

---

# **Ebury Audit Documentation**

***Release 0.4.0***

**José Antonio Perdiguero López**

January 19, 2016



<b>1</b>	<b>Installation</b>	<b>1</b>
1.1	Installation . . . . .	1
1.2	Configuration . . . . .	3
<b>2</b>	<b>Audit</b>	<b>7</b>
2.1	Signals . . . . .	7
2.2	Models . . . . .	8
2.3	Middleware . . . . .	10
2.4	Managers . . . . .	11
2.5	Commands . . . . .	13
<b>3</b>	<b>Additional information</b>	<b>15</b>
3.1	Custom providers . . . . .	15
<b>4</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



---

## Installation

---

### 1.1 Installation

To install *Audit* you need to follow the next steps:

1. Add MongoDB repository.
2. Install MongoDB.
3. Configure authentication for MongoDB.
4. Install Django Audit Tools package.

#### 1.1.1 Install MongoDB

##### Add repository

Import the public key used by the package management system:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

Create a list file for MongoDB repository:

```
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb-org-2.6.list
```

Reload local package database:

```
sudo apt-get update
```

##### Install MongoDB packages

Install the MongoDB packages:

```
sudo apt-get install mongodb-org=2.6.3 mongodb-org-server=2.6.3 mongodb-org-shell=2.6.3 mongodb-org-tools=2.6.3
```

(OPTIONAL) Pin a specific version of MongoDB:

```
echo "mongodb-org hold" | sudo dpkg --set-selections
echo "mongodb-org-server hold" | sudo dpkg --set-selections
echo "mongodb-org-shell hold" | sudo dpkg --set-selections
echo "mongodb-org-mongos hold" | sudo dpkg --set-selections
echo "mongodb-org-tools hold" | sudo dpkg --set-selections
```

Start MongoDB:

```
sudo service mongod start
```

### Configure authentication

Connect to MongoDB using client terminal:

```
mongo
```

Change to admin database:

```
use admin
```

Create admin user (replace <USER\_ADMIN> and <PASSWORD\_ADMIN>):

```
db.createUser({
  user: "<USER_ADMIN>",
  pwd: "<PASSWORD_ADMIN>",
  roles:
    [
      { role: "userAdminAnyDatabase", db: "admin" }
    ]
})
```

Activate authentication login uncommenting the next line in /etc/mongod.conf:

```
# auth = True
```

Restart MongoDB:

```
sudo service mongod restart
```

Connect to MongoDB as admin:

```
mongo admin -u <USER_ADMIN> -p <PASSWORD_ADMIN>
```

Change to audit database (replace <DATABASE>):

```
use <DATABASE>
```

Create audit user (replace <USER> and <PASSWORD>):

```
db.createUser({
  user: "<USER>",
  pwd: "<PASSWORD>",
  roles:
    [
      { role: "readWrite", db: "<DATABASE>" }
    ]
})
```

(OPTIONAL) Add environment variables appending next lines to virtualenv's activate file (replace <USER> and <PASSWORD>):

```
export DB_AUDIT_USER=<USER>
export DB_AUDIT_PASSWORD=<PASSWORD>
```

### 1.1.2 Install Audit

Use pip to install *Audit* from local file:

```
pip install django-audit-tools.tar.gz
```

Or install from pip repository:

```
pip install django-audit-tools
```

## 1.2 Configuration

To configure Audit Tools you need to follow the next steps:

1. Add *audit\_tools* to your **INSTALLED\_APPS** settings like this:

```
INSTALLED_APPS = (
    ...
    'audit_tools',
)
```

2. Add *audit.middleware.AuditMiddleware* to your **MIDDLEWARE\_CLASSES** settings like this:

```
MIDDLEWARE_CLASSES = (
    ...
    'audit_tools.audit.middleware.AuditMiddleware',
)
```

3. Configure blacklisted URLs in **AUDIT\_BLACKLIST** settings.
4. Register models that will be logged in **AUDIT\_LOGGED\_MODELS** settings.
5. Execute the next django command:

```
python manage.py prepare_audit
```

### 1.2.1 Settings

#### AUDIT\_ACTIVATE

Activate or deactivate audit.

