



EXPLORING MAYAN EDMS

THE DEFINITIVE GUIDE

From beginners to advanced users

1
Introduction

2
Tutorials

3
Reference

Roberto Rosario

Creator and lead developer of
Mayan EDMS

neeraj76@yahoo.com

neeraj76@yahoo.com

Exploring Mayan EDMS

For Mayan EDMS version 3.2

Roberto Rosario

Creator and lead developer of Mayan EDMS

First edition

Copyright Notice

Copyright 2019 Roberto Rosario, All Rights Reserved.

<https://robertorosario.com>

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including, but not limited to photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Mayan EDMS is a copyright of Roberto Rosario. The "Mayan EDMS" name and its logo are trademarks of Roberto Rosario. Trademarks may be registered in some jurisdictions. All other Trademarks are the property of their respective owners.

The book cover has been designed using resources from Freepik.com. <https://www.freepik.com/new7ducks>

First edition, August, 2019

TABLE OF CONTENTS

	Page
I. Preface	1
II. Getting started	9
1. Introduction	11
1.1 What is Mayan EDMS?	11
1.2 Features	11
1.3 Deployments	12
1.4 Open standards	12
1.5 Modular design	12
1.6 Where to get it	12
1.7 Browsers supported	13
1.8 Interface	13
III. Basic topics	17
2. Minimal concepts	19
3. Authentication	23
3.1 How-to	24
3.1.1 Login on	24
3.1.2 Login out	25
3.1.3 Resetting your password	25
3.2 Takeaways	26
4. Document types	29
4.1 How-to	31
4.1.1 Creating document types	31
4.2 Takeaways	31
5. Sources	33
5.1 How-to	35
5.1.1 Creating new sources	35
5.1.2 Testing sources	35
5.1.3 Uploading a document via a webform	35
5.2 Takeaways	36
5.3 Exercises	36
5.3.1 Upload your first document	36
6. Visualization	39

6.1	How-to	41
6.1.1	Viewing document thumbnails	41
6.1.2	Accessing quick document previews	41
6.1.3	Previewing documents	41
6.1.4	Viewing the document page list	42
6.1.5	Using the interactive page view	42
6.1.6	Clearing the document image cache	43
6.2	Takeaways	43
7.	Finding documents	45
7.1	How-to	47
7.1.1	Viewing the most recent documents	47
7.1.2	Viewing the most recent accessed documents	48
7.1.3	Performing a basic search	48
7.1.4	Performing an advanced search	48
7.2	Takeaways	49
8.	Quick labels	51
8.1	How-to	53
8.1.1	Creating quick labels	53
8.1.2	Using quick labels during upload	53
8.1.3	Using quick labels for existing documents	53
8.2	Takeaways	54
9.	Collaboration	57
9.1	Commenting	57
9.1.1	How-to	58
9.1.1.1	Viewing document comments	58
9.1.1.2	Adding a comment	59
9.1.1.3	Editing a comment	59
9.1.2	Takeaways	59
9.2	Checkouts	61
9.2.1	How-to	63
9.2.1.1	Checking out a document	63
9.2.1.2	Checking in a document	63
9.2.2	Takeaways	63
9.3	Mailing	66
9.3.1	How-to	67
9.3.1.1	Creating a mailing profile	67
9.3.1.2	Mailing a document as an attachment	67
9.3.1.3	Mailing a document as a link	68
9.3.2	Takeaways	68
10.	Deletions	71
10.1	How-to	72
10.1.1	Moving a document to the trash can	72
10.1.2	Viewing the documents in the trash can	73
10.1.3	Restoring documents in the trash can	73
10.2	Takeaways	73
11.	Message of the day	75
11.1	How-to	75
11.1.1	Create a new message	75
11.2	Takeaways	76

IV. Intermediate topics	79
12. Categorization	81
12.1 Cabinets	81
12.1.1 How-to	83
12.1.1.1 Creating a cabinet	83
12.1.1.2 Creating a subcabinet	83
12.1.1.3 Adding a document to a cabinet	83
12.1.1.4 Adding multiple document to a cabinet	84
12.1.2 Takeaways	84
12.2 Tags	86
12.2.1 How-to	87
12.2.1.1 Creating tags	87
12.2.1.2 Viewing all tags	87
12.2.1.3 Viewing the tags of a document	88
12.2.1.4 Attaching tags to a document	88
12.2.1.5 Attaching tags to a multiple documents	88
12.2.2 Takeaways	89
13. Versioning	91
13.1 How-to	93
13.1.1 View a document version list	93
13.1.2 Uploading a new document version	93
13.2 Takeaways	93
14. Deletion policies	95
14.1 How-to	96
14.1.1 Setting up deletion policies	96
14.2 Takeaways	97
15. User management	99
15.1 User accounts	101
15.1.1 How-to	101
15.1.1.1 Creating user accounts	101
15.1.1.2 Changing the password of a user	102
15.1.2 Takeaways	102
15.2 User groups	104
15.2.1 How-to	104
15.2.1.1 Creating user groups	104
15.2.1.2 Adding a user to a group	104
15.2.2 Takeaways	105
16. Access control	107
16.1 Permissions	107
16.1.1 How-to	108
16.1.1.1 Creating a role	108
16.1.1.2 Granting permissions to a role	108
16.1.1.3 Adding groups to a role	108
16.1.2 Takeaways	109
16.2 Access Control Lists	111
16.2.1 Inherited access control	111
16.2.2 How-to	112
16.2.2.1 Creating an Access Control List Entry	112
16.2.3 Takeaways	113

17. Document data	115
17.1 Document data levels	116
17.1.1 First level of document data	116
17.1.2 Second level of document data	116
17.1.3 Third level of document data	117
17.2 Document parsing	117
17.2.1 How-to	118
17.2.1.1 Configuring document parsing	118
17.2.1.2 Changing default document parsing	118
17.2.1.3 Viewing the parsed content of a document	119
17.2.1.4 Downloading the parsed content of a document	119
17.2.2 Takeaways	120
17.3 Optical Character Recognition (OCR)	122
17.3.1 How-to	123
17.3.1.1 Configuring document OCR processing	123
17.3.1.2 Changing default document OCR processing	123
17.3.1.3 Viewing the OCR content of a document	123
17.3.1.4 Downloading the parsed content of a document	124
17.4 File metadata	126
17.4.1 How-to	127
17.4.1.1 Configuring document file metadata processing	127
17.4.1.2 Changing default document file metadata processing	127
17.4.1.3 Viewing the file metadata of a document	127
17.4.2 Takeaways	128
17.5 Document metadata	130
17.5.1 How-to	130
17.5.1.1 Creating metadata types	130
17.5.1.2 Assigning a metadata type to a document type	131
17.5.2 Takeaways	132
18. Auditing	135
18.1 Events	135
18.1.1 How-to	136
18.1.1.1 Viewing the events of a document	136
18.1.1.2 Viewing all the events in the system	136
18.1.1.3 Viewing all the events of the current user	136
18.1.2 Takeaways	136
19. Notifications	139
19.1 How-to	140
19.1.1 Subscribing to system events	140
19.1.2 Subscribing to an object's events	140
19.2 Exercises	140
19.2.1 Subscribe to a document's view event	140
19.3 Takeaways	141
V. Advanced topics	143
20. Validation	145
20.1 Uniqueness	145
20.1.1 How-to	146
20.1.1.1 Viewing the UUID of a document	146
20.1.2 Takeaways	146

20.2	Checksum	148
20.2.1	How-to	148
20.2.1.1	Viewing the checksum of a document	148
20.2.2	Takeaways	148
20.3	Digital signatures	150
20.3.1	How-to	150
20.3.1.1	Uploading a key	150
20.3.1.2	Querying a keyserver for a public key	150
20.3.1.3	View the signatures of a document	151
20.3.1.4	Signing a document (embedded)	151
20.3.1.5	Signing a document (detached)	152
20.3.2	Takeaways	152
21.	Deduplication	155
21.1	How-to	155
21.1.1	Viewing the list of duplicated documents	155
21.1.2	Manually trigger the duplicate scan	156
21.1.3	Viewing the duplicates of a document	156
21.2	Takeaways	156
22.	Automatic Categorization	159
22.1	Indexing	159
22.1.1	How-to	160
22.1.1.1	Creating index templates	160
22.1.1.2	Editing the template code on an index template	161
22.1.1.3	Associating an index template with a document type	161
22.1.1.4	Triggering an index rebuild	162
22.1.2	Exercises	162
22.1.2.1	Create an index that categorizes documents by type	162
22.1.3	Takeaways	163
22.2	Mirroring	165
22.2.1	Mirroring	165
22.2.1.1	Introduction	165
22.2.2	How-to	166
22.2.2.1	Mirroring an index	166
22.2.3	Takeaways	167
23.	Referencing	169
23.1	Smart links	169
23.1.1	How-to	171
23.1.1.1	Creating smart links	171
23.1.1.2	Editing the conditions of a smart link	171
23.1.1.3	Associating a smart link with a document type	172
23.1.1.4	Viewing a document's smart links	172
23.1.2	Exercises	173
23.1.2.1	Create smart link to links documents by type	173
23.1.3	Takeaways	173
24.	Business processes	175
24.1	Workflows	175
24.1.1	How-to	176
24.1.1.1	Creating a new workflow template	176
24.1.1.2	Creating workflow template states	177
24.1.1.3	Creating workflow template transitions	177
24.1.1.4	Associating a workflow template with a document type	178

24.1.1.5	Viewing a workflow preview	179
24.1.1.6	Transitioning a workflow instance	179
24.1.2	Exercises	179
24.1.2.1	Create an approval workflow	179
24.1.3	Takeaways	180
25. Automation		183
25.1	Transition triggers	184
25.2	State actions	184
25.3	How-to	185
25.3.1	Creating state actions	185
25.3.2	Creating transition triggers	185
25.4	Exercises	186
25.4.1	Create a new document notifier workflow	186
25.4.2	Create a document rename workflow	187
25.5	Takeaways	188
VI. Operations		191
26. Installation		193
26.1	Minimum hardware requirements	193
26.2	Using Docker	194
26.2.1	Start a Mayan EDMS Docker image	194
26.2.2	Using a dedicated Docker network	195
26.3	Direct deployments	196
26.3.1	Basic deployment	196
26.3.1.1	1. Install binary dependencies:	196
26.3.1.2	2. Create the user account for the installation:	196
26.3.1.3	3. Create the parent directory where the project will be deployed:	196
26.3.1.4	4. Create the Python virtual environment:	197
26.3.1.5	5. Make the mayan user the owner of the installation directory:	197
26.3.1.6	6. Install Mayan EDMS from PyPI:	197
26.3.1.7	7. Install the Python client for PostgreSQL and Redis:	197
26.3.1.8	8. Create the database for the installation:	197
26.3.1.9	9. Initialize the project:	197
26.3.1.10	10. Collect the static files:	198
26.3.1.11	11. Create the supervisor file at /etc/supervisor/conf.d/mayan.conf:	198
26.3.1.12	12. Configure Redis:	198
26.3.1.13	13. Enable and restart the services :	198
26.3.1.14	14. Cleaning up:	199
26.3.2	Advanced deployment	199
26.3.2.1	1. Install RabbitMQ:	199
26.3.2.2	2. Install the Python client for RabbitMQ:	199
26.3.2.3	3. Create the RabbitMQ user and vhost:	199
26.3.2.4	4. Edit the supervisor file at /etc/supervisor/conf.d/mayan.conf:	199
26.3.2.5	5. Restart the services:	200
27. Settings		203
27.1	Configuration file	205
27.2	Environment variables	205
27.3	Python settings module	206
27.4	Setting namespaces	206
27.4.1	Appearance	206

27.4.2	Authentication	206
27.4.3	Auto administrator	206
27.4.4	Celery	207
27.4.5	Common	207
27.4.6	Converter	208
27.4.7	Django	209
27.4.8	Document parsing	214
27.4.9	Document signatures	214
27.4.10	Documents	215
27.4.11	File metadata	218
27.4.12	Lock manager	219
27.4.13	Mailing	219
27.4.14	Mayan	220
27.4.15	Metadata	220
27.4.16	Mirroring	220
27.4.17	OCR	220
27.4.18	Signatures	221
27.4.19	Sources	221
27.4.20	Storage	222
27.5	How-to	222
27.5.1	Updating the configuration file	222
27.5.1.1	Option 1: Via the user interface	222
27.5.1.2	Option 2: Via manual editing	223
27.5.2	Reverting the config file	223
27.5.3	Passing settings via environment variables	223
27.6	Takeaways	223
28. System emails		227
28.1	Sending administrative emails	227
28.2	Testing the emails configuration	227
28.3	Takeaways	227
29. Maintenance		229
29.1	Backups	229
30. Docker image		231
30.1	Environment Variables	231
30.2	Accessing outside data	232
30.3	Performing backups	232
30.4	Restoring from a backup	233
30.5	Upgrading	233
30.6	Customizing the image	233
30.7	Using Docker compose	234
30.8	Nightly images	234
31. Troubleshooting		235
31.1	Database	235
31.1.1	MySQL error: OperationalError: (1267, "Illegal mix of collations (latin1_swedish_ci, IMPLICIT) and (utf8_general_ci, COERCIBLE) for operation '='")	235
31.1.2	MySQL error: Incorrect string value: 'xE2x80x95rs6...' for column 'content' at row 1	236

31.1.3 MySQL error: django.db.utils.IntegrityError IntegrityError: (1452, 'Cannot add or update a child row: a foreign key constraint fails (`...`.', CONSTRAINT `..._refs_id_b0252274` FOREIGN KEY (`...`) REFERENCES `...` (`...'))')	236
31.2 PostgreSQL error: OperationalError: FATAL: sorry, too many clients already	236
31.3 Dependencies	237
31.4 Docker	237
31.4.1 MAYAN_APT_INSTALLS does not work for Archlinux with kernels > 4.14	237
31.5 Passwords	237
31.5.1 Admin password reset	237
31.5.2 Missing automatic admin account after installation	238
31.6 Watchfolders	238
31.6.1 Incomplete files uploaded	239

VII. Integration 241

32. API	243
32.1 API paths	243
32.1.1 auth	243
32.1.2 cabinets	243
32.1.3 checkouts	245
32.1.4 content_types	245
32.1.5 document_types	246
32.1.6 documents	248
32.1.7 event_type_namespaces	254
32.1.8 event_types	255
32.1.9 events	255
32.1.10 groups	255
32.1.11 indexes	256
32.1.12 keys	258
32.1.13 messages	259
32.1.14 metadata_types	259
32.1.15 notifications	260
32.1.16 objects	261
32.1.17 permissions	262
32.1.18 roles	262
32.1.19 search	263
32.1.20 search_models	263
32.1.21 smart_links	264
32.1.22 staging_folders	265
32.1.23 tags	266
32.1.24 templates	267
32.1.25 trashed_documents	267
32.1.26 users	268
32.1.27 workflows	270
32.2 API schema	273
32.2.1 AccessControlList	273
32.2.2 AccessControlListPermission	273
32.2.3 Cabinet	273
32.2.4 CabinetDocument	274
32.2.5 Comment	274

32.2.6	ContentType	275
32.2.7	DeletedDocument	275
32.2.8	Document	275
32.2.9	DocumentCheckout	276
32.2.10	DocumentMetadata	276
32.2.11	DocumentPage	276
32.2.12	DocumentPageContent	276
32.2.13	DocumentPageOCRContent	276
32.2.14	DocumentTag	277
32.2.15	DocumentType	277
32.2.16	DocumentTypeFilename	278
32.2.17	DocumentTypeMetadataType	278
32.2.18	DocumentVersion	278
32.2.19	Event	278
32.2.20	EventType	279
32.2.21	EventTypeNamespace	279
32.2.22	Group	279
32.2.23	Index	279
32.2.24	IndexInstanceNode	280
32.2.25	IndexTemplateNode	280
32.2.26	Key	280
32.2.27	Message	281
32.2.28	MetadataType	281
32.2.29	NewCabinetDocument	282
32.2.30	NewDocument	282
32.2.31	NewDocumentCheckout	282
32.2.32	NewDocumentMetadata	282
32.2.33	NewDocumentTag	283
32.2.34	NewDocumentTypeMetadataType	283
32.2.35	NewDocumentVersion	283
32.2.36	NewWorkflowDocumentType	283
32.2.37	Notification	283
32.2.38	Permission	283
32.2.39	RecentDocument	283
32.2.40	ResolvedSmartLink	284
32.2.41	ResolvedSmartLinkDocument	284
32.2.42	Role	285
32.2.43	SearchField	285
32.2.44	searchModel	285
32.2.45	SmartLink	285
32.2.46	SmartLinkCondition	285
32.2.47	StagingFolder	286
32.2.48	StagingFolderFile	286
32.2.49	Tag	287
32.2.50	Template	287
32.2.51	User	287
32.2.52	UserGroupList	288
32.2.53	Workflow	288
32.2.54	WorkflowDocumentType	288
32.2.55	WorkflowInstance	289
32.2.56	WorkflowInstanceLogEntry	289
32.2.57	WorkflowState	289
32.2.58	WorkflowTransition	290
32.2.59	WritableAccessControlList	290

32.2.60	WritableAccessControlListPermission	290
32.2.61	WritableCabinet	290
32.2.62	WritableComment	291
32.2.63	WritableDocument	291
32.2.64	WritableDocumentType	291
32.2.65	WritableDocumentTypeMetadataType	292
32.2.66	WritableDocumentVersion	292
32.2.67	WritableRole	293
32.2.68	WritableSmartLink	293
32.2.69	WritableTag	293
32.2.70	WritableWorkflow	294
32.2.71	WritableWorkflowInstanceLogEntry	294
32.2.72	WritableWorkflowTransition	294
32.3	How-to	294
32.3.1	Installing an HTTP client	294
32.3.2	Getting a list of document types	295
32.3.3	Uploading a new document	295
32.3.4	Obtaining API tokens	295
32.3.5	Adding API token to the request header	296
32.3.6	Using sessions	296
32.3.7	Searching documents	296
32.3.8	Searching documents by field	297

VIII. Internals 301

33. Project structure	303	
33.1	Follow PEP8	305
33.2	Imports	306
33.3	Dependencies	307
33.4	Variables	307
33.5	Strings	307
33.6	Migrations	307
33.7	General	308
33.7.1	Steps to deploy a development version	308
33.7.2	Contributing changes	308
33.7.3	Debugging	308
33.8	App modules	309
33.9	Views	311
33.10	Git branch structure	311
33.11	Commit messages	312
33.12	Source file package	313
33.13	Wheel package	313
33.13.1	Version numbering	314
33.13.2	Release checklist	314
33.14	Release using GitLab CI	315
33.15	Manual release	315
33.16	Docker image	316
33.16.1	Building the image	316
34. MERCs	319	
34.1	MERC 1: Purpose and Guidelines	319
34.1.1	What is a MERC?	319
34.1.2	MERC Types	320

34.1.3	MERC submission workflow	320
34.1.3.1	Pre-proposal	320
34.1.3.2	Submitting the draft	320
34.1.3.3	Implementation	320
34.1.4	MERC format	320
34.1.4.1	MERC Metadata	321
34.1.4.2	Auxiliary Files	322
34.1.5	Reporting MERC Bugs, or Submitting MERC Updates	322
34.2	MERC 2: Test writing	322
34.2.1	Abstract	322
34.2.2	Motivation	323
34.2.3	Specification	323
34.3	MERC 3: Using JavaScript libraries	324
34.3.1	Abstract	324
34.3.2	Rationale	324
34.3.3	Motivation	324
34.3.4	Backwards Compatibility	325
34.3.5	Specification	325
34.4	MERC 4: Support forum	325
34.4.1	Abstract	325
34.4.2	Motivation	326
34.4.3	Specification	326
34.5	MERC 5: Explicit arguments	326
34.5.1	Abstract	326
34.5.2	Motivation	327
34.5.3	Specification	327
34.5.4	Backwards Compatibility	327
34.5.5	Reference Implementation	327
34.6	MERC 6: Lower information disclose	328
34.6.1	Abstract	329
34.6.2	Motivation	329
34.6.3	Specification	329
34.7	MERC XX: Unify Roles and Groups	330
34.7.1	Abstract	330
34.7.2	Rationale	330
34.7.3	Motivation	331
34.7.4	Backwards Compatibility	331
34.7.5	Specification	331

IX. Appendix	333
---------------------	------------

Index	337
--------------	------------

neeraj76@yahoo.com

PART I.

PREFACE

neeraj76@yahoo.com

History

The more complex something is, the harder it is for us to reconcile it with its simple beginnings. Beginnings that are usually the result of a simple choice, most often a binary one, with no apparent discernible reward out of each outcome. But sometimes, we get a hunch, an ever so slight feeling, that one of the options is worth choosing. A subtle hint that said option might just change the course of our lives, Mayan EDMS is such a story.

This text was originally the “history” section of the introduction chapter. The purpose was to tell the genesis of Mayan EDMS as a project and its story. As I read up a bit on how to write technical manuals, I kept bumping on a persistent issue: you shouldn’t write in first person when writing a technical book. This was difficult to avoid because the story of Mayan EDMS is not isolated. The story of Mayan is also my personal story. If the conditions of my life at the time were different, Mayan would probably never come into existence. After struggling to fit the genesis of Mayan EDMS as part of the introduction, I decided to turn it into a preface and instead relate it to you as it should be, as a first person account.

Even though the official release of Mayan EDMS happened back in 2011 the roots of its creation go back to 2009. That year Act 161 of December 1, 2009, known as the “Act for the Reform of the Process of Permits of Puerto Rico”¹ came into effect.

This new act repealed act no. 76 of June 24, 1975², as amended, known as the “Organic Act of the Regulations and Permits Administration”, and created the new “Permit Management Office”³, attached to the existing Planning Board⁴.

But for me the most influential part of this Act was that it specified the new digital documentation requirements when working with permits. Physical paper was to be eliminated from the entire permit process.

As a technology person this was music to my ears, not so much for the rest of the agency. The new Act was only a legal document, its creators failed to establish guidelines, an implementation plan, and a myriad of other things outside the legal scope that were equally important to make the mandates of the act a reality.

Don’t mistake this with a “high horse” critique of government technology blunders, it was the pivotal aspect of this story. The shock waves of the Act 161 digitalization paradox reached our regional branch in 2010. It was our turn to enact it and solve the documentation problem by doing what every other branch was doing, nothing. If you’ve worked in government, you already know, it is usually the safest “solution”.

As the IT manager, I did my best to find a product to use for the digitalization problem. In the expected IT manager fashion, I search for pre-made solutions that would fit our needs. Government IT seems to always be the round hole of square peg solutions. I was not surprised that there were no existing solution able to fulfill the specific requirements of our predicament. The only solution would be one custom made. This is the moment my life branched, the superposition of my future was about to collapse into reality.

The agency didn’t have an official software development department and there was no budget at the time to hire a development company to do the work. The universe froze, and I found myself in a metaphorical isolation room. I took a leap of faith. Based on what I’ve seen so far from trying out different solutions, I decided to try my hand at the problem, “throw something together” as they say. Of course it would also have to be done on my spare time, for free.

Working on my spare time, meant I would be able to keep all the rights and creative control of anything that came out. The goal was a simple: a PDF manager with OCR and a basic search feature. That was it.

¹ Ley Núm. 161 de 1 de diciembre de 2009, según enmendada, conocida como la “Ley para la Reforma del Proceso de Permisos de Puerto Rico”

² Ley Núm. 76 de 24 de junio de 1975, según enmendada, conocida como “Ley Orgánica de la Administración de Reglamentos y Permisos”

³ Oficina de Gerencia de Permisos

⁴ Junta de Planificación

After using Linux/GNU for more than a decade as my only operating system, along with Python, and Django, I had glimpse of how far a change of paradigm can go. If I hadn't made the choice to change my stack of tools to a full set of Free Open Source ones, retaining the rights for such a simple software would not have seemed like much.

Database models done and while working on the business logic, I caught by chance a documentary about the Maya people. I've always been amazed at the level of scientific development of some pre-Columbian cultures, but this documentary stuck with me more than others. It was about something I didn't know about, the Maya books.

Folding books made out of bark paper, known as the Maya codices. Only three have survived. But it was more than enough to inspire me.

The Maya accomplished the creation of an incredible writing system. One that supported the development of math, astronomy, and history. Like other ancient cultures, the Maya took to stone to give form to their thoughts. Like very few others, they understood the power of the written word to pass on information through time and decided to give it its own special place in their society. The struggles with information modernization that I was trying to solve, made me feel history had been turned on its head, we were the ones living in the past.

An thus Mayan EDMS came into being. EDMS stand for Electronic Document Management System. An acronym that perfectly describes what the project does, impossible to elevator pitch and the bane of SEO optimization. But still, it was clear, Mayan EDMS would be more than a place to store PDFs, it would become an idea, an information preservation idea.

Next, time to deploy, or more precise, where to deploy. Since it was not an official project, Mayan EDMS couldn't be installed on the agency's servers. Not only that, they were running another operating system and that meant installing a virtual machine manager, which meant more unnecessary round trips of emails justifying more moving parts.

Have you noticed, how in TV series writing, when the main character is stuck, the solution always seems to come from someone else's mundane dialog? The main character lost in thought, frustrated at the seemingly unsolvable nature of the problem. Then accompanied by an angelic chorus, the obviously unrelated scripted dialog of the secondary character about the previous night's game, ushers the main character into a stream of inspiration. The problem's solution becomes obviously simple in a moment worthy of the price of best the *Deux ex machina* awards.

"One person's trash is another person's treasure".

From water cooler talk I learned about some equipment that was not being used and could be repurposed to host my software. Some phone calls with the property department, some of the required paperwork dance, and presto! Calling the computers vintage would have been a compliment. But I was confident I could make this work. Too many sleepless nights were spent to give up at that moment.

None of the provided computers had the power to run Mayan EDMS on their own. As a good practice I had segregated functionality into modules. It meant that with minor changes, I could split the different parts of Mayan EDMS and have them execute on the different computers, all acting as a single entity with just enough computational power to launch the software.

One ISO image download and 5 installations later, Mayan EDMS came online for the first time in production. It is a shame I didn't record the exact date and time of such an occasion.

The rollout on the branch office went without problems. To save time training the users, I purposely used office vocabulary for the different features of the program. When there were no direct options, simile and metaphors filled the gap. This managed to transport users from the physical reality into a new electronic one with the lowest learning curve possible.

A few days passed, and everything was working exceptionally well. Scanning documents was no longer the bottleneck in the registration or evaluation processes, the documents were organized, it was fast, the users loved it. Nothing is rarely this good.

Inspired by the initial success more features were quickly added. Staging folders allowed using multiple network scanners with ease. Indexes were added to automatically categorize documents by case number, registration date, employee name, project name. Index mirroring allowed access to the indexes without using the web interface.

Confidence in the software grew fast, this prompted the inclusion of more document types. Before long, in addition of the permit forms, many more document formats were being scanned into Mayan EDMS. Letters, photos, blueprints, forms from other government agencies.

The document count grew even faster. The distributed nature of the design supported further growth from the same hardware with minor database index tweaks.

Empowered by the successful deployment, I released Mayan EDMS to the rest of the world in April of 2011. It only took hours for the first patch to arrive. Knowing that something you built is being used by people on the other side of the world is a unique experience. It anchors you, gives you a sense of not just living in the world, but you are actually doing something of value too.

Every week I received a gift in the form a of an electronic thank you. Growth stabilized to five new users per week. This felt amazing and at the time I thought it was a good growth level, the project had exceeded my expectations, I was happy with how the story turned out. Little did I knew it was just the beginning.

Mayan EDMS took a life of its own. What followed was a blur of events, barely possible to recollect.

In just two years, opportunity knocked, big time! And it boiled down to simple binary decision. A decisions seemingly insignificant at the time. A new government job that came with creative freedom to replicate Mayan EDMS's success in the entire government.

Instead of the initial five new users per week, the Mayan EDMS user community, grew and still grows by several thousand per day. The Docker image reached one million downloads in a matter of months. The speed of growth is reaching numbers that take me some time to adjust to, before I can fully comprehend them and compare them internally.

When we create something, we do so expecting the influence to go just one way. We are the creators, the tinkerers, we influence the object. But as we work on our creation, a bond is created. Not just an emotional attachment but a real chain on influence that goes both ways. Just as I was creating Mayan EDMS, it too was creating a new life for me.

The project had its eight anniversary and has already exceeded even my most optimistic expectations. It has grown into the complete solution it is today thanks to the collaboration of thousands of developers spanning all continents of the world.

The time, monetary, and resource constraints of the environment for which and in which Mayan EDMS was created, gave the software its strong and unique advantages. Even today, after eight years, Mayan EDMS's advantages continue to make it consistently one of the best choices for many types of situations and deployments.

I knew about Maslow's hierarchy, but I never expected to climb it this way.

Book conventions

This book is divided in two basic units. The first is parts. Each part will group similar topics. Each topic is presented as a chapter. Each chapter will cover a topic or a functionality. Each chapter starts by introducing the feature, showing guided steps to use the feature, provides a summary, and may include optional exercises for the reader to perform.

Code samples will be show using a monospaced font as follows:

```
sudo -u mayan /opt/mayan-edms/bin/pip install --no-cache-dir --no-use-pep517 mayan-  
↳edms
```

Additional information that is not required but can be useful is presented in the following way:



Tip: Since a workflow transition is an action, use verbs for clarity. For example, use “Approve” instead of “Approved”.



Note: Platforms with the ARM CPU might also need additional requirements.

Information that is important or required in the context of the text but would be hard to find, is presented in the following way:



Important: If you are the administrator of the installation, a super user account was automatically created for you and the credentials of that account would have been displayed on the console during installation. The credentials will also be shown in the login screen until the password is changed.



Permissions required:

The “Edit comments” permission is required for this action, globally or via an ACL for the document.



Interrupting this step will cause errors!

Technical terminology will be presented as literals using a monospaced font as follows:

The MAYAN_MEDIA_ROOT environment variable.

Names of other technologies or products are presented in italics.

GNU/Linux operating system.

neeraj76@yahoo.com

PART II.

GETTING STARTED

neeraj76@yahoo.com

INTRODUCTION

1.1 What is Mayan EDMS?

Mayan EDMS is a modern, highly scalable, electronic document management system. It provides a secure vault for electronic documents.

1.2 Features

Concepts are intuitive with names and descriptions taken from common office vocabulary.

Mayan EDMS was designed from the ground up with a new approach of managing digital documents, as such it features:

- Automatic categorization
- Full text search
- Unlimited indexes
- Unlimited folders
- Optical character recognition
- File type agnostic
- Equipment agnostic
- Custom document metadata
- Maintenance free operation
- Dynamic metadata generation
- Embedded workflow engine
- Open Source API for third party integration
- Native web application interface for quick implementation and deployments
- Built on open standards, no vendor lock-in
- Robust access control
- Mature with 8 years in production
- Active community
- Growing team of developers and translators

1.3 Deployments

Mayan EDMS is designed to work with multiple platforms and deployment environments. Some of these are:

- bare metal servers (both small and enterprise grade)
- virtualized environments
- cloud providers
- various container technologies
- and even single board computers and battery powered appliances

1.4 Open standards

Most of the existing document management products require a brand match between the hardware and the software. The software will only work with equipment from the same vendor. Mayan EDMS is designed to be equipment agnostic and will work with any scanning equipment available.

Unlike other solutions, instead of creating a wall of PDF files, Mayan EDMS preserves the contextual relationship of scanned documents.

Most products store documents in a proprietary format. This makes retrieval of scanned documents impossible using 3rd party software. This also facilitates vendor locking. In essence users of proprietary solutions are giving away control of their documents. Mayan EDMS keeps all scanned documents in their original format for a low-risk and future proof solution.

1.5 Modular design

Mayan EDMS is coded in units of functionality called apps. Apps are like modules or plug-ins. Each app adds a specific functionality to whole system. The collections of all the apps make up the entire system known as Mayan EDMS. This design strategy allows for the quick addition of new functionality, either by the software creators or by third party developers with minimal modifications.

1.6 Where to get it

Mayan EDMS is distributed in several formats:

- Python format (sdist and wheel): <https://pypi.org/project/mayan-edms/>
- Docker image: <https://hub.docker.com/r/mayanedms/mayanedms/>
- Source code: <https://gitlab.com/mayan-edms/mayan-edms>

The official website of Mayan EDMS is <https://www.mayan-edms.com>.

1.7 Browsers supported

Mayan EDMS is developed using Firefox¹ as the reference browser. Aside from Firefox, Chrome² is also supported. The technologies from Firefox and Chrome are also found in other browsers like Opera³ and Brave⁴. These other browsers should also work properly.

Mayan EDMS does not support the Internet Explorer⁵ set of browsers.

1.8 Interface

To lower the learning curve, Mayan EDMS employs user interface elements that are very common to users, and that are used in many other applications.



Fig. 1: Located at the top of the page, this is the place from where the majority of the features are accessed.

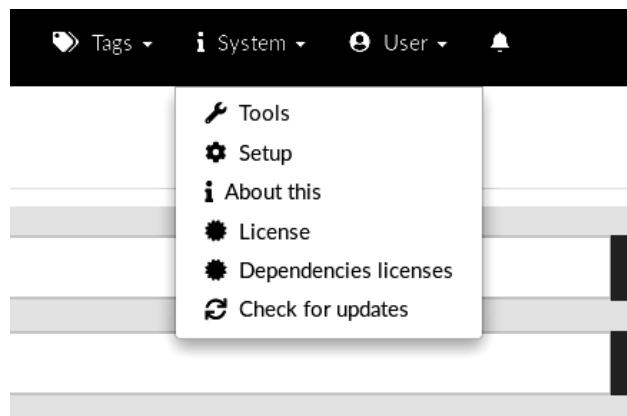


Fig. 2: Clicking on a menu entry will activate that entry and show a list of further options.

¹ Firefox is a trademark of the Mozilla Foundation in the U.S. and other countries.

² Chrome is a trademark of Google Inc.

³ Opera is a trademark of Opera Software AS.

⁴ Brave is a trademark of Brave Software Inc.

⁵ Internet Explorer and is a registered trademarks of Microsoft Corporation in the United States and/or other countries.

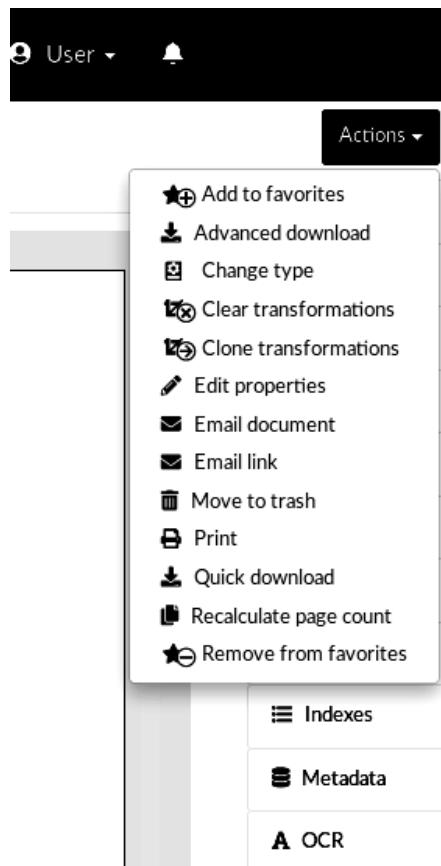


Fig. 3: Dropdowns are stand-alone menu entries with options that change depending on what is being shown in the current view.

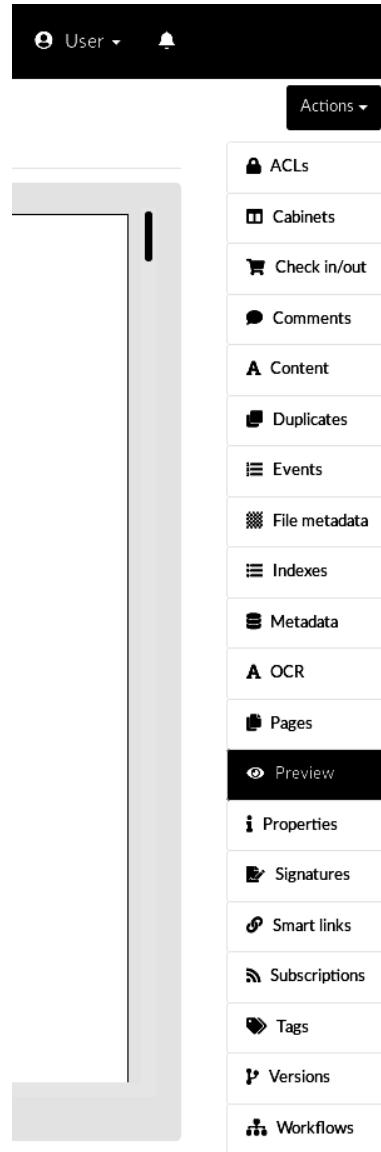


Fig. 4: When multiple subviews are available, a sidebar will be used.



Fig. 5: Button with an outline switch to a new view while color coded buttons trigger an action.

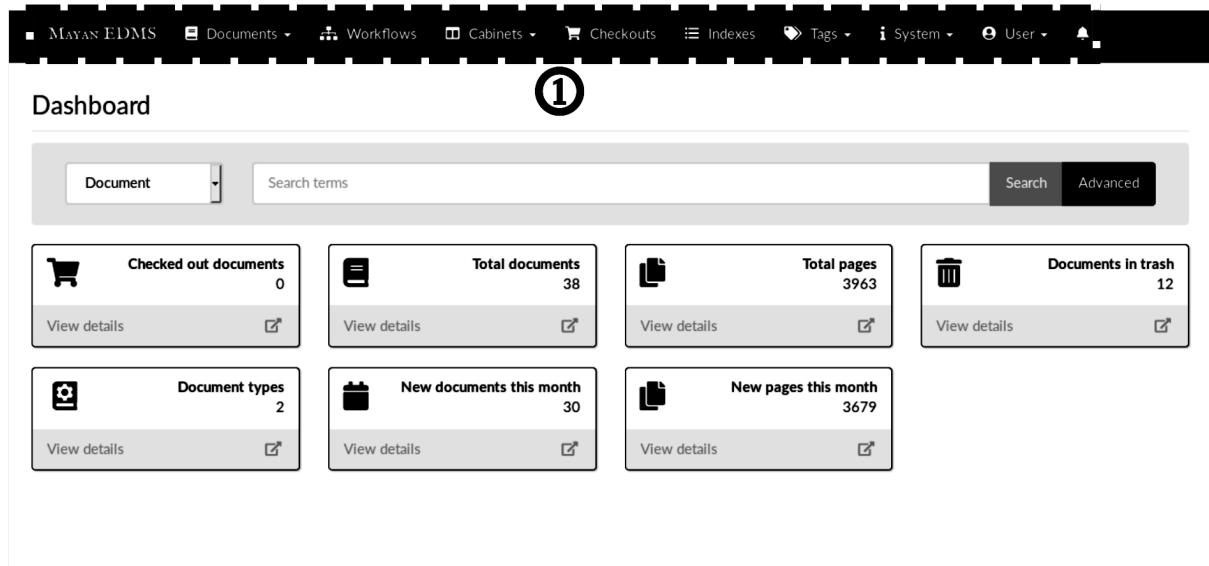


Fig. 6: The main menu shown in context, identified as #1.

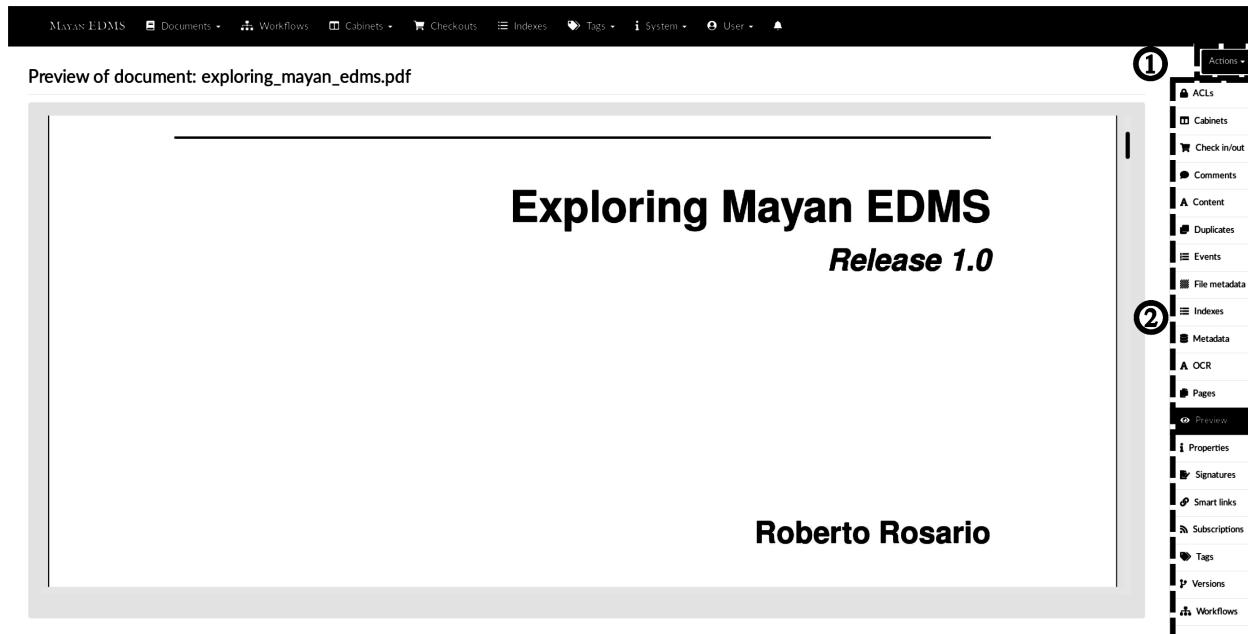


Fig. 7: The action dropdown identified as #1 and the side bar as #2, as they appear in the document preview view.

