emuoperations is scheduled to run once per day, it will commence any operation scheduled to run in the previous 24 hours: in theory an operation could have been scheduled to run 23 hours and 59 minutes earlier. If emuoperations is to be run once per day, it probably makes sense therefore to schedule operations close to the time at which emuoperations is run. Alternatively, emuoperations can be run at various times throughout the day.

emuoperations will also re-run any previous operations that did not complete.

## Accessing information from a Scheduled Operations record

Each type of Scheduled Operation registers a function that is called to process the operation. For example, the Delete Scheduled Operation is performed by a registered function called Delete.

```
sub
Register
{
    my $plugins = shift;
    #
    # We handle the "Delete" method.
    #
    $plugins->{"Delete"} = \&Delete;
}
```

The function is passed two parameters:

- An IMu session which allows access to EMu records to perform the operation.
- A hash of data from a Scheduled Operations record with details about this particular operation (i.e. when, what records are affected, what module, etc.).
   sub

```
Delete
{
    my $imusession = shift;
    my $record = shift;
    #
    # Run the "Delete" operation.
    #
    ...
}
```



The list of keys available in the hash are:

Irn	The IRN of a record in the Scheduled Operations module with details about this scheduled operation.
Name	The name of the operation.
Туре	The type of operation.
Module	The module the operation is to be performed on.
ActionIrn	The target IRN for the Merge operation.
IrnsToProcess	The list of IRNs that the operation needs to process.
IrnsProcessed	The list of IRNs that the operation has already processed.
	Typically this would be an empty list except when an operation failed to complete.
Directory	The directory which contains files / information required by an operation to process.
Identifier	An identifier to add to records updated as part of running the operation.

The values for the keys are accessed through the <code>\$record</code> parameter, e.g.:

```
$record->{Module}
```

-0R-

```
@{$record->{IrnsToProcess}}
```



#### An example operation

In this example a list of IRNs is deleted:

```
#!/usr/bin/perl
use strict;
use warnings;
use lib "$ENV{EMUPATH}/utils/imu/lib";
use IMu::Module;
#
  Registration function.
#
no warnings 'redefine';
sub
Register
{
        my $plugins = shift;
        #
        #
           We handle the "Delete" method.
        #
        $plugins->{"Delete"} = \&Delete;
use warnings 'redefine';
#
#
   The handler for the "Delete" operation
#
sub
Delete
{
        my ($imusession, $record) = @ ;
        my ($attachments, $start, @deleteirns, $irn, $i);
        #
        # Check that we have the required information
        #
        if (! defined($record->{IrnsToProcess}) ||
@{$record->{IrnsToProcess}} == 0)
        {
                FileLog("Error: no irns supplied for
deletion");
                return(1);
        }
        elsif (! defined($record->{Module}) or $record-
>{Module} eg "")
        {
                FileLog("Error: delete module is not
defined");
                return(1);
```

```
#
        #
         Get the other information that we need to process
        #
        $attachments = GetAttachmentFields($record-
>{Module});
        @deleteirns = @{$record->{IrnsToProcess}};
        $start = GetStartPosition($record);
        FileLog("Running DELETE plugin for $record-
>{Module}");
        FileLog("%d records scheduled for deletion, starting
at position $start", scalar(@deleteirns));
        # Now delete each record in turn
        #
        for ($i = $start; $i < @deleteirns; $i++)</pre>
        {
                $irn = $deleteirns[$i];
                FileLog("Deleting irn $irn...");
                last if (! ProcessDeletion($imusession,
$attachments, $irn, $record));
                AddToProcessed ($irn);
        }
        return($i != @deleteirns);
# Do the actual deletion work
sub
ProcessDeletion
{
        my ($imusession, $attachments, $irn, $record) = @ ;
        my ($table, $colname, $module, @matches, $hits,
%found, $key, $column);
        eval
        {
                %found = ();
                foreach $key (keys %{$attachments})
                 {
                         # The assignment here is unusual but
it gets around an
                         # odd foreach scoping problem after
an exception is thrown.
                         $table = $key;
                         $module = IMu::Module->new($table,
$imusession);
                         foreach $column (keys
%{$attachments->{$table}})
```