Default:

```
AUDIT_ACTIVATE = True
```

#### AUDIT\_DB\_ALIAS

Audit database connection alias.

Default:

```
AUDIT_DB_ALIAS = 'audit'
```

## AUDIT\_DB\_CONNECTION

Audit database connection parameters.

Default:

```
AUDIT_DB_CONNECTION = {
    'HOST': 'localhost',
    'PORT': 27017,
    'NAME': 'audit',
    'USER': '',
    'PASSWORD': '',
}
```

## AUDIT\_RUN\_ASYNC

Use Celery to run in async mode.

**Important:** Celery concurrency level must be configure to 1 (`--concurrency=1` parameter in `celeryd` start)

Default:

```
AUDIT_RUN_ASYNC = False
```

## AUDIT\_CELERY\_QUEUE

Celery queue name.

Default:

```
AUDIT_CELERY_QUEUE = 'audit'
```

## AUDIT\_LOGGED\_MODELS

List of models that will be logged for audit. Each entry consists in a string that represents a model using “`<module>.<model>`” format.

Example:

```
AUDIT_LOGGED_MODELS = (
    'audit_tools.audit.models.Access',
)
```

Default:

```
AUDIT_LOGGED_MODELS = ()
```

## AUDIT\_BLACKLIST

Blacklisted URLs. Each application may have a tuple of regex patterns. If an URL matches a pattern will not be logged.

Example:



```
AUDIT_BLACKLIST = {
    'api': (
        r'^/api/.*',
        r'^/API/.*',
    )
}
```

Default:

```
AUDIT_BLACKLIST = {}
```

## AUDIT\_ACCESS\_INDEXES

Custom indexes for the accesses. There is the possibility to add new custom indexes to the Audit database.

Example:

```
AUDIT_ACCESS_INDEXES = [
    'custom.pools.names',
    'custom.pools.num_polls',
    ('custom.pools.names', 'custom.pools.num_polls'),
]
```

## AUDIT\_PROCESS\_INDEXES

Custom indexes for the processes. There is the possibility to add new custom indexes to the Audit database.

## AUDIT\_MODEL\_ACTION\_INDEXES

Custom indexes for the model actions. There is the possibility to add new custom indexes to the Audit database.

## AUDIT\_CUSTOM\_PROVIDER

Custom data provider. Each application may add custom data to Access entries using own functions.

Default:

```
AUDIT_CUSTOM_PROVIDER = {
    'audit_tools': 'audit_tools.audit.middleware.custom_provider',
}
```

## AUDIT\_TRANSLATE\_URLS

Translate Audit URLs:

Default:

```
AUDIT_TRANSLATE_URLS = False
```



## 2.1 Signals

`audit_tools.audit.signals.register(model)`

Register a model to the audit code.

**Parameters** `model` (*object*) – Model to register.

`audit_tools.audit.signals.register_models()`

Register all models listed in `settings.LOGGED_MODELS`.