PART III.

BASIC TOPICS

neeraj76@yahoo.com

MINIMAL CONCEPTS

Think of Mayan EDMS as a filing cabinet. It also has many features to improve productivity and reduce workload, but at its core, it is a sophisticated filing cabinet.

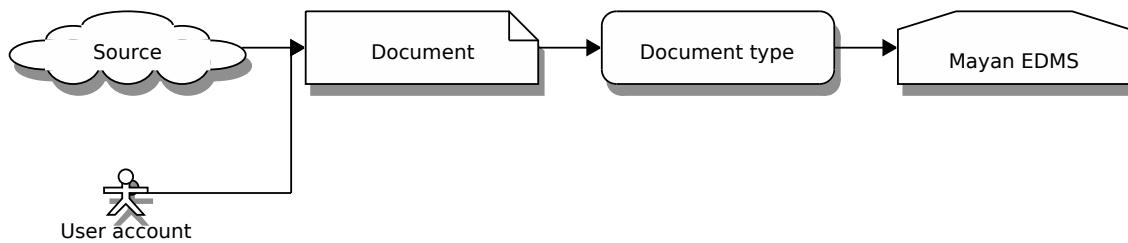


Fig. 1: Minimal elements needed to start using Mayan EDMS.

To start using Mayan EDMS you will need some basics elements. These are:

- A user account
- A source of documents
- The type of documents

If you've used computers before, the concept of "user accounts" will be very familiar. If not, a user account identifies you in the system. This is similar to what a bank account number, the social security number, or an employee number accomplishes.

Following the analogy of Mayan EDMS being a filing cabinet, then the "source of documents" are the objects that are producing or storing the new documents you want to move to the filing cabinet. These sources exists in the real world and they can be scanning equipment, smartphones, emails, etc. It is anything that produces, or is currently holding, electronic document files that will be managed by Mayan EDMS.

Finally the "type of the documents", is just that, a word or phrase that describes, or identifies, the type of document that will be filed.

The action of filing new documents, is known as "uploading" in Mayan EDMS.

To start uploading documents from source, you need at least one of each. One user account, source, and one document type. You need at least one of each but you can define as many as you need later on.

To make things easier, during the installation, an initial setup consisting of a user account with a random password, a default document type (named "Default"), and a default source (also named "Default") will be created automatically. You can to use these, rename them, or even delete them when you no longer need them.

The next three chapters will expand on these elements and in very little time you will be uploading your first document.

NOTES

neeraj76@yahoo.com

AUTHENTICATION



Fig. 1: "Richardson" standard glass panel door. Thorp fire proof door Co. 1914.

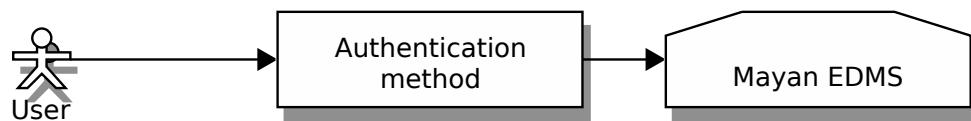


Fig. 2: Units of authentication: User, method, resource.

Authentication is the act of verifying the identity of a user in order to allow usage of the system. Authentication only ensures that the user is who they claim they are. Other systems can then build on top of the authentication system to provide additional functionality based on the verified identity of the user.

In its most common form, it uses two pieces of information:

- An identifier - Assigned to you by the administrator. Usually a username based on the combination of your first name and last name, or your e-mail address.
- A password - Initially created for you by the administrators, but can later be changed to anything you want.



Fig. 3: The default authentication employs a username and password combination.

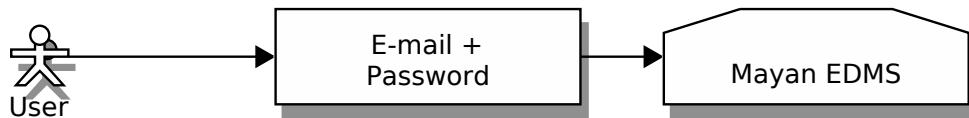


Fig. 4: Mayan EDMS also support authentication using the user's email and password combination.

3.1 How-to

3.1.1 Login on¹



Important: If you are the administrator of the installation, a super user account was automatically created for you and the credentials of that account would have been displayed on the console during installation. The credentials will also be shown in the login screen until the password is changed.

1. Enter the Mayan EDMS URL in your browser's URL bar and press Enter.
2. On the login screen enter your username in the Username field. Your installation may have been configured to use the e-mail instead to identify users, if that is the case enter your e-mail in the E-mail.
3. Enter your password in the Password field.

¹ As with many niche technical vocabulary whose popularity has exploded over time, the origin and correct use of the term used denote authentication to a multi user computer, is a much debated topic. No attempt is made to use any perceived authoritative or canonical form. The form used here is the one most familiar to the author.

-
4. Optional: Select the checkbox named **Remember me** to keep you logged in for an extended amount of time, even after you close the browser.
 5. Click the **Sign in** button.
-



- The view that will be used to authenticate users is controlled by the *LOGIN_URL* (page 213) setting.
 - The login method is controlled by the *AUTHENTICATION_LOGIN_METHOD* (page 206) setting.
 - The login session time length is controlled by the *AUTHENTICATION_MAXIMUM_SESSION_LENGTH* (page 206) setting.
 - The view where users are redirected after login in is controlled by the *LOGIN_REDIRECT_URL* (page 213) setting.
 - The view where users are redirected after login out is controlled by the *LOGOUT_REDIRECT_URL* (page 213) setting.
-



If a custom login view is used, several setting will not have effect as they depend on the default authentication view. You will need to implement these on your custom login view.

3.1.2 Login out¹

1. Go to the *Users → Logout* menu.
2. Your session will be closed and you will be redirected to the login view or another view the administrator may have configured. Login out from one browser window will log you out from all other windows or tabs too.

3.1.3 Resetting your password



Important: To use the password reset feature, *System emails* (page 227) need to be already configured. These emails are sent by the system itself and not by the users. Their usage and configuration is different than *Mailing* (page 66).

1. Enter the Mayan EDMS URL in your browser's URL bar and press **Enter**.
2. On the login screen, click on the link titled **Forgot your password?**.
3. Enter your e-mail address in the form.
4. A message will be sent to your e-mail account. Open the message and click on the password reset link provided.
5. Enter a new password twice that complies with the password validation rules shown.
6. Click the **Submit** button.
7. Your password has now been changed and you may proceed to login again.

3.2 Takeaways

- Mayan EDMS support username and email authentication.
- The authentication system is decoupled from the rest of the system and can be independently configured.
- To be able to reset your password via e-mail, in case it is forgotten, the system must first be configured to send e-mails.

NOTES

neeraj76@yahoo.com

DOCUMENT TYPES

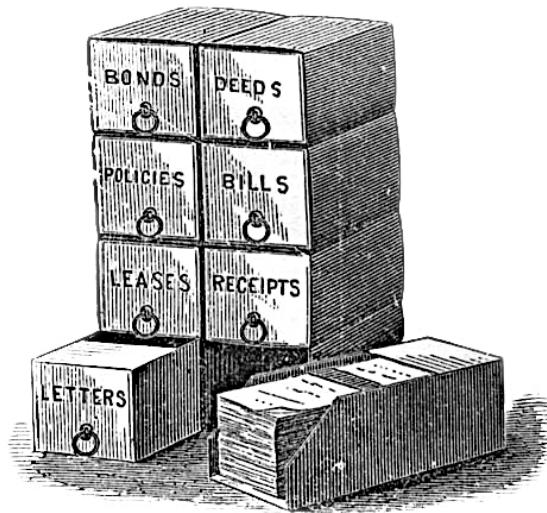


Fig. 1: *C. C. & A. L. Chamberlain Document and Box Envelope, 1877 ad*

Document types are the basic unit of data in Mayan EDMS. A document type can be interpreted also as a document category, a document class, or a document template.

Every other aspect of the system will rely on, or be tied to, one or more document type. Create one document type for each type or class of document you intend to upload into Mayan EDMS.

Document types need to be created before documents can be uploaded. It is not possible to upload documents without assigning them a document type.

Document types usually mirror the type of physical, paper document they represent.

Once create a document type may be used for an unlimited number of documents.

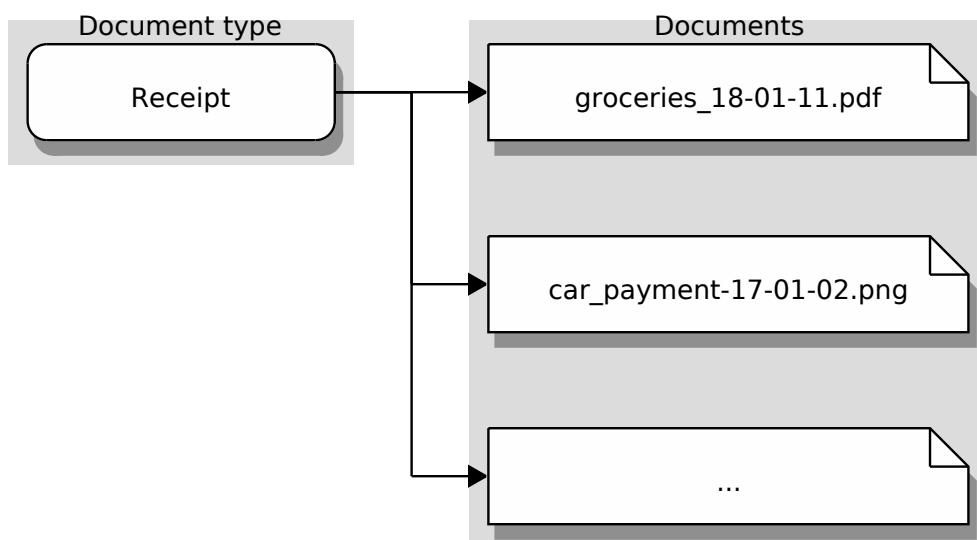
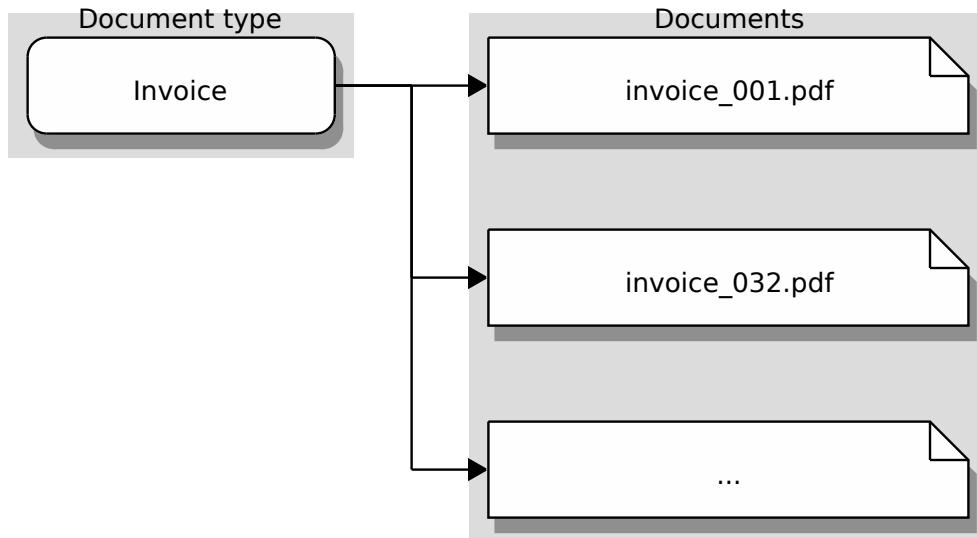
Example document types:

- Letter
- Invoice
- Time sheet
- Blueprint
- Receipt



Tip: Although a document type represents an unlimited number of documents, when deciding on a name, using the singular form improves clarity.

Examples:



Settings and attributes are applied to document types and documents will inherit those settings and attributes based on the document type they were assigned when uploaded.

A document can only be of one type at a given moment, but if needed, the type of a document can be changed.

Upon changing its type, the document will lose its previous settings and attributes, and will inherit the settings and attributes of its new type.

4.1 How-to

4.1.1 Creating document types



Permissions required:

The “Create document types” permission is required for this action.

1. Go to the *System* → *Setup* → *Document types* menu.
2. From the **Actions** dropdown select **Create document type**.
3. Enter a label to be shown to users when using this document type.
4. Press **Submit**.

4.2 Takeaways

- Document type are the most basic level of classification.
- Every document needs to be assigned to a document type.
- Documents can only be of one document type.
- The type of the document is chosen during upload but can also be changed afterwards.

NOTES

SOURCES



Fig. 1: Post office boxes, Collier Stationary Co. 1909

Document sources (or sources for short) define places from which documents can be uploaded or gathered. Mayan EDMS provides multiple strategies to feed new documents to the system.

Sources are divided into two main categories:

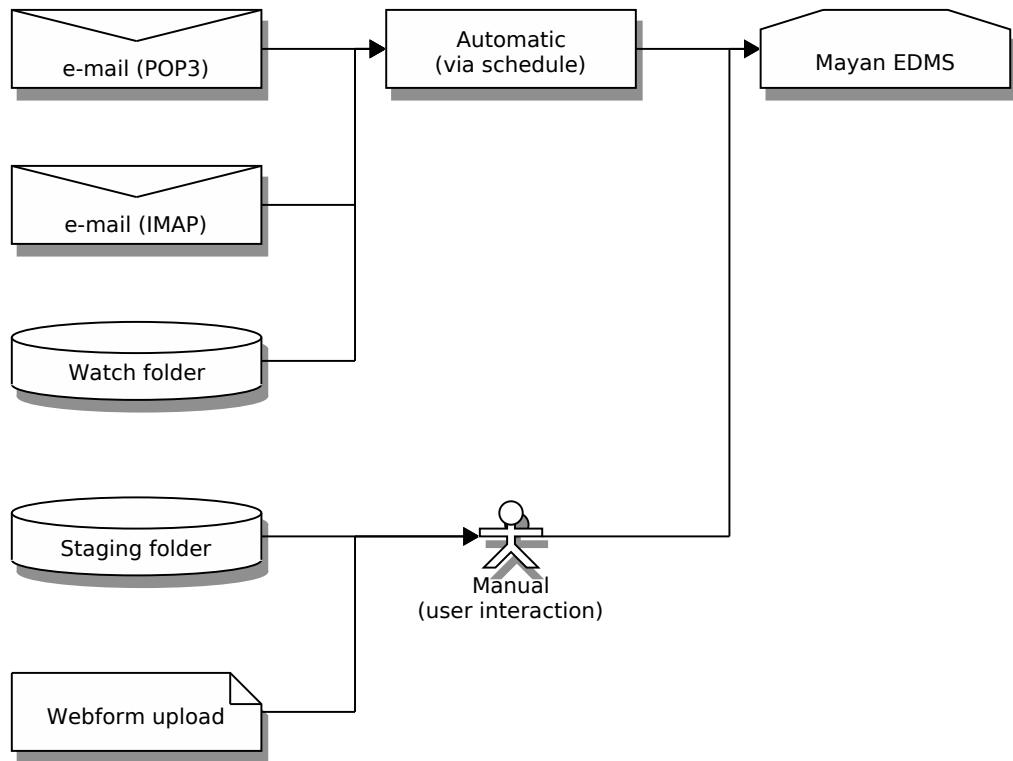
- Manual - Require user intervention.
- Automatic - Are activated by the system using a time interval.

Manual sources:

- Webform - These are HTML forms with a **Browse** button or a drag and drop area that will open the file dialog when clicked to allow selection of files from the user's computer.
- Staging folder - Are file system folders where networked attached scanners can save image files. The files in these staging folders are read and a preview is generated to help the process of upload. Documents in a Staging folder remain there for an indefinite amount of time until a user uploads them. An example use of Staging folders use is when multiple users are scanning documents using networked equipment, but only one user must be allowed to upload those documents. This one user examines the scans quality and decides what to upload, what to reject, and what must be re-scanned.

Automatic sources:

- *POP3* email - Provide the email, server and credential of a POP3 based email to be scanned periodically for email. The body of the email is uploaded as a document and the attachments of the email are uploaded as separate documents.
- *IMAP* email - Same as the POP3 email source but for email accounts using the IMAP protocol.
- Watch folder - A filesystem folder that is scanned periodically for files. Any file in the watch folder is automatically uploaded and subsequently deleted. Watch folders can be used when the quality of the scans is irrelevant or when they will be known to be of good quality, such as when receiving e-faxes as PDFs.



Document sources may be configured to allow document bundles to be uploaded as compressed files which are then decompressed and their content uploaded as separate documents. This feature is useful when migrating from another document manager system.

The variety of source type allow choosing the better option for a particular use. For example, staging folders and watched folders are better suited for interfacing with scanning equipment. To receive documents from the field from a portable scanner or a smart phone, an email source may be the better choice.



Note: If you deployed a Mayan EDMS Docker container and want to use watched folders or staging folder, refer to the Docker chapter *Accessing outside data* (page 232).

5.1 How-to

5.1.1 Creating new sources



Permissions required:

The “Create new document sources” permission is required for this action.

1. Go to the **System → Setup → Sources** menu.
2. From the **Actions** dropdown select the new type of source to create.
3. Each source type will have different fields to customize its behavior. Enter the required information in each field based on the help text provided.
4. Press **Save**.

5.1.2 Testing sources



Permissions required:

The “Create new document sources” permission is required for this action.

1. Go to the **System → Setup → Sources** menu.
2. Click on the button **Check now** of the source to test.
3. Click on the button **Logs** to examine the error log for the source. If the error log is empty the source is correctly configured and may be used in production.

5.1.3 Uploading a document via a webform



Permissions required:

The “Create documents” permission is required for this action.



Note: Make sure a webform type source exists or create one before proceeding.

1. Make sure a webform type source exists.
2. Go to the **Documents → New document** menu.
3. The document upload wizard will start. Select a document type for the document that will be uploaded.
4. Click the button **Next step** or double-click on the document type selection to continue.

-
5. If there are metadata types associated with the selected document type, this next step will show a form allowing data entry for each metadata type for the new document. If there are required metadata types, a value must be entered before the wizard will allow progressing further. Click the button **Next step** to continue.
 6. If tags have been created, this next step will show a form allowing the selection of multiple tags to attach to the new document. Click the button **Next step** to continue.
 7. If cabinets have been created, this next step will show a form allowing the selection of multiple cabinets to which the new document will be added. Click the button **Next step** to continue.
 8. The upload form will be shown. Multiple new documents may be uploaded by dragging the files from another windows and dropping them on the box with the text **Drop files or click here to upload files**. Alternatively clicking the box with the text **Drop files or click here to upload files** will open the browser's file selection window allowing files to be uploaded.
 9. No further action is required after the last step, the changes are automatic and is completed in the background.
-



Settings:

- The image storage defaults to a folder inside the media folder of the installation with the name `document_storage`. The behavior of the document storage is controlled by the settings `DOCUMENTS_STORAGE_BACKEND` (page 217) and `DOCUMENTS_STORAGE_BACKEND_ARGUMENTS` (page 217).
-

5.2 Takeaways

- Sources are how new documents are added to the system.
- There are manual and automatic sources that execute on a time schedule.
- Some sources, like the webform and staging folders, provide a “Wizard” with steps to expedite some tasks like attaching tags or adding to cabinets.

5.3 Exercises

5.3.1 Upload your first document



Permissions required:

The “Create documents” permission is required for this action.



Important: You need a user account created and ready to log in.

1. Login on Mayan EDMS as shown in *Login on 1* (page 24). If you are not able to log in, reset your password following the steps in *Resetting your password* (page 25).

-
2. Upload a document using the webform as shown in *Uploading a document via a webform* (page 35).
 3. Verify that your document uploaded correctly.

NOTES

VISUALIZATION



Fig. 1: Windsor clerk lamp. Cole Steel

Mayan EDMS includes features that extend beyond the typical set of features expected from a document management system. One of these features is document visualization.

What document visualization does is generate previews of documents automatically. This allows usage of the document without having to download the document's file to the user's computer.

The visualization system supports most of the commonly used images formats (PNG, JPEG, TIFF, PBM, PGM, PPM, PNM, WEBP) as well as PDFs and office documents.

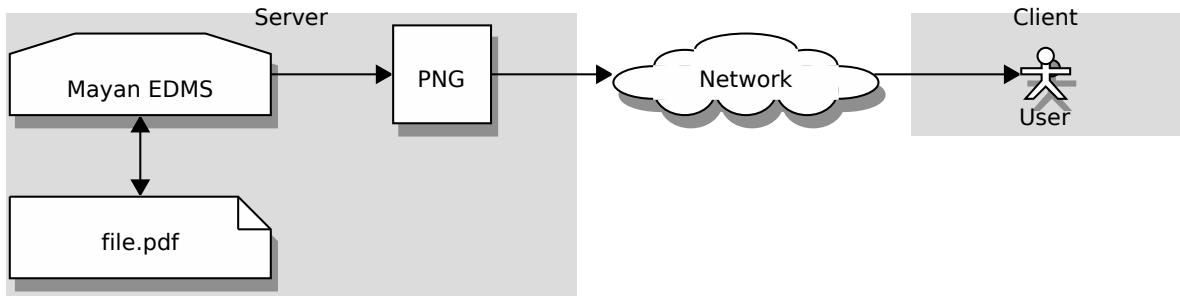


Fig. 2: The visualization system only sends images over the network to the user. The actual document always remains in the server.

Documents can be viewed using several display methods:

- Thumbnails: On any view that displays a list of documents, every document entry will show a thumbnail preview of the first page for quick identification.
- Quick preview: While remaining of the view displaying a list documents, shows a higher resolution preview of the document's first page. This view also enables a gallery mode and allows switching the preview from one document to the next or the previous one.
- Default view: Shows all pages as a vertical stream or sequential images which the user scrolls through. This methods is useful to obtain a quick view of any of the document pages.
- Page list view: This views shows a list of thumbnails of all the document's pages. This view is useful for documents with a large number of pages.
- Interactive page view: Allows performing transformations to a page for closer inspection. The transformations support are zoom and rotation. This view is useful for large format documents like blueprint or high resolution scans.

Images are generated on demand and there will be a processing delay the first time a document's image is accessed. Once accessed the image is retained in the system for faster preview in the future. The images are saved in a designated storage called the image cache. Over time the size of this folder will increase. A tool is provided to clear the document cache.

Settings:

- The image cache defaults to a folder inside the media folder of the installation with the name `document_cache`. The behavior of the document cache storage is controlled by the settings `DOCUMENTS_CACHE_STORAGE_BACKEND` (page 215) and `DOCUMENTS_CACHE_STORAGE_BACKEND_ARGUMENTS` (page 215).
- It is possible to disable the caching of images by changing the settings `DOCUMENTS_DISABLE_BASE_IMAGE_CACHE` (page 215) and `DOCUMENTS_DISABLE_TRANSFORMED_IMAGE_CACHE` (page 215) to True.
- In addition to caching the images in the server, Mayan EDMS also instruct the browser to cache the images. The browser cache time is controlled by the setting `DOCUMENTS_PAGE_IMAGE_CACHE_TIME` (page 216).
- The resolution of the thumbnails is controlled by the settings `DOCUMENTS_THUMBNAIL_HEIGHT` (page 218) and `DOCUMENTS_THUMBNAIL_WIDTH` (page 218).

-
- The resolution of the quick preview is controlled by the settings *DOCUMENTS_PREVIEW_HEIGHT* (page 216) and *DOCUMENTS_PREVIEW_WIDTH* (page 217).
 - The resolution of the sequential preview is controlled by the settings *DOCUMENTS_DISPLAY_HEIGHT* (page 215) and *DOCUMENTS_DISPLAY_WIDTH* (page 215).
-

6.1 How-to

6.1.1 Viewing document thumbnails



Permissions required:

The “View documents” permission is required for this action, either globally or via an ACL for the document or document type.

1. Go to the **Documents → All documents** menu.
2. A list of documents will be displayed. Underneath each document’s title a thumbnail image of the first page will be shown. Images are generated on demand and there will be a processing delay the first time a document’s image is accessed. Once accessed the image is retained in the system for faster preview in the future.

6.1.2 Accessing quick document previews



Permissions required:

The “View documents” permission is required for this action, either globally or via an ACL for the document or document type.

1. Go to the **Documents → All documents** menu.
2. A list of documents will be displayed. Click on the thumbnail of a document.
3. A middle resolution, floating preview image of the document’s first page will be shown. Images are generated on demand and there will be a processing delay the first time a document’s image is accessed. Once accessed the image is retained in the system for faster preview in the future.
4. To exit the quick preview mode, press the Escape key or click on another part of the window.

6.1.3 Previewing documents



Permissions required:

The “View documents” permission is required for this action, either globally or via an ACL for the document or document type.

-
1. Go to the **Documents** → **All documents** menu.
 2. A list of documents will be displayed.
 3. First method: Click on the document's title.
 4. Second method: Click on the document's thumbnail. Click on the document's title underneath the quick preview.
 5. The document preview view will be shown.

6.1.4 Viewing the document page list



Permissions required:

The “View documents” permission is required for this action, either globally or via an ACL for the document or document type.

1. Navigate to a document's preview view.
2. From the sidebar click on the **Pages** button.
3. A list of the document's pages will be shown. A thumbnail will be shown for page.
4. Optional: Click the page thumbnail to show a quick preview of the document page in gallery mode. To exit the quick preview mode, press the `Escape` key or click on another part of the window.

6.1.5 Using the interactive page view



Permissions required:

The “View documents” permission is required for this action, either globally or via an ACL for the document or document type.

1. Navigate to a document's preview view.
2. First method: Click on any of the document page preview from the sequential preview.
3. Second method: From the sidebar click on the **Pages** button. Click the page title or click thumbnail to show a quick preview of the page and click on the page title underneath the quick preview.
4. The interactive page view will be shown.
5. From this view it is possible to:
 - Change the active page
 - Rotate the page
 - Zoom the page
 - Reset the view
6. To return to the document preview view, from the sidebar click on the **Document** button.



Settings:

- The rotation degree increments is controlled by the setting *DOCUMENTS_ROTATION_STEP* (page 217).
- The zoom behavior is controlled by the settings *DOCUMENTS_ZOOM_MAX_LEVEL* (page 218), *DOCUMENTS_ZOOM_MIN_LEVEL* (page 218), and *DOCUMENTS_ZOOM_PERCENT_STEP* (page 218).

6.1.6 Clearing the document image cache



Permissions required:

The “Execute document modifying tools” permission is required for this action.

1. Go to the **Documents → Tools → Clear document image cache** menu.
2. Click the button **Yes** from the confirmation window.
3. The document image cache will be cleared in the background.

6.2 Takeaways

- The visualization system allows viewing documents directly from the user interface.
- It also helps save on licensing costs as no software is required on the client to view the documents.
- Visualization also removes the need to create a hard copy of the document for reading.
- Many of the most popular file formats are supported for visualization.
- Visualization is only a requirement for OCR, every other part of the system will work as normal even when the document file format is not supported for visualization.

NOTES

FINDING DOCUMENTS



Fig. 1: Cole's Revolving "full view" bins, Cole Steel, 1959

Mayan EDMS provides a very comprehensive search system. It allows finding documents regardless of their categorization, or even if they have not been categorized in anyway.

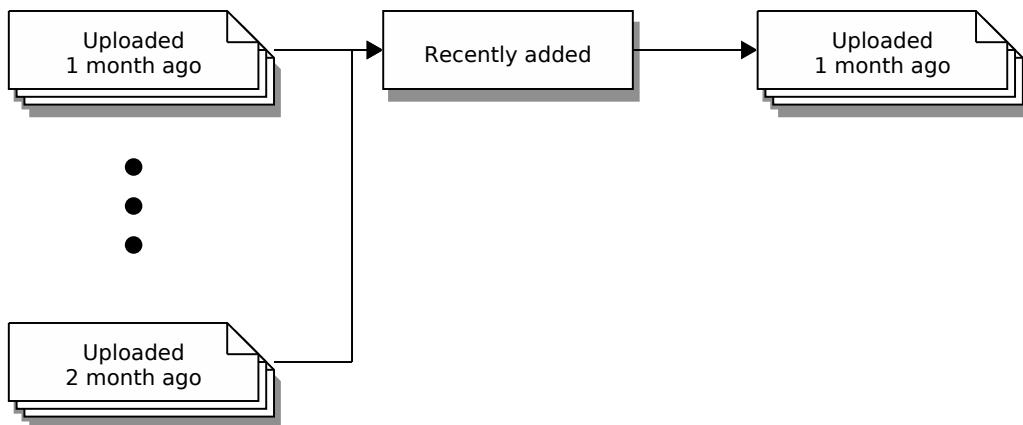


Fig. 2: The Recently added list will filter all documents in the system and provide a list of the most recent ones.

The simplest way to look for a document is to use the Recently added and Recently accessed document lists. The Recently added list will show the most recently created or uploaded documents in the system. The documents will be sorted by the date and time of the interaction, with the document at the top of the list being the most recent.

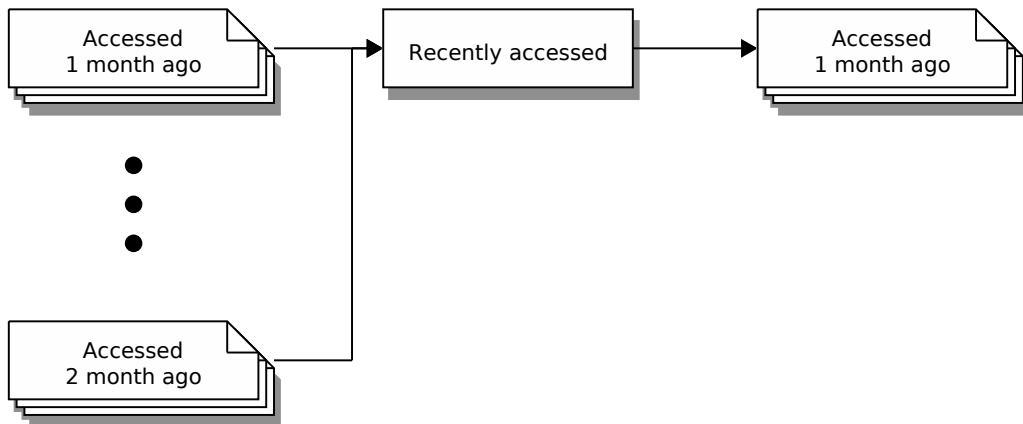


Fig. 3: The Recently accessed list will filter all documents in the system and show a list of the ones you have most recently interacted with.

The Recently accessed list will show the document with which the user has most recently interacted with. The documents will be sorted by the access time, with the document at the top of the list being the most recent.

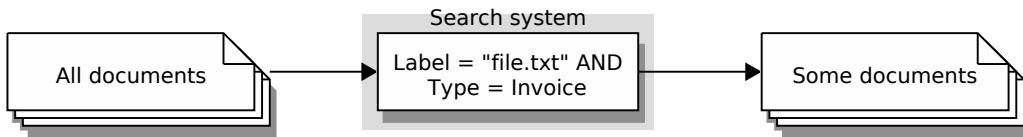


Fig. 4: The unified search system supports full text searching, searching based on the multiple properties of documents, and a search syntax to file tune the search criteria.

To perform a more general document search, the full text search function should be used. It allows displaying a resulting list of documents based on text criteria for the different properties of a document.

The search system has support for a syntax to provide additional control over the search criteria. In its default mode, the text entered as search terms are split into words and the search will return hits only if they contain all the words. This mode is described as an AND type search. For example entering the text:

```
blue red
```

Will only return instances with the words “blue” and “red”. To return results containing any word use the fragment OR as follows:

```
blue OR red
```

It is also possible to exclude a term for the search results by added a minus (-) sign prefix to the term that is to be excluded. This criteria:

```
blue -red
```

will return only results with the word “blue” that also doesn’t contain the word “red”. The criteria:

```
blue OR -red
```

will return documents that contain the word “blue” and any other results that doesn’t contain the word “red”.

To specify compounded terms, enclose the term in quotes. The criteria:

```
"blue read"
```

will only return results with the exact text fragment “blue red”.

7.1 How-to

7.1.1 Viewing the most recent documents



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

1. Go to the **Documents → Recently added** menu.

-
2. A list of the most recently created documents will be shown.
-



Settings:

The number of documents shown in the recently added document list is controlled by the *DOCUMENTS_RECENT_ADDED_COUNT* (page 217) setting.

7.1.2 Viewing the most recent accessed documents



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

1. Go to the **Documents → Recently accessed** menu.
 2. A list of the most recently accessed documents will be shown.
-



Settings:

The number of documents shown in the recently added document list is controlled by the *DOCUMENTS_RECENT_ACCESS_COUNT* (page 217) setting.

7.1.3 Performing a basic search



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

1. Navigate to the home view by clicking the **Mayan EDMS** brand button at the left of the main menu.
2. To retrieve document pages, enter the text to be searched for in the **Search pages** field.
3. To retrieve documents, enter the text to be searched for in the **Search documents** field.
4. Click on the **Search** button of the corresponding field.
5. A list of results will be shown.

7.1.4 Performing an advanced search



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

-
1. Navigate to the home view by clicking the **Mayan EDMS** brand button located to the left of the main menu.
 2. To perform an advanced search that retrieves document pages, click on the **Advanced** button of the field named **Search pages**.
 3. To perform an advanced search that retrieves documents, click on the **Advanced** button of the field named **Search documents**.
 4. A form will be displayed containing all the fields supported for searching the selected type of result.
 5. Enter the text to use as the search criteria in the corresponding field. Multiple entries are supported and each will act as optional criteria. This means that results will be shown if they match at least one of the search criteria. To show only results that match all the search criteria, select the option named **Match all** by clicking on the checkbox.
 6. Click on the **Search** button.
 7. A list of results will be shown.
 8. To repeat the search with slightly modified criteria, from the **Actions** dropdown, click the **Search again** button.
 9. The advanced search form will be displayed and the criteria used in the last search will be already entered in the corresponding fields.

7.2 Takeaways

- Documents can be retrieved regardless of their categorization method.
- Single or multiple criteria can be employed to perform broad or targeted searches.

NOTES

CHAPTER 8

QUICK LABELS

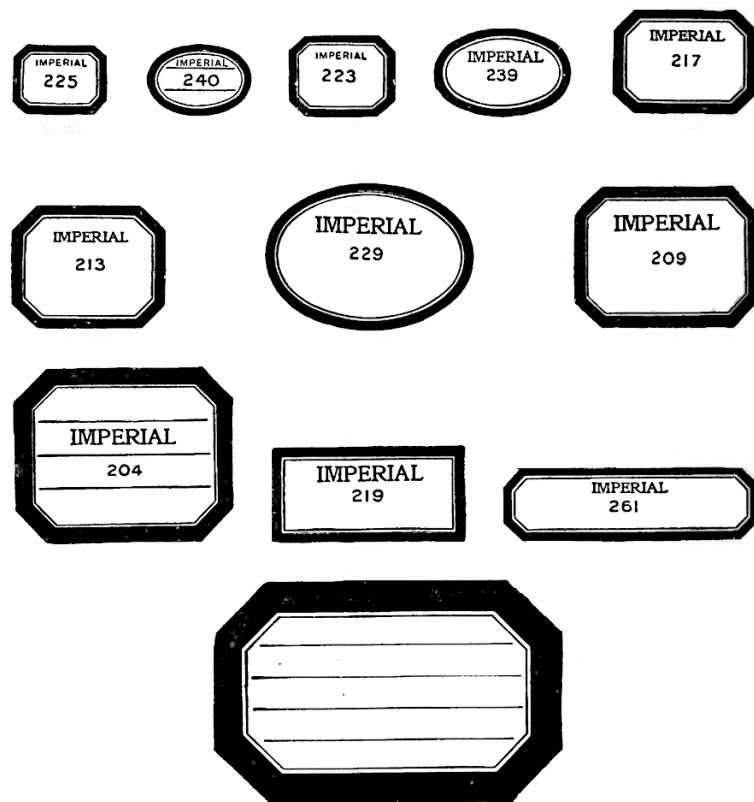


Fig. 1: Gum Labels, W. J. GAGE & Co. Limited, 1911

Quick labels are predetermined filenames that allow the quick renaming of documents as they are uploaded or after they have been uploaded.

Quick labels are added and associated to a document type.

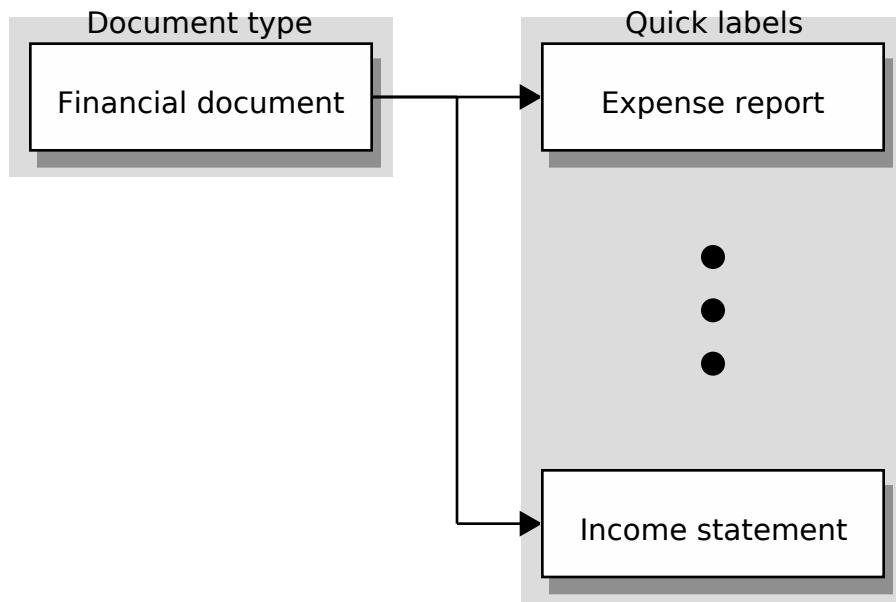


Fig. 2: Use quick labels when a document type can itself be represented by many types of files.

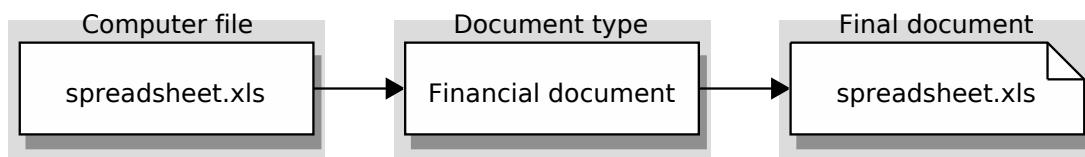


Fig. 3: Documents uploaded without the use of quick labels retain their file name.

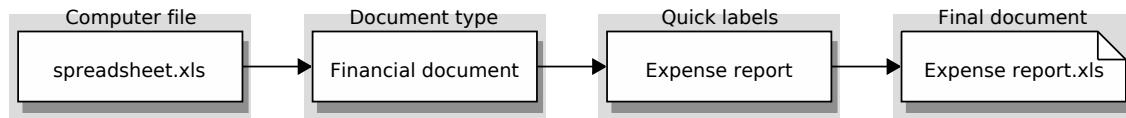


Fig. 4: Quick labels allow for the fast and simple renaming of files into meaningful document labels.

It is possible to preserve the file extension when using quick labels. Extensions are required for some operating system to be able to detect the correct file type to access the content.

For example if a document file is named "file0001.pdf" and the quick label "Receipt from X store" is applied, the resulting document label will be "Receipt from X store.pdf".

8.1 How-to

8.1.1 Creating quick labels



Permissions required:

The “Edit document types” permission is required for this action, either globally or via an ACL for a document type.

Since quick labels are associated with document types, creating quick labels must be done from the document type view.

1. Go to the *System* → *Setup* → *Document types* menu.
2. In the document type list, click on the Quick labels button of the document type for which you wish to create a quick label.
3. In the view titled “Quick labels for document type: <your document type>”, from the Actions dropdown select Add quick label to document type.
4. At the quick label creation form enter the desired label and press Save.

8.1.2 Using quick labels during upload

1. Use the new document upload wizard from *Documents* → *New document*.
2. Select a document type and navigate to the penultimate step, where you have the option to drag and drop files to upload.
3. Select a an option from the Quick document rename dropdown.
4. Optionally select the Preserve extension checkbox to keep the file extension.
5. Upload your documents.

8.1.3 Using quick labels for existing documents



Permissions required:

The “Edit document properties” permission is required for this action, either globally or via an ACL for a document or document type.

1. Navigate to the document preview view of the document to rename. Make sure quick labels for the document type of the document select have been created.
2. From the Actions dropdown select Edit Properties.
3. Select a an option from the Quick document rename dropdown.
4. Optionally select the Preserve extension checkbox to keep the file extension.
5. Press Save to rename the document.

8.2 Takeaways

- Quick labels help keep document labels uniform and in accordance to organization policy.
- Can be used during the upload of new documents or for existing documents.
- Quick labels are associated to document types.

NOTES

neeraj76@yahoo.com

COLLABORATION

Collaboration is a basic feature of any system. It allows keeping discussions and interactions between users in the framework of the system. The track of the decisions regarding the documents is kept alongside the documents in the system for future review.