**Scheduled Operations** 

```
{
                                  # Find records which match
this irn
                                 $colname = $column;
                                 $hits = $module-
>findTerms([$colname, $irn]);
                                 next if ($hits <= 0);</pre>
                                  #
                                  # Add records to found hash
                                  #
                                 FileLog("Found $hits matches
for $colname in $table");
                                 push(@{$found{$table}-
>{$colname}}, GetMatches($module));
                         }
                 }
        };
        if ($@)
        {
                FileLog("Error: failed to process $colname
in $table for irn $irn: $@");
                return(0);
        }
        @matches = keys %found;
        if (@matches)
        {
                 # Log that we cannot delete the record
                FileLog("Unable to delete irn $irn because
it is attached in the following places:");
                foreach $table (@matches)
                 {
                         foreach $colname (keys
%{$found{$table}})
                         {
                                 FileLog("\tModule: $table,
Column: $colname, Record(s): ".join(", ",
@{$found{$table}->{$colname}}));
                         }
                 }
        }
        else
        {
                 # Delete the record
                 DeleteRecord($imusession, $irn, $record);
        }
        #
        # Add irn to processed
```

```
#
        return(1);
}
# Delete the record
sub
DeleteRecord
{
        my ($imusession, $irn, $record) = @ ;
        my ($module, $hits, $result);
        eval
        {
                $module = IMu::Module->new($record-
>{Module}, $imusession);
                $hits = $module->findKey($irn);
                if ($hits > 0)
                 {
                         $result = $module->remove("start",
0, 1);
                         if (\$result == 0)
                         {
                                 FileLog("Failed to delete
irn $irn from $record->{Module}");
                         }
                }
                else
                 {
                         FileLog("Failed to find irn $irn in
$record->{Module}");
                }
        };
        if ($@)
        {
               FileLog("Failed to delete $irn from $record-
>{Module}: $@");
       }
}
# Get all the records that match the attachment query
sub
GetMatches
{
        my (\$module) = @;
        my ($result, @matches, $row);
        #
        # Get all of the records at once
        #
```

```
Qmatches = ();
        $result = $module->fetch("start", 0, -1, "irn");
        if ($result->{count})
        {
                 #
                # Get the irn for each row and push it to
the list of matches
                 #
                foreach $row (@{$result->{rows}})
                 {
                         push(@matches, $row->{irn});
                 }
        }
        return(@matches);
}
1;
```

# Useful functions that may be called from within an operation

The following functions are available to be called for use within an operation:

OpenLogFile (page 31)	Opens a results log file and adds it to the list of Result Files.
FileLog (page 31)	Writes a message to the standard operation Result File.
GetStartPosition (page 32)	Determines from what position to start processing the IrnsToProcess list.
AddToProcessed (page 32)	Adds the processed IRN to the <code>IrnsProcessed</code> list.
GetAttachmentFields (page 33)	Returns a hash of all attachment fields for a module.



#### **OpenLogFile**

Input parameters: Filename

Returns:

File Handle for writing and an indication if the file already exists (from a previous attempt to run the operation)

```
sub
DoSomething
{
    my $handle;
    my $exists;
    #
    # Open a file for logging results.
    #
    ($handle, $exists) = OpenLogFile("results.csv");
    if ($exists)
    {
        print $handle "...Resuming processing...";
    }
    ...
    close($handle);
```

#### **FileLog**

Input parameters:	Format string and parameters
Returns:	Nothing

```
sub
DoSomething
{
        #
        #
           Log a message.
        #
        FileLog("Error: no irns supplied for deletion");
        •••
        #
        #
           Log a formatted message.
        #
        FileLog("%d records scheduled for deletion, starting
at position $start", scalar(@deleteirns));
}
```

ł

#### **GetStartPosition**

Input parameters:	Record hash passed to operation
Returns:	Index into IrnsToProcess

```
sub
Operation
{
    my $imusession = shift;
    my $record = shift;
    my $start;
    #
    # Get the start position for processing the
    operation.
    #
    $start = GetStartPosition($record);
    ...
}
```

#### **AddToProcessed**

Input parameters:IRNReturns:Nothing

```
sub
Operation
{
    my $imusession = shift;
    my $record = shift;
    my $irn;
    ...
    #
    # Finished processing the operation on an irn.
    #
    AddToProcessed($irn);
    ...
}
```