`audit_tools.audit.signals.unregister(model)`

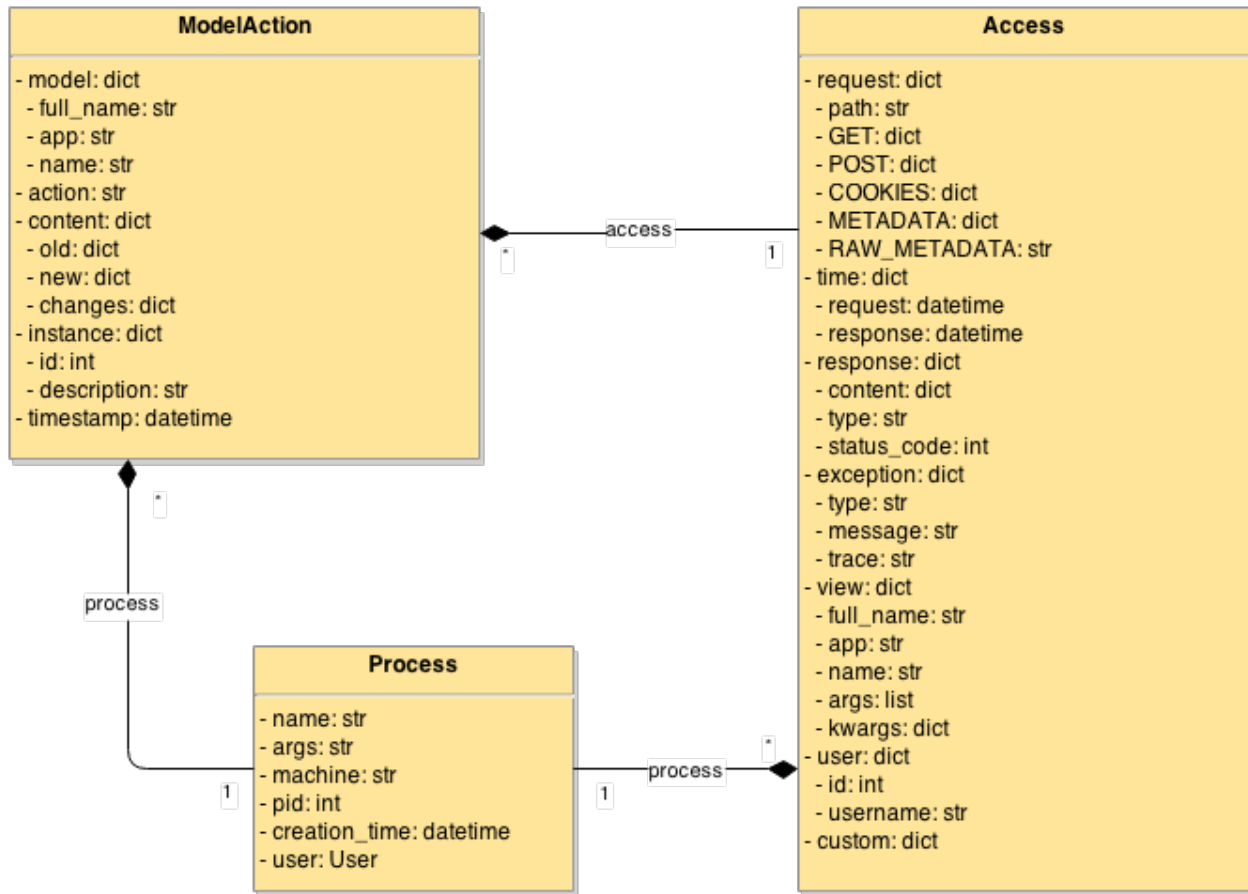
Unregister a model to the audit code.

**Parameters** `model` (*object*) – Model to unregister.

`audit_tools.audit.signals.unregister_models()`

Unregister all models listed in `settings.LOGGED_MODELS`.

## 2.2 Models



Ebury Audit models

**class** `audit_tools.audit.models.Access(*args, **values)`

Information gathered from a request and response objects. Contains the following structure:

### Variables

- **interlink\_id** – Interlink id.
- **request** – Request.
- **response** – Response content, type and status code.
- **exception** – Exception raised.
- **time** – Request and response time.
- **view** – View full\_name, app, name, args and kwargs.
- **user** – User id and name.
- **custom** – Custom providers field.
- **process** – Reference to Process object.

**get\_user()**

Obtains User object that corresponds to this Access instance.

**Returns** User object.

**Return type** django.contrib.auth.models.User

**get\_view()**

Obtains view object that corresponds to this Access instance.