9.1 Commenting

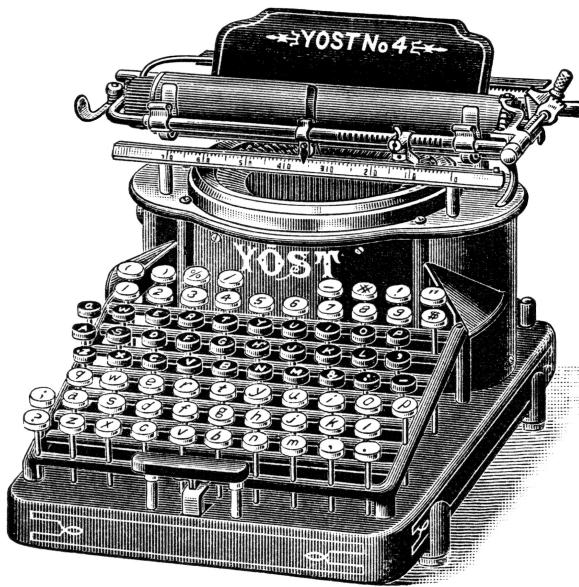


Fig. 1: 1896 Yost Typewriter

Document comments are timestamped, text entries relating to a document. Comments are the most basic form of collaboration in Mayan EDMS and allow for discussions to be recorded for the entire lifetime of a document.

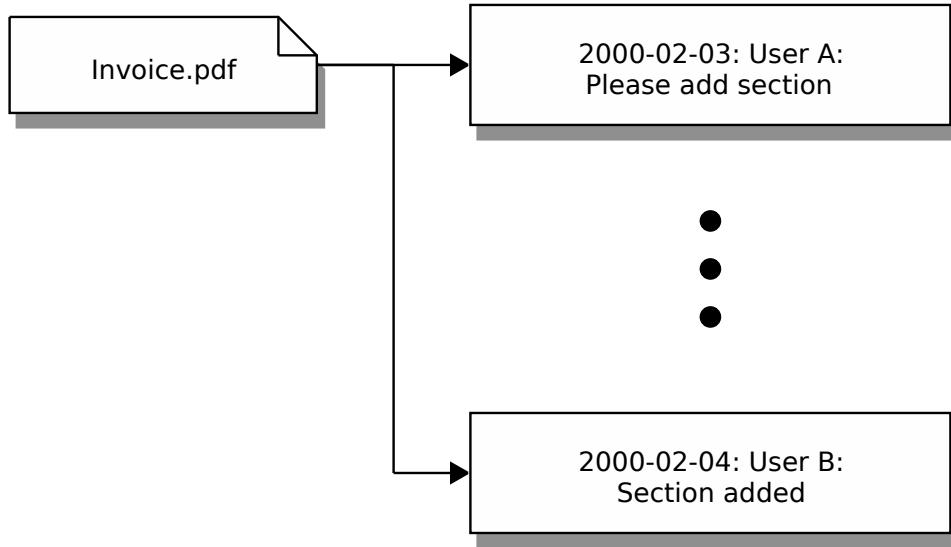


Fig. 2: Comments provide a simple and natural collaboration method.

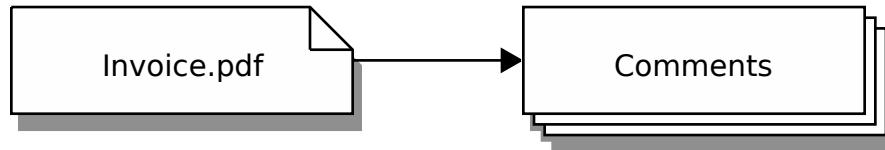


Fig. 3: A document may contain an unlimited number of comments.

Besides being tied to a document and being timestamped, comments show the user making the comments for quick identification.

9.1.1 How-to

9.1.1.1 Viewing document comments



Permissions required:

The “View comments” permission is required for this action, globally or via an ACL for the document.

-
1. Navigate to the selected documents preview view.
 2. In the sidebar click on the **Comments** button.
 3. A list showing the date and time, user, and text of the comments will be shown.

9.1.1.2 Adding a comment



Permissions required:

The “Create new comments” permission is required for this action, globally or via an ACL for the document.

1. Navigate to the selected documents preview view.
2. In the sidebar click on the **Comments** button.
3. From the **Actions** dropdown, select **Add comment**.
4. Enter a comment in the “Comment” field of the form.
5. Click **Submit**.

9.1.1.3 Editing a comment



Permissions required:

The “Edit comments” permission is required for this action, globally or via an ACL for the document.

1. Navigate to the selected documents preview view.
2. In the sidebar click on the **Comments** button.
3. Click on the **Edit** button of the selected comment.
4. Make changes to the comment using the “Comment” field of the form.
5. Click **Save**.

9.1.2 Takeaways

- Comments are the most simple method of collaboration.
- Comments remain attached to the document for the entire lifetime of the document.
- Comments can be edited or deleted.

NOTES

9.2 Checkouts



Fig. 4: Montgomery Model M Program Clock. American Seating Company, 1952

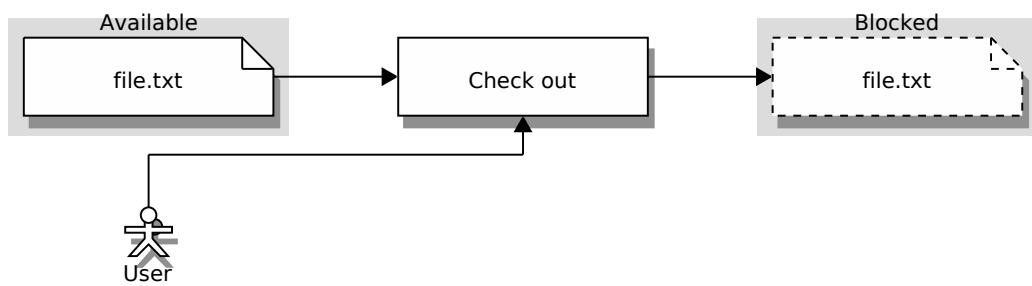


Fig. 5: Checking out a document restricts some functions of the document.

Checkouts allow user to block certain actions being performed on a document, while other work is being performed on the document.

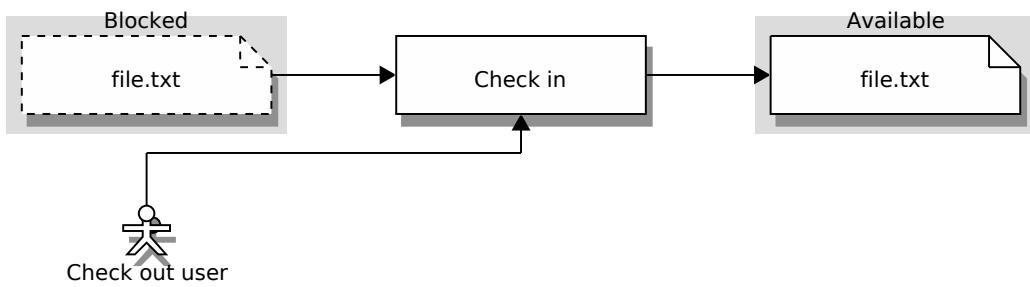


Fig. 6: Checking in a document removes all document restrictions.

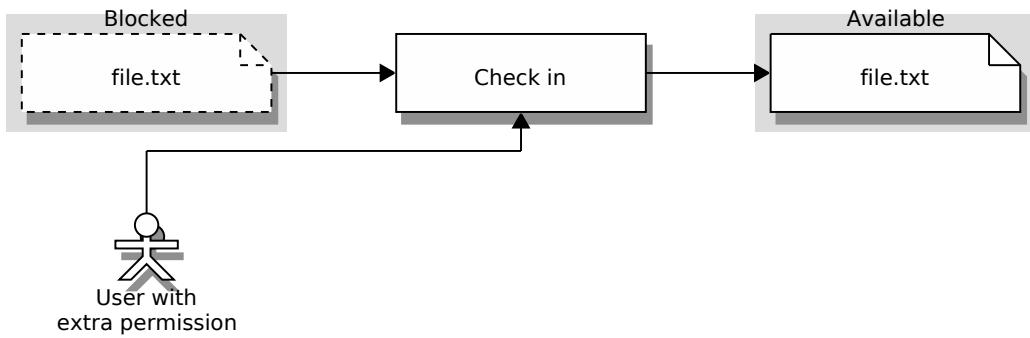


Fig. 7: Users with the Forcefully check in documents permission can check in documents that they didn't check out.

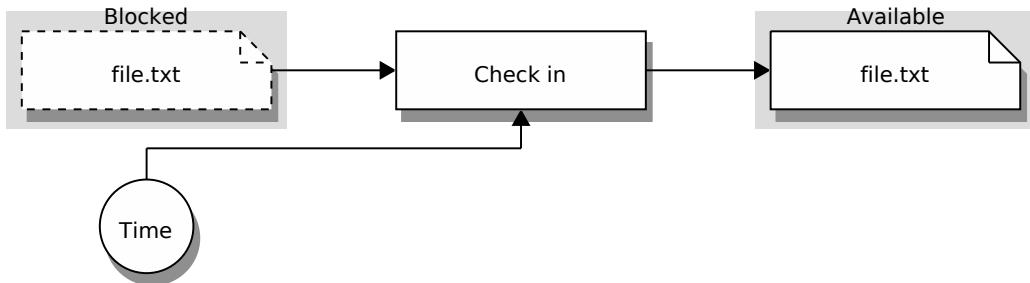


Fig. 8: If not checked in by the user that checked them out, documents are checked in automatically when their check out time expires. This avoids the situation where the documents would be blocked forever.

Checkouts are time-based, meaning it is not possible to block a document permanently. When a document is checked out, a time duration for the checkout is specified.

Checkouts end when one of the following happen:

-
- The user that initiated the checkout, checks in the document.
 - An administrator checks in the document.
 - The checkout expires and the system automatically checks in the document.

For convenience, an entry in the main menu named ***Checkouts***, will show a list of documents currently checked out.

9.2.1 How-to

9.2.1.1 Checking out a document



Permissions required:

The “Check out documents” permission is required for this action, globally or via and ACL for the document.

1. Navigate to the selected documents preview view.
2. In the sidebar click on the **Check in/out** button.
3. From the **Actions** dropdown select **Check out document**.
4. Set the checkout expiration by selecting the time unit and the unit value. Valid time units are: Days, Hours, and Minutes.
5. Optional: Select the checkbox named **Block new version upload** to block other users from uploading a new version of the document.
6. Click the **Save** button.

9.2.1.2 Checking in a document



Permissions required:

- The “Check in documents” permission is required for this action, globally or via and ACL for the document.
- The “Forcefully check in documents” permission is required for this action, globally or via and ACL for the document, to check in documents checked out by other users.

-
1. Navigate to the selected documents preview view.
 2. In the sidebar click on the **Check in/out** button.
 3. From the **Actions** dropdown select **Check in document**.
 4. Click the **Yes** button on the confirmation dialog.

9.2.2 Takeaways

- Checkouts reserve or block a document while a user works on it.
- Checkouts are time based, and if no action is taken, expire on their own.

-
- Only the user that checked out the document may check it in.
 - Super users or users with the “Forcefully check in documents” permission can check in documents that they have not checked out in the first place.

NOTES

9.3 Mailing

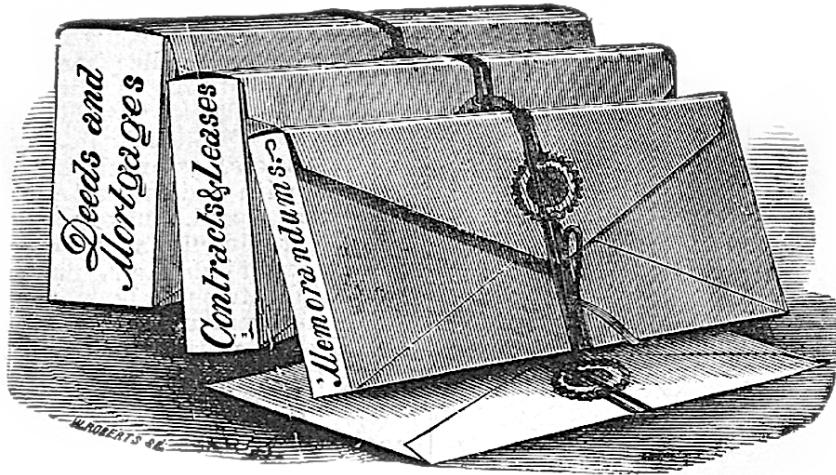


Fig. 9: 1883 Congress Tie Envelope

Mailing documents is used when collaboration with users outside the system is required or to send URL references to documents.

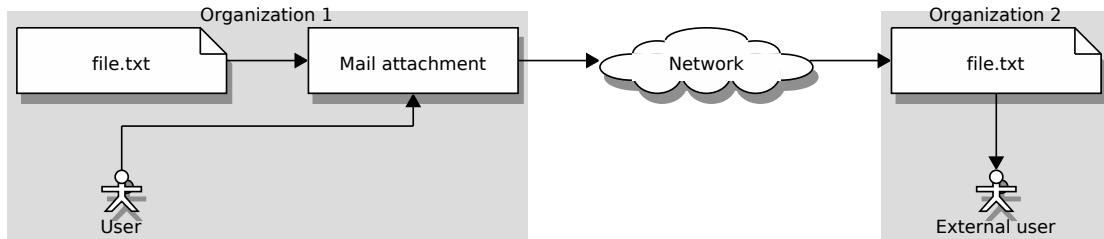


Fig. 10: Mailing documents as attachments allows sharing documents with users that are not part of the Mayan EDMS installation. Documents sent this way are no longer under the control of Mayan EDMS.

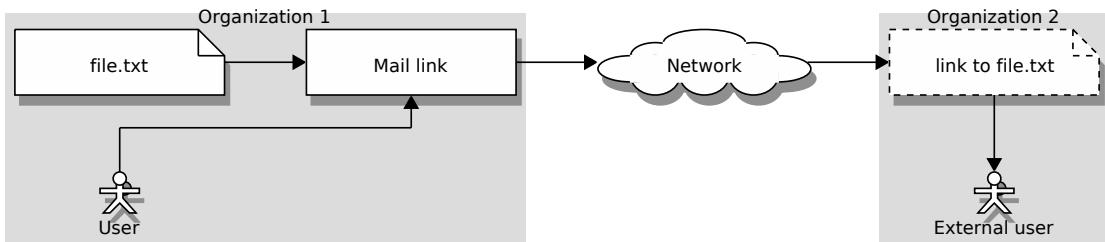


Fig. 11: Mail documents as links to send references to the documents. To access the documents, the users receiving the links will need to log into Mayan EDMS.

There are two ways to send document collaboration emails:

- Send an e-mail with the document as an attachment. The receiving user will be able to download the document from the e-mail. The document at the receiving user's end is no longer under control of the system.
- Send an e-mail with a link reference to the document. The receiving user will not be able to download the document from the e-mail. Instead the user will click on the link and be redirected to the installation where authentication will be required in order to access the document. The document remains under the control of the system.

Because sending e-mails require specialized server software not included with Mayan EDMS, and are part of an organization's information systems, mailing profiles need to be created before this feature can be used. Mailing profiles are created by administrators or staff and allow Mayan EDMS to interface with existing e-mails systems.

9.3.1 How-to

9.3.1.1 Creating a mailing profile



Permissions required:

The “Create a mailing profile” permission is required for this action.

1. Go to the **System → Setup → Mailing profiles** menu.
2. From the **Actions** dropdown select the **Create user mailer**.
3. Select the backend to use for the new mailing profile. Backends allow interfacing with different e-mails systems.
4. Fill the fields to configure the backend for the new mailing profile. These fields will change depending on the backend selected and the configuration of the organization's e-mail systems.
5. Press **Submit**.

9.3.1.2 Mailing a document as an attachment



Permissions required:

-
- The “Send document via email” permission is required for this action, globally or via an ACL for the document.
 - The “Use a mailing profile” permission is required for this action, globally or via an ACL for the mailing profile to be used.
-

1. Navigate to the selected documents preview view.
2. From the **Actions** dropdown select **Email document**.
3. Enter the e-mail address of the recipient. Multiple recipients can be specified.
4. Enter the subject of the email. A default subject is added to faster use. The document being sent can be referenced with the “{{ document }}” marker. When the e-mail is sent this marker will be replaced with the document’s label.
5. Enter the body of the email. A default body message is added to faster use. The document being sent can be referenced with the “{{ document }}” marker. When the e-mail is sent this marker will be replaced with the document’s label.
6. Select the mailing profile to use to send the email from the **Mailing profile** field.
7. Click **Send**.

9.3.1.3 Mailing a document as a link



Permissions required:

- The “Send document link via email” permission is required for this action, globally or via an ACL for the document.
 - The “Use a mailing profile” permission is required for this action, globally or via an ACL for the mailing profile to be used.
-

1. Navigate to the selected documents preview view.
2. From the **Actions** dropdown select **Email link**.
3. Enter the e-mail address of the recipient. Multiple recipients can be specified.
4. Enter the subject of the email. A default subject is added to faster use. The document being sent can be referenced with the “{{ document }}” marker. When the e-mail is sent this marker will be replaced with the document’s label.
5. Enter the body of the email. A default body message is added to faster use. The document being sent can be referenced with the “{{ document }}” marker. When the e-mail is sent this marker will be replaced with the document’s label.
6. Select the mailing profile to use to send the email from the **Mailing profile** field.
7. Click **Send**.

9.3.2 Takeaways

- Mailing allows collaboration with users outside of the system.
- One type of mailing action send the actual document as an attachment while the other sends just a link to the document.

-
- At least one mailing profile needs to be created before this feature can be used.

NOTES

DELETIONS

Office Baskets.



Fig. 1: Office Baskets, 1883

As a precaution against data loss, deletion of a document must be performed as a two step process. Documents to be deleted must first be moved to the trash can. Then, documents are deleted from the trash.

If a document was moved by error to the trash can it can be restored at anytime. Documents deleted from the trash can are deleted permanently and it is not possible to recover them using Mayan EDMS.

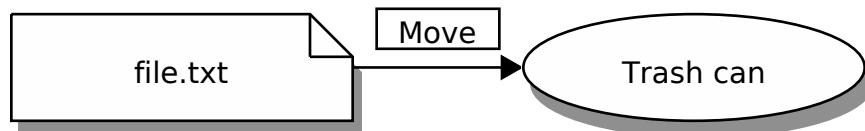


Fig. 2: Before it can be deleted, a document must be moved to the trash can.

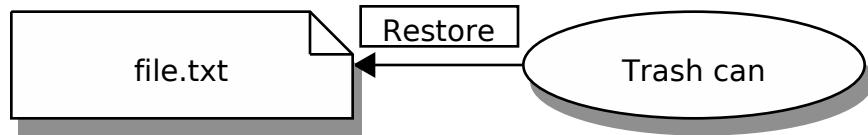


Fig. 3: Documents in the trash can, may be restored at any time.

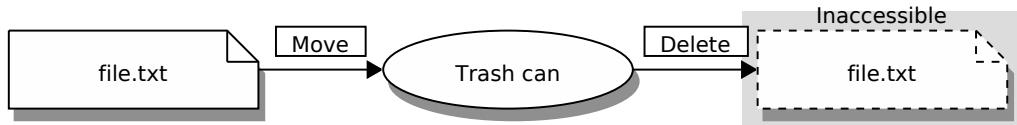


Fig. 4: Documents deleted from the trash can are no longer accessible.

Mayan EDMS can run on many types of systems, some of which are virtualized and don't allow direct access to the computer storage hardware. Because of this no expectations about the underlying storage system are made. This means that when a document is deleted from the trash can, a deletion command is issued for the document file to the computer filesystem, and that is the extent to which Mayan EDMS is involved in the deletion of the file. From that point forward it is the responsibility of the computer system to destroy the content of the document file and reclaim the space used.

What does this mean to you as a user? If you accidentally delete a document from the trash can and wish to recover the file, you must contact your system administrator to perform recovery operations on the computer system. On the opposite side of the spectrum, this means that document files deleted from Mayan EDMS might still be recoverable using specialized computer forensic tools.

Mayan EDMS cannot guarantee a completely safe destruction of document files and if this is a requirement for you, or your organization, additional procedures and tools may be required.

10.1 How-to

10.1.1 Moving a document to the trash can



Permissions required:

The “Trash documents” permission is required for this action, globally or via ACL for the document or document type.

1. Navigate to the document's preview view.
2. From the **Actions** dropdown select **Move to trash**.

-
3. Click the button **Yes** from the confirmation window.
 4. The document is now in the trash can and cannot be accessed anymore from any view.

10.1.2 Viewing the documents in the trash can

**Permissions required:**

The “View documents” permission is required for this action, globally or via ACL for the document type.

1. Go to the ***Documents* → *Trash can*** menu.
2. A list of all the documents in the trash can will be shown.

10.1.3 Restoring documents in the trash can

**Permissions required:**

The “Restore trashed document” permission is required for this action, globally or via ACL for the document type.

1. Go to the ***Documents* → *Trash can*** menu.
2. Select multiple documents by clicking on the checkbox in their title.
3. From the multi-item dropdown select **Restore**.
4. Click the button **Yes** from the confirmation window.
5. The document is now restored and fully accessible.

10.2 Takeaways

- The trash can feature protects against accidental deletion.
- Documents in the trash can may be restored or permanently deleted.

NOTES

MESSAGE OF THE DAY



Fig. 1: Iron signs, The Fred J. Meyers Manufacturing Co., Hamilton, Ohio.

Messages of the day, allow administrators to display messages in the login screen. These can be any kind of message: information, instructions, warnings.

The messages can be configured to become active at a certain date and time, and become inactive after a certain date and time.

11.1 How-to

11.1.1 Create a new message



Permissions required:

The “Create messages” permission is required for this action.

-
1. Go to the *System* → *Setup* → *Message of the day* menu.
 2. From the Actions dropdown select Create messages.
 3. Enter a label to describe the message in the Label field.
 4. Enter the actual message text to be displayed in the Message field.
 5. Check the Enable checkbox to enable the message.
 6. Optionally, select a start date and time, and/or an end date and time.
 7. Click the Submit button.

11.2 Takeaways

- Messages of the day increase the usability of the login screen as an information screen.
- Messages can be always active, active after a certain date, and/or inactive after a certain date.

NOTES

neeraj76@yahoo.com

PART IV.

INTERMEDIATE TOPICS

neeraj76@yahoo.com

CATEGORIZATION

Mayan EDMS provides several ways to organize or categorize documents. Categorization is useful when there is a need to group together documents by a topic, a value, or a purpose.

Example categorizations:

- By customer
- By project
- By case number

12.1 Cabinets

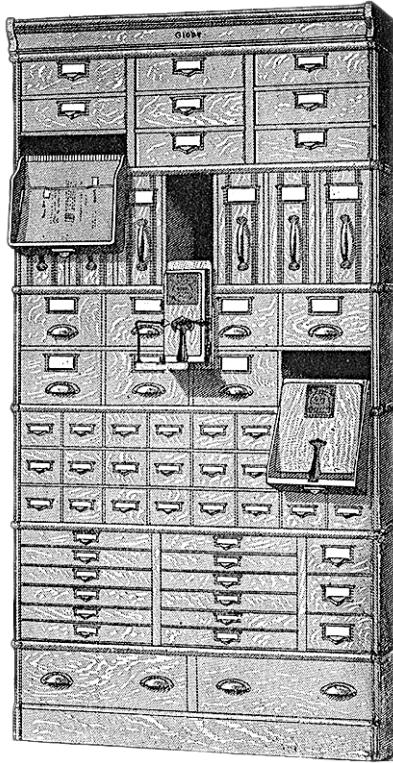


Fig. 1: 1897 Globe Wernicke Cabinet, The Globe Co Cincinnati, OH

Cabinets are a multi-level method to organize documents. Each cabinet can contain documents as well as other sub level cabinets. Cabinet are homologous to filesystem folders in a computer.

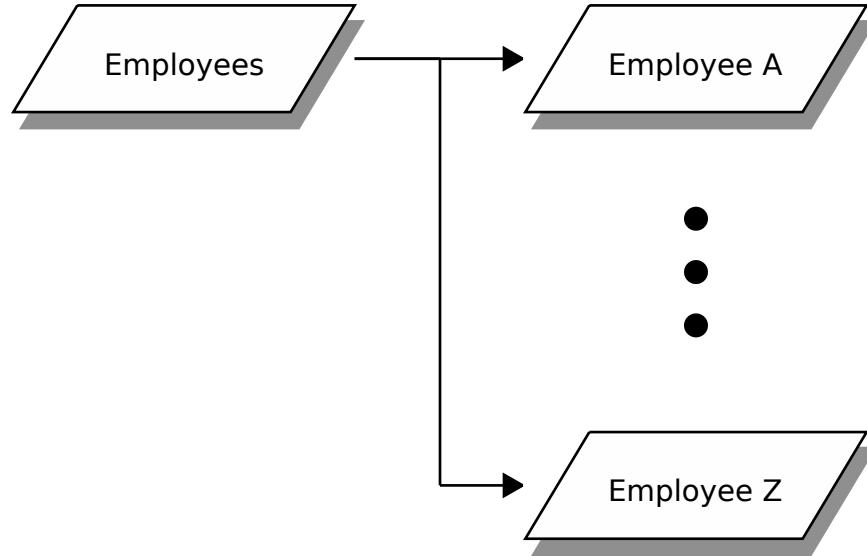


Fig. 2: Cabinets can contain other cabinets.

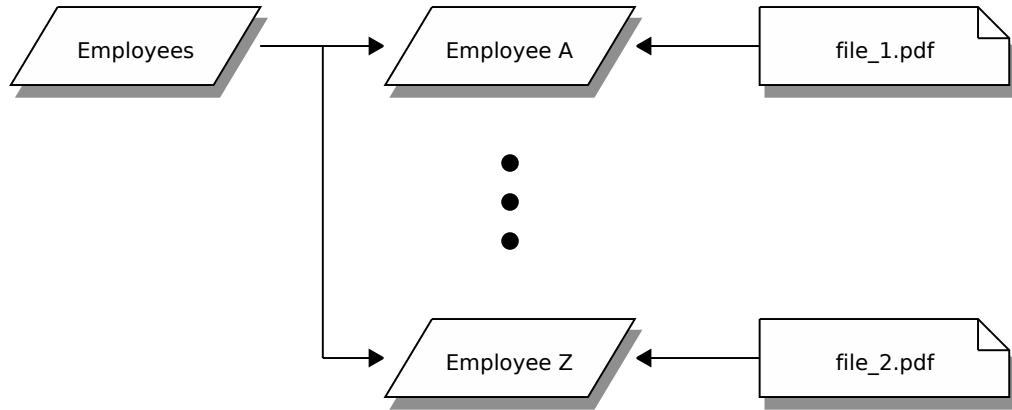


Fig. 3: Add documents to cabinets to organize them.

12.1.1 How-to

12.1.1.1 Creating a cabinet



Permissions required:

The “Create cabinets” permission is required for this action.

1. Go to the *Cabinets* → *Create cabinet* menu.
2. Enter a label to identify the cabinet.
3. Click Submit

12.1.1.2 Creating a subcabinet



Permissions required:

The “Create cabinets” permission is required for this action, globally or via an ACL for the parent cabinet.

1. Go to the *Cabinets* → *All* menu.
2. Navigate to a cabinet.
3. Click on the Add new level link from the cabinet list or from the Action menu.
4. Enter a label to identify the cabinet.
5. Click Submit

12.1.1.3 Adding a document to a cabinet



Permissions required:

- The “Add documents to cabinets” permission is required for this action, globally or via an ACL for the cabinet to which documents will be added, and for the the document that is to be added to the cabinet.
-

1. Go to preview view of the selected document.
2. From the sidebar click on the Cabinets button. This will switch the view to the list of cabinets that contain the document.
3. From the Actions dropdown select Add to cabinets.
4. Select the cabinets that will contain the document. Multiple cabinet can be selected.
5. Click Add.

12.1.1.4 Adding multiple document to a cabinet



Permissions required:

- The “Add documents to cabinets” permission is required for this action, globally or via an ACL for the cabinet to which documents will be added, and for the the document that is to be added to the cabinet.
-

1. Navigate to any view that show a list documents.
2. Select multiple document by clicking on the checkbox in their title.
3. From the multi-item dropdown select **Add to cabinets**.
4. Select the cabinets that will contain the documents. Multiple cabinets can be selected.
5. Click **Add**.

12.1.2 Takeaways

- Cabinet are like filesystem folders.
- A cabinet must be created before it can be populated with documents.
- Unlike a filesystem file with regard to folders, a document may be placed in multiple cabinets at the same time.

NOTES

12.2 Tags

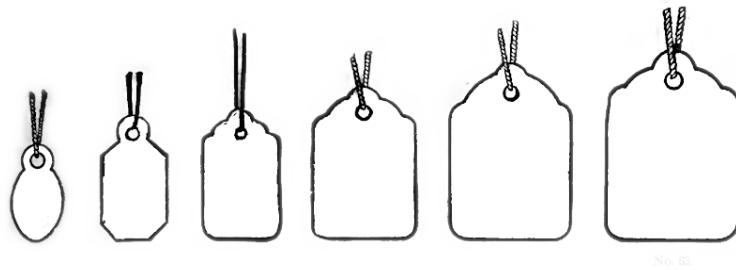


Fig. 4: String tags, W. J. GAGE & Co. Limited, 1911

Tags are color coded properties that can be attached or removed from documents. Tags allow giving documents a binary property. Documents can also be tagged with more than one tag.

Once tagged, documents can be searched by their tags. Conversely, is it also possible to show all the documents tagged with a particular tag.

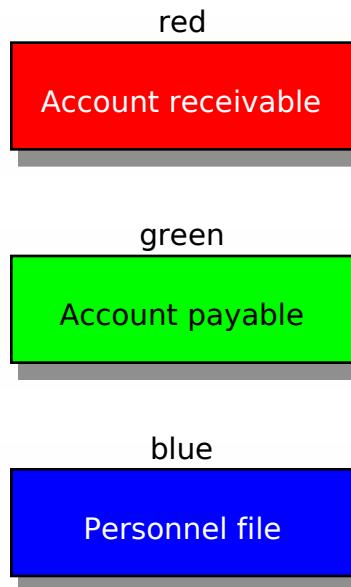


Fig. 5: Any number of tags can be created with different colors and labels.

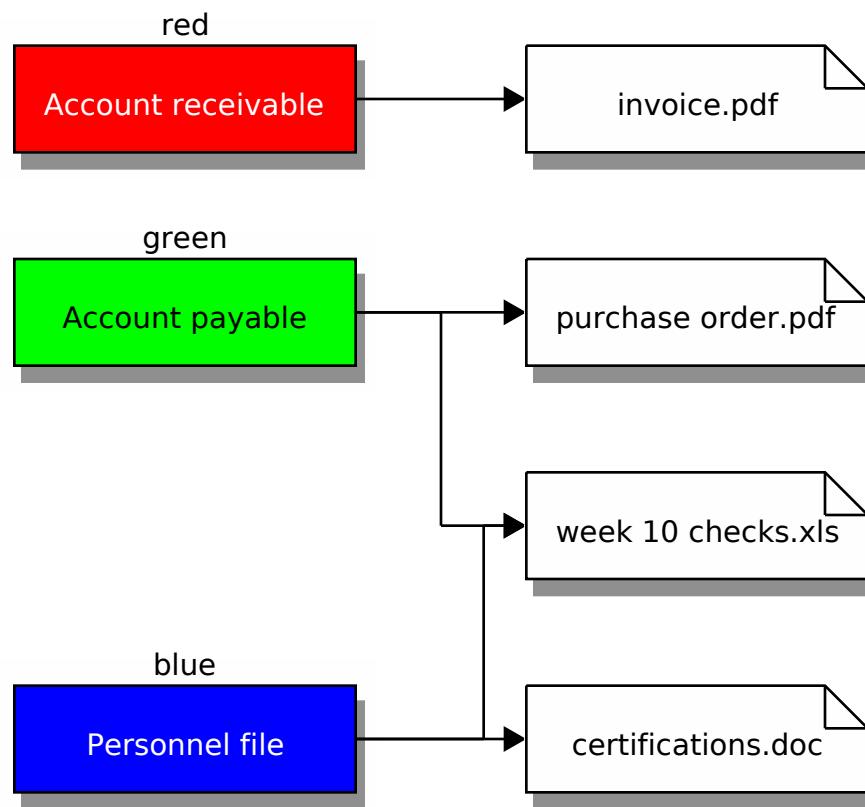


Fig. 6: Attach tags to documents to convey meaning.

12.2.1 How-to

12.2.1.1 Creating tags



Permissions required:

The “Create new tags” permission is required for this action.

1. Go to the **Tags → Create** menu.
2. Enter a label to identify the tag.
3. Select a color for the tag.
4. Press **Submit**.

12.2.1.2 Viewing all tags



Permissions required:

The “View tags” permission is required for this action, globally or via an ACL for a tag.

1. Go to the **Tags** → *All* menu.

12.2.1.3 Viewing the tags of a document



Permissions required:

- The “View tags” permission is required for this action, either globally or via an ACL for the document or document type.
- The “View tags” permission is required for this action, either globally or via an ACL for the tag.

1. Navigate to the document preview view of the selected document.
2. From the sidebar click the **Tags** button.
3. A list of all the tags attached to the document will be shown.

12.2.1.4 Attaching tags to a document



Permissions required:

- The “Attach tags to documents” permission is required for this action, either globally or via an ACL for the document or document type.
- The “Attach tags to documents” permission is required for this action, either globally or via an ACL for the tag.

1. Navigate to the document preview view of the selected document.
2. From the sidebar click the **Tags** button.
3. From the **Actions** dropdown click the button named **Attach tags**.
4. Select the tags to attach to the document from **Tag** field of the form.
5. Click the **Attach** button.

12.2.1.5 Attaching tags to a multiple documents



Permissions required:

- The “Attach tags to documents” permission is required for this action, either globally or via an ACL for the document or document type.
- The “Attach tags to documents” permission is required for this action, either globally or via an ACL for the tag.

-
1. Navigate to any view that show a list documents.
 2. Select multiple documents by clicking on the checkbox in their title.
 3. From the multi-item dropdown select **Attach tags**.
 4. Select the tags to attach to the document from **Tag** field of the form.
 5. Click the **Attach** button.

12.2.2 Takeaways

- Tags are color coded properties.
- Can be attached or removed from documents.
- Multiple tags can be attached to the same document.
- Tags are a quick way to identify documents for any purpose.

NOTES

VERSIONING



Fig. 1: Paper file. Collier Stationery Co. 1909

Mayan EDMS has the ability to store different versions of the same document. Versioning is important for documents that change over time.

A comment field is provided to allow users to summarize the new version changes in comparison with the previous one. If a new version was uploaded by mistake or such new version is no longer necessary, it is possible to revert to a previous version of the document.

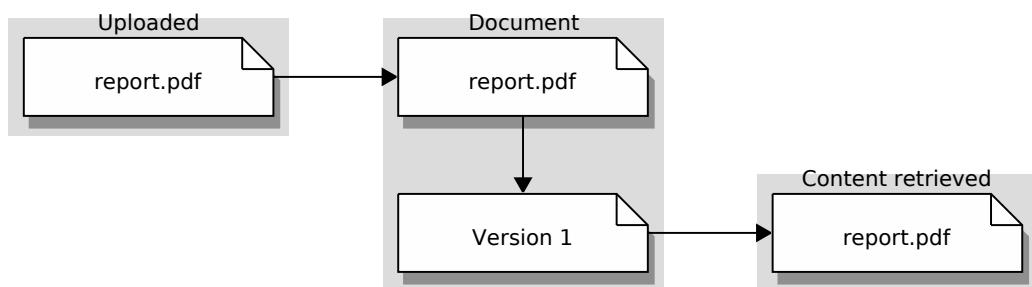


Fig. 2: Every document created contains at least one version.

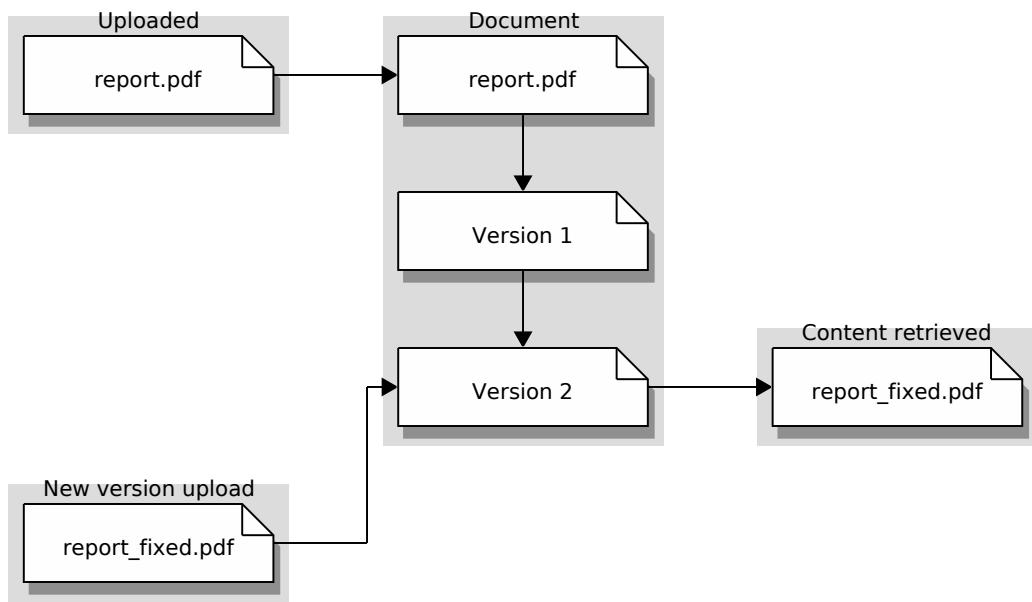


Fig. 3: Adding new version to a document only change the content of the document, every other property (label, metadata, indexes, tags) remains unchanged.

Only the interactive document sources (Web and Staging folders) are available to upload new document versions.

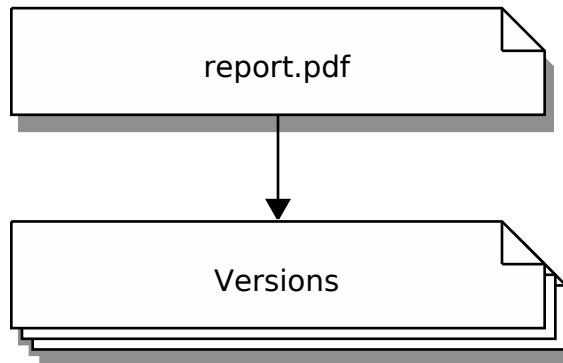


Fig. 4: There is no limit to the number of versions a document can have.

By default, the most recent version will be showed when working with the document, but any of the other versions can be viewed at any time.

13.1 How-to

13.1.1 View a document version list



Permissions required:

The “View documents’ versions list” permission is required for this action, either globally or via an ACL for a document or document type.

1. Navigate to the select document’s preview view.
2. Click on the sidebar’s **Versions** button.

13.1.2 Uploading a new document version



Permissions required:

The “Create new document versions” permission is required for this action, either globally or via an ACL for a document or document type.

1. Navigate to the select document’s versions list view.
2. From the **Actions** dropdown select **Upload new version**.
3. Optionally type a comment explaining the changes in the new version.
4. Press the **Browse** button and select a new file.
5. Press **Save** upload the new version.

13.2 Takeaways

- Versioning helps keep track of document changes.
- Versioning keeps data up to date while using the same document reference.
- The number of versions a document can have are unlimited.
- Document can be reverted to a previous version.

NOTES

DELETION POLICIES

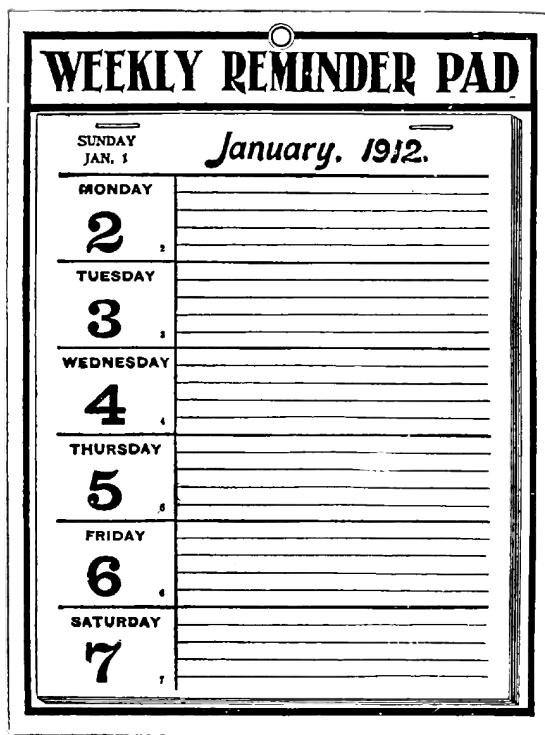


Fig. 1: Weekly reminder pad, Frank A. Weeks MFG Co. 1912

How long should a document be stored? When should a document be destroyed? These are the questions a retention and deletion policy answers.

"A document retention and destruction policy identifies the record retention responsibilities of staff, volunteers, board members, and outsiders for maintaining and documenting the storage and destruction of the organization's documents and records"

—Instructions for IRS form 990⁶

A document's life span is determined by several factors. Such as:

- Value to the organization or the individual. Examples of these documents are datasheets, reference material, manuals.