#### GetAttachmentFields

Input parameters: Module

**Returns**:

A hash of modules with attachment columns to the requested module

```
sub
Operation
{
        my $imusession = shift;
        my $record = shift;
        my $attachments;
        my $module;
        my $column;
        ...
         #
           Get the attachment fields for the operation
         #
module.
         #
        $attachments = GetAttachmentFields($record-
>{Module});
        ...
         #
           Process the attachment fields.
         #
         #
        foreach $module (keys %{$attachments})
         {
                 foreach $column (keys %{$attachments-
>{$module}})
                 {
                          •••
                 }
         }
        ...
}
```

#### SECTION 5

## emuoperations

emuoperations is a script used to execute scheduled operations.

A date and time specified in a Scheduled Operations record is the **earliest** that the operation will be run. The actual time at which an operation is run will depend on when the emuoperations script is scheduled to run (page 22). When run, emuoperations looks for any operations that were scheduled to run prior to the current date and time and commences them. Thus, if emuoperations is scheduled to run in the previous 24 hours (in theory an operation could have been scheduled to run 23 hours and 59 minutes earlier). If emuoperations is to be run once per day, it probably makes sense therefore to schedule operations close to the time at which emuoperations is run. Alternatively, emuoperations can be run at various times throughout the day.

### **Using emuoperations**

emuoperations may be used in two ways:

#### 1. Run all Scheduled Operations

#### Usage: emuoperations

Any Scheduled Operations required to be run will be executed. The current date and time is used to determine what operations are required. This form of the command is used by cron on a daily basis to ensure all Scheduled Operations for the given day are performed.

#### 2. Run a specific Scheduled Operation

#### Usage: emuoperations -iirn

The *irn* argument is the IRN of a Scheduled Operations record to be executed. This form of emuoperations is useful for testing new operations as it allows a specific operation to be run without waiting for the Scheduled Operations date and time to arrive.



## **Configuring emuoperations**

The emuoperations script connects to an imuserver to perform the scheduled operations. This connection needs to be made on a specific port. By default the standard EMu configuration port for IMu is the port number 20,000 greater than EMu's client connection port. For example, if the standard EMu client connection port is 20000, the standard imuserver connection port is 40000.

The emuoperations imuserver must run on a different port to perform the scheduled operations. The eoperations load starts the imuserver for handling operation requests. The port for emuoperations to connect on is defined by the EMUSERVERPORT environment variable plus 30000. EMUSERVERPORT is the port the EMu client uses to connect to the EMu server and corresponds to the Service value entered in the EMu Client login box.

It is recommended that the Administrator sets the EMUSERVERPORT environment variable in the etc/config file on the EMu server. Add the following text to the end of the etc/config file (if it does not exist already):

```
#
# EMUSERVERPORT is the port the EMu client uses to connect to the
# EMu server.
# The port corresponds to the "Service" value entered in the EMu
# Client Login box.
EMUSERVERPORT=port
export EMUSERVERPORT
```

where *port* is the service name used to connect to this EMu server.



# Index

Scenario 2 • 14

Storage of Scheduled Operations scripts • 20

#### Т

A The Operation tab • 3, 6, 10 Accessing information from a Scheduled Operations record • 23 AddToProcessed • 30, 32 An example operation • 25 operation • 30 Using emuoperations • 35 С Configuring emuoperations • 36 View Result Files • 17 D Viewing Operation Results • 17 Delete Operation the Delete tab  $\bullet$  3, 6, 11 Е emuoperations • 4, 22, 35 Examples • 12 F FileLog • 30, 31 G GetAttachmentFields • 30, 33 GetStartPosition • 30, 32 Η How to create an additional type of Scheduled Operation • 2, 4, 19 How to schedule an operation • 3 I

Image Import Operation the Image Import tab • 3, 8

Invoking a scheduled operation • 4, 22, 35

#### Μ

Merge Operation the Merge tab • 3, 10

#### 0

OpenLogFile • 30, 31

Overview • 1

#### S

Save a Result File • 18 Save all Result Files • 18

Scenario 1 • 12



#### U

Useful functions that may be called from within an

#### V