**Returns** View.

**Return type** object

**is\_exception()**

Check if this Access contains a exception.

**Returns** True if contains exception.

**Return type** bool

**is\_response()**

Check if this Access contains a response.

**Returns** True if contains response.

**Return type** bool

**items()**

List items in form (key, value).

**Returns** list(tuple())

**url**

Property that gives access to request\_\_path field.

**Type** str

**verbose\_str()**

Verbose string representation.

**Returns** Verbose string representation.

**Return type** str

**class** audit\_tools.audit.models.**Process**(\*args, \*\*values)

Represents a process that launch a django management command. Contains the following structure:

#### Variables

- **name** – Process name.
- **args** – Process args.
- **machine** – Machine that launched this process.
- **user** – User that launched this process.
- **pid** – Process pid.
- **creation\_time** – Time when process was launched.

**items()**

List items in form (key, value).

**Returns** list(tuple())

**verbose\_str()**

Verbose string representation.

**Returns** Verbose string representation.

**Return type** str

**class** `audit_tools.audit.models.ModelAction(*args, **values)`

Information from create, update or delete operations over a model. Contains the following structure:

**Variables**

- **model** – Model full\_name, app and name.
- **action** – Create, update or delete action.
- **content** – Old and new values and changes.
- **instance** – Instance object id and description.
- **timestamp** – Time when action occurs.
- **access** – Reference to Access object.
- **process** – Reference to Process object.

**changes**

Property that gives access to content\_\_changes field.

**Type** dict

**get\_instance()**

Obtains instance object that corresponds to this ModelAction instance.

**Returns** Instance object.

**Return type** object

**get\_model()**

Obtains Model class that corresponds to this ModelAction instance.

**Returns** Model class.

**Return type** object

**items()**

List items in form (key, value).

**Returns** list(tuple())

**verbose\_str()**

Verbose string representation.

**Returns** Verbose string representation.

**Return type** str

## 2.3 Middleware

**class** `audit_tools.audit.middleware.AuditMiddleware(*args, **kwargs)`

Middleware for audit logging

**process\_view** (*self, request, view\_func, view\_args, view\_kwargs*)

Preprocess request.

**Parameters**

- **request** (*django.http.HttpRequest*) – Http request.
- **view\_func** (*callable*) – View.
- **view\_args** (*list*) – View arguments.

- **view\_kwargs** (*dict*) – View keyword arguments.

**Returns** None

**process\_response** (*self, request, response*)

Postprocess response.

**Parameters**

- **request** (*django.http.HttpRequest*) – Http request.
- **response** (*django.http.HttpResponse*) – Response.

**Returns** None

**process\_exception** (*self, request, exception*)

Postprocess exception.

**Parameters**

- **request** (*django.http.HttpRequest*) – Http request.
- **exception** (*Exception*) – Response.

**Returns** None

## 2.4 Managers

### 2.4.1 Access

The following methods are added to the default Access manager:

**class** `audit_tools.audit.managers.AccessQuerySet` (*document, collection*)

Custom manager for Access.

**filter\_by\_exception** (*\*args, \*\*kwargs*)

Filter object by exception.

**Parameters** **kwargs** – exc kwarg required.

**Returns** QuerySet

**filter\_by\_url** (*\*args, \*\*kwargs*)

Filtered queryset by url. Url accept all modifiers, including \_\_regex.

**Parameters** **kwargs** – url kwarg required.

**Returns** QuerySet

**filter\_by\_view** (*\*args, \*\*kwargs*)

Filtered queryset by view object or function.

**Parameters** **kwargs** – fview kwarg required.

**Returns** QuerySet

**get\_by\_exception** (*\*args, \*\*kwargs*)

Get object by exception.

**Parameters** **kwargs** – exc kwarg required.

**Returns** Access

**get\_by\_url** (\*args, \*\*kwargs)

Get object by url. Url accept all modifiers, including \_\_regex.