⁶ <https://www.irs.gov/pub/irs-pdf/f990.pdf>

- Possible future need. Examples are invoices, receipts, tax documents, journals.
- Sentimental or historical value.
- Privacy or security aspects of the documents. Here we can mention documents with personal information, military or corporate trade secrets, human resource documents, and legal documents.

This is by no means an exhaustive list and the priorities will be different for every organization.



Fig. 2: Deletion policies allow moving a document to the trash can after a period of time, and deleting the document after a second period of time.

Mayan EDMS provides a two-step deletion policy setup. The first step determines the time after which the documents of a specific type will be moved to the trash can automatically. The second step determines the amount of time after which the documents in the trash can of the specific type will be permanently deleted. It is possible to configure one step of the deletion policy and not the other. The step that is not configured must be executed manually by the user.

For practical reasons, a default deletion policy is set for new document types. This default policy will delete documents left in the trash can after 30 days. No policy to move documents to the trash can by default is applied. This policy may be changed or disabled at any moment.

14.1 How-to

14.1.1 Setting up deletion policies



Permissions required:

The “Edit document types” permission is required for this action, globally or via ACL for the document type.

1. Go to the **System → Setup → Document types** menu.
2. Click the button named **Deletion policies** for the selected document type.
3. Select the unit of time for the period after which the document will be moved to the trash.
4. Enter a period of time after which documents of this type will be moved to the trash.
5. Select the unit of time for the period of time after which the documents moved to the trash will be permanently deleted
6. Enter a period of time after which documents moved to the trash will be permanently deleted.
7. Press **Submit**.
8. No further user action is required. The deletion policies will be executed automatically in the background by the system.

14.2 Takeaways

- Deletion policies help comply with the organization's policies or with legal requirements.
- It is an automated system that only needs to be configured once.
- Two policies are supported per document type: move to trash, and final deletion.

NOTES

USER MANAGEMENT

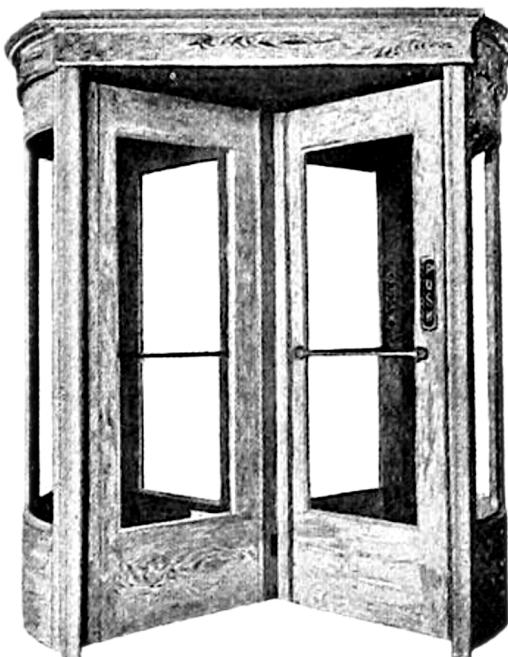


Fig. 1: Revolving doors, The Berlin Interior Hardwood Co. 1914

Like most multi user systems, Mayan EDMS uses the user account paradigm. Besides user accounts, groups are also provided to organize users into units.

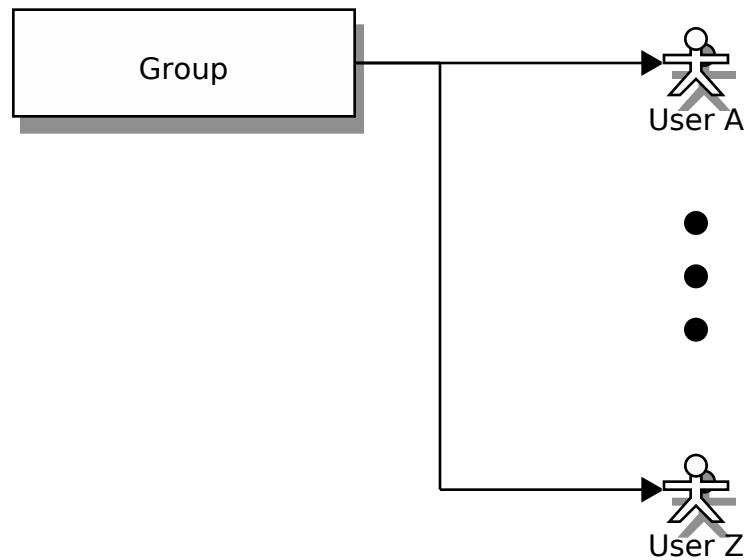


Fig. 2: Each group may contain any number of users.

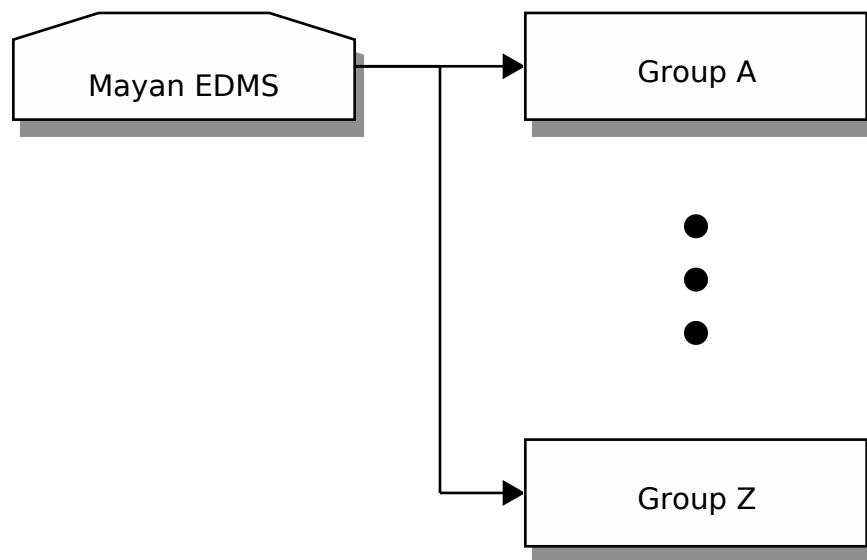


Fig. 3: Each installation of Mayan EDMS may contain any number of groups.

15.1 User accounts

Since user accounts are one of the elements of the access control system, and of the auditing system, it is important that they are assigned to only one person. Allowing more than one person to use the same user account, will negate the purpose of said systems, and possibly others more.

By default the user accounts are managed by Mayan EDMS and kept integrated into the database. However the user accounts system is modular, and if needed, can be adapted to take user accounts information for external systems like LDAP and Microsoft's Active Directory.

To ensure that best practices are followed when setting or changing password, password validators can be enabled. The password validators will verify that the password is not failing to comply with practices that weaken the effectiveness of the password. By default some password validators will be enabled. Examples password validators enabled by default:

- Validator to ensure the password is not too similar to other user personal information.
- Validator to ensure the password is at least 8 characters long.
- Validator to ensure the password is not a commonly used password.
- Validator to ensure the password is not entirely numeric.

15.1.1 How-to

15.1.1.1 Creating user accounts



Permissions required:

The “Create new users” permission is required for this action.

1. Go to the **System** → **Setup** → **Users** menu.
2. From the **Actions** dropdown select **Create new user**.
3. Provide a username to identify the user. Usernames must be unique for entire system.
4. Optional: Enter the first name and last of the user.
5. Optional: Enter the e-mail of the user.
6. Click the **Active** checkbox to mark the user as active or inactive. This checkbox will be selected by default.
Inactive users are not able to login and use the system.
7. Press **Submit**.
8. The user account will be created and you will be redirected to the password change view for the user account.
Provide a password for initial access according to your organization's password security policies. The password will be verified by the password validators that are enabled at the moment. The password will not be visible during data entry.
9. Enter the password in the two fields of the form.
10. Press **Submit**.

15.1.1.2 Changing the password of a user



Permissions required:

The “Edit existing users” permission is required for this action, globally or via ACL for the user.

1. Go to the **System → Setup → Users** menu.
2. Click on the **Set password** button for the selected user.
3. Enter a new password that complies with the shown requisites in the two fields of the form. The new password will not be visible during data entry.
4. Press **Submit**.

15.1.2 Takeaways

- User accounts should be associated to only one user.
- Password validators ensure that more secure passwords are used.
- User accounts are created in the system and remain embedded in the same database by default, but this can be customized so that accounts from external sources are used instead.

NOTES

15.2 User groups

Besides user accounts, Mayan EDMS also supports user groups, or just groups for short. These are collections of user accounts. A user account may belong to more than one group.



Tip: Since groups are organizational units, the groups created in the system should follow the naming convention used by the organization for its personnel workgroups.

15.2.1 How-to

15.2.1.1 Creating user groups



Permissions required:

The “Create new groups” permission is required for this action.

1. Go to the *System* → *Setup* → *Groups* menu.
2. From the Actions dropdown select Create new group.
3. Provide a name to identify the group.
4. Press Submit.

15.2.1.2 Adding a user to a group



Permissions required:

- The “Edit existing users” permission is required for this action, globally or via an ACL for the user.
 - Also the “Edit existing groups” permission is required, globally or via an ACL for a document type.
-

This action can be performed in two ways.

Option 1: Via the user account view

1. Go to the *System* → *Setup* → *Users* menu.
2. Click on the button Groups of the selected user account.
3. Select from the left panel the groups to which the user will be added. Multiple groups can be selected by holding the *Control* or the *Shift* to select many at a time.
4. Click the Add button.
5. Optional: Click the Add all button to add the user to all the groups available a single action.

-
6. No further action is required after the last step, the changes are instantaneous.

Option 2: Via the group view

1. Go to the *System* → *Setup* → *Groups* menu.
2. Click on the button **Users** of the selected group.
3. Select from the left panel the users to be added. Multiple users can be selected by holding the Control or the Shift to select many at a time.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to add all the users available in a single action.
6. No further action is required after the last step, the changes are instantaneous.

15.2.2 Takeaways

- Groups are organization units for user accounts.
- There is no limit to the number of groups that can be created.
- Groups should mirror the organization's personnel workgroups.
- A user account may belong to multiple groups.

NOTES

ACCESS CONTROL



Fig. 1: Cash box, Collier Stationery Co, 1909

In computer systems, once the identity of the user has been confirmed, the level of access or “authorization” (not to be confused with *Authentication* (page 23)) to resources can then be specified.

Mayan EDMS uses an extensible, role-based, access control system to control how access to objects and system functions are defined.

The access control system is a policy-neutral that revolves around roles, permissions, and objects. Since it is policy-neutral, it can address many authorization scenarios used by commercial and government organizations.

The access control system provides two modes of operation. A simpler one using two elements to define the access control and the second one providing very specific access control using three elements.

16.1 Permissions

The first method uses a permission and a role. A permission is granted to a role. Then users are added to a group, and the group is added to the role. This will cause all users to inherit the role’s permissions. These permissions will be inherited for all the objects and actions of the system. An object is anything that can be created, edited, or deleted. Examples are documents, tags, cabinets.

Since there is a limited number of actions that can be performed, there is a limited number of permission. These are grouped by topic or by the app that declares them. Since the system is policy-neutral, there are no predefined roles and system administrator are free to create an unlimited number of roles.

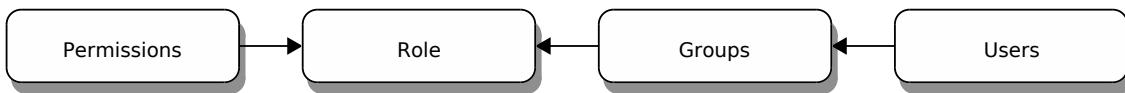


Fig. 2: Permissions are granted to the role. Users are added to groups which are added to the role.

16.1.1 How-to

16.1.1.1 Creating a role



Permissions required:

The “Create roles” permission is required for this action.

1. Go to the **System** → **Setup** → **Roles** menu.
2. From the **Actions** dropdown select **Create new role**.
3. Enter a label to identify the role.
4. Press **Submit**.

16.1.1.2 Granting permissions to a role



Permissions required:

The “Edit roles” permission is required for this action, globally or via an ACL for the role.

1. Go to the **System** → **Setup** → **Roles** menu.
2. Click on the link **Role permissions** for the role to which permissions will be granted.
3. Select from the left panel the permission to be granted. Multiple permissions can be selected by holding the Control or the Shift key.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to grant all the permissions to the role with a single action.
6. No further action is required after the last step, the changes are instantaneous.

16.1.1.3 Adding groups to a role



Permissions required:

- The “Edit roles” permission is required for this action, globally or via an ACL for the role.

-
- The “Edit groups” permission is required for this action, globally or via an ACL for the groups to be added.
-

1. Go to the *System* → *Setup* → *Roles* menu.
2. Click on the link **Role groups** for the role to which will receive the new groups.
3. Select from the left panel the groups to be added. Multiple groups can be selected by holding the **Control** or the **Shift** key.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to add all the groups to the role with a single action.
6. No further action is required after the last step, the changes are instantaneous.

16.1.2 Takeaways

- Permissions are not granted to users or groups, they are granted to roles.
- An unlimited number of roles can be created.
- Users are added to groups, groups are added to roles.
- Granting a permission directly to a role, grants it for the entire system and all objects.

NOTES

16.2 Access Control Lists

The second method involves three elements. A permission is granted to a role in association with an object. Then users are added to a group which in turn is added to the role. This will cause all users to inherit the role's permissions but only for the object specified. This access control method will allow an administrator to grant editing access to a set of documents to a group of users while allowing only viewing access to a different set of users. The combination of a permission, a role, and an object is called an Access Control List. Access Control Lists can be created for almost all of the objects in the system.

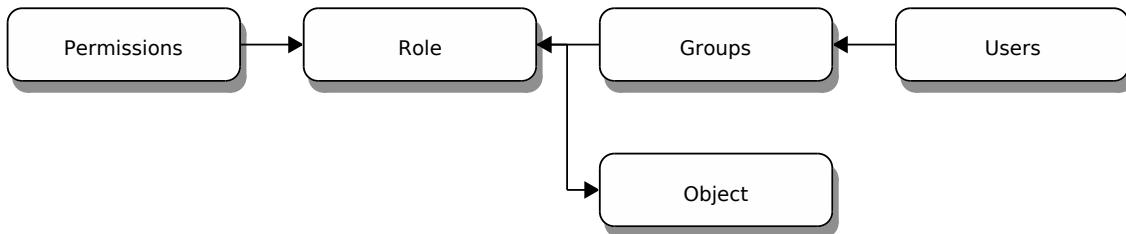


Fig. 3: Permissions are granted to a role but only for an object. Users are added to groups which are added to roles. This constitutes an Access Control List Entry.

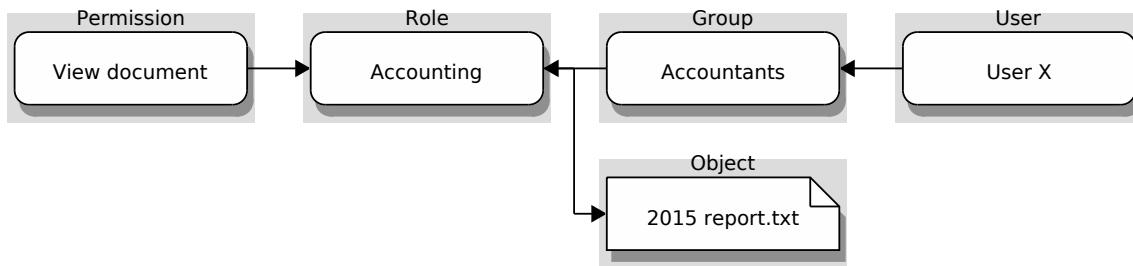


Fig. 4: Example of an Access Control List entry. Users in groups belonging to the Accounting role would be able to view the 2015 report.txt document.

16.2.1 Inherited access control

Some objects are associated and related to other objects. For example, documents need to have a document type before they can be created. If the Access Control List entry is created for a document type, every document of that type will also inherit the Access Control List Entry.

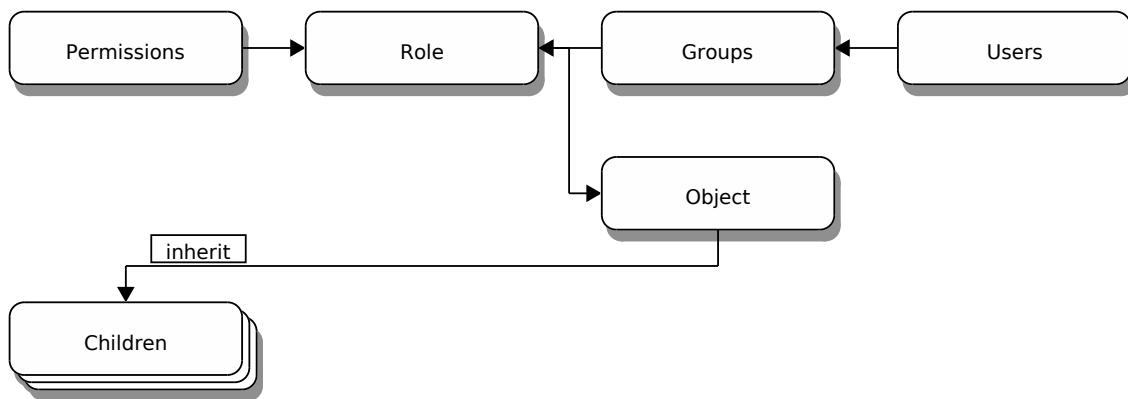


Fig. 5: Children objects will inherit the Access Control List Entry of their parent objects.

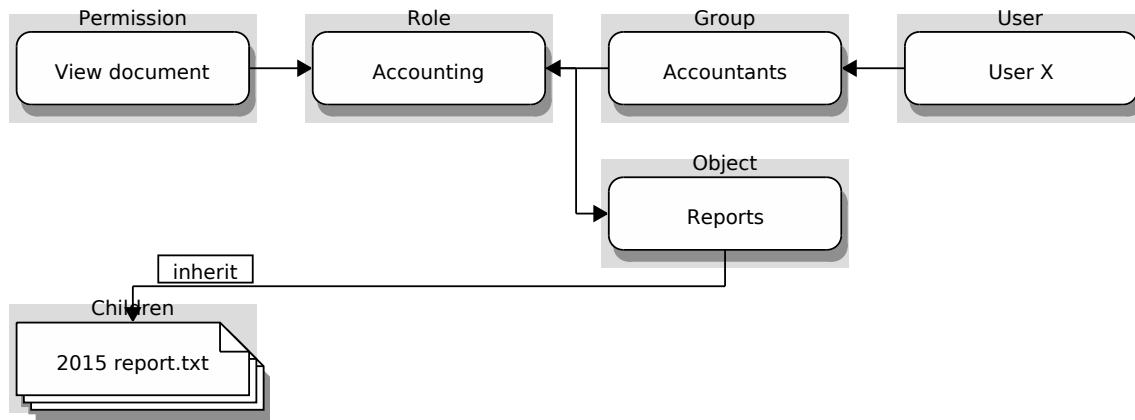


Fig. 6: Example of an inherited Access Control List entry. Users in groups belonging to the Accounting role would be able to view the all documents of type Reports which include the document 2015 report.txt document among others. If access control needs to be updated, it only needs to be done for the document type and not for each document of the type Reports.

16.2.2 How-to

16.2.2.1 Creating an Access Control List Entry



Permissions required:

- The “Create roles” permission is required for this action if a new role will be created. The “Edit roles” permission is required for this action, globally, or for the role, if the role will be updated.

-
- The “Edit ACLs” permission is required for this action, globally or via and ACL for the object for which the ACL will be created.
-

1. Go to the *System* → *Setup* → *Roles* menu.
2. From the **Actions** dropdown select **Create new role**.
3. Enter a label to identify the role.
4. Don’t grant any permissions to the role.
5. Press **Submit**.
6. Go to the *System* → *Setup* → *Document types* menu.
7. From the sidebar click on the **ACLs** button.
8. From the **Actions** dropdown select **New ACL**.
9. Choose the role from the selection form.
10. Press **Save**.
11. Select the permissions to grant from the left panel and click **Add**.
12. Select from the left panel the permission to be granted. Multiple permissions can be selected by holding the Control or the Shift key.
13. Click the **Add** button.
14. Optional: Click the **Add all** button to grant all the permissions to the role with a single action.
15. No further action is required after the last step, the changes are instantaneous.

16.2.3 Takeaways

- An ACL entry is the result of a set of permissions granted to a role for an object.
- ACLs allow for very specific access control to objects.
- ACLs can be created for parent objects like document types, which will be inherited by all child objects, in this case documents.

NOTES

DOCUMENT DATA

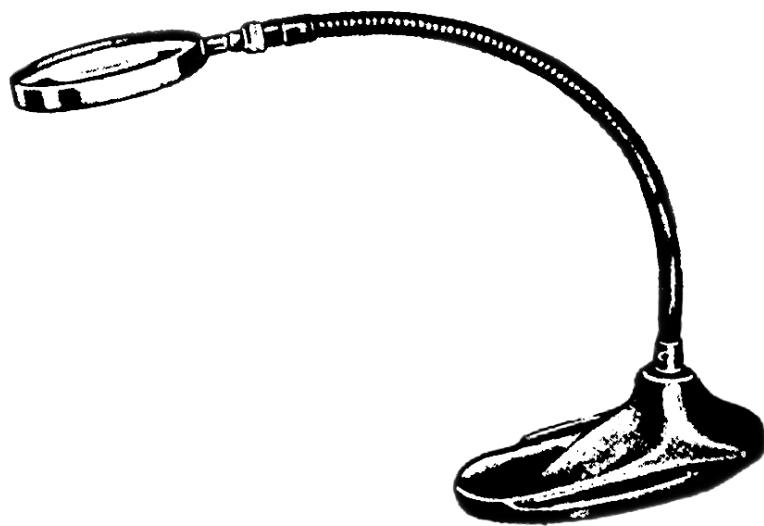


Fig. 1: Gooseneck magnifier. J P Roberts Co

Document data is a very broad topic, as we'll see below. Mayan EDMS supports working with several types of data associated with a document. It extracts this information and makes it available for several purposes like reading, searching, or indexing.

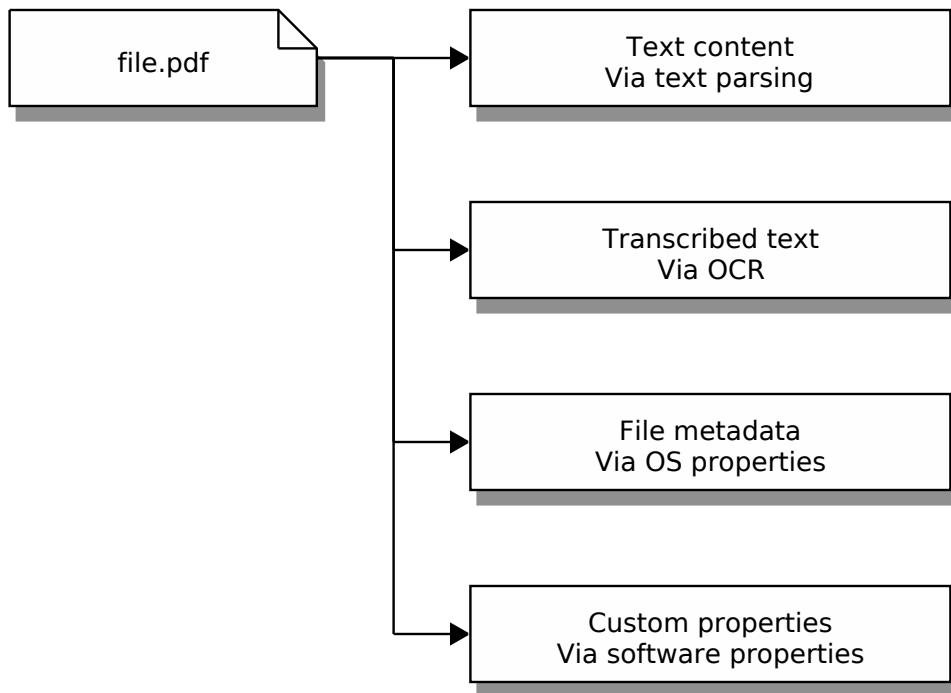


Fig. 2: A single document has many types of data associated with it. Each type of data needs to be accessed by a different method.

17.1 Document data levels

17.1.1 First level of document data

The first level of document data is the one carried by the document. In essence the document itself.

Sometimes this data is readily accessible as text data and can be extracted mechanically by a process called parsing. Other times the document does not contain text data but images. Data can be extracted from images by a process called Optical Character Recognition or OCR.

Text data and image data are part of the document and unless edited on purpose, won't change. This immutability of the document data remains regardless of the software used to access them, the operating system, or the storage medium.

17.1.2 Second level of document data

The second level of document data is the one obtained by the format and method used to compose the document. We are talking about the file's properties.

The document file's properties change based on the format used to create the document file and the way it is stored. These kind of file attributes are collectively described in Mayan EDMS as file metadata. File metadata is another layer of information about the document and needs to be extracted using different methods. File metadata may contain information such as:

- EXIF field: Use for JPEG images, it contains aperture, exposure, ISO and other properties at the moment the photo was taken. If enabled, might even contain the GPS coordinates where the picture was taken.
- Authorship information: Office documents may store the name of the official author of a document, but may also contain more technical information such as the username who saved the document, filesystem directory where it was originally saved to, creation and modification times, among others.
- MP3 ID tags: Contain information for music files. May include track number, author, copyrights, album, image art, license information, etc.

Improper use or configuration of file metadata may lead to severe security and privacy problems. Properly used, it is an invaluable tool to determine the provenance of a document, check authenticity, validate licensing, follow copyright, perform forensic investigation, rebuild or rescue files.

17.1.3 Third level of document data

The third level of document data is the data that is associated to a document by a third party that doesn't change the document itself. This is data that is added to the collective knowledge about the document, but is not part of the document itself. This is data created about a document via a filing or archiving system, or a software.

It can be argued that besides the character data, the OCR, and the file metadata, everything else in Mayan EDMS is third level document data.

While this is semantically true, if we accept this premise, every part of this written material would need to be filed under this paragraph. For the practical purpose of feature classification, we will limit the scope of third level document data, to the system of custom document attributes that Mayan EDMS possess called simple document metadata.

17.2 Document parsing



Fig. 3: Egry Autographic Register. Egry Autographic Register Co.

Parsing is the act of reading the text information from a document and making it accessible to the user and, to the rest of the system.

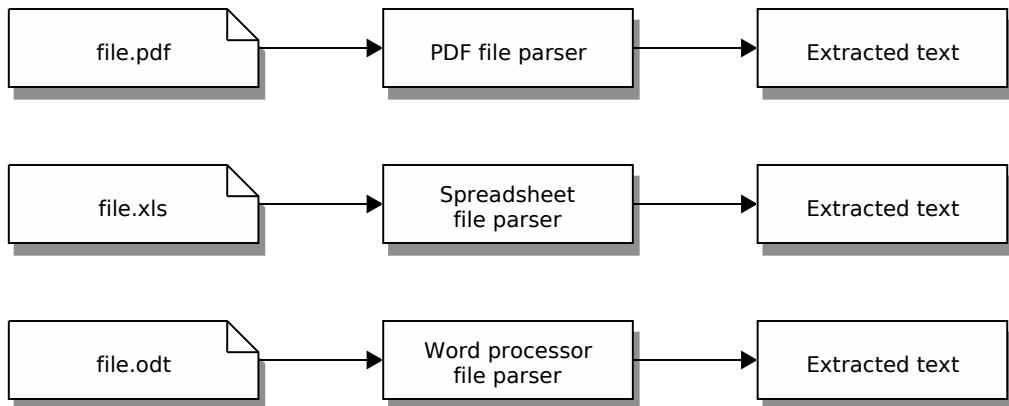


Fig. 4: Each document file format needs a different text parser.

This action is not always straight straightforward and for some documents in unsupported, or proprietary formats, might not be possible at all.

Data extracted by parsing is stored in relation to the page from where it was stored. This means that parsed data may be accessed as a whole for the document or inspected by page.

17.2.1 How-to

17.2.1.1 Configuring document parsing



Permissions required:

The “Change document type parsing settings” permission is required for this action, globally or via ACL for the document type.

1. Go to the **System → Setup → Document types** menu.
2. Click the **Setup parsing** button of the selected document type.
3. Select the **Automatically queue newly created documents for parsing**. checkbox to have every new document of this type be submitted for content parsing.

17.2.1.2 Changing default document parsing



Permissions required:

The “Edit settings” permission is required for this action.



Important: Changes to the settings require a system restart to take effect. Mistakes in the format or value of a setting may cause the system to not start properly or start at all. These are fixable but require a roll back using the `revertsettings` command.

1. Go to the `System → Setup → Settings` menu.
2. In the `Settings namespaces` click on the `Settings` button of the `Document parsing` line.
3. Click `Edit` button of the `DOCUMENT PARSING AUTO PARSING` line.
4. Change the value in the `Value` field of the form. A value of `true` will cause all new document types created to have the `Automatically queue newly created documents for parsing` checkbox selected. A value of `false` will cause all new document types created to have the `Automatically queue newly created documents for parsing` checkbox unselected.
5. Click `Save`.
6. Restart the installation.

17.2.1.3 Viewing the parsed content of a document



Permissions required:

The “View the content of a document” permission is required for this action, globally or via an ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on `Content` button.
3. The document parsed text will be shown, divided by page.

17.2.1.4 Downloading the parsed content of a document



Permissions required:

The “View the content of a document” permission is required for this action, globally or via an ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on `Content` button.
3. From the `Actions` dropdown, click on `Download content` button.
4. The text content of the document will be downloaded as a text file with a filename composed of the document’s label plus the ‘-content’ suffix.

17.2.2 Takeaways

- Parsed content is the actual text data from a document.
- Document types can be configured so that new documents are parse automatically or not.
- The document's file format will determine if the text document can be parsed or not.

NOTES

17.3 Optical Character Recognition (OCR)

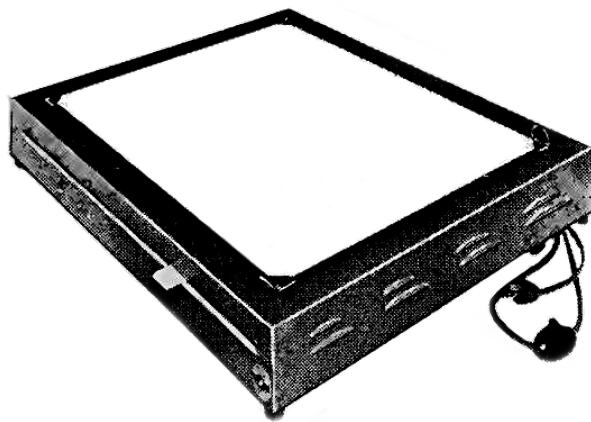


Fig. 5: Portable tracing unit. Carl Mfg Co. 1967

OCR processing allows generating text information from documents (or document pages) that do not contain text that can be normally parsed. OCR is like transcribing an image. In essence, it gives the computer the ability to understand letters and words, so that the computer is able to produce text out of images.

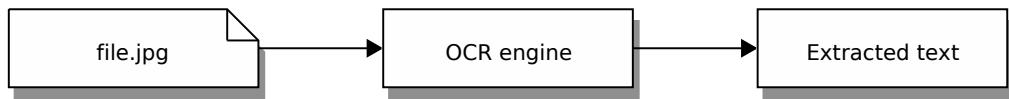


Fig. 6: The OCR engine “reads” the images in a document and transcribes them into text.

This process is very complex and never produces text that is 100% exact. As such, OCR text should never be used as the sole input for any document processing function.

Documents can be parsed and/or processed for OCR, the features are non exclusive. This means that a document with text content and image content will produce text from parsed from the internal text of the document and text from the OCR processing of the document images.

17.3.1 How-to

17.3.1.1 Configuring document OCR processing



Permissions required:

The “Change document type OCR settings” permission is required for this action, globally or via ACL for the document type.

1. Go to the *System → Setup → Document types* menu.
2. Click the Setup OCR button of the selected document type.
3. Select the Automatically queue newly created documents for OCR checkbox to have every new document of this type be submitted for OCR processing.

17.3.1.2 Changing default document OCR processing



Permissions required:

The “Edit settings” permission is required for this action.



Important: Changes to the settings require a system restart to take effect. Mistakes in the format or value of a setting may cause the system to not start properly or start at all. These are fixable but require a roll back using the `revertsettings` command.

1. Go to the *System → Setup → Settings* menu.
2. In the Settings namespaces click on the Settings button of the OCR line.
3. Click Edit button of the OCR_AUTO_OCR line.
4. Change the value in the Value field of the form. A value of `true` will cause all new document types created to have the Automatically queue newly created documents for OCR checkbox selected. A value of `false` will cause all new document types created to have the Automatically queue newly created documents for OCR checkbox unselected.
5. Click Save.
6. Restart the installation.

17.3.1.3 Viewing the OCR content of a document



Permissions required:

The “View the transcribed text from document” permission is required for this action, globally or via an ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on **OCR** button.
3. The transcribed text from the document images will be shown, divided by page.

17.3.1.4 Downloading the parsed content of a document



Permissions required:

The “View the transcribed text from document” permission is required for this action, globally or via an ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on **OCR** button.
3. From the **Actions** dropdown, click on **Download OCR text** button.
4. The transcribed text from the document will be downloaded as a text file with a filename composed of the document’s label plus the ‘-OCR’ suffix.
 - OCR transcribed the images of a document to text.
 - It is not a 100% accurate process and should not be relied on for use in automatic functions like categorization.
 - Document types can be configured to process new documents for OCR automatically, or to skip them.
 - As long as an image preview can be generated for a document, it will be processed for OCR.

NOTES

17.4 File metadata



Fig. 7: US Postal scale. Collier Stationery Co. 1909

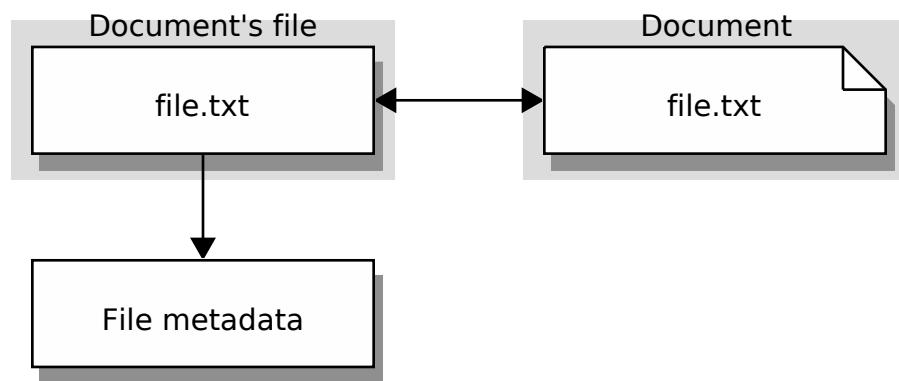


Fig. 8: File metadata are the properties of the actual file that composes the document.

File metadata extraction allows property and value pairs from a document's file. File metadata introspects the file that composes the document for embedded information that is not normally accessible.

The amount of information extract will depend on the extraction driver being used and the format of the document's file. Some file type have little information others have a greater amount of information.

Like parsing and OCR, information obtained from the file metadata processing can be used for search or for system features like categorization.

Parsing and OCR do not interfere with file metadata extraction and can be performed on the same document.

17.4.1 How-to

17.4.1.1 Configuring document file metadata processing



Permissions required:

The “Change document type file metadata settings” permission is required for this action, globally or via ACL for the document type.

1. Go to the *System* → *Setup* → *Document types* menu.
2. Click the Setup file metadata button of the selected document type.
3. Select the Automatically queue newly created documents for processing checkbox to have every new document of this type be submitted for file metadata processing.

17.4.1.2 Changing default document file metadata processing



Permissions required:

The “Edit settings” permission is required for this action.



Important: Changes to the settings require a system restart to take effect. Mistakes in the format or value of a setting may cause the system to not start properly or start at all. These are fixable but require a roll back using the `revertsettings` command.

1. Go to the *System* → *Setup* → *Settings* menu.
2. In the Settings namespaces click on the Settings button of the File metadata line.
3. Click Edit button of the FILE_METADATA_AUTO_PROCESS line.
4. Change the value in the Value field of the form. A value of `true` will cause all new document types created to have the Automatically queue newly created documents for processing checkbox selected. A value of `false` will cause all new document types created to have the Automatically queue newly created documents for processing checkbox unselected.
5. Click Save.
6. Restart the installation.

17.4.1.3 Viewing the file metadata of a document



Permissions required:

The “View file metadata” permission is required for this action, globally or via an ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on **File metadata** button.
3. A list of drivers that may have produced any results during the document’s file metadata analysis will be shown. A column will show the number of attributes produced by each driver.
4. Click on the **Attributes** button of the driver entry.
5. A list of attributes and value pair will be shown for each attribute of file metadata that the driver was able to extract from the document’s file.

17.4.2 Takeaways

- File metadata are attributes from the document’s file.
- The format of the document’s file will determine which file metadata attributes can be extracted.
- Each document file type will produce a different list of file metadata attributes.

NOTES

17.5 Document metadata

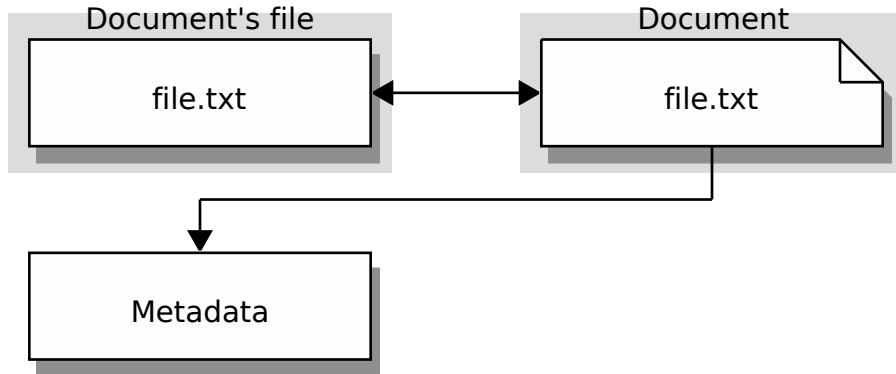


Fig. 9: Metadata are custom properties that are added to documents.

Metadata is the name of the attribute of a document. The concept of metadata is divided in two: **metadata types** (size, color, distance) and **metadata values** for those types. Metadata types are defined in the setup menu and associated with document types. Then, when a document is uploaded, a value for that metadata can be entered. There are two kinds of metadata type to document type relations: optional and required. When a metadata type is optional for a document type, it can be left blank for a document being uploaded and the upload will still be successful. On the other hand, required metadata type must be given a value or it will not be possible to upload the document at hand.

Examples of metadata type: Invoice number, color, employee id.

The data entry of metadata types can be set to allow any value to be provided (the default) or a list of possible values can be entered in the `Lookup` configuration option and users will be presented with a drop down list of options instead of the default text entry box.

If metadata types are setup to allow any value to be entered a `validation` option can be chosen to block the entry of invalid data. Metadata types also provide `parsers` which will not block the entry of data but are able to interpret and modify the value provided by the user to conform to a specific format. An example of a provided parser is the date parser which will interpret and correct dates provided by users regardless of the format in which they are entered.

17.5.1 How-to

17.5.1.1 Creating metadata types



Permissions required:

The “Create new metadata types” permission is required for this action.

1. Go to the `System → Setup → Metadata types` menu.
2. From the `Actions` dropdown select `Create new`.

-
3. Provide a name to reference this metadata type in other parts of the system.
 4. Enter a label to be shown to users when using this metadata type.
 5. Optional: Enter a default value for the metadata type.
 6. Optional: Provide a comma separated list of options to restrict the data entry when using this metadata type.
 7. Optional: Select a validator and a parser to validate and clean up the data entry when not using a predetermined list of values.
 8. Press **Submit**.

17.5.1.2 Assigning a metadata type to a document type



Permissions required:

- The “Edit metadata types” permission is required for this action, globally, or via an ACL, for a metadata type.
 - Also the “Edit document type” permission is required, globally or via an ACL for a document type.
-

This action can be performed in two ways.

Option 1: Via the metadata type view

1. Go to the **System → Setup → Metadata types** menu.
2. Click on the button **Document types** of the metadata type you wish to associate.
3. From the list of existing document types press either:
 - **None** if this metadata type will not be available for the type of documents.
 - **Optional** if this metadata type will be available and is optional to provide a value for the type of documents.
 - **Required** if this metadata type will be available and is required to provide a value for the type of documents.
4. Press **Save**.

Option 2: Via the document type view

1. Go to the **System → Setup → Document types** menu.
2. Click on the button **Metadata types** of the metadata type you wish to associate.
3. From the list of existing metadata types press either:
 - **None** if this metadata type will not be available for the type of documents.
 - **Optional** if this metadata type will be available and is optional to provide a value for the type of documents.
 - **Required** if this metadata type will be available and is required to provide a value for the type of documents.
4. Press **Save**.

17.5.2 Takeaways

- Metadata are custom properties. They extend the amount of information about a document.
- Metadata types are first created and associated to a document type before they can be used with documents.
- Both types of data entry are supported: free style and constrained.
- Metadata can be defined as optional or required.

NOTES

neeraj76@yahoo.com

AUDITING

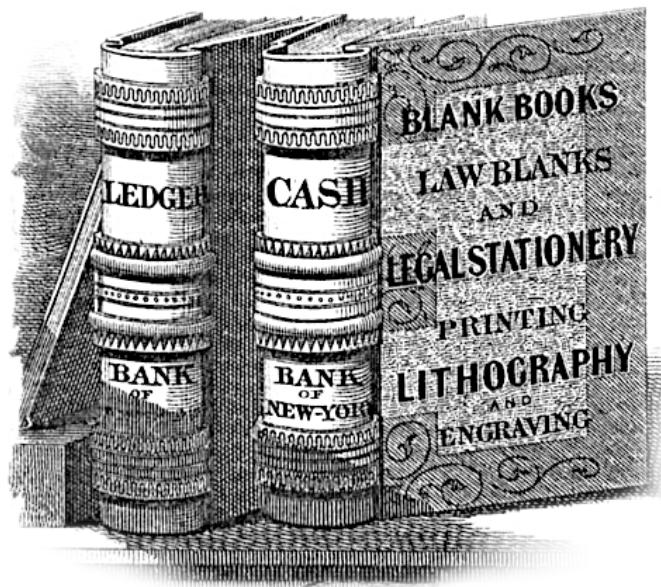


Fig. 1: Blank Ledger Books, 1855

Mayan EDMS records every action performed in the system. This is done by creating a journal and recording entries of the actions performed in the system. The actions are recorded regardless of the user that performed them, or if they were performed by the system itself as an automated function.

These journals are crucial to perform audits. Knowing who performed what action on what object is crucial when trying to determine the cause of an undesirable result.

In Mayan EDMS this journal of actions is called the “Events” system.

18.1 Events

The events system is implemented as a timestamped collection of event entries. Every event entry contains between 4 and 5 pieces of information:

- Date and time when the event happened.
- The user that committed the event, called the “Actor”.
- The type of action that was performed, called the “Event”.

-
- The object that is directly affected by the action, called the “Target”.
 - Any additional object used to perform the action on the target, called the “Action object”.

The event system is a centralized one, but provides many “views” to filter, or restrict, the list of events by a topic of interest. These can be:

- A list of events by a user.
- A list of events for an object.
- A list of events of a specific type.

18.1.1 How-to

18.1.1.1 Viewing the events of a document



Permissions required:

The “Access the events of an object” permission is required for this action, globally, or via ACL, for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on **Events** button.
3. A list of the events relating to the document will be displayed. The list will include events where the document was either the “Target” or the “Action object”

18.1.1.2 Viewing all the events in the system

1. Go to the **System → Tools → Events** menu.
2. A list of the events for the entire system will be displayed. Clicking on any of the actor or event entries, will filter the list and show only events relating to the actor or of the event type. Clicking on any target or action object will change the view and show the main detail view for the object.

18.1.1.3 Viewing all the events of the current user

1. Go to the **User → My events** menu.
2. A list of events relating to the currently logged on user will be shown.

18.1.2 Takeaways

- Events is the name of the auditing system in Mayan EDMS.
- User and system events are tracked.
- Target are objects for which an action was performed.
- Action objects are objects that were part of the action performed for a target.

NOTES

neeraj76@yahoo.com

NOTIFICATIONS

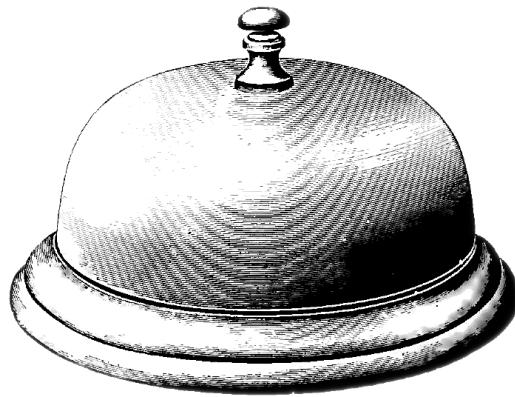


Fig. 1: 5-inch desk bell. Frank A. Weeks MFG Co. 1912

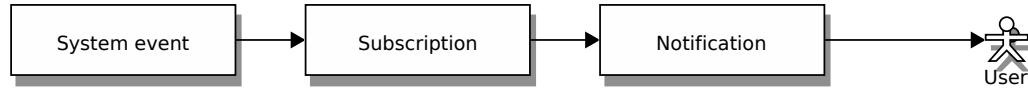


Fig. 2: In order to receive notifications, the user must first be subscribed to an event.



Fig. 3: It is also possible to subscribe to the events of specific objects.

Notifications are alerts shown in the user interface based on subscriptions to event. Besides subscribing to system events, it is also possible to subscribe to the events of specific objects.

19.1 How-to

19.1.1 Subscribing to system events

1. Go to the *User → Event subscriptions* menu.
2. Click on the **Subscribed** button of the event for which you wish to receive notifications. Multiple events can be selected.
3. Press **Save**.
4. Whenever an event of the type being subscribed occurs for this point forward a notification will be generated.



Note: While it is possible to subscribe to any system event, only the events of an object for which you have the “Access the events of an object” permission via an ACL will result in a notification entry.

19.1.2 Subscribing to an object's events



Permissions required:

The “Access the events of an object” permission is required for this action, globally or via ACL for the selected object.

1. Navigate to the selected documents preview view.
2. From the sidebar click the **Subscriptions** button.
3. Click on the **Subscribed** button of the event for which you wish to receive notifications. Multiple events can be selected.
4. Press **Save**.

19.2 Exercises

19.2.1 Subscribe to a document's view event

1. Navigate to the selected document's preview view.
2. From the sidebar click the **Subscriptions** button.
3. Click on the **Subscribed** button of the **Documents → Document viewed** event.
4. Press **Save**.
5. Navigate to the home screen by clicking on the **Mayan EDMS** button.
6. Navigate again to the selected document's preview view.
7. The notification icon (in the shape of a bell) next to the user menu will show an unread notification counter.
8. Click on the notifications icon.
9. A notification of the type “Documents: Document viewed” events for the selected document will be shown.

-
10. Click the button **Mark as seen** to clear the unread notification counter. To mark all the notification as “seen”, from the **Actions** dropdown, click the **Mark all as seen** button.
 11. Navigate again to the selected document’s preview view.
 12. From the sidebar click the **Subscriptions** button.
 13. Click on the **None** button of the **Documents** → **Document viewed** event.
 14. Press **Save**.

19.3 Takeaways

- Notifications allow you to remain informed on important events.
- It is possible to subscribe to system wide events, or events, for a specific object.

NOTES

PART V.

ADVANCED TOPICS

neeraj76@yahoo.com

VALIDATION

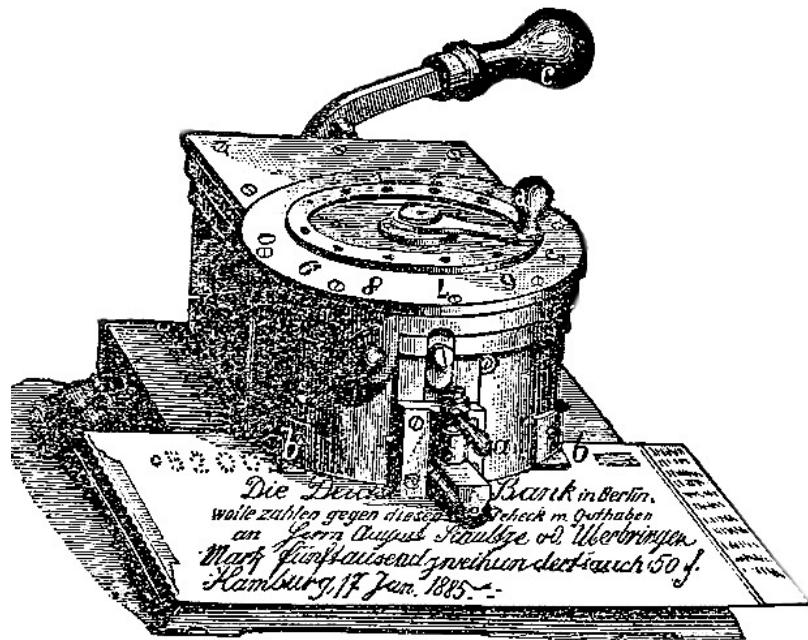


Fig. 1: 1889 Automatic Check Perforator

Mayan EDMS includes systems to help users validate several aspects of a document. These range from uniqueness to content integrity.

20.1 Uniqness

The first layer of integrity consists on assigning a universally unique identifier to each document in the system. For this Mayan EDMS uses version 4 of the Universally Unique Identifier (UUID) specification outlined in the Internet Engineering Task Force document RFC 4122.

It is mathematically impossible to create a unique number assignment system, where the same identifier is not assigned to two or more documents. However, with the use of UUID version 4, 2.71 quintillion IDs need to be generated (or documents uploaded, as each document is assigned a single UUID), in order to have a 50% chance of a single duplicate.

Example of a document UUID:

```
215799de-48e1-459b-b090-2c0fbd1d9073
```

20.1.1 How-to

20.1.1.1 Viewing the UUID of a document



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on the **Properties** button.
3. A form showing the different properties of the selected document will be shown. The UUID of the documents will be one of the items in the form.

20.1.2 Takeaways

- UUID are identifiers, generated with an algorithm that makes them very unlikely to ever repeat.
- Every document is assigned its own UUID.
- Document UUIDs cannot be changed or edited.

NOTES

20.2 Checksum

Operating system file integrity is supported by creating a fingerprint of the document when it is created. This is known as hashing, and will produce a number sequence that is unique based on the content of the document. This means that two documents with the exact same content will produce the same hash. This hash is stored in the database and can be used to verify that the content of the document has not been altered or damaged after it was uploaded. For this, Mayan EDMS uses the hashing algorithm known as SHA256.

Hashing works in such a way, that even changing a single byte, or character of the document will cause a mismatch between the stored hash and the new hash of the file. This will indicate that the document has been tampered with or has been damaged.

Example of a document checksum:

```
af52c3c87fc7ef03a773dfa654ef010bf2d82a24c2e4fd0ee3736a58daf622
```

20.2.1 How-to

20.2.1.1 Viewing the checksum of a document



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on the **Properties** button.
3. A form showing the different properties of the selected document will be shown. The checksum of the document’s latest version file will be one of the items in the form.

20.2.2 Takeaways

- Checksums are fingerprints generated from the documents content.
- If two documents were created with the exact same file, they will have the same checksum.
- Changing even a single letter will return a completely new checksum.
- Unlike the UUID, the checksum is dependent on the content of the file, therefore uploading a new version of a document will produce a new checksum.

NOTES

20.3 Digital signatures

File integrity allows for validation of the documents from the moment it is uploaded. In order to provide validation of the document, from even before it was uploaded, Mayan EDMS uses digital signatures. Digital signatures are hashes that have been encoded with the document's author private key. By using the corresponding public key, it is possible to validate that not only the document has not been altered since it was first created, but that the hash itself has not been updated either.

Just like the stored file hash, changing a single byte or character of the document will render the digital signature invalid.

Since the used to generate the digital signature is secret, it also validates the authorship of the document. If your organization uses a private key assignment policy, the authorship could be extended to provide also the document's author identity.

For these features, Mayan EDMS uses GNU privacy guard signatures. These are provided by the GNU privacy guard or GPG software. GPG is a free, open source alternative to the commercial PGP signature software.

Mayan EDMS supports embedded and detached signatures. Documents with embedded signatures are automatically decoded and usable.

20.3.1 How-to

20.3.1.1 Uploading a key



Permissions required:

The “Upload keys” permission is required for this action.

1. Export the GPG key you wish to upload into Mayan EDMS, in ASCII version, and into a file.
2. Go to the **Setup → Key management** menu.
3. From the sidebar click the **Private keys** or **Public keys** button.
4. From the **Actions** dropdown click the button named **Upload key**.
5. Copy of the content of the exported key into the field name

Key data.

1. Press **Submit**.

20.3.1.2 Querying a keyserver for a public key



Permissions required:

The “Query keyservers” permission is required for this action.



Settings:

-
- The keyserver to be queries is determined by the setting *SIGNATURES_KEYSERVER* (page 221).
-

1. Go to the *Setup → Key management* menu.
2. From the **Actions** dropdown click the button named **Query keyservers**.
3. Enter an e-mail, proper name, or key ID into the field named **Term**.
1. Press **Search**.

20.3.1.3 View the signatures of a document



Permissions required:

The “View details of document signatures” permission is required for this action, either globally or via an ACL for a document or document type.

1. Navigate to the select document’s preview view.
2. Click on the sidebar’s **Signatures** button.
3. A list of signatures for the document’s latest version will be shown. The list will include basic information like the signature’s date, key ID, and type. Further information about the signature can be accessed by clicking the **Details** button.

20.3.1.4 Signing a document (embedded)



Important: A private key must be previously uploaded in order to be able to sign documents.



Permissions required:

The “Sign documents with embedded signatures” permission is required for this action, either globally or via an ACL for a document or document type.

The “Use keys to sign content” permission is required for this action, either globally or via an ACL for the selected private key.

1. Navigate to the select document’s preview view.
2. Click on the sidebar’s **Signatures** button.
3. From the **Actions** dropdown select **Sign embedded**.
4. Select the key to use from the **Key** field.
5. In the **Passphrase** field, enter the key’s secret passphrase if it has one.
6. Click the **Submit** button.

-
7. A new document version will be created that has a signature embedded as part of the file content.

20.3.1.5 Signing a document (detached)



Important: A private key must be previously uploaded in order to be able to sign documents.



Permissions required:

The “Sign documents with detached signatures” permission is required for this action, either globally or via an ACL for a document or document type.

The “Use keys to sign content” permission is required for this action, either globally or via an ACL for the selected private key.

1. Navigate to the select document’s preview view.
2. Click on the sidebar’s **Signatures** button.
3. From the **Actions** dropdown select **Sign detached**.
4. Select the key to use from the **Key** field.
5. In the **Passphrase** field, enter the key’s secret passphrase if it has one.
6. Click the **Submit** button.
7. A new signature file will be created and uploaded. No new version of the document are created with this method.

20.3.2 Takeaways

- Keys must be generated by using GPG and then uploaded into Mayan EDMS.
- If the private key is uploaded, it can then be used to sign documents.
- If the public key is uploaded, it can only be used to validate signatures.

NOTES

neeraj76@yahoo.com

DEDUPLICATION



Fig. 1: Double bevel Eberhard Faber eraser. American Seating Company, 1952

Even under the most careful operation, mistakes can happen and the same documents may be uploaded several times. Duplication is not as a severe problems like say accidental deletion, but must still be addressed.

Some problems caused by duplication are:

- Multiple authoritative documents: It is hard when retrieving multiple copies of the same document to know which one is the copy with the most up-to-date information or properties.
- Wasted storage space: Each document uses storage space individually. For small duplicated documents the effect might not be evident, but duplicating large documents can quickly exhaust storage space.
- Wasted CPU processing time: Each document must be processed and introspected individually. This incurs in processing time. On heavily used systems with a large number of documents and users, duplicated documents might be keeping the processing queues full and blocking access to important documents.
- It is not necessary: Unlike traditional filing systems, in Mayan EDMS the same document might be part of different categorization units, all from a single uploaded file.

Mayan EDMS keeps track of duplicate documents automatically. User intervention is only needed to decide which of the duplicate copies should be removed.

The *Checksum* (page 148) system is used to track duplicates ensuring that only exact copies are tagged as duplicates.

21.1 How-to

21.1.1 Viewing the list of duplicated documents



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

-
1. Go to the **Documents → Duplicated documents** menu.

-
2. A list of the duplicate documents will be shown. The count of duplicate copies for each result will be shown underneath the document preview with the text **Duplicates**.

21.1.2 Manually trigger the duplicate scan



Note: This is normally performed automatically by the system. Use this function if a large number of documents were added by means outside the control of the system. For example a custom import app, or a direct copy of document files.



Permissions required:

The “Execute document modifying tools” permission is required for this action.

1. Go to the **Tools → Duplicated document scan** menu.
2. Click the button **Yes** from the confirmation window.
3. The duplication scan has been triggered and will be executed in the background. The list of duplicated documents will be updated periodically.

21.1.3 Viewing the duplicates of a document



Permissions required:

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

1. Navigate to the document’s preview view.
2. From the sidebar, click on the **Duplicates** button.
3. A list of exact duplicates of the documents selected will be shown.

21.2 Takeaways

- Duplicated documents waste space, processing time, and don’t provide any benefits.
- Mayan EDMS tracks duplicate documents automatically but doesn’t perform any action on them.

NOTES

neeraj76@yahoo.com

AUTOMATIC CATEGORIZATION

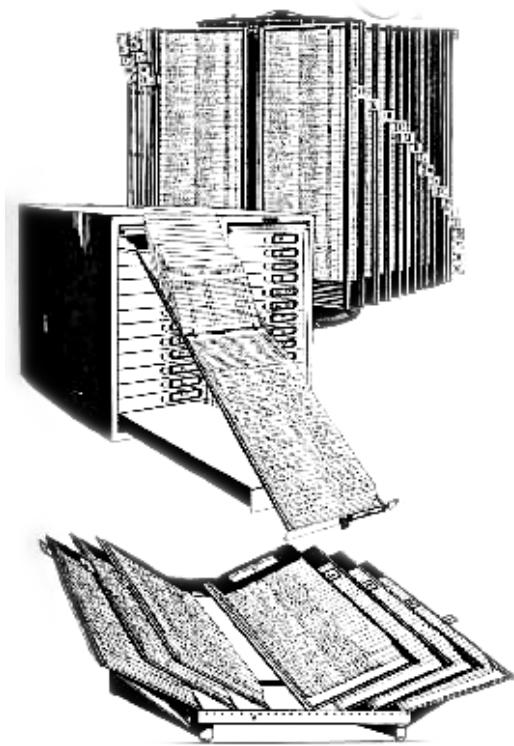


Fig. 1: Acme Visible Records, 1921 ad

Besides the manual categorization features presented in chapter *Categorization* (page 81), Mayan EDMS also provides automatic categorization methods that, once configured, will work on the background.

These methods update their structures as documents are added, modified, or deleted.

22.1 Indexing

Indexes are an automatic method to hierarchically organize documents. It is similar to cabinets in this regard. Indexes are updated automatically as the properties of the documents they have been set to organize change.

Any property of a document (metadata, label, MIME type) can be used to feed the index and perform the categorization.

Indexes work by first creating a template. This template is then automatically filled with the values of the document properties it specifies. The resulting values are used as the categorization units to store the document references in.

Indexes need to be associated with a document type.

Indexes are updated automatically as documents are uploaded or their properties updated. If an index's template is updated and is an index meant for documents that already exists in the system, a manual index rebuild must be triggered manually.



Fig. 2: Example index template to categorize invoices per year. The string `document.metadata_value_of.invoice_year` will be replaced with the value of the metadata `invoice_year` for each document.

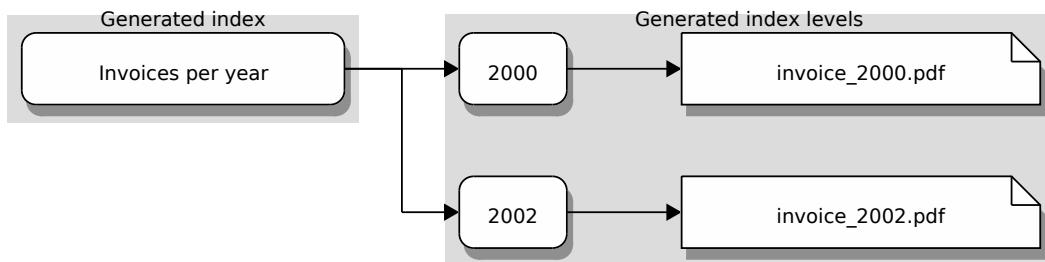


Fig. 3: Generated index from the template above for two invoice documents in the system called `invoice_2000.pdf` and `invoice_2002.pdf` with the years 2000 and 2002 as their `invoice_year` metadata values.

22.1.1 How-to

22.1.1.1 Creating index templates



Permissions required:

The “Create new document indexes” permission is required for this action.

-
1. Go to the **System** → **Setup** → **Indexes** menu.
 2. From the **Actions** dropdown select **Create index**.
 3. Enter a label to be shown to users in the system.
 4. Enter a slug to be used to reference this index by other parts of the system.

-
5. New index templates are enabled by default.
 6. Press **Submit**.

22.1.1.2 Editing the template code on an index template



Permissions required:

The “Edit document indexes” permission is required for this action, globally or via ACL for the selected index template.

1. Go to the **System → Setup → Indexes** menu.
2. Click the button **Tree template** of the selected index template.
3. Click the button **New child node** to create a new template index level.
4. Enter an indexing expression that will be replaced with the actual values of the document properties to generate an index. The available attributes that can be used for the indexing expression are presented below.
5. Click to select the checkbox **Link documents** to generate an index level that will have documents references. To generate nested index levels that will not contain documents unselect this checkbox.
6. Click **Submit**.

22.1.1.3 Associating an index template with a document type



Permissions required:

- The “Edit document indexes” permission is required for this action, globally or via an ACL for the selected index template.
 - Also the “Edit document type” permission is required, globally or via an ACL for the selected document type.
-

This action can be performed in two ways.

Option 1: Via the index template view

1. Go to the **System → Setup → Indexes** menu.
2. Click on the **Document types** button of the selected index template.
3. Select from the left panel the document types to be associated. Multiple document types can be selected by holding the **Control** or the **Shift** key.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to associate all the document types to the index template with a single action.
6. No further action is required after the last step, the changes are instantaneous.

Option 2: Via the document type view

1. Go to the **System → Setup → Document types** menu.
2. Click on the **Index templates** button of the selected document type.
3. Select from the left panel the index templates to be associated. Multiple index templates can be selected by holding the **Control** or the **Shift** key.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to associate all the index templates to the document type with a single action.
6. No further action is required after the last step, the changes are instantaneous.

22.1.1.4 Triggering an index rebuild



Permissions required:

The “Rebuild document indexes” permission is required for this action, globally or via an ACL for the index template to be rebuilt.

1. Go to the **System → Tools → Rebuild indexes** menu.
2. Select the indexes to rebuild from the list. Multiple indexes may be selected.
3. Click **Submit**.
4. The rebuild task will be scheduled and will be executed in the background.

22.1.2 Exercises

22.1.2.1 Create an index that categorizes documents by type



Permissions required:

- The “Create new document indexes” permission is required for this action.
- The “Edit document indexes” permission is required for this action, globally or via ACL for the index template.
- The “Edit document types” permission is required for this action, globally or via ACL for the document type.

1. Make sure at least two document types exists to associate with this index template.
2. Create an index template as explained in *Creating index templates* (page 160), with the label “Document types” and the name “document_types”.
3. Add the following template code, following the instructions in ref:*indexes-edit*:

```
{ { document.document_type } }
```
4. Associate the index template to the document types following the steps in *Associating an index template with a document type* (page 161).

-
5. Upload at least two new documents of the types associated to the index template. Or if the index template was associated with existing document types, trigger the index template rebuild as shown in *Triggering an index rebuild* (page 162)
 6. Examine the index instance from the main menu.

22.1.3 Takeaways

- Indexes are like cabinet but managed by the system.
- They work by means of templates that are replaced by document properties.
- Changing the template of an index already in use requires a complete rebuild.

NOTES

22.2 Mirroring



Fig. 4: Tower slide projector. Sears business equipment, 1963.

22.2.1 Mirroring

22.2.1.1 Introduction

Since indexes are hierarchical units that resemble filesystem folders, they can be exported via the mirroring app and will behave as a read-only filesystem structure. This functionality allows access to the generated index structure of documents without having to use the web graphical user interface.

Once indexes are mirrored, they behave like any other filesystem directory and can even be further shared via the network with network file system software like *Samba* or *NFS*.

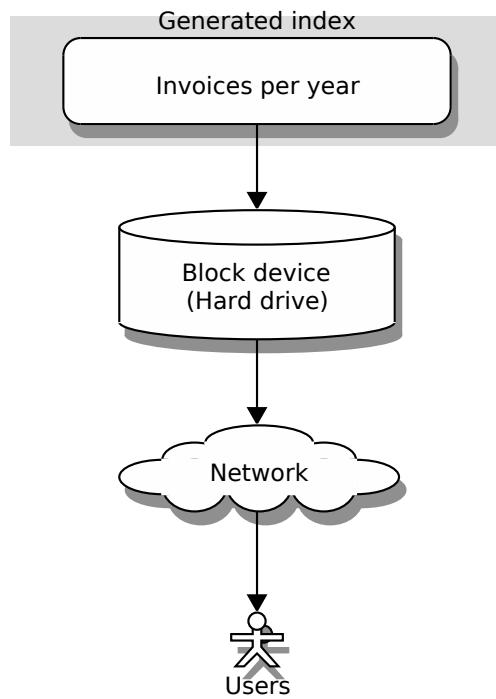


Fig. 5: Mirroring allows presenting generated indexes as filesystem folders. Mirrored indexes can also be shared on the network as normal shared folders.

22.2.2 How-to

22.2.2.1 Mirroring an index



Permissions required:

No permissions are required as this operation is performed in the command line and outside the control of Mayan EDMS's access control system.

1. Open a terminal window to enter the following commands.
2. Create a blank folder to be used to mirror the index:

```
mkdir ~/mirrored_index
```

3. Execute the mountindex management command:

```
mayan-edms.py mountindex <index slug to mirror> ~/mirrored_index
```

22.2.3 Takeaways

- Indexes can be exported as disk folders.
- The process of exporting the index is done in the command line.
- Exported indexes are read-only.
- Exported indexes can be shared over the network.

NOTES

REFERENCING

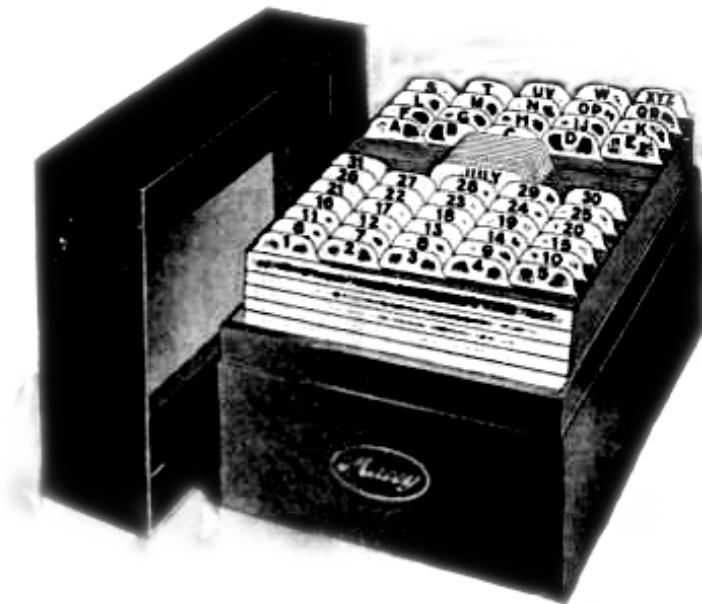


Fig. 1: Index Cards. Collier Stationery Co, 1909

Referencing allows linking two or more documents regardless of their type or current categorization. It is useful when the documents are related but must remain in their respective categorization structures.

23.1 Smart links

Smart links are an automatic referencing feature. They use rules which are evaluated against the properties of the documents to find matches. The matches are displayed in the document's sidebar.

Smart links need to be associated to document types before they can be used. Multiple smart links may be associated to a document type.

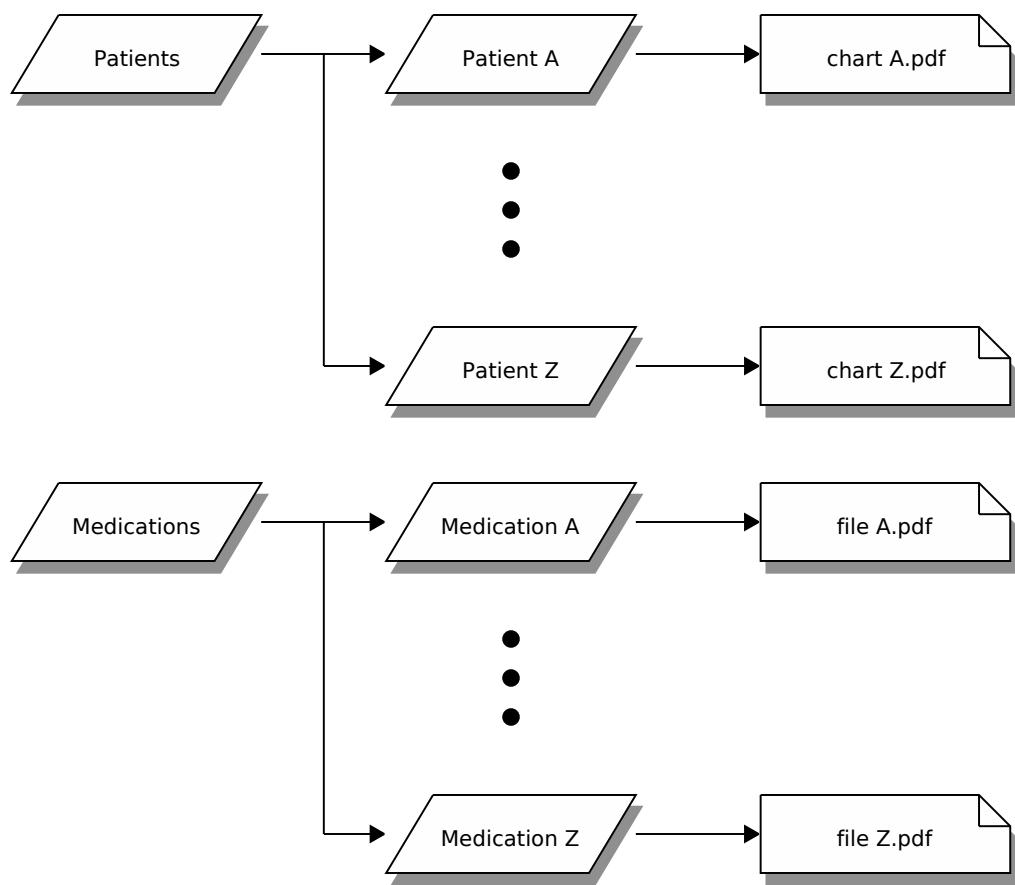


Fig. 2: Two different categorization structures. It would not make sense to merge them into a single one.

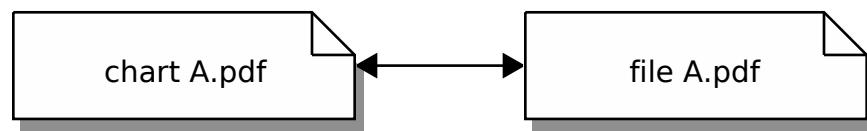


Fig. 3: A smart link makes it possible to establish a relationship between documents of the different categorization structures without changing the structures or the location of the documents.

23.1.1 How-to

23.1.1.1 Creating smart links



Permissions required:

The “Create new smart links” permission is required for this action.

1. Go to the **System → Setup → Smart links** menu.
2. From the **Actions** dropdown select **Create new smart link**.
3. Enter a label to be shown to users in the system.
4. Optional: Enter a dynamic label template. This template will be evaluated and its result shown as the smart link’s title. This allows the title of the smart link to change depending on the document that is accessing the smart link. The list of available document properties for the dynamic label template will be shown below the form field.
5. Use the **Enabled** checkbox to enable or disable the smart link. New smart links are enabled by default.
6. Press **Submit**.

23.1.1.2 Editing the conditions of a smart link



Permissions required:

The “Edit smart links” permission is required for this action, globally or via ACL for the selected smart link.

1. Go to the **System → Setup → Smart links** menu.
2. Click the button **Conditions** of the selected smart link.
3. Select the **Inclusion** of the condition. This field specifies the logic relationship between this condition and its previous condition. The inclusion for the first condition is always **And** regardless of the selection as there are no previous conditions for an inclusion relationship.
4. Select the **Foreign document field**. This is the property from the other documents against which the properties of the smart link’s document will be compared against.
5. Select the **Operator**. This is the operator for the comparison between the document’s properties.
6. Enter the **Expression**. This is a template expression that will generate the property for the document of the smart link. It is this resulting data that will be compared against the foreign document property.
7. Optional: Select the **Negated** checkbox to invert the logic of the comparison.
8. New smart link are enabled by default.
9. Use the **Enabled** checkbox to enable or disable the condition. New conditions are enabled by default.
10. Click the **Save** button.

23.1.1.3 Associating a smart link with a document type

**Permissions required:**

- The “Edit smart links” permission is required for this action, globally or via an ACL for the selected smart link.
 - Also the “Edit document type” permission is required, globally or via an ACL for the selected document type.
-

This action can be performed in two ways.

Option 1: Via the smart link view

1. Go to the *System* → *Setup* → *Smart links* menu.
2. Click on the button **Document types** of the selected smart link.
3. Select from the left panel the document types to be associated. Multiple document types can be selected by holding the Control or the Shift key.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to associate all the document types to the smart link with a single action.
6. No further action is required after the last step, the changes are instantaneous.

Option 2: Via the document type view

1. Go to the *System* → *Setup* → *Document types* menu.
2. Click on the button **Smart links** of the selected document type.
3. Select from the left panel the smart links to be associated. Multiple smart links can be selected by holding the Control or the Shift key.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to associate all the smart links to the document type with a single action.
6. No further action is required after the last step, the changes are instantaneous.

23.1.1.4 Viewing a document’s smart links

**Permissions required:**

The “View documents” permission is required for this action, globally or via ACL for the document or document type.

1. Navigate to the preview view of the selected document.
2. From the sidebar click on the **Smart links** button.
3. A list of smart links associated to the document’s type will be shown.
4. Click on the button **Documents** of a smart link to view the resulting document list.

23.1.2 Exercises

23.1.2.1 Create smart link to links documents by type



Permissions required:

- The “Create new smart links” permission is required for this action.
 - The “Edit smart links” permission is required for this action, globally or via ACL for the smart link.
 - The “Edit document types” permission is required for this action, globally or via ACL for the document type.
-

1. Make sure at least two document types exists to associate with this smart link.
2. Upload at least two documents for each type.
3. Create a smart link as explained in *Creating smart links* (page 171), with the label “By document types” and the dynamic label template:

```
Other documents of type "{{ document.document_type }}"
```

4. Add the following condition, following the instructions in [ref:smart-links-edit](#):

- Foreign document field: “Document type label”
- Operator: “is equal to”
- Expression:

```
{{ document.document_type }}
```

1. Associate the smart link to the document types following the steps in *Associating an index template with a document type* (page 161).
2. Examine the smart links of the uploaded documents as shown in [ref:smart-links-document-smart-list](#):

23.1.3 Takeaways

- Smart links allow creating relationships between documents without affecting their respective organization structures.
- Smart links are configured once, then they execute on their own to establish the document relationships.
- Smart links need to be associated to a document type.

NOTES

BUSINESS PROCESSES



Fig. 1: Belay Binder. Barret Bindery Company

A Business Processes Manager (or BPM) allows the electronic definition of structured collections of business activities and the states of such business activities. Simply put, it allows for the electronic representation of business logic and processes. This capability closes the gap between electronic systems and business needs. It also allows electronics system to conform to the business needs and not the other way around.

24.1 Workflows

Mayan EDMS includes support for BPM by means of a document workflow engine.

This workflow engine provides a structured method for storing a sequence of states over which the a document will progress. Workflows keep track of how a document has been processed so far and what other steps needs to be performed to fulfill the goal of a business process.

Workflows work by storing a series of states to help you know the “status” of a document. To move a workflow from one state to another, transitions are used.

Transitions connect two different states and help provide context to know which state is possible to transition to, from a previous state. Transitions provide an order for the sequence of possible states changes.

Transitions can be executed manually by users if they have the required access level as configured by the system administrator.

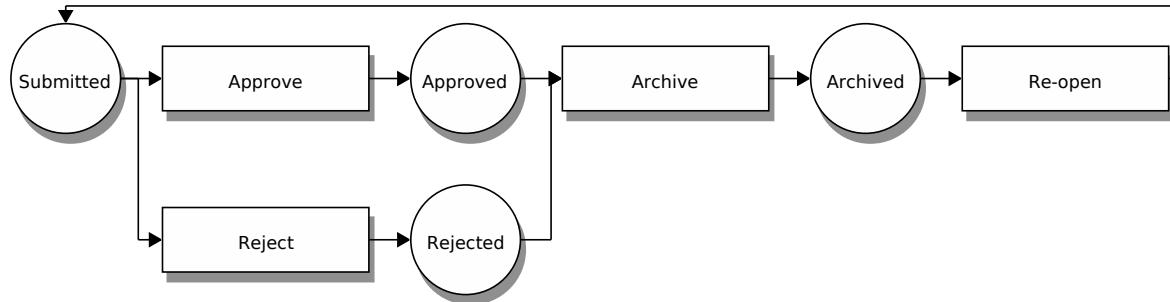


Fig. 2: Example workflow. Circles represent states, rectangles represent transitions.

After creation, workflow templates need to be associated to a document type before they are able to be used. Once associated they will execute automatically whenever a new document is uploaded.

For convenience, an entry in the main menu named **Workflows**, will show a list of active workflows and the documents currently executing them. The menu will also show a list of active workflow states and document lists for which the selected workflow state is the currently active state.

24.1.1 How-to

24.1.1.1 Creating a new workflow template



Permissions required:

The “Create workflows” permission is required for this action.

1. Go to the **System** → **Setup** → **Workflows** menu.
 2. From the **Actions** dropdown select **Create workflow**.
 3. Enter a **Label** to display to users and identify the workflow template.
 4. Enter an **Internal name** to allow other areas to reference this workflow template.
 5. Click on the **Submit** button.
-



Tip: For ease of administration, use the same value for the **Internal name** as you used for the

Label, but change it to lowercase and user an underscore instead of spaces.

24.1.1.2 Creating workflow template states



Permissions required:

The “Edit workflows” permission is required for this action, globally or via ACL for the selected workflow template.

1. Go to the **System → Setup → Workflows** menu.
2. Click on the button **States** of the selected workflow template.
3. From the **Actions** dropdown select **Create state**.
4. Optional: Make this state at the initial state of the workflow by selecting the **Initial** checkbox.
5. Enter a **Label** to display to users and identify this workflow template state.
6. Optional: Provide a number in the **Completion** to identify the level of completion for the workflow template that this state marks.
7. Click on the **Save** button.



Tip: Since a workflow state signify a property of a document, use adjectives for clarity. For example, use “Approved” instead of “Approve”.

24.1.1.3 Creating workflow template transitions



Permissions required:

The “Edit workflows” permission is required for this action, globally or via ACL for the selected workflow template.

1. Go to the **System → Setup → Workflows** menu.
2. Click on the button **Transitions** of the selected workflow template.
3. From the **Actions** dropdown select **Create transition**.
4. Enter a **Label** to display to users and identify this workflow template transitions.
5. Select the workflow template state that will cause this transition to be available from the **Origin state** list. Only states from this workflow template will be shown.
6. Select the workflow template state that will become the active state when this transition is executed from the **Destination state** list. Only states from this workflow template will be shown.
7. Click on the **Save** button.



Tip: Since a workflow transition is an action, use verbs for clarity. For example, use “Approve” instead of “Approved”.

24.1.1.4 Associating a workflow template with a document type



Permissions required:

- The “Edit workflows” permission is required for this action, globally or via an ACL for the selected workflow.
 - Also the “Edit document type” permission is required, globally or via an ACL for the selected document type.
-

This action can be performed in two ways.

Option 1: Via the workflow template view

1. Go to the *System* → *Setup* → *Workflows* menu.
2. Click on the button **Document types** of the selected workflow template.
3. Click the sidebar button named **Document types** to associate this workflow template to a document type.
4. Select from the left panel the document types to be associated. Multiple document types can be selected by holding the **Control** or the **Shift** key.
5. Click the **Add** button.
6. Optional: Click the **Add all** button to associate all the document types to the workflow template with a single action.
7. No further action is required after the last step, the changes are instantaneous.

Option 2: Via the document type view

1. Go to the *System* → *Setup* → *Document types* menu.
2. Click on the button **Workflows** of the selected document type.
3. Select from the left panel the workflow templates to be associated. Multiple workflow templates can be selected by holding the **Control** or the **Shift** key.
4. Click the **Add** button.
5. Optional: Click the **Add all** button to associate all the workflow templates to the document type with a single action.
6. No further action is required after the last step, the changes are instantaneous.

24.1.1.5 Viewing a workflow preview

**Permissions required:**

The “View workflows” permission is required for this action, globally or via ACL for the workflow template.

1. Go to the **System → Setup → Workflows** menu.
2. Click on the button named **Preview** of the selected workflow template.
3. A graphical representation of the workflow template will be shown. The graphical representation of a workflow is similar to a flowchart. The states are represented with circles. The transitions are represented with arrows. Circle with a double border represent the initial state of the workflow.

24.1.1.6 Transitioning a workflow instance

**Permissions required:**

The “Transition workflow” permission is required for this action, either globally or via an ACL for the selected workflow template.

1. Navigate to the selected document’s preview view.
2. From the **Actions** dropdown select **Workflows**. A list of active workflows will be shown.
3. Click the button **Transition** from the selected workflow instance.
4. Select the transition to execute from the **Transition** selection.
5. Optional: Enter a comment for the transition. The comment will be shown in the workflow instance detail view.
6. Click the **Submit** button.

24.1.2 Exercises

24.1.2.1 Create an approval workflow

**Permissions required:**

- The “Create workflows” permission is required for this action.
- The “Edit workflows” permission is required for this action, globally or via ACL for the workflow template.
- The “Edit document types” permission is required for this action, globally or via ACL for the document type.
- The “Transition workflows” permission is required for this action, globally or via ACL for the document or document type.

-
1. Make sure a document type exists to associate with this workflow template.