**Parameters** **kwargs** – url kwarg required.

**Returns** Access

**get\_by\_view** (\*args, \*\*kwargs)

Get object by view.

**Parameters** **kwargs** – fview kwarg required.

**Returns** Access

Examples:

```
# URL /
Access.objects.filter_by_url(url='/')
# Filter URLs using regular expression
Access.objects.filter_by_url(url=r'^/polls/\w*$')
# Accesses to /polls/ done by user with id 23
Access.objects.filter_by_url(url='/polls/', user__id=23)

# Accesses to poll's index
from polls.views import index
Access.objects.filter_by_view(fview=index)

# Accesses that raises an AttributeError exception
Access.objects.filter_by_exception(exc=AttributeError)
```

## 2.4.2 ModelAction

The following methods are added to the default ModelAction manager:

**class** audit\_tools.audit.managers.**ModelActionQuerySet** (document, collection)

Custom manager for ModelAction.

**filter\_by\_instance** (\*args, \*\*kwargs)

Filtered queryset by object instance.

**Parameters** **kwargs** – obj kwarg required.

**Returns** QuerySet

**filter\_by\_model** (\*args, \*\*kwargs)

Filtered queryset by model.

**Parameters** **kwargs** – klass kwarg required.

**Returns** QuerySet

**filter\_by\_model\_list** (\*args, \*\*kwargs)

Filtered queryset by model list.

**Parameters** **kwargs** – klass kwarg required.

**Returns** QuerySet

**get\_by\_instance** (\*args, \*\*kwargs)

Get object by instance.

**Parameters** **kwargs** – obj kwarg required.

**Returns** ModelAction



**get\_by\_model** (\*args, \*\*kwargs)

Get object by model.

**Parameters** **kwargs** – klass kwarg required.

**Returns** ModelAction

**get\_by\_model\_list** (\*args, \*\*kwargs)

Get object by model list.

**Parameters** **kwargs** – klass kwarg required.

**Returns** ModelAction

Examples:

```
# Actions done over all polls
from polls.models import Poll
ModelAction.objects.filter_by_model(klass=Poll)

# Actions done over a single poll
poll = Poll.objects.get(id=1)
ModelAction.objects.filter_by_instance(obj=poll)

# Actions done over all polls and users
from django.contrib.auth.models import User
ModelAction.objects.filter_by_model_list(klass=[Poll, User])
```

## 2.5 Commands

Django commands for Audit application.

### 2.5.1 drop\_audit

Drop audit database. If only initial date is given, data from that day to now. If the two dates are provided, data will be deleted in that range. Delete all data otherwise.

Syntax:

```
python manage.py drop_audit [init_date] [end_date]
```

Example:

```
python manage.py drop_audit
python manage.py drop_audit 01/01/2000
python manage.py drop_audit 01/01/2000 01/02/2000
```

### 2.5.2 prepare\_audit

Prepare system for Audit application adding needed permissions, tables...

Syntax:

```
python manage.py prepare_audit
```

### 2.5.3 remove\_audit

Remove Audit application from system.

Syntax:

```
python manage.py remove_audit
```

---

## Additional information

---

### 3.1 Custom providers

Custom provider is a mechanism that permits an application to add custom data to *Access* logs. A single provider can be specified for each app.

The next example define a provider that returns polls associated to current user:

```
def poll_provider(request):
    user = User.objects.get(pk=request.user.id)
    polls = Poll.objects.filter(user=user)
    polls_names = [p.name for p in polls]

    res = {
        'names': poll_names,
        'num_polls': len(poll_names),
    }

    return res
```

If this provider is defined inside `polls.utils` then must be set in `settings.AUDIT_CUSTOM_PROVIDER`:

```
AUDIT_CUSTOM_PROVIDER = {
    'polls': 'polls.utils.poll_provider',
}
```