2. Create a workflow template as explained in *Creating a new workflow template* (page 176), with the label “Approval workflow” and the name “approval_workflow”.
3. Add the following four (4) states:
 - “Submitted”, completion: 0 (mark this as the initial state).
 - “Approved”, completion: 90.
 - “Rejected”, completion: 90.
 - “Archived”, completion: 100.

Use the steps provided in *Creating workflow template states* (page 177).

4. Add the following five (5) transitions:
 - “Approve”, states “Submitted” to “Approved”.
 - “Reject”, states “Submitted” to “Rejected”.
 - “Archive”, states “Approved” to “Archived”.
 - “Archive”, states “Rejected” to “Archived”.
 - “Re-open”, states “Archived” to “Submitted”.

Use the steps provided in *Creating workflow template transitions* (page 177).

5. Associate the workflow template to a document type following the steps in *Associating a workflow template with a document type* (page 178).
6. Obtain a preview of the workflow template as explained in *Viewing a workflow preview* (page 179). It should look similar to the following:

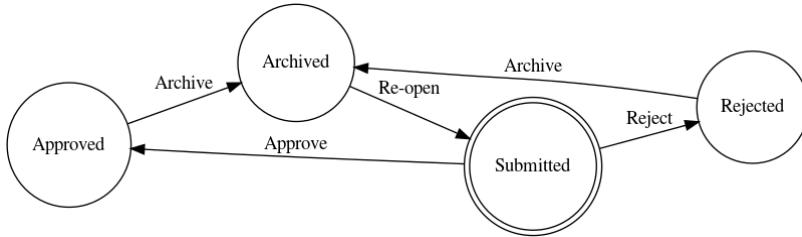


Fig. 3: Preview of an approval workflow.

7. Upload a new document of the type associated to the workflow template.
8. Transition the document’s workflow.

24.1.3 Takeaways

- Workflows allows for the execution of business logic.
- Workflows are composed of states and transitions.
- Workflows are defined as templates. Each template is then instantiated many times.
- Workflows templates are associated to a document type.
- States and transitions are not shared between workflow templates.

NOTES

neeraj76@yahoo.com

AUTOMATION



Fig. 1: Portable Tower Adder. Sears business equipment, 1963.

Workflows are mainly used to represent business processes. But they can also be used as an automation system to customize Mayan EDMS and have it perform some tasks automatically.

It is possible to have a workflow transition automatically as a response to a document's event.

Likewise, it is possible to have a workflow execute an action when a certain state is reached or left.

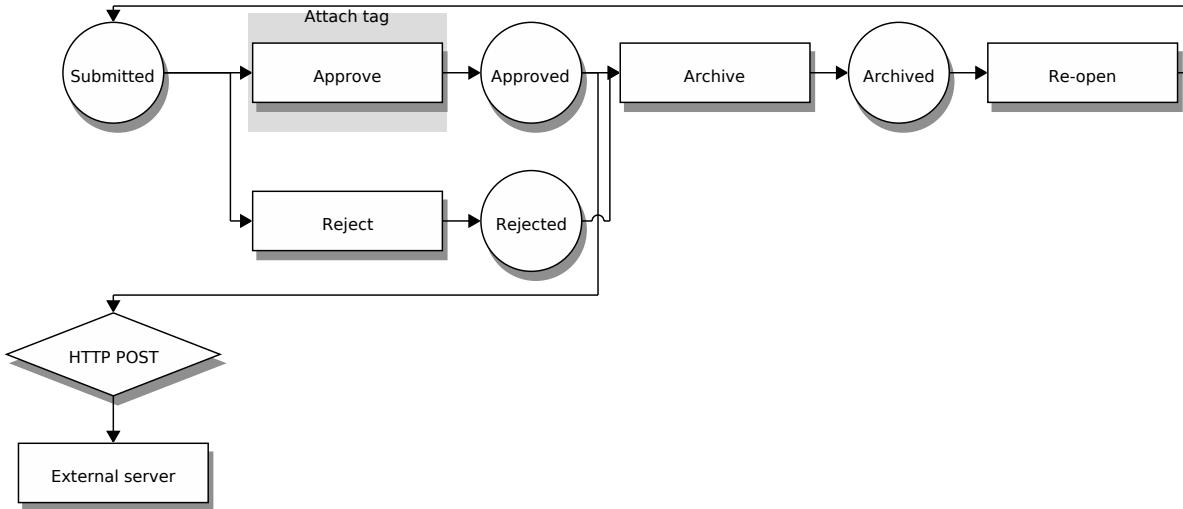


Fig. 2: Example workflow with a trigger for the “Approve” transition and an action for the “Approved” state.

25.1 Transition triggers

Besides being able to be executed manually by users, transitions can also be programmed to execute automatically based on system events. In Mayan EDMS, this is called transition triggering and is one of the tools available to automate business processes.

For example:

- Move a document from a “scanned” state to a “billed” state when a tag is attached to the document.
- Move a document from a “uploaded” state to a “OCR ready” state when the OCR engine finishes processing the document.

25.2 State actions

The other tool provided for process automation is being able to execute an action when a workflow state is reached or left. These are called state events.

Some examples of state actions currently provided are:

- Attach a tag to a document
- Remove a tag from a document
- Do an HTTP POST request to an external IP address
- Edit the label or the description of a document.

Over time more state actions are being added. Some state actions, like the one that creates an HTTP POST request, allow Mayan EDMS to trigger processes in external systems based on the state of a document. One example of this is triggering the billing process of an accounting system when an invoice is scanned and uploaded to Mayan EDMS.

Workflow state actions and transitions triggers are new features and are still evolving.

Workflows allow translating business logic into a series of states. With the addition of state actions and transition triggers, the workflows in Mayan EDMS are no longer just informative, but can be part of your actual business automation process.

25.3 How-to

25.3.1 Creating state actions



Permissions required:

The “Edit workflows” permission is required for this action, globally or via ACL for the selected workflow template.

1. Go to the **System** → **Setup** → **Workflows** menu.
2. Click on the button **States** of the selected workflow template.
3. Click on the button **Actions** of the selected workflow template state.
4. From the **Actions** dropdown select **Create action**.
5. Select the type of action to create from the **Action** form field.
6. Click on the **Save** button.
7. A form will be shown with fields to configure the action selected. The fields will vary depending on the action selected. Regardless of the type of action, there are three fields that will always be shown: **Label**, **When**, **Enabled**.
8. On the **Label** field, enter text to identify the action.
9. On the **When** field, select at which moment the action will be executed. This can be when the workflow enters the state or when the workflow exits the state.
10. New state actions are enabled by default. The **Enabled** checkbox may be used to disable this state without having to delete it.
11. Fill the rest for the fields specific to the type of action selected.
12. Press the **Save** button.

25.3.2 Creating transition triggers



Permissions required:

The “Edit workflows” permission is required for this action, globally or via ACL for the selected workflow template.

1. Go to the **System** → **Setup** → **Workflows** menu.
2. Click on the button **Transitions** of the selected workflow template.
3. Click on the button **Transition triggers** of the selected workflow template state.
4. On the form, select which events will cause the transition to be executed automatically. This is done by clicking on the **Yes** button of the event.

-
5. Press the **Save** button.

25.4 Exercises

25.4.1 Create a new document notifier workflow



Permissions required:

- The “Create workflows” permission is required for this action.
- The “Edit workflows” permission is required for this action, globally or via ACL for the workflow template.
- The “Edit document types” permission is required for this action, globally or via ACL for the document type.

1. Create a test HTTP server with the following code:

```
from http.server import HTTPServer, BaseHTTPRequestHandler
import json

class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        content_length = int(self.headers['Content-Length'])
        body = self.rfile.read(content_length)
        print(json.loads(body))
        self.send_response(200)
        self.end_headers()

httpd = HTTPServer(('localhost', 8080), SimpleHTTPRequestHandler)
httpd.serve_forever()
```

2. Execute the test HTTP server with:

```
python3 httpserver.py
```

3. Make sure a document type exists to associate with this workflow template.
4. Create a workflow template as explained in *Creating a new workflow template* (page 176), with the label “New document notifier” and the name “new_document_notifier”.
5. Add the following state:

- “Created”, completion: 100 (mark this as the initial state).

Use the steps provided in *Creating workflow template states* (page 177).

6. Add an action of the state as explained in *Creating state actions* (page 185). Use the action type “Perform a POST request”. Use “Send new document data” for the Label. Set the action to occur on the on entry. Use `http://127.0.0.1:8080` for the URL. For the payload use the following:

```
{"document_pk": "{{document.uuid}}", "document_type": "{{document.document_type}}"}  
  ↳label}}
```

7. No transition is needed for this workflow:

-
8. Associate the workflow template to a document type following the steps in *Associating a workflow template with a document type* (page 178).
 9. Obtain a preview of the workflow template as explained in *Viewing a workflow preview* (page 179). It should look similar to the following:

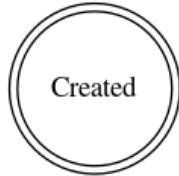


Fig. 3: Preview of the workflow.

10. Upload a new document of the type associated to the workflow template.
11. Observe the HTTP request on the test server when a new document is uploaded:

```
{ 'document_pk': '25e35b95-48dc-403e-b9d3-74aee965e101', 'document_type': 'Default' }  
127.0.0.1 - - [01/Jun/2019 21:10:36] "POST / HTTP/1.1" 200 -
```

25.4.2 Create a document rename workflow



Permissions required:

- The “Create workflows” permission is required for this action.
- The “Edit workflows” permission is required for this action, globally or via ACL for the workflow template.
- The “Create document types” permission is required for this action.
- The “Edit document types” permission is required for this action, globally or via ACL for the document type.
- The “Create metadata types” permission is required for this action.
- The “Edit metadata types” permission is required for this action, globally or via ACL for the metadata type type.

-
1. Create a metadata type with the label “Invoice number” and the name “invoice_number”. Follow the instruction in *Creating metadata types* (page 130).
 2. Create a document type labeled “Invoice” as explained in *Creating document types* (page 31).
 3. Associate the “Invoice number” metadata type as optional for the “Invoice” document type. Follow the steps in *Assigning a metadata type to a document type* (page 131).
 4. Create a workflow template as explained in *Creating a new workflow template* (page 176), with the label “Invoice rename” and the name “invoice_rename”.
 5. Add the following two (2) states:
 - “Created”, completion: 0 (mark this as the initial state).
 - “Renamed”, completion: 100.

Use the steps provided in *Creating workflow template states* (page 177).

6. Add an action of the “Renamed” state as explained in *Creating state actions* (page 185). Use the action type “Modify the properties of the document”. Use “Rename invoice label” for the label. Set the action to occur on entry. For the “Document label” template, use the following:

```
{% if document.metadata_value_of.invoice_number %} {{ document.document_type.
  -label }}-{{ document.metadata_value_of.invoice_number }} {%- else %} {{ document.
  -label }} {%- endif %}
```

7. Add the following two (2) transitions:

- Label: “Rename”, origin state: “Created”, destination state: “Renamed”.
- Label: “Reset workflow”, origin state: “Renamed”, destination state: “Created”.

Use the steps provided in *Creating workflow template transitions* (page 177).

8. Set the triggers of the “Rename” transition to be the events:

- Metadata, Document metadata added
- Metadata, Document metadata edited

Use the steps provided in *Creating transition triggers* (page 185).

9. Set the triggers of the “Reset workflow” transition to be the event:

- Documents, Document properties edited

Use the steps provided in *Creating transition triggers* (page 185).

10. Obtain a preview of the workflow template as explained in *Viewing a workflow preview* (page 179). It should look similar to the following:

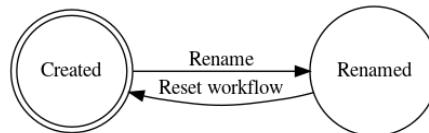


Fig. 4: Preview of the workflow.

11. Associate the workflow template to the “Invoice” document type following the steps in *Associating a workflow template with a document type* (page 178).

12. Execute the workflow using one of two options:

- Option 1: Upload a new document with the “Invoice” type, but do not set a value for the “Invoice number” metadata. The document label will remain unchanged. Add a metadata to the document, use type “Invoice number”, and set a value. The label of the document will be renamed to include the new metadata value.
- Option 2: Upload a new document with the “Invoice” type, and set a value for the “Invoice number” metadata. The label of the document will be renamed to include the invoice number as it finishes uploading.

25.5 Takeaways

- Automation extend workflows to also react to events and perform actions.
- With automation business processes can not only be represented but also automated.
- State actions may work on the document, the system, or even external systems.

NOTES

neeraj76@yahoo.com

PART VI.

OPERATIONS

neeraj76@yahoo.com

INSTALLATION

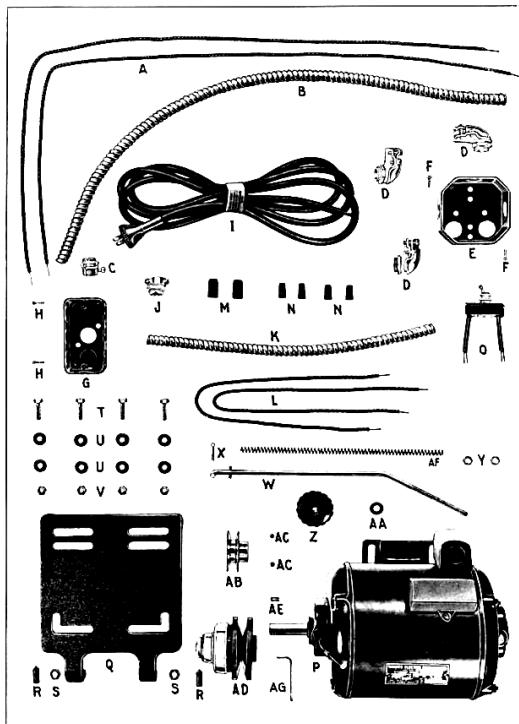


Fig. 1: Davidson dual duplicator instruction manual. Davidson Corporation, 1950.

Mayan EDMS can be installed in several ways. The two recommended ways are: by using *Docker*, and by doing a direct installation.

The Docker method provides the easiest installation process while the direct installation provides better performance and customization.

26.1 Minimum hardware requirements

- 1 Gigabytes of RAM (500 Megabytes if OCR is turned off).
- Multiple core CPU (64 bit, faster than 1 GHz recommended).
- Unix-like operating system like Linux and OpenBSD. For other operating systems use container technologies like Docker or virtual machines.

26.2 Using Docker

Docker can be installed using their automated script:

```
wget -qO- https://get.docker.com/ | sh
```

This installs the latest versions of Docker. If you don't want to run an automated script, follow the instructions outlined in their documentation: <https://docs.docker.com/install/>

Once the Docker installation is finished, proceed to install the Docker image for Mayan EDMS.

26.2.1 Start a Mayan EDMS Docker image

Download version 9.6 of the Docker PostgreSQL image:

```
docker pull postgres:9.6
```

Download the Mayan EDMS image using the command:

```
docker pull mayanedms/mayanedms:<version>
```



Note: Instead of a specific version tag you may use then generic :latest tag to the get latest version available automatically. If you use the :latest tag here, remember to do so in the next steps also.

```
docker pull mayanedms/mayanedms:latest
```

Create and run a PostgreSQL container:

```
docker run -d \
--name mayan-edms-postgres \
--restart=always \
-p 5432:5432 \
-e POSTGRES_USER=mayan \
-e POSTGRES_DB=mayan \
-e POSTGRES_PASSWORD=mayanuserpass \
-v /docker-volumes/mayan-edms/postgres:/var/lib/postgresql/data \
-d postgres:9.6
```

The PostgreSQL container will have one database named mayan, with a user named mayan too, with a password of mayanuserpass. The container will expose its internal 5432 port (PostgreSQL's default port) via the host's 5432 port. The data of this container will reside on the host's /docker-volumes/mayan-edms/postgres folder.

Finally create and run a Mayan EDMS container:

```
docker run -d \
--name mayan-edms \
--restart=always \
-p 80:8000 \
-e MAYAN_DATABASE_ENGINE=django.db.backends.postgresql \
-e MAYAN_DATABASE_HOST=172.17.0.1 \
-e MAYAN_DATABASE_NAME=mayan \
-e MAYAN_DATABASE_PASSWORD=mayanuserpass \
```

(continues on next page)

```
-e MAYAN_DATABASE_USER=mayan \
-v /docker-volumes/mayan-edms/media:/var/lib/mayan \
mayanedms/mayanedms:<version>
```

The Mayan EDMS container will connect to the PostgreSQL container via the 172.17.0.1 IP address (the Docker host's default IP address). It will connect using the `django.db.backends.postgresql` database driver and connect to the `mayan` database using the `mayan` user with the password `mayanuserpass`. The container will keep connections to the database for up to 60 seconds in an attempt to reuse them increasing response time and reducing memory usage. The files of the container will be stored in the host's `/docker-volumes/mayan-edms/media` folder. The container will expose its web service running on port 8000 on the host's port 80.

The container will be available by browsing to `http://localhost` or to the IP address of the computer running the container.

If another web server is running on port 80 use a different port in the `-p` option. For example: `-p 81:8000`.

26.2.2 Using a dedicated Docker network

Use this method to avoid having to expose PostgreSQL port to the host's network or if you have other PostgreSQL instances but still want to use the default port of 5432 for this installation.

Create the network:

```
docker network create mayan
```

Launch the PostgreSQL container with the network option and remove the port binding (`-p 5432:5432`):

```
docker run -d \
--name mayan-edms-postgres \
--network=mayan \
--restart=always \
-e POSTGRES_USER=mayan \
-e POSTGRES_DB=mayan \
-e POSTGRES_PASSWORD=mayanuserpass \
-v /docker-volumes/mayan-edms/postgres:/var/lib/postgresql/data \
-d postgres:9.6
```

Launch the Mayan EDMS container with the network option and change the database hostname to the PostgreSQL container name (`mayan-edms-postgres`) instead of the IP address of the Docker host (172.17.0.1):

```
docker run -d \
--name mayan-edms \
--network=mayan \
--restart=always \
-p 80:8000 \
-e MAYAN_DATABASE_ENGINE=django.db.backends.postgresql \
-e MAYAN_DATABASE_HOST=mayan-edms-postgres \
-e MAYAN_DATABASE_NAME=mayan \
-e MAYAN_DATABASE_PASSWORD=mayanuserpass \
-e MAYAN_DATABASE_USER=mayan \
-e MAYAN_DATABASE_CONN_MAX_AGE=0 \
-v /docker-volumes/mayan-edms/media:/var/lib/mayan \
mayanedms/mayanedms:<version>
```

26.3 Direct deployments

For users with knowledge of *Python*, *Django*, *Linux*, and databases.

Mayan EDMS should be deployed like any other Django project and preferably using *virtualenv*. Below are some ways to deploy and use Mayan EDMS. Do not use more than one method.

Being a Django and a Python project, familiarity with these technologies is recommended to better understand why Mayan EDMS does some of the things it does.

Compilers and development libraries will be installed to compile runtime libraries. LibreOffice and Poppler utils will also be installed as they are used to convert document files. *Supervisord* will be used to monitor and keep all Mayan processes running.

26.3.1 Basic deployment

This setup uses less memory and CPU resources at the expense of some speed. For another setup that offers more performance and scalability refer to the *Advanced deployment* (page 199) below.

26.3.1.1 1. Install binary dependencies:

If using a Debian, or Ubuntu based Linux distribution, get the executable requirements using:

```
sudo apt-get install g++ gcc ghostscript gnupg1 graphviz libfuse2 \
libjpeg-dev libmagic1 libpq-dev libpng-dev libreoffice libtiff-dev \
poppler-utils postgresql python-dev python-virtualenv redis-server \
sane-utils supervisor tesseract-ocr zlib1g-dev -y
```



Note: Platforms with the ARM CPU might also need additional requirements.

```
apt-sudo get libffi-dev libssl-dev -y
```

26.3.1.2 2. Create the user account for the installation:

This will create an unprivileged user account that is also unable to login.

```
sudo adduser mayan --disabled-password --disabled-login --no-create-home --
--gecos ""
```

26.3.1.3 3. Create the parent directory where the project will be deployed:

/opt/ is a good choice as it is meant is for “software and add-on packages that are not part of the default installation”. (<https://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/opt.html>)

```
sudo mkdir /opt
```

26.3.1.4 4. Create the Python virtual environment:

This will keep all the Python packages installed here isolated from the rest of the Python packages in the system.

```
sudo virtualenv /opt/mayan-edms
```

26.3.1.5 5. Make the mayan user the owner of the installation directory:

```
sudo chown mayan:mayan /opt/mayan-edms -R
```

26.3.1.6 6. Install Mayan EDMS from PyPI:

```
sudo -u mayan /opt/mayan-edms/bin/pip install --no-cache-dir --no-use-pep517  
↳mayan-edms
```

26.3.1.7 7. Install the Python client for PostgreSQL and Redis:

```
sudo -u mayan /opt/mayan-edms/bin/pip install --no-cache-dir --no-use-pep517  
↳psycopg2==2.7.3.2 redis==2.10.6
```



Note: Platforms with the ARM CPU might also need additional requirements.

```
sudo -u mayan /opt/mayan-edms/bin/pip install --no-cache-dir --no-use-pep517  
↳psutil==5.6.2
```

26.3.1.8 8. Create the database for the installation:

```
sudo -u postgres psql -c "CREATE USER mayan WITH password 'mayanuserpass';"  
sudo -u postgres createdb -O mayan mayan
```

26.3.1.9 9. Initialize the project:

This step will create all the database structures, download static media files like JavaScript libraries and HTML frameworks, and create an initial admin account with a random password.



Note: For simplicity, the MAYAN_MEDIA_ROOT folder is set to be a subfolder of the installation. If you want to keep your files separated from the installation files, change the value of the MAYAN_MEDIIR_ROOT variable in this and all subsequent steps. Be sure to first create the folder and give ownership of it to the mayan user with the chown command.

**DANGER
HIGH VOLTAGE**

If this step is interrupted, even if it is later resumed, in some cases, will cause the automatic admin user to no be created. Make sure all environment variable and values are correct. If this happens, refer to the troubleshooting chapters: *Missing automatic admin account after installation* (page 238) and *Admin password reset* (page 237).

```
sudo -u mayan MAYAN_DATABASE_ENGINE=django.db.backends.postgresql MAYAN_
˓→DATABASE_NAME=mayan \
MAYAN_DATABASE_PASSWORD=mayanuserpass MAYAN_DATABASE_USER=mayan \
MAYAN_DATABASE_HOST=127.0.0.1 MAYAN_MEDIA_ROOT=/opt/mayan-edms/media \
/opt/mayan-edms/bin/mayan-edms.py initialsetup
```

26.3.1.10 10. Collect the static files:

This step merges and compressed static media files so they can be served more effectively.

```
sudo -u mayan MAYAN_MEDIA_ROOT=/opt/mayan-edms/media \
/opt/mayan-edms/bin/mayan-edms.py preparestatic --noinput
```

26.3.1.11 11. Create the supervisor file at /etc/supervisor/conf.d/mayan.conf:

```
MAYAN_DATABASE_ENGINE=django.db.backends.postgresql MAYAN_DATABASE_
˓→NAME=mayan \
MAYAN_DATABASE_PASSWORD=mayanuserpass MAYAN_DATABASE_USER=mayan \
MAYAN_DATABASE_HOST=127.0.0.1 MAYAN_MEDIA_ROOT=/opt/mayan-edms/media \
/opt/mayan-edms/bin/mayan-edms.py platformtemplate supervisord > /etc/
˓→supervisor/conf.d/mayan.conf
```

26.3.1.12 12. Configure Redis:

Configure Redis to discard data when it runs out of memory, not save its database and only keep 1 database:

```
echo "maxmemory-policy allkeys-lru" >> /etc/redis/redis.conf
echo "save \"\"\" >> /etc/redis/redis.conf
echo "databases 1" >> /etc/redis/redis.conf
systemctl restart redis
```

26.3.1.13 13. Enable and restart the services⁷:

```
systemctl enable supervisor
systemctl restart supervisor
```

⁷ <https://bugs.launchpad.net/ubuntu/+source/supervisor/+bug/1594740>.

26.3.1.14 14. Cleaning up:

The following operating system dependencies are only needed during installation and can be removed.

```
apt-get remove --purge libjpeg-dev libpq-dev libpng-dev libtiff-dev zlib1g-dev
```

26.3.2 Advanced deployment

This variation uses RabbitMQ as the message broker. RabbitMQ consumes more memory but scales to thousands of messages per second. RabbitMQ messages are also persistent by default, this means that pending tasks are not lost in the case of a restart or power failure. The Gunicorn workers are increased to 3.

26.3.2.1 1. Install RabbitMQ:

If using a Debian or Ubuntu based Linux distribution, get the executable requirements using:

```
sudo apt-get install rabbitmq-server -y
```

26.3.2.2 2. Install the Python client for RabbitMQ:

```
sudo -u mayan /opt/mayan-edms/bin/pip install --no-cache-dir --no-use-pep517 librabbitmq==2.0.0
```

26.3.2.3 3. Create the RabbitMQ user and vhost:

```
sudo rabbitmqctl add_user mayan mayanrabbitmqpassword
sudo rabbitmqctl add_vhost mayan
sudo rabbitmqctl set_permissions -p mayan mayan ".*" ".*" ".*"
```

26.3.2.4 4. Edit the supervisor file at /etc/supervisor/conf.d/mayan.conf:

Replace (paying attention to the comma at the end):

```
MAYAN_BROKER_URL="redis://127.0.0.1:6379/0",
```

with:

```
MAYAN_BROKER_URL="amqp://mayan:mayanuserpass@localhost:5672/mayan",
```

increase the number of Gunicorn workers to 3 in the line (-w 2 section):

```
command = /opt/mayan-edms/bin/gunicorn -w 2 mayan.wsgi --max-requests 1000 --
max-requests-jitter 50 --worker-class gevent --bind 0.0.0.0:8000 --timeout=120
```

remove the concurrency limit (or increase it) of the fast worker (remove --concurrency=1).

26.3.2.5 5. Restart the services:

```
supervisorctl restart all
```

NOTES

neeraj76@yahoo.com

SETTINGS

In Mayan EDMS settings are grouped in to units called namespaces. The most common criteria is to group the settings based on the app that defined them. In most cases, the name of settings will contain as a prefix the name of the namespace.

The settings can be changed using several methods. These methods are:

- By a configuration file
- By environment variables
- By Python settings modules



Fig. 1: Users can choose to edit the `config.yml` directly.

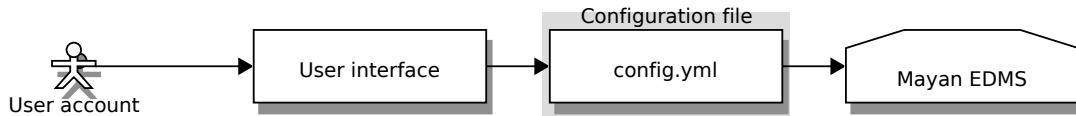


Fig. 2: The `config.yml` file is updated every time the settings are modified via the user interface.

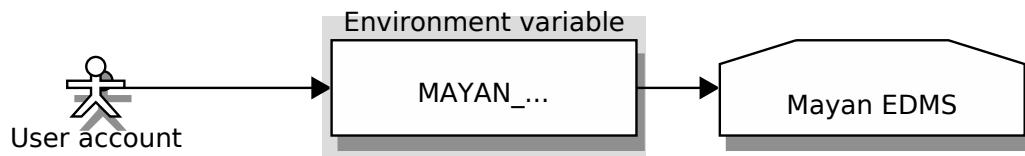


Fig. 3: When using environment variables, the variable must start with the prefix MAYAN_.

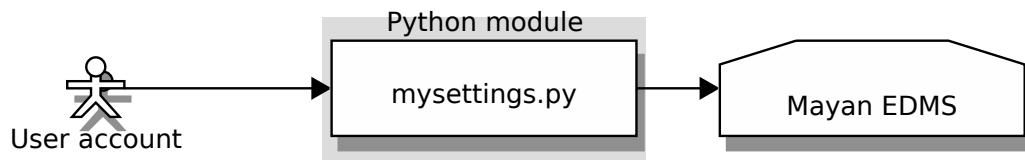


Fig. 4: Using a Python settings module provides the most flexibility, as normal Python code can be used to create a dynamic settings module.

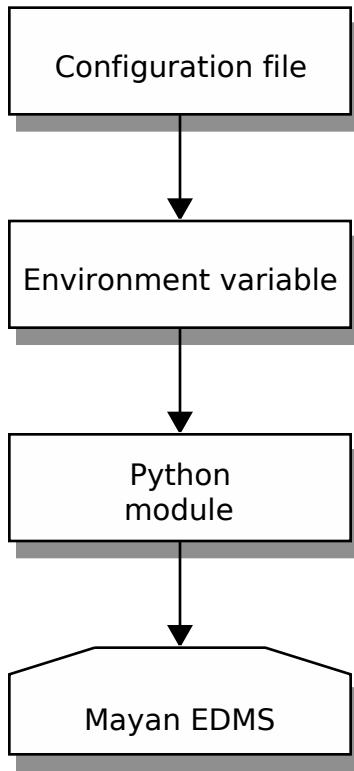


Fig. 5: Order of settings evaluation.

27.1 Configuration file

It is also possible to configure Mayan EDMS by means of a configuration file that used the YAML format. This file resides in the `media` folder and is named `config.yml`. The configuration file is read during startup and if the file format is found to be valid, a backup file is created with the name `config_backup.yml`. The purpose of the backup file is to allow users to revert the configuration. The configuration may be reverted because the user may wish to return to a previous configuration or because an invalid configuration change may have been introduced which stop Mayan EDMS from starting up.

27.2 Environment variables

When using environment variables for configuration, there is no need to manipulate configuration files. Because environment variables operate almost the same way regardless of operating system, platform, or application, they offer an advantage over other methods when Mayan EDMS is operated as part of a stack of software. With environment variables, the entire stack can be configured using a single configuration method.

27.3 Python settings module

The final configuration method, is via a Python settings module. This module must be created by the user and it must be a valid Python executable module with a .py extension. The module needs to reside in the `mayan_settings` folder, which itself resides in the `media` folder. The settings modules is not applied automatically, the user must instruct Mayan EDMS which module to import.

27.4 Setting namespaces

27.4.1 Appearance

- APPEARANCE_MAXIMUM_TITLE_LENGTH

Default:

120

Maximum number of characters that will be displayed as the view title.

27.4.2 Authentication

- AUTHENTICATION_LOGIN_METHOD

Default:

username

Controls the mechanism used to authenticated user. Options are: username, email

- AUTHENTICATION_MAXIMUM_SESSION_LENGTH

Default:

2592000

Maximum time an user clicking the “Remember me” checkbox will remain logged in. Value is time in seconds.

27.4.3 Auto administrator

- AUTOADMIN_EMAIL

Default:

autoadmin@example.com

Sets the email of the automatically created super user account.

- AUTOADMIN_PASSWORD

Default:

None

The password of the automatically created super user account. If it is equal to None, the password is randomly generated.

- AUTOADMIN_USERNAME

Default:

admin

The username of the automatically created super user account.

27.4.4 Celery

- BROKER_URL

Default:

Default: “amqp://”. Default broker URL. This must be a URL in the form of: transport://userid:password@hostname:port/virtual_host Only the scheme part (transport://) is required, the rest is optional, and defaults to the specific transports default values.

- CELERY_RESULT_BACKEND

Default:

Default: No result backend enabled by default. The backend used to store task results (tombstones). Refer to <http://docs.celeryproject.org/en/v4.1.0/userguide/configuration.html#result-backend>

27.4.5 Common

- COMMON_AUTO_LOGGING

Default:

Automatically enable logging to all apps.

- COMMON_DB_SYNC_TASK_DELAY

Default:

Time to delay background tasks that depend on a database commit to propagate.

- COMMON_HOME_VIEW

Default:

Name of the view attached to the brand anchor in the main menu. This is also the view to which users will be redirected after log in.

-
- COMMON_PAGINATE_BY

Default:

```
40
```

The number objects that will be displayed per page.

- COMMON_PRODUCTION_ERROR_LOGGING

Default:

```
False
```

Enable error logging outside of the system error logging capabilities.

- COMMON_PRODUCTION_ERROR_LOG_PATH

Default:

```
/home/rosarior/.virtualenvs/mayan-edms-book/lib/python3.7/site-packages/mayan/  
↳ media/error.log
```

Path to the logfile that will track errors during production.

- COMMON_PROJECT_TITLE

Default:

```
Mayan EDMS
```

Name to be displayed in the main menu.

- COMMON_PROJECT_URL

Default:

```
https://www.mayan-edms.com
```

URL of the installation or homepage of the project.

- COMMON_SHARED_STORAGE

Default:

```
django.core.files.storage.FileSystemStorage
```

A storage backend that all workers can use to share files.

- COMMON_SHARED_STORAGE_ARGUMENTS

Default:

```
{location: /home/rosarior/.virtualenvs/mayan-edms-book/lib/python3.7/site-  
↳ packages/mayan/media/shared_files}
```

None

27.4.6 Converter

- CONVERTER_GRAPHICS_BACKEND

Default:

```
mayan.apps.converter.backends.python.Python
```

Graphics conversion backend to use.

- CONVERTER_GRAPHICS_BACKEND_CONFIG

Default:

```
{  
    libreoffice_path: /usr/bin/libreoffice, pdftoppm_  
    dpi: 300, pdftoppm_format: jpeg, pdftoppm_path: /usr/bin/  
    pdftoppm, pdfinfo_path: /usr/bin/pdfinfo, pillow_format:  
    JPEG  
}
```

Configuration options for the graphics conversion backend.

27.4.7 Django

- ALLOWED_HOSTS

Default:

```
['127.0.0.1', 'localhost', '[::1]']
```

A list of strings representing the host/domain names that this site can serve. This is a security measure to prevent HTTP Host header attacks, which are possible even under many seemingly-safe web server configurations. Values in this list can be fully qualified names (e.g. ‘www.example.com’), in which case they will be matched against the request’s Host header exactly (case-insensitive, not including port). A value beginning with a period can be used as a subdomain wildcard: ‘.example.com’ will match example.com, www.example.com, and any other subdomain of example.com. A value of ‘*’ will match anything; in this case you are responsible to provide your own validation of the Host header (perhaps in a middleware; if so this middleware must be listed first in MIDDLEWARE).

- APPEND_SLASH

Default:

```
True
```

When set to True, if the request URL does not match any of the patterns in the URLconf and it doesn’t end in a slash, an HTTP redirect is issued to the same URL with a slash appended. Note that the redirect may cause any data submitted in a POST request to be lost. The APPEND_SLASH setting is only used if CommonMiddleware is installed (see Middleware). See also PREPEND_WWW.

- AUTH_PASSWORD_VALIDATORS

Default:

```
[{'NAME': 'django.contrib.auth.password_validation.  
UserAttributeSimilarityValidator'}, {'NAME': 'django.contrib.auth.password_  
validation.MinimumLengthValidator'}, {'NAME': 'django.contrib.auth.password_  
validation.CommonPasswordValidator'}, {'NAME': 'django.contrib.auth.password_  
validation.NumericPasswordValidator'}]
```

The list of validators that are used to check the strength of user’s passwords.

- DATABASES

Default:

```
{'default': {'ENGINE': 'django.db.backends.sqlite3', 'NAME': '/home/rosario/.  
˓→virtualenvs/mayan-edms-book/lib/python3.7/site-packages/mayan/media/db.sqlite3',  
˓→ 'ATOMIC_REQUESTS': False, 'AUTOCOMMIT': True, 'CONN_MAX_AGE': 0, 'OPTIONS': {}  
˓→, 'TIME_ZONE': None, 'USER': '', 'PASSWORD': '', 'HOST': '', 'PORT': '', 'TEST  
˓→': {'CHARSET': None, 'COLLATION': None, 'NAME': None, 'MIRROR': None}}}
```

A dictionary containing the settings for all databases to be used with Django. It is a nested dictionary whose contents map a database alias to a dictionary containing the options for an individual database. The DATABASES setting must configure a default database; any number of additional databases may also be specified.

- DATA_UPLOAD_MAX_MEMORY_SIZE

Default:

```
2621440
```

Default: 2621440 (i.e. 2.5 MB). The maximum size in bytes that a request body may be before a SuspiciousOperation (RequestDataTooBig) is raised. The check is done when accessing request.body or request.POST and is calculated against the total request size excluding any file upload data. You can set this to None to disable the check. Applications that are expected to receive unusually large form posts should tune this setting. The amount of request data is correlated to the amount of memory needed to process the request and populate the GET and POST dictionaries. Large requests could be used as a denial-of-service attack vector if left unchecked. Since web servers don't typically perform deep request inspection, it's not possible to perform a similar check at that level. See also FILE_UPLOAD_MAX_MEMORY_SIZE.

- DEFAULT_FROM_EMAIL

Default:

```
webmaster@localhost
```

Default: ‘webmaster@localhost’ Default email address to use for various automated correspondence from the site manager(s). This doesn't include error messages sent to ADMINS and MANAGERS; for that, see SERVER_EMAIL.

- DISALLOWED_USER_AGENTS

Default:

```
[]
```

Default: [] (Empty list). List of compiled regular expression objects representing User-Agent strings that are not allowed to visit any page, systemwide. Use this for bad robots/crawlers. This is only used if CommonMiddleware is installed (see Middleware).

- EMAIL_BACKEND

Default:

```
django.core.mail.backends.smtp.EmailBackend
```

Default: ‘django.core.mail.backends.smtp.EmailBackend’. The backend to use for sending emails.

- EMAIL_HOST

Default:

localhost

Default: ‘localhost’. The host to use for sending email.

- EMAIL_HOST_PASSWORD

Default:

--

Default: “” (Empty string). Password to use for the SMTP server defined in EMAIL_HOST. This setting is used in conjunction with EMAIL_HOST_USER when authenticating to the SMTP server. If either of these settings is empty, Django won’t attempt authentication.

- EMAIL_HOST_USER

Default:

--

Default: “” (Empty string). Username to use for the SMTP server defined in EMAIL_HOST. If empty, Django won’t attempt authentication.

- EMAIL_PORT

Default:

25

Default: 25. Port to use for the SMTP server defined in EMAIL_HOST.

- EMAIL_TIMEOUT

Default:

None

Default: None. Specifies a timeout in seconds for blocking operations like the connection attempt.

- EMAIL_USE_SSL

Default:

False

Default: False. Whether to use an implicit TLS (secure) connection when talking to the SMTP server. In most email documentation this type of TLS connection is referred to as SSL. It is generally used on port 465. If you are experiencing problems, see the explicit TLS setting EMAIL_USE_TLS. Note that EMAIL_USE_TLS/EMAIL_USE_SSL are mutually exclusive, so only set one of those settings to True.

- EMAIL_USE_TLS

Default:

False

Default: False. Whether to use a TLS (secure) connection when talking to the SMTP server. This is used for explicit TLS connections, generally on port 587. If you are experiencing hanging connections, see the implicit TLS setting EMAIL_USE_SSL.

- FILE_UPLOAD_MAX_MEMORY_SIZE

Default:

```
2621440
```

Default: 2621440 (i.e. 2.5 MB). The maximum size (in bytes) that an upload will be before it gets streamed to the file system. See Managing files for details. See also DATA_UPLOAD_MAX_MEMORY_SIZE.

- INSTALLED_APPS

Default:

```
('mayan.apps.appearance', 'django.contrib.admin', 'django.contrib.admindocs',
 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.messages',
 'django.contrib.sessions', 'django.contrib.sites', 'whitenoise.runserver_nostatic',
 'django.contrib.staticfiles', 'actstream', 'colorful', 'corsheaders',
 'djcelery', 'formtools', 'mathfilters', 'mptt', 'pure_pagination', 'rest_framework',
 'rest_framework.authtoken', 'solo', 'stronghold', 'widget_tweaks',
 'mayan.apps.acls', 'mayan.apps.authentication', 'mayan.apps.autoadmin', 'mayan.apps.common',
 'mayan.apps.converter', 'mayan.apps.dashboards', 'mayan.apps.dependencies',
 'mayan.apps.django_gpg', 'mayan.apps.dynamic_search', 'mayan.apps.events',
 'mayan.apps.lock_manager', 'mayan.apps.mimetype', 'mayan.apps.navigation',
 'mayan.apps.permissions', 'mayan.apps.platform', 'mayan.apps.rest_api',
 'mayan.apps.smart_settings', 'mayan.apps.task_manager', 'mayan.apps.user_management',
 'mayan.apps.motd', 'mayan.apps.cabinets', 'mayan.apps.checkouts',
 'mayan.apps.document_comments', 'mayan.apps.document_indexing', 'mayan.apps.document_parsing',
 'mayan.apps.document_signatures', 'mayan.apps.document_states',
 'mayan.apps.documents', 'mayan.apps.file_metadata', 'mayan.apps.linked',
 'mayan.apps.mailer', 'mayan.apps.mayan_statistics', 'mayan.apps.metadata',
 'mayan.apps.mirroring', 'mayan.apps.ocr', 'mayan.apps.sources', 'mayan.apps.storage',
 'mayan.apps.tags', 'drf_yasg')
```

A list of strings designating all applications that are enabled in this Django installation. Each string should be a dotted Python path to: an application configuration class (preferred), or a package containing an application.

- INTERNAL_IPS

Default:

```
('127.0.0.1',)
```

A list of IP addresses, as strings, that: Allow the debug() context processor to add some variables to the template context. Can use the admindocs bookmarklets even if not logged in as a staff user. Are marked as “internal” (as opposed to “EXTERNAL”) in AdminEmailHandler emails.

- LANGUAGES

Default:

```
((('ar', 'Arabic'), ('bg', 'Bulgarian'), ('bs', 'Bosnian'), ('cs', 'Czech'), ('da',
'Danish'), ('de', 'German'), ('el', 'Greek'), ('en', 'English'), ('es',
'Spanish'), ('fa', 'Persian'), ('fr', 'French'), ('hu', 'Hungarian'), ('id',
'Indonesian'), ('it', 'Italian'), ('lv', 'Latvian'), ('nl', 'Dutch'), ('pl',
'Polish'), ('pt', 'Portuguese'), ('pt-br', 'Portuguese (Brazil)'), ('ro',
'Romanian'), ('ru', 'Russian'), ('sl', 'Slovenian'), ('tr', 'Turkish'), ('vi',
'Vietnamese'), ('zh', 'Chinese')))
```

A list of all available languages. The list is a list of two-tuples in the format (language code, language name) for example, ('ja', 'Japanese'). This specifies which languages are available for language selection. Generally,

the default value should suffice. Only set this setting if you want to restrict language selection to a subset of the Django-provided languages.

- LANGUAGE_CODE

Default:

```
en-us
```

A string representing the language code for this installation. This should be in standard language ID format. For example, U.S. English is “en-us”. It serves two purposes: If the locale middleware isn’t in use, it decides which translation is served to all users. If the locale middleware is active, it provides a fallback language in case the user’s preferred language can’t be determined or is not supported by the website. It also provides the fallback translation when a translation for a given literal doesn’t exist for the user’s preferred language.

- LOGIN_REDIRECT_URL

Default:

```
common:root
```

Default: ‘/accounts/profile/’ The URL where requests are redirected after login when the contrib.auth.login view gets no next parameter. This is used by the login_required() decorator, for example. This setting also accepts named URL patterns which can be used to reduce configuration duplication since you don’t have to define the URL in two places (settings and URLconf).

- LOGIN_URL

Default:

```
authentication:login_view
```

Default: ‘/accounts/login/’ The URL where requests are redirected for login, especially when using the login_required() decorator. This setting also accepts named URL patterns which can be used to reduce configuration duplication since you don’t have to define the URL in two places (settings and URLconf).

- LOGOUT_REDIRECT_URL

Default:

```
authentication:login_view
```

Default: None. The URL where requests are redirected after a user logs out using LogoutView (if the view doesn’t get a next_page argument). If None, no redirect will be performed and the logout view will be rendered. This setting also accepts named URL patterns which can be used to reduce configuration duplication since you don’t have to define the URL in two places (settings and URLconf).

- STATICFILES_STORAGE

Default:

```
whitenoise.storage.CompressedManifestStaticFilesStorage
```

The file storage engine to use when collecting static files with the collectstatic management command. A ready-to-use instance of the storage backend defined in this setting can be found at django.contrib.staticfiles.storage.staticfiles_storage.

- STATIC_URL

Default:

```
/static/
```

URL to use when referring to static files located in STATIC_ROOT. Example: “/static/” or “http://static.example.com/” If not None, this will be used as the base path for asset definitions (the Media class) and the staticfiles app. It must end in a slash if set to a non-empty value.

- TIME_ZONE

Default:

```
UTC
```

A string representing the time zone for this installation. Note that this isn’t necessarily the time zone of the server. For example, one server may serve multiple Django-powered sites, each with a separate time zone setting.

- WSGI_APPLICATION

Default:

```
mayan.wsgi.application
```

The full Python path of the WSGI application object that Django’s built-in servers (e.g. runserver) will use. The django-admin startproject management command will create a simple `wsgi.py` file with an application callable in it, and point this setting to that application.

27.4.8 Document parsing

- DOCUMENT_PARSING_AUTO_PARSING

Default:

```
True
```

Set new document types to perform parsing automatically by default.

- DOCUMENT_PARSING_PDFTOTEXT_PATH

Default:

```
/usr/bin/pdftotext
```

File path to poppler’s pdftotext program used to extract text from PDF files.

27.4.9 Document signatures

- SIGNATURES_STORAGE_BACKEND

Default:

```
django.core.files.storage.FileSystemStorage
```

Path to the Storage subclass to use when storing detached signatures.

- SIGNATURES_STORAGE_BACKEND_ARGUMENTS

Default:

```
{location: /home/rosarior/.virtualenvs/mayan-edms-book/lib/python3.7/site-  
↳ packages/mayan/media/document_signatures}
```

Arguments to pass to the SIGNATURE_STORAGE_BACKEND.

27.4.10 Documents

- DOCUMENTS_CACHE_STORAGE_BACKEND

Default:

```
django.core.files.storage.FileSystemStorage
```

Path to the Storage subclass to use when storing the cached document image files.

- DOCUMENTS_CACHE_STORAGE_BACKEND_ARGUMENTS

Default:

```
{location: /home/rosarior/.virtualenvs/mayan-edms-book/lib/python3.7/site-  
↳ packages/mayan/media/document_cache}
```

Arguments to pass to the DOCUMENT_CACHE_STORAGE_BACKEND.

- DOCUMENTS_DISABLE_BASE_IMAGE_CACHE

Default:

```
False
```

Disables the first cache tier which stores high resolution, non transformed versions of documents's pages.

- DOCUMENTS_DISABLE_TRANSFORMED_IMAGE_CACHE

Default:

```
False
```

Disables the second cache tier which stores medium to low resolution, transformed (rotated, zoomed, etc) versions of documents' pages.

- DOCUMENTS_DISPLAY_HEIGHT

Default:

None

- DOCUMENTS_DISPLAY_WIDTH

Default:

```
3600
```

None

- DOCUMENTS_FAVORITE_COUNT

Default:

400

Maximum number of favorite documents to remember per user.

- DOCUMENTS_FIX_ORIENTATION

Default:

False

Detect the orientation of each of the document's pages and create a corresponding rotation transformation to display it rightside up. This is an experimental feature and it is disabled by default.

- DOCUMENTS_HASH_BLOCK_SIZE

Default:

65535

Size of blocks to use when calculating the document file's checksum. A value of 0 disables the block calculation and the entire file will be loaded into memory.

- DOCUMENTS_LANGUAGE

Default:

eng

Default documents language (in ISO639-3 format).

- DOCUMENTS_LANGUAGE_CODES

Default:

```
('ilo', 'run', 'uig', 'hin', 'pan', 'pnb', 'wuu', 'msa', 'kxd', 'ind', 'zsm',
↳ 'jax', 'meo', 'kvr', 'xmm', 'min', 'mui', 'zmi', 'max', 'mfa', 'cji', 'nan',
↳ 'pus', 'pbu', 'pbt', 'wne', 'hsn', 'hak', 'ful', 'fuc', 'fuf', 'ffm', 'fue',
↳ 'fuh', 'fuq', 'fuv', 'fub', 'fui', 'nep', 'npi', 'dty', 'sin', 'khm', 'kxm',
↳ 'ell', 'grc', 'cpq', 'gmy', 'pnt', 'tsd', 'yej', 'nya', 'mnp', 'dhd', 'cdo',
↳ 'hil', 'bcc', 'bgn', 'bgp', 'cmn', 'kok', 'spa', 'eng', 'ara', 'por', 'ben',
↳ 'rus', 'jpn', 'deu', 'jav', 'tel', 'vie', 'kor', 'fra', 'mar', 'tam', 'ura',
↳ 'tur', 'ita', 'yue', 'tha', 'guz', 'fas', 'pol', 'kan', 'mal', 'sun', 'hau',
↳ 'ory', 'mya', 'ukr', 'bho', 'tgl', 'yor', 'mai', 'uzb', 'snd', 'amh', 'ron',
↳ 'orm', 'ibo', 'aze', 'awa', 'gan', 'ceb', 'nld', 'kur', 'hbs', 'mlg', 'skr',
↳ 'ctg', 'zha', 'tuk', 'asm', 'mad', 'som', 'mwr', 'mag', 'bgc', 'hun', 'hne',
↳ 'dcc', 'aka', 'kaz', 'syl', 'zul', 'ces', 'kin', 'hat', 'que', 'swe', 'hmn',
↳ 'sna', 'mos', 'xho', 'bel')
```

List of supported document languages. In ISO639-3 format.

- DOCUMENTS_PAGE_IMAGE_CACHE_TIME

Default:

31556926

Time in seconds that the browser should cache the supplied document images. The default of 31559626 seconds corresponds to 1 year.

- DOCUMENTS_PREVIEW_HEIGHT

Default:

None

- DOCUMENTS_PREVIEW_WIDTH

Default:

None

- DOCUMENTS_PRINT_HEIGHT

Default:

None

- DOCUMENTS_PRINT_WIDTH

Default:

None

- DOCUMENTS_RECENT_ACCESS_COUNT

Default:

Maximum number of recently accessed (created, edited, viewed) documents to remember per user.

- DOCUMENTS_RECENT_ADDED_COUNT

Default:

Maximum number of recently created documents to show.

- DOCUMENTS_ROTATION_STEP

Default:

Amount in degrees to rotate a document page per user interaction.

- DOCUMENTS_STORAGE_BACKEND

Default:

Path to the Storage subclass to use when storing document files.

- DOCUMENTS_STORAGE_BACKEND_ARGUMENTS

Default:

```
{location: /home/rosarior/.virtualenvs/mayan-edms-book/lib/python3.7/site-  
↳ packages/mayan/media/document_storage}
```

Arguments to pass to the DOCUMENT_STORAGE_BACKEND.

- DOCUMENTS_THUMBNAIL_HEIGHT

Default:

```
800
```

Height in pixels of the document thumbnail image.

- DOCUMENTS_THUMBNAIL_WIDTH

Default:

```
800
```

Width in pixels of the document thumbnail image.

- DOCUMENTS_ZOOM_MAX_LEVEL

Default:

```
300
```

Maximum amount in percent (%) to allow user to zoom in a document page interactively.

- DOCUMENTS_ZOOM_MIN_LEVEL

Default:

```
25
```

Minimum amount in percent (%) to allow user to zoom out a document page interactively.

- DOCUMENTS_ZOOM_PERCENT_STEP

Default:

```
25
```

Amount in percent zoom in or out a document page per user interaction.

27.4.11 File metadata

- FILE_METADATA_AUTO_PROCESS

Default:

```
True
```

Set new document types to perform file metadata processing automatically by default.

- FILE_METADATA_DRIVERS_ARGUMENTS

Default:

```
{  
    exif_driver: {exiftool_path: /usr/bin/exiftool},  
}
```

Arguments to pass to the drivers.

27.4.12 Lock manager

- LOCK_MANAGER_BACKEND

Default:

```
mayan.apps.lock_manager.backends.file_lock.FileLock
```

Path to the class to use when to request and release resource locks.

- LOCK_MANAGER_DEFAULT_LOCK_TIMEOUT

Default:

```
30
```

Default amount of time in seconds after which a resource lock will be automatically released.

27.4.13 Mailing

- MAILER_DOCUMENT_BODY_TEMPLATE

Default:

```
Attached to this email is the document: {{ document }}  
-----  
This email has been sent from %(project_title)s (%(project_website)s)
```

Template for the document email form body text. Can include HTML.

- MAILER_DOCUMENT_SUBJECT_TEMPLATE

Default:

```
Document: {{ document }}
```

Template for the document email form subject line.

- MAILER_LINK_BODY_TEMPLATE

Default:

```
To access this document click on the following link: {{ link }}  
-----  
This email has been sent from %(project_title)s (%(project_website)s)
```

Template for the document link email form body text. Can include HTML.

- MAILER_LINK_SUBJECT_TEMPLATE

Default:

```
Link for document: {{ document }}
```

Template for the document link email form subject line.

27.4.14 Mayan

- MAYAN_CELERY_CLASS

Default:

```
celery.Celery
```

The class used to instantiate the main Celery app.

27.4.15 Metadata

- METADATA_AVAILABLE_PARSERS

Default:

```
[ 'mayan.apps.metadata.parsers.DateTimeParser', 'mayan.apps.metadata.parsers.  
˓→DateParser', 'mayan.apps.metadata.parsers.TimeParser' ]
```

None

- METADATA_AVAILABLE_VALIDATORS

Default:

```
[ 'mayan.apps.metadata.validators.DateTimeValidator', 'mayan.apps.metadata.  
˓→validators.DateValidator', 'mayan.apps.metadata.validators.TimeValidator' ]
```

None

27.4.16 Mirroring

- MIRRORING_DOCUMENT_CACHE_LOOKUP_TIMEOUT

Default:

```
10
```

Time in seconds to cache the path lookup to a document.

- MIRRORING_NODE_CACHE_LOOKUP_TIMEOUT

Default:

```
10
```

Time in seconds to cache the path lookup to an index node.

27.4.17 OCR

- OCR_AUTO_OCR

Default:

True

Set new document types to perform OCR automatically by default.

- OCR_BACKEND

Default:

mayan.apps.ocr.backends.tesseract.Tesseract

Full path to the backend to be used to do OCR.

- OCR_BACKEND_ARGUMENTS

Default:

None

27.4.18 Signatures

- SIGNATURES_GPG_HOME

Default:

/home/rosarior/.virtualenvs/mayan-edms-book/lib/python3.7/site-packages/mayan/ ↳ media/gpg_home
--

Home directory used to store keys as well as configuration files.

- SIGNATURES_GPG_PATH

Default:

/usr/bin/gpg1

Path to the GPG binary.

- SIGNATURES_KEYSERVER

Default:

pool.sks-keyservers.net

Keyserver used to query for keys.

27.4.19 Sources

- SOURCES_SCANIMAGE_PATH

Default:

/usr/bin/scainimage

File path to the scainimage program used to control image scanners.

-
- SOURCES_STAGING_FILE_CACHE_STORAGE_BACKEND

Default:

```
django.core.files.storage.FileSystemStorage
```

Path to the Storage subclass to use when storing the cached staging_file image files.

- SOURCES_STAGING_FILE_CACHE_STORAGE_BACKEND_ARGUMENTS

Default:

```
{location: /home/rosarior/.virtualenvs/mayan-edms-book/lib/python3.7/site-packages/mayan/media/staging_file_cache}
```

Arguments to pass to the SOURCES_STAGING_FILE_CACHE_STORAGE_BACKEND.

27.4.20 Storage

- STORAGE_TEMPORARY_DIRECTORY

Default:

```
/tmp
```

Temporary directory used site wide to store thumbnails, previews and temporary files.

27.5 How-to

27.5.1 Updating the configuration file



Permissions required:

- The “View settings” permission is required for this action.
- The “Edit settings” permission is required for this action.

This action can be performed in two ways.

27.5.1.1 Option 1: Via the user interface

1. Go to the *System* → *Setup* → *Settings* menu.
2. Click on the **Settings** button of the selected namespace.
3. Click on the **Edit** button of the selected setting.
4. Update the value of the setting.
5. Click the **Save** button.
6. Restart the installation for the new value to take effect.

27.5.1.2 Option 2: Via manual editing

1. Open the file `config.yml` located in the `media` folder of your installation using a text editor.
2. Update the value of the selected setting. The value must be a YAML valid value.
3. Save the file.
4. Restart Mayan EDMS for the new value to take effect.

27.5.2 Reverting the config file

1. Execute the `revertsettings` management command:

```
./manage.py revertsettings
```

If your installation uses a Python virtualenv, execute the command by specifying the full path to the `mayan-edms.py` module. For example, if your installation is located in the `/opt` folder, use:

```
/opt/bin/mayan-edms.py revertsettings
```

If using a Docker image, execute:

```
docker exec -ti mayan-edms /opt/mayan-edms/bin/mayan-edms.py revertsettings
```

2. If is valid backup configuration file to use for the `revertsettings` command, the following message will be displayed:

```
There is no last valid version to restore.
```

3. Restart Mayan EDMS for the new value to take effect.

27.5.3 Passing settings via environment variables

1. Obtain the name of the setting you wish to change. This can be done by navigating to the setting in the user interface via the **System → Setup → Settings**. It can also be done using the management command `showsettings`:

```
./manage.py showsettings
```

2. Add the prefix `MAYAN_` to the name of the settings option. For example, to change the value of the `COMMON_PAGINATE_BY` setting use:

```
export MAYAN_COMMON_PAGINATE_BY=10
```

3. Restart Mayan EDMS for the new value to take effect.
4. Verify that the setting took effect by navigating to the setting using the **System → Setup → Settings** menu.

27.6 Takeaways

- Settings are YAML formatted.
- Settings can be specified via a Python module, a YAML config file, or via environment variables.

-
- Each app specifies its settings.
 - Settings are prefixed with the name of the app that defined them.

NOTES

neeraj76@yahoo.com

SYSTEM EMAILS

The system email is used for sending password reset emails. It must be configured as shown in the chapter named *Settings* (page 203).

28.1 Sending administrative emails

Example:

```
EMAIL_BACKEND: django.core.mail.backends.smtp.EmailBackend
EMAIL_HOST: '<your smtp ip address or hostname>'
EMAIL_HOST_PASSWORD: '<your smtp password>'
EMAIL_HOST_USER: '<your smtp username>'
EMAIL_PORT: 25 # or 587 or your server's SMTP port
EMAIL_TIMEOUT:
EMAIL_USE_SSL: true
EMAIL_USE_TLS: false
```

To change the reference URL in the password reset emails in the default document mailing template modify the *COMMON_PROJECT_URL* (page 208) setting. For information on the different ways to change a setting check the *Settings* (page 203) chapter.

28.2 Testing the emails configuration

To test the email settings use the management command `sendtestemail`. Example:

```
mayan-edms.py sendtestemail myself@example.com
```

28.3 Takeaways

- System emails have no relation to the mailing app profiles.
- Mailing profiles are configure via the user interface. System emails are configured via the *Settings* (page 203) system.

NOTES

MAINTENANCE

29.1 Backups

Two pieces of data need to be collected to backup an installation of Mayan EDMS:

- The `media` folder
- The database

The `media` folder holds all the files for the uploaded documents, cache files, configuration files, and document signatures.

The database holds all the data associated with the document files. All the features provided by Mayan EDMS are live in the database.

To back up the `media` folder just copy the actual folder like any other filesystem folder. If you are using the default storage backend, the document files should be found in the `media` folder of your installation. If the installation was performed as shown in the deployment section of the installation chapter, the complete path of the `media` folder should be `/opt/mayan-edms/media`.

To backup the database refer to your database manager documentation. Mayan EDMS is agnostic on the type of method used to backup the database and works with live or dumped styles of database backups.

Here is an example of how to perform a backup and a restore using a PostgreSQL database manager.

To dump the database into an SQL text file use:

```
pg_dump -h <host> -U <database user> -c <database name> -W > `date +%Y-%m-%d"_"%H-%M-%S`.sql
```

Example:

```
pg_dump -h 127.0.0.1 -U mayan -c mayan -W > `date +%Y-%m-%d"_"%H-%M-%S`.sql
```

To restore the database from the SQL text file:

```
psql -h <host> -U <database user> -d <database name> -W -f <sql dump file>
```

Example:

```
psql -h 127.0.0.1 -U mayan -d mayan -W -f 2018-06-07_18-10-56.sql
```

Here is an example of how to perform a backup and restore of a PostgreSQL Docker container using a compressed dump file. A dump file is not compatible or interchangeable with an SQL text file.

To backup a PostgreSQL Docker container:

```
docker exec <container name> pg_dump -U <database user> -Fc -c <database name> > _
↳ `date +%Y-%m-%d"_"%H-%M-%S` .dump
```

Example:

```
docker exec mayan-edms-db pg_dump -U mayan -Fc -c mayan > `date +%Y-%m-%d"_"%H-%M-%S` .
↳ dump
```

This will produce a compressed dump file with the current date and time as the filename.

To restore a PostgreSQL Docker container from a dump file:

```
docker exec -i <container name> pg_restore -U <database user> -d <database name> <
↳ <dump file>
```

Since it is not possible to drop a currently open PostgreSQL database, this command must be used on a new and empty PostgreSQL container.

Example:

```
docker run -d \
--name mayan-edms-pg-new \
--restart=always \
-p 5432:5432 \
-e POSTGRES_USER=mayan \
-e POSTGRES_DB=mayan \
-e POSTGRES_PASSWORD=mayanuserpass \
-v /docker-volumes/mayan-edms/postgres-new:/var/lib/postgresql/data \
-d postgres:9.6

docker exec -i mayan-edms-pg-new pg_restore -U mayan -d mayan < 2018-06-07_17-09-34.
↳ dump
```

More information at:

- PostgreSQL: <https://www.postgresql.org/docs/current/static/backup.html>
- MySQL: <https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html>

DOCKER IMAGE

30.1 Environment Variables

In addition to the standard Mayan EDMS, the Docker image adds some that are specific to it, or that behave somewhat differently to provide additional functionality.

MAYAN_BROKER_URL

This optional environment variable determines the broker that Celery will use to relay task messages between the frontend code and the background workers. For more information read the pertinent Celery Kombu documentation page: Broker URL⁸

The Docker image supports using Redis and RabbitMQ as brokers.

Caveat: If the *MAYAN_BROKER_URL* and *MAYAN_CELERY_RESULT_BACKEND* environment variables are specified, the built-in Redis server inside the container will be disabled.

MAYAN_CELERY_RESULT_BACKEND

This optional environment variable determines the results backend that Celery will use to relay result messages from the background workers to the frontend code. For more information read the pertinent Celery Kombu documentation page: Task result backend settings⁹

The Docker image supports using Redis and RabbitMQ as result backends.

Caveat: If the *MAYAN_BROKER_URL* and *MAYAN_CELERY_RESULT_BACKEND* environment variables are specified, the built-in Redis server inside the container will be disabled.

MAYAN_SETTINGS_MODULE

Optional. Allows loading an alternate settings file.

MAYAN_GUNICORN_WORKERS

Optional. This environment variable controls the number of frontend workers that will be executed. If not specified the default is 2. For heavier loads, user a higher number. A formula recommended for this setting is the number of CPU cores + 1.

MAYAN_WORKER_FAST_CONCURRENCY

Optional. Changes the concurrency (number of child processes) of the Celery worker consuming the queues in the fast (low latency, short tasks) category. Default is 1. Use 0 to disable hardcoded concurrency and allow the Celery worker to launch its default number of child processes (equal to the number of CPUs detected).

MAYAN_WORKER_MEDIUM_CONCURRENCY

⁸ <http://kombu.readthedocs.io/en/latest/userguide/connections.html#connection-urls>

⁹ <http://docs.celeryproject.org/en/3.1/configuration.html#celery-result-backend>

Optional. Changes the concurrency (number of child processes) of the Celery worker consuming the queues in the medium (medium latency, long running tasks) category. Default is 1. Use 0 to disable hardcoded concurrency and allow the Celery worker to launch its default number of child processes (equal to the number of CPUs detected).

MAYAN_WORKER_SLOW_CONCURRENCY

Optional. Changes the concurrency (number of child processes) of the Celery worker consuming the queues in the slow (high latency, very long running tasks) category. Default is 1. Use 0 to disable hardcoded concurrency and allow the Celery worker to launch its default number of child processes (equal to the number of CPUs detected).

MAYAN_USER_UID

Optional. Changes the UID of the mayan user internal to the Docker container. Defaults to 1000.

MAYAN_USER_GUID

Optional. Changes the GUID of the mayan user internal to the Docker container. Defaults to 1000.

30.2 Accessing outside data

To use Mayan EDMS's staging folders or watch folders from Docker, the data for these source must be made accessible to the container. This is done by mounting the folders in the host computer to folders inside the container. This is necessary because Docker containers do not have access to host data on purpose. For example, to make a folder in the host accessible as a watch folder, add the following to the Docker command line when starting the container:

```
-v /opt/scanned_files:/scanned_files
```

The command line would look like this:

```
docker run ... -v /opt/scanned_files:/scanned_files mayanedms/mayanedms:latest
```

Now create a watch folder in Mayan EDMS using the path /scanned_files and the documents from the host folder /opt/scanned_files will be automatically available. Use the same procedure to mount host folders to be used as staging folders. In this example /scanned_files was used as the container directory, but any path can be used as long as:

- the path not an already existing path
- the path is not used by any other program
- the path is a single level path

30.3 Performing backups

To backup the existing data, stop the image and copy the content of the volume. For the example:

```
docker run -d --name mayan-edms --restart=always -p 80:8000 \
-v /docker-volumes/mayan:/var/lib/mayan \
-v /opt/scanned_files:/srv/watch_folder mayanedms/mayanedms:latest
```

That would be the /docker-volumes/mayan folder:

```
sudo tar -zcvf backup.tar.gz /docker-volumes/mayan
sudo chown `whoami` backup.tar.gz
```

If using an external PostgreSQL or MySQL database or database containers, these too need to be backed up using their respective procedures. A simple solution is to copy the entire database container volume after the container has been stopped.

30.4 Restoring from a backup

Uncompress the backup archive in the original docker volume using:

```
sudo tar -xvzf backup.tar.gz -C /
```

30.5 Upgrading

Upgrading a Mayan EDMS Docker container is actually a matter of stopping and deleting the container, downloading the most recent version of the image and starting a container again. The container will take care of updating the database structure to the newest version if necessary.

IMPORTANT! Do not delete the volume storing the data, only the container.

Stop the container to be upgraded:

```
docker stop mayan-edms
```

Remove the container:

```
docker rm mayan-edms
```

Pull the new image version:

```
docker pull mayanedms/mayanedms:latest
```

Start the container again with the new image version:

```
docker run -d --name mayan-edms --restart=always -p 80:8000 -v /docker-volumes/mayan:/var/lib/mayan mayanedms/mayanedms:latest
```

30.6 Customizing the image

If you just need to add a few Ubuntu or Python packages to your installation, you can use the following environment variables:

MAYAN_APT_INSTALLS

Specifies a list of Ubuntu .deb packages to be installed via APT when the container is first created. The installed packages are not lost when the image is stopped. Example: To install the Tesseract OCR language packs for German and Spanish add the following in your docker start command line:

```
-e MAYAN_APT_INSTALLS="tesseract-ocr-deu tesseract-ocr-spa"
```

MAYAN_PIP_INSTALLS

Specifies a list of Python packages to be installed via pip. Packages will be downloaded from the Python Package Index (<https://pypi.python.org>) by default.

30.7 Using Docker compose

To deploy a complete production stack using the included Docker compose file execute:

```
docker-compose -f docker-compose.yml up -d
```

This Docker compose file will provision four containers:

- Postgres as the database
- Redis as the Celery result storage
- RabbitMQ as the Celery broker
- Mayan EDMS using the above service containers

To stop the stack use:

```
docker-compose -f docker-compose.yml stop
```

The stack will also create four volumes to store the data of each container. These are:

- mayan_app - The Mayan EDMS data container, normally called *mayan_data* when not using Docker compose.
- mayan_broker - The broker volume, in this case RabbitMQ.
- mayan_db - The database volume, in this case Postgres.
- mayan_results - The celery result backend volume, in this case Redis.

30.8 Nightly images

The continuous integration pipeline used for testing development builds also produces a resulting Docker image. These are build automatically and their stability is not guaranteed. They should never be used in production. If you want to try out the Docker images the development uses or want a sneak peek at the new features being worked on checkout the container registry at: https://gitlab.com/mayan-edms/mayan-edms/container_registry

TROUBLESHOOTING

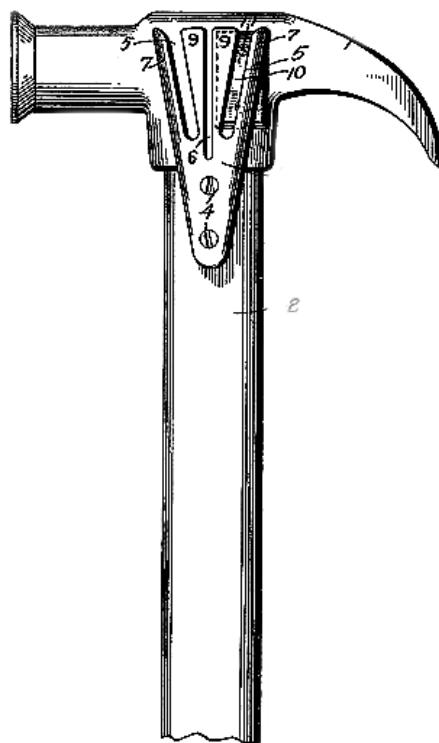


Fig. 1: Claw hammer patent illustration, 1902

31.1 Database

31.1.1 MySQL error: `OperationalError: (1267, "Illegal mix of collations (latin1_swedish_ci, IMPLICIT) and (utf8_general_ci, COERCIBLE) for operation '='")`

```
$ mayan-edms.py shell
>>> from django.db import connection
>>> cursor = connection.cursor()
```

(continues on next page)

(continued from previous page)

```
>>> cursor.execute('SHOW TABLES')
>>> results=[]
>>> for row in cursor.fetchall(): results.append(row)
>>> for row in results: cursor.execute('ALTER TABLE %s CONVERT TO CHARACTER SET utf8
↪COLLATE utf8_general_ci;' % (row[0]))
```

References:

- <http://stackoverflow.com/questions/1073295/django-character-set-with-mysql-weirdness>

31.1.2 MySQL error: Incorrect string value: 'xE2xE80x95rs6...' for column 'content' at row 1

When using MySQL and doing OCR on languages other than English

Use utf-8 collation on MySQL server, or at least in table ‘documents_documentpage’, ‘content’ field

References:

- http://groups.google.com/group/django-users/browse_thread/thread/429447086fca6412
- <http://markmail.org/message/bqajx2utvmtriixi>

31.1.3 MySQL error: django.db.utils.IntegrityError IntegrityError: (1452, 'Cannot add or update a child row: a foreign key constraint fails (`...`.`...`, CONSTRAINT `..._refs_id_b0252274` FOREIGN KEY (`...`) REFERENCES `...`(`...`))')

Solution: Convert all MySQL tables to the same type, either all MyISAM or InnoDB

31.2 PostgreSQL error: OperationalError: FATAL: sorry, too many clients already

Set MAYAN_DATABASE_CONN_MAX_AGE to 0

This setting keeps a database connection alive. It allows reuse of database connections. When Mayan EDMS is deployed with Gunicorn a microthreads backend, the database connections are not shared and this setting has the reverse effect of exhausting the available PostgreSQL connections available. To avoid this, Setting MAYAN_DATABASE_CONN_MAX_AGE to 0 will cause all microthreads to release their connections, by closing them when finished.

References:

- <https://serverfault.com/questions/635100/django-conn-max-age-persists-connections-but-doesnt-reuse-them-with-postgresq>
- <https://github.com/benoitc/gunicorn/issues/996>

31.3 Dependencies

Due to OS differences some binaries might reside in different locations. Use environment variables or the configuration file to tell Mayan EDMS where to file these binaries.

Example: OpenBSD. Add the following entries to supervisor configuration files.

```
MAYAN_DOCUMENT_PARSING_PDFTOTEXT_PATH=/usr/local/bin/pdftotext,  
MAYAN_SIGNATURES_GPG_PATH=/usr/local/bin/gpg,  
MAYAN_SOURCES_SCANIMAGE_PATH: /usr/local/bin/scanimage,
```

Alternatively a symlink from the actual binary location to where Mayan EDMS is expecting them to be found by default also works for some users:

```
ln -s /usr/local/bin/gpg /usr/bin/gpg1
```

Example 2: Ubuntu 16.04. Add the following entries to supervisor configuration files.

```
MAYAN_SIGNATURES_GPG_PATH=/usr/bin/gpg1,
```

Or add a symlink:

```
ln -s /usr/bin/gpg /usr/bin/gpg1
```

31.4 Docker

31.4.1 MAYAN_APT_INSTALLS does not work for Archlinux with kernels > 4.14

This is caused by a change from kernel 4.18 - 4.19. Metacopy on these kernels is set to yes in archlinux kernels (/sys/module/overlay/parameters/metacopy) and overlayfs should override this which it does not at the moment.

The workaround is to disable metacopy:

```
echo N | sudo tee /sys/module/overlay/parameters/metacopy
```

References:

- <https://bbs.archlinux.org/viewtopic.php?id=241866>
- <https://www.spinics.net/lists/linux-unionfs/msg06316.html>

31.5 Passwords

31.5.1 Admin password reset

To reset the password of the admin account use the following command:

```
MAYAN_MEDIA_ROOT=<your Mayan media root setting> <installation directory>/bin/mayan-  
edms.py changepassword admin
```

If you followed the deploying instructions from the documentation your MAYAN_MEDIA_ROOT will be /opt/mayan-edms/media.

If using a Docker image, execute the command inside the container. First you need to know the name of the Docker container running Mayan EDMS on your setup with:

```
docker ps
```

Then execute the password reset command inside the Docker container:

```
docker exec -ti <your docker container name> /opt/mayan-edms/bin/mayan-edms.py  
→changepassword admin
```

Another way to do this is to execute a shell inside the container to get a command prompt:

```
docker exec -ti <your docker container name> /bin/bash
```

And then execute the command:

```
/opt/mayan-edms/bin/mayan-edms.py changepassword admin
```

31.5.2 Missing automatic admin account after installation

This is caused when the `initialsetup` command is interrupted as the admin user is created outside of the database migrations.

To create an admin super user account manually use the command:

```
MAYAN_MEDIA_ROOT=<your Mayan media root setting> <installation directory>/bin/mayan-  
→edms.py createsuperuser
```

If you followed the deploying instructions from the documentation your MAYAN_MEDIA_ROOT will be /opt/mayan-edms/media.

If using a Docker image, execute the command inside the container. First find you container name with:

```
docker ps
```

Then execute the command inside the container:

```
docker exec -ti <your docker container name> /opt/mayan-edms/bin/mayan-edms.py  
→createsuperuser
```

Another way to do this is to execute a shell inside the container to get a command prompt:

```
docker exec -ti <your docker container name> /bin/bash
```

And then execute the command:

```
/opt/mayan-edms/bin/mayan-edms.py createsuperuser
```

31.6 Watchfolders

31.6.1 Incomplete files uploaded

To avoid uploading files are they are being copied to the watchfolder, copy the files to a temporary directory on the same partition as the watchfolder first. Then move the files to the watchfolder. The move will be executed as an atomic operation and will prevent the files to be uploaded in the middle of the copying process.

NOTES

PART VII.

INTEGRATION

neeraj76@yahoo.com

API

Mayan EDMS provides an HTTP REST Application Program Interface (or API). This API allows integration with 3rd party software using simple HTTP requests.

Several API authentication methods are provided: **Session**, **Token**, and **HTTP Basic**.

The URL for the API can be found via the **Tools → REST API** menu. The API is also self-documenting. The live API documentation can be found in the **Tools → API Documentation (Swagger)** menu for the Swagger version and in the **Tools → API Documentation (ReDoc)** menu for the ReDoc version.

There are a few ways to structure REST APIs. In the case of Mayan EDMS, API endpoints are structured by resource type. Examples:

- /cabinets - To view or create new cabinets
- /cabinets/<id> - To view the details, edit, or delete an existing cabinet.
- /cabinets/<id>/documents - To view, add, or remove documents from an existing cabinet.
- /cabinets/<id>/documents/<id> - To view, add, or remove one document from an existing cabinet.

The API supports the HTTP verbs: **GET**, **POST**, **PUT**, **PATCH**, and **DELETE**.

32.1 API paths

32.1.1 auth

- POST /auth/token/obtain/

auth_token_obtain_create

Obtain an API authentication token.

32.1.2 cabinets

- GET /cabinets/

cabinets_list

Returns a list of all the cabinets.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

-
- POST /cabinets/

cabinets_create

Create a new cabinet

Parameters:

- data (*body*) (**required**)

[Schema: *WritableCabinet* (page 290)]

- DELETE /cabinets/{id}/

cabinets_delete

Delete the selected cabinet.

- GET /cabinets/{id}/

cabinets_read

Returns the details of the selected cabinet.

- PATCH /cabinets/{id}/

cabinets_partial_update

Edit the selected cabinet.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableCabinet* (page 290)]

- PUT /cabinets/{id}/

cabinets_update

Edit the selected cabinet.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableCabinet* (page 290)]

- GET /cabinets/{id}/documents/

cabinets_documents_list

Returns a list of all the documents contained in a particular cabinet.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /cabinets/{id}/documents/

cabinets_documents_create

Add a document to the selected cabinet.

Parameters:

- data (*body*) (**required**)

[Schema: *NewCabinetDocument* (page 282)]

-
- DELETE /cabinets/{id}/documents/{document_pk}/

cabinets_documents_delete

Remove a document from the selected cabinet.

- GET /cabinets/{id}/documents/{document_pk}/

cabinets_documents_read

Returns the details of the selected cabinet document.

32.1.3 checkouts

- GET /checkouts/

checkouts_list

Returns a list of all the documents that are currently checked out.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /checkouts/

checkouts_create

Checkout a document.

Parameters:

- data (*body*) (**required**)

[Schema: *NewDocumentCheckout* (page 282)]

- DELETE /checkouts/{id}/checkout_info/

checkouts_checkout_info_delete

Checkin a document.

- GET /checkouts/{id}/checkout_info/

checkouts_checkout_info_read

Retrieve the details of the selected checked out document entry.

32.1.4 content_types

- GET /content_types/

content_types_list

Returns a list of all the available content types.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

32.1.5 document_types

- GET /document_types/

document_types_list

Returns a list of all the document types.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /document_types/

document_types_create

Create a new document type.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableDocumentType* (page 291)]

- GET /document_types/{document_type_pk}/metadata_types/

document_types_metadata_types_list

Returns a list of selected document type's metadata types.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /document_types/{document_type_pk}/metadata_types/

document_types_metadata_types_create

Add a metadata type to the selected document type.

Parameters:

- data (*body*) (**required**)

[Schema: *NewDocumentTypeMetadataType* (page 283)]

- DELETE /document_types/{document_type_pk}/metadata_types/{metadata_type_pk}/

document_types_metadata_types_delete

Remove a metadata type from a document type.

- GET /document_types/{document_type_pk}/metadata_types/{metadata_type_pk}/

document_types_metadata_types_read

Retrieve the details of a document type metadata type.

- PATCH /document_types/{document_type_pk}/metadata_types/{metadata_type_pk}/

document_types_metadata_types_partial_update

Edit the selected document type metadata type.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableDocumentTypeMetadataType* (page 292)]

- PUT /document_types/{document_type_pk}/metadata_types/{metadata_type_pk}/
document_types_metadata_types_update

Edit the selected document type metadata type.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableDocumentTypeMetadataType* (page 292)]

- DELETE /document_types/{id}/
document_types_delete

Delete the selected document type.

- GET /document_types/{id}/

document_types_read

Return the details of the selected document type.

- PATCH /document_types/{id}/
document_types_partial_update

Edit the properties of the selected document type.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableDocumentType* (page 291)]

- PUT /document_types/{id}/

document_types_update

Edit the properties of the selected document type.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableDocumentType* (page 291)]

- GET /document_types/{id}/documents/
document_types_documents_list

Returns a list of all the documents of a particular document type.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- GET /document_types/{id}/workflows/
document_types_workflows_list

Returns a list of all the document type workflows.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

32.1.6 documents

- GET /documents/

documents_list

Returns a list of all the documents.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /documents/

documents_create

Create a new document.

Parameters:

- data (*body*) (**required**)
[Schema: *NewDocument* (page 282)]

- GET /documents/recent/

documents_recent_list

Return a list of the recent documents for the current user.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- GET /documents/{document_pk}/comments/

documents_comments_list

Returns a list of all the document comments.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /documents/{document_pk}/comments/

documents_comments_create

Create a new document comment.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableComment* (page 291)]

-
- DELETE /documents/{document_pk}/comments/{comment_pk} /
documents_comments_delete

Delete the selected document comment.

- GET /documents/{document_pk}/comments/{comment_pk} /
documents_comments_read

Returns the details of the selected document comment.

- PATCH /documents/{document_pk}/comments/{comment_pk} /
documents_comments_partial_update

Parameters:

- data (*body*) (**required**)

[Schema: *Comment* (page 274)]

- PUT /documents/{document_pk}/comments/{comment_pk} /
documents_comments_update

Parameters:

- data (*body*) (**required**)

[Schema: *Comment* (page 274)]

- GET /documents/{document_pk}/metadata/ /
documents_metadata_list

Returns a list of selected document's metadata types and values.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /documents/{document_pk}/metadata/ /
documents_metadata_create

Add an existing metadata type and value to the selected document.

Parameters:

- data (*body*) (**required**)

[Schema: *NewDocumentMetadata* (page 282)]

- DELETE /documents/{document_pk}/metadata/{metadata_pk} /
documents_metadata_delete

Remove this metadata entry from the selected document.

- GET /documents/{document_pk}/metadata/{metadata_pk} /
documents_metadata_read

Return the details of the selected document metadata type and value.

-
- PATCH /documents/{document_pk}/metadata/{metadata_pk}/
documents_metadata_partial_update

Edit the selected document metadata type and value.

Parameters:

- data (*body*) (**required**)

[Schema: *DocumentMetadata* (page 276)]

- PUT /documents/{document_pk}/metadata/{metadata_pk}/
documents_metadata_update

Edit the selected document metadata type and value.

Parameters:

- data (*body*) (**required**)

[Schema: *DocumentMetadata* (page 276)]

- GET /documents/{document_pk}/tags/
documents_tags_list

Returns a list of all the tags attached to a document.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /documents/{document_pk}/tags/
documents_tags_create

Attach a tag to a document.

Parameters:

- data (*body*) (**required**)

[Schema: *NewDocumentTag* (page 283)]

- DELETE /documents/{document_pk}/tags/{id}/
documents_tags_delete

Remove a tag from the selected document.