This provider will result in an additional field inside *Access* named 'polls':

```
{
    ...
    custom = {
        polls = {
            names = [ "poll_1", "poll_2" ],
            num_polls = 2
        }
    }
}
```



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## **a**

`audit_tools.audit.managers`, [11](#)  
`audit_tools.audit.middleware`, [10](#)  
`audit_tools.audit.models`, [8](#)  
`audit_tools.audit.signals`, [7](#)





## A

Access (class in `audit_tools.audit.models`), 8  
 AccessQuerySet (class in `audit_tools.audit.managers`), 11  
`audit_tools.audit.managers` (module), 11  
`audit_tools.audit.middleware` (module), 10  
`audit_tools.audit.models` (module), 8  
`audit_tools.audit.signals` (module), 7  
 AuditMiddleware (class in `audit_tools.audit.middleware`), 10

## C

changes (attribute of `audit_tools.audit.models.ModelAction`), 10

## F

`filter_by_exception()` (method of `audit_tools.audit.managers.AccessQuerySet`), 11  
`filter_by_instance()` (method of `audit_tools.audit.managers.ModelActionQuerySet`), 12  
`filter_by_model()` (method of `audit_tools.audit.managers.ModelActionQuerySet`), 12  
`filter_by_model_list()` (method of `audit_tools.audit.managers.ModelActionQuerySet`), 12  
`filter_by_url()` (method of `audit_tools.audit.managers.AccessQuerySet`), 11  
`filter_by_view()` (method of `audit_tools.audit.managers.AccessQuerySet`), 11

## G

`get_by_exception()` (method of `audit_tools.audit.managers.AccessQuerySet`), 11  
`get_by_instance()` (method of `audit_tools.audit.managers.ModelActionQuerySet`), 12  
`get_by_model()` (method of `audit_tools.audit.managers.ModelActionQuerySet`), 12

`get_by_model_list()` (method of `audit_tools.audit.managers.ModelActionQuerySet`), 13  
`get_by_url()` (method of `audit_tools.audit.managers.AccessQuerySet`), 11  
`get_by_view()` (method of `audit_tools.audit.managers.AccessQuerySet`), 12  
`get_instance()` (method of `audit_tools.audit.models.ModelAction`), 10  
`get_model()` (method of `audit_tools.audit.models.ModelAction`), 10  
`get_user()` (method of `audit_tools.audit.models.Access`), 8  
`get_view()` (method of `audit_tools.audit.models.Access`), 9

## I

`is_exception()` (method of `audit_tools.audit.models.Access`), 9  
`is_response()` (method of `audit_tools.audit.models.Access`), 9  
`items()` (method of `audit_tools.audit.models.Access`), 9  
`items()` (method of `audit_tools.audit.models.ModelAction`), 10  
`items()` (method of `audit_tools.audit.models.Process`), 9

## M

ModelAction (class in `audit_tools.audit.models`), 9  
 ModelActionQuerySet (class in `audit_tools.audit.managers`), 12

## P

Process (class in `audit_tools.audit.models`), 9  
`process_exception()` (method of `audit_tools.audit.middleware.AuditMiddleware`), 11  
`process_response()` (method of `audit_tools.audit.middleware.AuditMiddleware`), 11  
`process_view()` (method of `audit_tools.audit.middleware.AuditMiddleware`), 10

## R

`register()` (in module `audit_tools.audit.signals`), [7](#)

`register_models()` (in module `audit_tools.audit.signals`), [7](#)

## U

`unregister()` (in module `audit_tools.audit.signals`), [7](#)

`unregister_models()` (in module `audit_tools.audit.signals`), [7](#)

`url` (`audit_tools.audit.models.Access` attribute), [9](#)

## V

`verbose_str()` (`audit_tools.audit.models.Access` method), [9](#)

`verbose_str()` (`audit_tools.audit.models.ModelAction` method), [10](#)

`verbose_str()` (`audit_tools.audit.models.Process` method), [9](#)