- GET /documents/{document_pk}/tags/{id}/
documents_tags_read

Returns the details of the selected document tag.

- POST /documents/{document_pk}/versions/{version_pk}/ocr/
documents_versions_ocr_create

Submit a document version for OCR.

- GET /documents/{document_pk}/versions/{version_pk}/pages/{page_pk}/content/
documents_versions_pages_content_read

Returns the content of the selected document page.

- GET /documents/{document_pk}/versions/{version_pk}/pages/{page_pk}/ocr/
documents_versions_pages_ocr_read

Returns the OCR content of the selected document page.

- DELETE /documents/{id}/
documents_delete

Move the selected document to the thrash.

- GET /documents/{id}/
documents_read

Return the details of the selected document.

- PATCH /documents/{id}/
documents_partial_update

Edit the properties of the selected document.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableDocument* (page 291)]

- PUT /documents/{id}/
documents_update

Edit the properties of the selected document.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableDocument* (page 291)]

- GET /documents/{id}/cabinets/
documents_cabinets_list

Returns a list of all the cabinets to which a document belongs.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- GET /documents/{id}/download/
documents_download_read

Download the latest version of a document.

- GET /documents/{id}/indexes/
documents_indexes_list

Returns a list of all the indexes to which a document belongs.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- GET /documents/{id}/resolved_smart_links/
documents_resolved_smart_links_list

Returns a list of the smart links that apply to the document.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- GET /documents/{id}/resolved_smart_links/{smart_link_pk}/
documents_resolved_smart_links_read

Return the details of the selected resolved smart link.

- GET /documents/{id}/resolved_smart_links/{smart_link_pk}/documents/
documents_resolved_smart_links_documents_list

Returns a list of the smart link documents that apply to the document.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /documents/{id}/submit/
documents_submit_create

Submit a document for OCR.

- GET /documents/{id}/versions/
documents_versions_list

Return a list of the selected document's versions.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /documents/{id}/versions/
documents_versions_create

Create a new document version.

Parameters:

- data (*body*) (**required**)
[Schema: *NewDocumentVersion* (page 283)]

- DELETE /documents/{id}/versions/{version_pk}/
documents_versions_delete

Delete the selected document version.

-
- GET /documents/{id}/versions/{version_pk} /
documents_versions_read
Returns the selected document version details.
 - PATCH /documents/{id}/versions/{version_pk} /
documents_versions_partial_update
Edit the selected document version.
Parameters:
 - data (*body*) (**required**)
[Schema: *WritableDocumentVersion* (page 292)]
 - PUT /documents/{id}/versions/{version_pk} /
documents_versions_update
Edit the selected document version.
Parameters:
 - data (*body*) (**required**)
[Schema: *WritableDocumentVersion* (page 292)]
 - GET /documents/{id}/versions/{version_pk}/download /
documents_versions_download_read
Download a document version.
 - GET /documents/{id}/versions/{version_pk}/pages /
documents_versions_pages_list
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]
 - GET /documents/{id}/versions/{version_pk}/pages/{page_pk} /
documents_versions_pages_read
Returns the selected document page details.
 - PATCH /documents/{id}/versions/{version_pk}/pages/{page_pk} /
documents_versions_pages_partial_update
Edit the selected document page.
Parameters:
 - data (*body*) (**required**)
[Schema: *DocumentPage* (page 276)]
 - PUT /documents/{id}/versions/{version_pk}/pages/{page_pk} /
documents_versions_pages_update
Edit the selected document page.
Parameters:

- `data (body) (required)`

[Schema: *DocumentPage* (page 276)]

- GET `/documents/{id}/versions/{version_pk}/pages/{page_pk}/image/`
documents_versions_pages_image_read

Returns an image representation of the selected document.

- GET `/documents/{id}/workflows/`
documents_workflows_list

Returns a list of all the document workflows.

Parameters:

- `page (query)` A page number within the paginated result set.
[integer]

- GET `/documents/{id}/workflows/{workflow_pk}/`
documents_workflows_read

Return the details of the selected document workflow.

- GET `/documents/{id}/workflows/{workflow_pk}/log_entries/`
documents_workflows_log_entries_list

Returns a list of all the document workflows log entries.

Parameters:

- `page (query)` A page number within the paginated result set.
[integer]

- POST `/documents/{id}/workflows/{workflow_pk}/log_entries/`
documents_workflows_log_entries_create

Transition a document workflow by creating a new document workflow log entry.

Parameters:

- `data (body) (required)`

[Schema: *WritableWorkflowInstanceLogEntry* (page 294)]

32.1.7 event_type_namespaces

- GET `/event_type_namespaces/`
event_type_namespaces_list

Returns a list of all the available event type namespaces.

Parameters:

- `page (query)` A page number within the paginated result set.
[integer]

-
- GET /event_type_namespaces/{name} /
event_type_namespaces_read
Returns the details of an event type namespace.
 - GET /event_type_namespaces/{name}/event_types /
event_type_namespaces_event_types_list
Returns a list of all the available event types from a namespaces.
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]

32.1.8 event_types

- GET /event_types /
event_types_list
Returns a list of all the available event types.
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]

32.1.9 events

- GET /events /
events_list
Returns a list of all the available events.
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]

32.1.10 groups

- GET /groups /
groups_list
Returns a list of all the groups.
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]

-
- POST /groups/

groups_create

Create a new group.

Parameters:

- data (*body*) (**required**)

[Schema: *Group* (page 279)]

- DELETE /groups/{id}/

groups_delete

Delete the selected group.

- GET /groups/{id}/

groups_read

Return the details of the selected group.

- PATCH /groups/{id}/

groups_partial_update

Partially edit the selected group.

Parameters:

- data (*body*) (**required**)

[Schema: *Group* (page 279)]

- PUT /groups/{id}/

groups_update

Edit the selected group.

Parameters:

- data (*body*) (**required**)

[Schema: *Group* (page 279)]

32.1.11 indexes

- GET /indexes/

indexes_list

Returns a list of all the defined indexes.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /indexes/

indexes_create

Create a new index.

Parameters:

- `data (body) (required)`

[Schema: *Index* (page 279)]

- GET /indexes/node/{id}/documents/

indexes_node_documents_list

Returns a list of all the documents contained by a particular index node instance.

Parameters:

- `page (query) A page number within the paginated result set.`

[integer]

- DELETE /indexes/template/{id}/

indexes_template_delete

Delete the selected index template node.

- GET /indexes/template/{id}/

indexes_template_read

Returns the details of the selected index template node.

- PATCH /indexes/template/{id}/

indexes_template_partial_update

Partially edit an index template node.

Parameters:

- `data (body) (required)`

[Schema: *IndexTemplateName* (page 280)]

- PUT /indexes/template/{id}/

indexes_template_update

Edit an index template node.

Parameters:

- `data (body) (required)`

[Schema: *IndexTemplateName* (page 280)]

- DELETE /indexes/{id}/

indexes_delete

Delete the selected index.

- GET /indexes/{id}/

indexes_read

Returns the details of the selected index.

- PATCH /indexes/{id}/

indexes_partial_update

Partially edit an index.

Parameters:

- `data (body) (required)`
[Schema: *Index* (page 279)]

- `PUT /indexes/{id}/`

indexes_update

Edit an index.

Parameters:

- `data (body) (required)`

[Schema: *Index* (page 279)]

- `GET /indexes/{id}/template/`

indexes_template_list

Returns a list of all the template nodes for the selected index.

Parameters:

- `page (query) A page number within the paginated result set.`

[integer]

32.1.12 keys

- `GET /keys/`

keys_list

Returns a list of all the keys.

Parameters:

- `page (query) A page number within the paginated result set.`

[integer]

- `POST /keys/`

keys_create

Upload a new key.

Parameters:

- `data (body) (required)`

[Schema: *Key* (page 280)]

- `DELETE /keys/{id}/`

keys_delete

Delete the selected key.

- `GET /keys/{id}/`

keys_read

Return the details of the selected key.

32.1.13 messages

- GET /messages/

messages_list

Returns a list of all the messages.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /messages/

messages_create

Create a new message.

Parameters:

- data (*body*) (**required**)
[Schema: *Message* (page 281)]

- DELETE /messages/{id}/

messages_delete

Delete the selected message.

- GET /messages/{id}/

messages_read

Return the details of the selected message.

- PATCH /messages/{id}/

messages_partial_update

Edit the selected message.

Parameters:

- data (*body*) (**required**)
[Schema: *Message* (page 281)]

- PUT /messages/{id}/

messages_update

Edit the selected message.

Parameters:

- data (*body*) (**required**)
[Schema: *Message* (page 281)]

32.1.14 metadata_types

- GET /metadata_types/

metadata_types_list

Returns a list of all the metadata types.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /metadata_types/

metadata_types_create

Create a new metadata type.

Parameters:

- data (*body*) (**required**)
[Schema: *MetadataType* (page 281)]

- DELETE /metadata_types/{metadata_type_pk}/

metadata_types_delete

Delete the selected metadata type.

- GET /metadata_types/{metadata_type_pk}/

metadata_types_read

Return the details of the selected metadata type.

- PATCH /metadata_types/{metadata_type_pk}/

metadata_types_partial_update

Edit the selected metadata type.

Parameters:

- data (*body*) (**required**)
[Schema: *MetadataType* (page 281)]

- PUT /metadata_types/{metadata_type_pk}/

metadata_types_update

Edit the selected metadata type.

Parameters:

- data (*body*) (**required**)
[Schema: *MetadataType* (page 281)]

32.1.15 notifications

- GET /notifications/

notifications_list

Return a list of notifications for the current user.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

32.1.16 objects

- GET /objects/{app_label}/{model}/{object_id}/acls/

objects_acls_list

Returns a list of all the object's access control lists

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /objects/{app_label}/{model}/{object_id}/acls/

objects_acls_create

Create a new access control list for the selected object.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableAccessControlList* (page 290)]

- DELETE /objects/{app_label}/{model}/{object_id}/acls/{id}/

objects_acls_delete

Delete the selected access control list.

- GET /objects/{app_label}/{model}/{object_id}/acls/{id}/

objects_acls_read

Returns the details of the selected access control list.

- GET /objects/{app_label}/{model}/{object_id}/acls/{id}/permissions/

objects_acls_permissions_list

Returns the access control list permission list.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /objects/{app_label}/{model}/{object_id}/acls/{id}/permissions/

objects_acls_permissions_create

Add a new permission to the selected access control list.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableAccessControlListPermission* (page 290)]

- DELETE /objects/{app_label}/{model}/{object_id}/acls/{id}/permissions/{permission_pk}/

objects_acls_permissions_delete

Remove the permission from the selected access control list.

-
- GET /objects/{app_label}/{model}/{object_id}/acls/{id}/permissions/{permission_pk}/

objects_acls_permissions_read

Returns the details of the selected access control list permission.

- GET /objects/{app_label}/{model}/{object_id}/events/

objects_events_list

Return a list of events for the specified object.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

32.1.17 permissions

- GET /permissions/

permissions_list

Returns a list of all the available permissions.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

32.1.18 roles

- GET /roles/

roles_list

Returns a list of all the roles.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- POST /roles/

roles_create

Create a new role.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableRole* (page 293)]

- DELETE /roles/{id}/

roles_delete

Delete the selected role.

-
- GET /roles/{id}/

roles_read

Return the details of the selected role.

- PATCH /roles/{id}/

roles_partial_update

Edit the selected role.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableRole* (page 293)]

- PUT /roles/{id}/

roles_update

Edit the selected role.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableRole* (page 293)]

32.1.19 search

- GET /search/advanced/{search_model}/

search_advanced_read

Perform an advanced search operation

- GET /search/{search_model}/

search_read

Perform a search operation

32.1.20 search_models

- GET /search_models/

search_models_list

Returns a list of all the available search models.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

32.1.21 smart_links

- GET /smart_links/

smart_links_list

Returns a list of all the smart links.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /smart_links/

smart_links_create

Create a new smart link.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableSmartLink* (page 293)]

- DELETE /smart_links/{id}/

smart_links_delete

Delete the selected smart link.

- GET /smart_links/{id}/

smart_links_read

Return the details of the selected smart link.

- PATCH /smart_links/{id}/

smart_links_partial_update

Edit the selected smart link.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableSmartLink* (page 293)]

- PUT /smart_links/{id}/

smart_links_update

Edit the selected smart link.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableSmartLink* (page 293)]

- GET /smart_links/{id}/conditions/

smart_links_conditions_list

Returns a list of all the smart link conditions.

Parameters:

-
- page (*query*) A page number within the paginated result set.
[integer]
- POST /smart_links/{id}/conditions/
smart_links_conditions_create
Create a new smart link condition.
Parameters:
 - data (*body*) (**required**)
[Schema: *SmartLinkCondition* (page 285)]
 - DELETE /smart_links/{id}/conditions/{condition_pk}/
smart_links_conditions_delete
Delete the selected smart link condition.
 - GET /smart_links/{id}/conditions/{condition_pk}/
smart_links_conditions_read
Return the details of the selected smart link condition.
 - PATCH /smart_links/{id}/conditions/{condition_pk}/
smart_links_conditions_partial_update
Edit the selected smart link condition.
Parameters:
 - data (*body*) (**required**)
[Schema: *SmartLinkCondition* (page 285)]
 - PUT /smart_links/{id}/conditions/{condition_pk}/
smart_links_conditions_update
Edit the selected smart link condition.

32.1.22 staging_folders

- GET /staging_folders/
staging_folders_list
Returns a list of all the staging folders and the files they contain.
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]

-
- GET /staging_folders/file/{staging_folder_pk}/{encoded_filename} /
staging_folders_file_read
Details of the selected staging file.
 - GET /staging_folders/file/{staging_folder_pk}/{encoded_filename}/image/ /
staging_folders_file_image_read
Returns an image representation of the selected document.
 - GET /staging_folders/{id} /
staging_folders_read
Details of the selected staging folders and the files it contains.

32.1.23 tags

- GET /tags /
tags_list
Returns a list of all the tags.
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]
- POST /tags /
tags_create
Create a new tag.
Parameters:
 - data (*body*) (**required**)
[Schema: *WritableTag* (page 293)]
- DELETE /tags/{id} /
tags_delete
Delete the selected tag.
- GET /tags/{id} /
tags_read
Return the details of the selected tag.
- PATCH /tags/{id} /
tags_partial_update
Edit the selected tag.
Parameters:
 - data (*body*) (**required**)
[Schema: *WritableTag* (page 293)]

-
- PUT /tags/{id}/

tags_update

Edit the selected tag.

Parameters:

- data (*body*) (**required**)

[Schema: *WritableTag* (page 293)]

- GET /tags/{id}/documents/

tags_documents_list

Returns a list of all the documents tagged by a particular tag.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

32.1.24 templates

- GET /templates/

templates_list

Returns a list of all the available templates.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- GET /templates/{name}/

templates_read

Retrieve the details of the partial template.

32.1.25 trashed_documents

- GET /trashed_documents/

trashed_documents_list

Returns a list of all the trashed documents.

Parameters:

- page (*query*) A page number within the paginated result set.

[integer]

- DELETE /trashed_documents/{id}/

trashed_documents_delete

Delete the trashed document.

-
- GET /trashed_documents/{id} /
trashed_documents_read
Retreive the details of the trashed document.
 - POST /trashed_documents/{id}/restore/
trashed_documents_restore_create
Restore a trashed document.

32.1.26 users

- GET /users/
users_list
Returns a list of all the users.
Parameters:
 - page (*query*) A page number within the paginated result set.
[integer]
- POST /users/
users_create
Create a new user.
Parameters:
 - data (*body*) (**required**)
[Schema: *User* (page 287)]
- DELETE /users/current/
users_current_delete
Delete the current user.
- GET /users/current/
users_current_read
Return the details of the current user.
- PATCH /users/current/
users_current_partial_update
Partially edit the current user.
Parameters:
 - data (*body*) (**required**)
[Schema: *User* (page 287)]
- PUT /users/current/
users_current_update
Edit the current user.
Parameters:

- `data (body) (required)`

[Schema: *User* (page 287)]

- `DELETE /users/{id}/`

users_delete

Delete the selected user.

- `GET /users/{id}/`

users_read

Return the details of the selected user.

- `PATCH /users/{id}/`

users_partial_update

Partially edit the selected user.

Parameters:

- `data (body) (required)`

[Schema: *User* (page 287)]

- `PUT /users/{id}/`

users_update

Edit the selected user.

Parameters:

- `data (body) (required)`

[Schema: *User* (page 287)]

- `GET /users/{id}/groups/`

users_groups_list

Returns a list of all the groups to which an user belongs.

Parameters:

- `page (query) A page number within the paginated result set.`

[integer]

- `POST /users/{id}/groups/`

users_groups_create

Add a user to a list of groups.

Parameters:

- `data (body) (required)`

[Schema: *UserGroupList* (page 288)]

32.1.27 workflows

- GET /workflows/

workflows_list

Returns a list of all the workflows.

Parameters:

- page (*query*) A page number within the paginated result set.
[integer]

- POST /workflows/

workflows_create

Create a new workflow.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableWorkflow* (page 294)]

- DELETE /workflows/{id}/

workflows_delete

Delete the selected workflow.

- GET /workflows/{id}/

workflows_read

Return the details of the selected workflow.

- PATCH /workflows/{id}/

workflows_partial_update

Edit the selected workflow.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableWorkflow* (page 294)]

- PUT /workflows/{id}/

workflows_update

Edit the selected workflow.

Parameters:

- data (*body*) (**required**)
[Schema: *WritableWorkflow* (page 294)]

- GET /workflows/{id}/document_types/

workflows_document_types_list

Returns a list of all the document types attached to a workflow.

Parameters:

- `page` (*query*) A page number within the paginated result set.
[integer]

- POST /workflows/{id}/document_types/

workflows_document_types_create

Attach a document type to a specified workflow.

Parameters:

- `data` (*body*) (**required**)

[Schema: *NewWorkflowDocumentType* (page 283)]

- DELETE /workflows/{id}/document_types/{document_type_pk}/

workflows_document_types_delete

Remove a document type from the selected workflow.

- GET /workflows/{id}/document_types/{document_type_pk}/

workflows_document_types_read

Returns the details of the selected workflow document type.

- GET /workflows/{id}/states/

workflows_states_list

Returns a list of all the workflow states.

Parameters:

- `page` (*query*) A page number within the paginated result set.

[integer]

- POST /workflows/{id}/states/

workflows_states_create

Create a new workflow state.

Parameters:

- `data` (*body*) (**required**)

[Schema: *WorkflowState* (page 289)]

- DELETE /workflows/{id}/states/{state_pk}/

workflows_states_delete

Delete the selected workflow state.

- GET /workflows/{id}/states/{state_pk}/

workflows_states_read

Return the details of the selected workflow state.

- PATCH /workflows/{id}/states/{state_pk}/

workflows_states_partial_update

Edit the selected workflow state.

Parameters:

-
- **data (body) (required)**

[Schema: *WorkflowState* (page 289)]

- PUT /workflows/{id}/states/{state_pk}/

workflows_states_update

Edit the selected workflow state.

Parameters:

- **data (body) (required)**

[Schema: *WorkflowState* (page 289)]

- GET /workflows/{id}/transitions/

workflows_transitions_list

Returns a list of all the workflow transitions.

Parameters:

- **page (query)** A page number within the paginated result set.

[integer]

- POST /workflows/{id}/transitions/

workflows_transitions_create

Create a new workflow transition.

Parameters:

- **data (body) (required)**

[Schema: *WritableWorkflowTransition* (page 294)]

- DELETE /workflows/{id}/transitions/{transition_pk}/

workflows_transitions_delete

Delete the selected workflow transition.

- GET /workflows/{id}/transitions/{transition_pk}/

workflows_transitions_read

Return the details of the selected workflow transition.

- PATCH /workflows/{id}/transitions/{transition_pk}/

workflows_transitions_partial_update

Edit the selected workflow transition.

Parameters:

- **data (body) (required)**

[Schema: *WritableWorkflowTransition* (page 294)]

- PUT /workflows/{id}/transitions/{transition_pk}/

workflows_transitions_update

Edit the selected workflow transition.

Parameters:

-
- `data (body) (required)`
[Schema: *WritableWorkflowTransition* (page 294)]

32.2 API schema

32.2.1 AccessControlList

- `content_type` [Schema: *ContentType* (page 275)]
- `id (read only)` [integer]
- `object_id (required)` [integer]
- `permissions_url (read only)` [string]
API URL pointing to the list of permissions for this access control list.
- `role` [Schema: *Role* (page 285)]
- `url (read only)` [string]

32.2.2 AccessControlListPermission

- `namespace (read only)` [string]
- `pk (read only)` [string]
- `label (read only)` [string]
- `acl_permission_url (read only)` [string]
API URL pointing to a permission in relation to the access control list to which it is attached. This URL is different than the canonical workflow URL.
- `acl_url (read only)` [string]

32.2.3 Cabinet

- `children (read only)` [string]
List of children cabinets.
- `documents_count (read only)` [string]
Number of documents on this cabinet level.
- `documents_url (read only)` (format: uri) [string]
URL of the API endpoint showing the list documents inside this cabinet.
- `full_path (read only)` [string]
The name of this cabinet level appended to the names of its ancestors.
- `id (read only)` [integer]
- `label (required)` (max length: 128) [string]
- `parent (required)` [integer]

-
- parent_url (**read only**) [string]
 - url (**read only**) (format: uri) [string]

32.2.4 CabinetDocument

- date_added (**read only**) (format: date-time) [string]
The server date and time when the document was finally processed and added to the system.
- description (**read only**) [string]
An optional short text describing a document.
- document_type (**required**) [Schema: *DocumentType* (page 277)]
- id (**read only**) [integer]
- label (**read only**) [string]
The name of the document.
- language (**read only**) [string]
The dominant language in the document.
- latest_version [Schema: *DocumentVersion* (page 278)]
- url (**read only**) (format: uri) [string]
- uuid (**read only**) (format: uuid) [string]
UUID of a document, universally Unique ID. An unique identifier generated for each document.
- versions_url (**read only**) (format: uri) [string]
- cabinet_document_url (**read only**) [string]
API URL pointing to a document in relation to the cabinet storing it. This URL is different than the canonical document URL.

32.2.5 Comment

- comment (**required**) [string]
- document [Schema: *Document* (page 275)]
- document_comments_url (**read only**) [string]
- id (**read only**) [integer]
- submit_date (**read only**) (format: date-time) [string]
- url (**read only**) [string]
- user [Schema: *User* (page 287)]

32.2.6 ContentType

- app_label (**required**) (max length: 100) [string]
- id (**read only**) [integer]
- model (**required**) (max length: 100) [string]

32.2.7 DeletedDocument

- date_added (**read only**) (format: date-time) [string]

The server date and time when the document was finally processed and added to the system.

- deleted_date_time (**read only**) (format: date-time) [string]

The server date and time when the document was moved to the trash.

- description (**read only**) [string]

An optional short text describing a document.

- document_type (**read only**) (format: uri) [string]

- document_type_label (**read only**) [string]

- id (**read only**) [integer]

- label (**read only**) [string]

The name of the document.

- language (**read only**) [string]

The dominant language in the document.

- restore (**read only**) (format: uri) [string]

- url (**read only**) (format: uri) [string]

- uuid (**read only**) (format: uuid) [string]

UUID of a document, universally Unique ID. An unique identifier generated for each document.

32.2.8 Document

- date_added (**read only**) (format: date-time) [string]

The server date and time when the document was finally processed and added to the system.

- description [string]

An optional short text describing a document.

- document_type (**required**) [Schema: *DocumentType* (page 277)]

- id (**read only**) [integer]

- label (max length: 255) [string]

The name of the document.

-
- `language` (max length: 8) [string]
The dominant language in the document.
 - `latest_version` [Schema: *DocumentVersion* (page 278)]
 - `url` (**read only**) (format: uri) [string]
 - `uuid` (**read only**) (format: uuid) [string]
UUID of a document, universally Unique ID. An unique identifier generated for each document.
 - `versions_url` (**read only**) (format: uri) [string]

32.2.9 DocumentCheckout

- `document` (**required**) [Schema: *Document* (page 275)]

32.2.10 DocumentMetadata

- `document` [Schema: *Document* (page 275)]
- `id` (**read only**) [integer]
- `metadata_type` [Schema: *MetadataType* (page 281)]
- `url` (**read only**) [string]
- `value` (max length: 255) [string]

The actual value stored in the metadata type field for the document.

32.2.11 DocumentPage

- `document_version_url` (**read only**) [string]
- `image_url` (**read only**) [string]
- `page_number` (**read only**) [integer]
- `url` (**read only**) [string]

32.2.12 DocumentPageContent

- `content` [string]
The actual text content as extracted by the document parsing backend.

32.2.13 DocumentPageOCRContent

- `content` [string]
The actual text content extracted by the OCR backend.

32.2.14 DocumentTag

- color (**read only**) [string]

The RGB color values for the tag.

- documents_count (**read only**) [string]

- documents_url (**read only**) (format: uri) [string]

- id (**read only**) [integer]

- label (**read only**) [string]

A short text used as the tag name.

- url (**read only**) (format: uri) [string]

- document_tag_url (**read only**) [string]

API URL pointing to a tag in relation to the document attached to it. This URL is different than the canonical tag URL.

32.2.15 DocumentType

- delete_time_period [integer]

Amount of time after which documents of this type in the trash will be deleted.

- delete_time_unit [string]

- days

- hours

- minutes

- documents_url (**read only**) (format: uri) [string]

- documents_count (**read only**) [string]

- id (**read only**) [integer]

- label (**required**) (max length: 96) [string]

The name of the document type.

- filenames (**read only**) [array]

- trash_time_period [integer]

Amount of time after which documents of this type will be moved to the trash.

- trash_time_unit [string]

- days

- hours

- minutes

- url (**read only**) (format: uri) [string]

32.2.16 DocumentTypeFilename

- filename (**required**) (max length: 128) [string]

32.2.17 DocumentTypeMetadataType

- document_type [Schema: *DocumentType* (page 277)]
- id (**read only**) [integer]
- metadata_type [Schema: *MetadataType* (page 281)]
- required [boolean]
- url (**read only**) [string]

32.2.18 DocumentVersion

- checksum (**read only**) [string]

A hash/checkdigit/fingerprint generated from the document's binary data. Only identical documents will have the same checksum.

- comment [string]

An optional short text describing the document version.

- document_url (**read only**) [string]
- download_url (**read only**) [string]
- encoding (**read only**) [string]

The document version file encoding. binary 7-bit, binary 8-bit, text, base64, etc.

- file (**read only**) [string]
- mimetype (**read only**) [string]

The document version's file mimetype. MIME types are a standard way to describe the format of a file, in this case the file format of the document. Some examples: "text/plain" or "image/jpeg".

- pages_url (**read only**) [string]
- size (**read only**) [string]
- timestamp (**read only**) (format: date-time) [string]

The server date and time when the document version was processed.

- url (**read only**) [string]

32.2.19 Event

- id (**read only**) [integer]
- actor (**read only**) [string]
- target (**read only**) [string]

-
- actor_content_type [Schema: *ContentType* (page 275)]
 - target_content_type [Schema: *ContentType* (page 275)]
 - verb [Schema: *EventType* (page 279)]
 - actor_object_id (**required**) (max length: 255) [string]
 - description [string]
 - target_object_id (max length: 255) [string]
 - timestamp (format: date-time) [string]
 - public [boolean]

32.2.20 EventType

- label (**required**) [string]
- name (**required**) [string]
- id (**required**) [string]
- event_type_namespace_url (**read only**) [string]

32.2.21 EventTypeNamespace

- label (**required**) [string]
- name (**required**) [string]
- url (**read only**) [string]
- event_types_url (**read only**) (format: uri) [string]

32.2.22 Group

- id (**read only**) [integer]
- name (**required**) (max length: 80) [string]
- url (**read only**) (format: uri) [string]
- users_count (**read only**) [string]

32.2.23 Index

- document_types (**required**) [array]
- enabled [boolean]

Causes this index to be visible and updated when document data changes.

- id (**read only**) [integer]
- instance_root [Schema: *IndexInstanceNode* (page 280)]
- label (**required**) (max length: 128) [string]

-
- node_templates (**read only**) [array]

32.2.24 IndexInstanceNode

- documents (**read only**) (format: uri) [string]
- documents_count (**read only**) [string]
- children (**read only**) [string]
- id (**read only**) [integer]
- level (**read only**) [integer]
- parent [integer]
- value (max length: 128) [string]

32.2.25 IndexTemplateNode

- enabled [boolean]

Causes this node to be visible and updated when document data changes.

- expression (**required**) [string]

Enter a template to render. Use Django's default templating language (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>)

- id (**read only**) [integer]
- index (**required**) [integer]
- level (**read only**) [integer]
- link_documents [boolean]

Check this option to have this node act as a container for documents and not as a parent for further nodes.

- parent [integer]

32.2.26 Key

- algorithm (**read only**) [integer]
 - creation_date (**read only**) (format: date) [string]
 - expiration_date (**read only**) (format: date) [string]
 - fingerprint (**read only**) [string]
 - id (**read only**) [integer]
 - key_data (**required**) [string]
- ASCII armored version of the key.
- key_type (**read only**) [string]
 - length (**read only**) [integer]
 - url (**read only**) (format: uri) [string]

-
- `user_id` (**read only**) [string]

32.2.27 Message

- `end_datetime` (format: date-time) [string]

Date and time until when this message is to be displayed.

- `enabled` [boolean]

- `label` (**required**) (max length: 32) [string]

Short description of this message.

- `message` (**required**) [string]

The actual message to be displayed.

- `start_datetime` (format: date-time) [string]

Date and time after which this message will be displayed.

- `id` (**read only**) [integer]

- `url` (**read only**) (format: uri) [string]

32.2.28 MetadataType

- `default` (max length: 128) [string]

Enter a template to render. Use Django's default templating language (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>)

- `id` (**read only**) [integer]

- `label` (**required**) (max length: 48) [string]

- `lookup` [string]

Enter a template to render. Must result in a comma delimited string. Use Django's default templating language (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>).

- `name` (**required**) (max length: 48) [string]

Name used by other apps to reference this metadata type. Do not use python reserved words, or spaces.

- `parser` [string]

- `mayan.apps.metadata.parsers.DateTimeParser`

- `mayan.apps.metadata.parsers.DateParser`

- `mayan.apps.metadata.parsers.TimeParser`

The parser will reformat the value entered to conform to the expected format.

- `url` (**read only**) (format: uri) [string]

- `validation` [string]

- `mayan.apps.metadata.validators.DateTimeValidator`

- `mayan.apps.metadata.validators.DateValidator`

- `mayan.apps.metadata.validators.TimeValidator`

The validator will reject data entry if the value entered does not conform to the expected format.

32.2.29 NewCabinetDocument

- documents_pk_list (**required**) [string]

Comma separated list of document primary keys to add to this cabinet.

32.2.30 NewDocument

- description [string]

An optional short text describing a document.

- document_type (**required**) [integer]

- id (**read only**) [integer]

- file (**read only**) (**required**) (format: uri) [string]

- label (max length: 255) [string]

The name of the document.

- language (max length: 8) [string]

The dominant language in the document.

32.2.31 NewDocumentCheckout

- block_new_version (**required**) [boolean]

- document (**read only**) [integer]

- document_pk (**required**) [integer]

Primary key of the document to be checked out.

- expiration_datetime (**required**) (format: date-time) [string]

- id (**read only**) [integer]

32.2.32 NewDocumentMetadata

- id (**read only**) [integer]

- metadata_type_pk (**required**) [integer]

Primary key of the metadata type to be added to the document.

- url (**read only**) [string]

- value (max length: 255) [string]

The actual value stored in the metadata type field for the document.

32.2.33 NewDocumentTag

- tag_pk (**required**) [integer]

Primary key of the tag to be added.

32.2.34 NewDocumentTypeMetadataType

- id (**read only**) [integer]
- metadata_type_pk (**required**) [integer]
Primary key of the metadata type to be added.
- required [boolean]
- url (**read only**) [string]

32.2.35 NewDocumentVersion

- comment (**required**) [string]
- file (**read only**) (**required**) [string]

32.2.36 NewWorkflowDocumentType

- document_type_pk (**required**) [integer]
Primary key of the document type to be added.

32.2.37 Notification

- action [Schema: *Event* (page 278)]
- read [boolean]
- user [Schema: *User* (page 287)]

32.2.38 Permission

- namespace (**read only**) [string]
- pk (**read only**) [string]
- label (**read only**) [string]

32.2.39 RecentDocument

- document (**read only**) [integer]
- datetime_accessed (**read only**) (format: date-time) [string]

32.2.40 ResolvedSmartLink

- conditions_url (**read only**) (format: uri) [string]
- document_types (**read only**) [array]
- dynamic_label (**read only**) [string]

Enter a template to render. Use Django's default templating language (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>). The {{ document }} context variable is available.

- enabled (**read only**) [boolean]
- label (**read only**) [string]
- id (**read only**) [integer]
- url (**read only**) (format: uri) [string]
- resolved_dynamic_label (**read only**) [string]
- resolved_smart_link_url (**read only**) [string]
- resolved_documents_url (**read only**) [string]

32.2.41 ResolvedSmartLinkDocument

- date_added (**read only**) (format: date-time) [string]

The server date and time when the document was finally processed and added to the system.

- description (**read only**) [string]

An optional short text describing a document.

- document_type (**required**) [Schema: *DocumentType* (page 277)]

- id (**read only**) [integer]

- label (**read only**) [string]

The name of the document.

- language (**read only**) [string]

The dominant language in the document.

- latest_version [Schema: *DocumentVersion* (page 278)]

- url (**read only**) (format: uri) [string]

- uuid (**read only**) (format: uuid) [string]

UUID of a document, universally Unique ID. An unique identifier generated for each document.

- versions_url (**read only**) (format: uri) [string]

- resolved_smart_link_url (**read only**) [string]

32.2.42 Role

- groups (**read only**) [array]
- id (**read only**) [integer]
- label (**required**) (max length: 64) [string]
- permissions (**read only**) [array]
- url (**read only**) (format: uri) [string]

32.2.43 SearchField

- field (**read only**) [string]
- label (**read only**) [string]

32.2.44 SearchModel

- app_label (**read only**) [string]
- model_name (**read only**) [string]
- pk (**read only**) [string]
- search_fields (**read only**) [array]

32.2.45 SmartLink

- conditions_url (**read only**) (format: uri) [string]
- document_types (**read only**) [array]
- dynamic_label (max length: 96) [string]

Enter a template to render. Use Django's default templating language (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>). The {{ document }} context variable is available.

- enabled [boolean]
- label (**required**) (max length: 96) [string]
- id (**read only**) [integer]
- url (**read only**) (format: uri) [string]

32.2.46 SmartLinkCondition

- enabled [boolean]
- expression (**required**) [string]

Enter a template to render. Use Django's default templating language (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>). The {{ document }} context variable is available.

-
- `foreign_document_data` (**required**) (max length: 128) [string]

This represents the metadata of all other documents.

- `inclusion` [string]

– &

– |

The inclusion is ignored for the first item.

- `id` (**read only**) [integer]

- `negated` [boolean]

Inverts the logic of the operator.

- `operator` (**required**) [string]

– exact

– iexact

– contains

–icontains

– in

– gt

– gte

– lt

– lte

– startswith

– istartswith

– endswith

– iendswith

– regex

– iregex

- `smart_link_url` (**read only**) [string]

- `url` (**read only**) [string]

32.2.47 StagingFolder

- `files` (**read only**) [string]

32.2.48 StagingFolderFile

- `filename` (**required**) (max length: 255) [string]

- `image_url` (**read only**) [string]

- `url` (**read only**) [string]

32.2.49 Tag

- `color (required) (max length: 7) [string]`
The RGB color values for the tag.
- `documents_count (read only) [string]`
- `documents_url (read only) (format: uri) [string]`
- `id (read only) [integer]`
- `label (required) (max length: 128) [string]`
A short text used as the tag name.
- `url (read only) (format: uri) [string]`

32.2.50 Template

- `hex_hash (read only) [string]`
- `name (read only) [string]`
- `html (read only) [string]`
- `url (read only) (format: uri) [string]`

32.2.51 User

- `first_name (max length: 30) [string]`
- `date_joined (read only) (format: date-time) [string]`
- `email (max length: 254) (format: email) [string]`
- `groups (read only) [array]`
- `groups_pk_list [string]`

List of group primary keys to which to add the user.

- `id (read only) [integer]`
- `is_active (read only) [boolean]`

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

- `last_login (read only) (format: date-time) [string]`
- `last_name (max length: 30) [string]`
- `password [string]`
- `url (read only) (format: uri) [string]`
- `username (required) (max length: 150) [string]`

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

32.2.52 UserGroupList

- group_pk_list (**required**) [string]
Comma separated list of group primary keys to assign this user to.

32.2.53 Workflow

- document_types_url (**read only**) (format: uri) [string]
- id (**read only**) [integer]
- internal_name (**required**) (max length: 255) [string]
This value will be used by other apps to reference this workflow. Can only contain letters, numbers, and underscores.
- label (**required**) (max length: 255) [string]
- states [array]
- transitions [array]
- url (**read only**) (format: uri) [string]

32.2.54 WorkflowDocumentType

- delete_time_period (**read only**) [integer]
Amount of time after which documents of this type in the trash will be deleted.
- delete_time_unit (**read only**) [string]
 - days
 - hours
 - minutes
- documents_url (**read only**) (format: uri) [string]
- documents_count (**read only**) [string]
- id (**read only**) [integer]
- label (**read only**) [string]
The name of the document type.
- filenames (**read only**) [array]
- trash_time_period (**read only**) [integer]
Amount of time after which documents of this type will be moved to the trash.
- trash_time_unit (**read only**) [string]
 - days
 - hours
 - minutes
- url (**read only**) (format: uri) [string]

-
- `workflow_document_type_url` (**read only**) [string]

API URL pointing to a document type in relation to the workflow to which it is attached. This URL is different than the canonical document type URL.

32.2.55 WorkflowInstance

- `current_state` [Schema: *WorkflowState* (page 289)]

- `document_workflow_url` (**read only**) [string]

API URL pointing to a workflow in relation to the document to which it is attached. This URL is different than the canonical workflow URL.

- `last_log_entry` [Schema: *WorkflowInstanceLogEntry* (page 289)]

- `log_entries_url` (**read only**) [string]

A link to the entire history of this workflow.

- `transition_choices` (**read only**) [array]

- `workflow` [Schema: *Workflow* (page 288)]

32.2.56 WorkflowInstanceLogEntry

- `comment` [string]

- `datetime` (**read only**) (format: date-time) [string]

- `document_workflow_url` (**read only**) [string]

- `transition` [Schema: *WorkflowTransition* (page 290)]

- `user` [Schema: *User* (page 287)]

32.2.57 WorkflowState

- `completion` [integer]

Enter the percent of completion that this state represents in relation to the workflow. Use numbers without the percent sign.

- `id` (**read only**) [integer]

- `initial` [boolean]

Select if this will be the state with which you want the workflow to start in. Only one state can be the initial state.

- `label` (**required**) (max length: 255) [string]

- `url` (**read only**) [string]

- `workflow_url` (**read only**) [string]

32.2.58 WorkflowTransition

- destination_state (**required**) [Schema: *WorkflowState* (page 289)]
- id (**read only**) [integer]
- label (**required**) (max length: 255) [string]
- origin_state (**required**) [Schema: *WorkflowState* (page 289)]
- url (**read only**) [string]
- workflow_url (**read only**) [string]

32.2.59 WritableAccessControlList

- content_type [Schema: *ContentType* (page 275)]
- id (**read only**) [integer]
- object_id (**read only**) [integer]
- permissions_pk_list [string]
Comma separated list of permission primary keys to grant to this access control list.
- permissions_url (**read only**) [string]
API URL pointing to the list of permissions for this access control list.
- role_pk (**required**) [integer]
Primary keys of the role to which this access control list binds to.
- url (**read only**) [string]

32.2.60 WritableAccessControlListPermission

- namespace (**read only**) [string]
- pk (**read only**) [string]
- label (**read only**) [string]
- acl_permission_url (**read only**) [string]
API URL pointing to a permission in relation to the access control list to which it is attached. This URL is different than the canonical workflow URL.
- acl_url (**read only**) [string]
- permission_pk (**required**) [string]
Primary key of the new permission to grant to the access control list.

32.2.61 WritableCabinet

- documents_pk_list [string]
Comma separated list of document primary keys to add to this cabinet.

-
- `label` (**required**) (max length: 128) [string]
 - `id` (**read only**) [integer]
 - `parent` [integer]

32.2.62 WritableComment

- `comment` (**required**) [string]
- `document` [Schema: *Document* (page 275)]
- `document_comments_url` (**read only**) [string]
- `id` (**read only**) [integer]
- `submit_date` (**read only**) (format: date-time) [string]
- `url` (**read only**) [string]
- `user` [Schema: *User* (page 287)]

32.2.63 WritableDocument

- `date_added` (**read only**) (format: date-time) [string]
The server date and time when the document was finally processed and added to the system.
- `description` [string]
An optional short text describing a document.
- `document_type` [Schema: *DocumentType* (page 277)]
- `id` (**read only**) [integer]
- `label` (max length: 255) [string]
The name of the document.
- `language` (max length: 8) [string]
The dominant language in the document.
- `latest_version` [Schema: *DocumentVersion* (page 278)]
- `url` (**read only**) (format: uri) [string]
- `uuid` (**read only**) (format: uuid) [string]
UUID of a document, universally Unique ID. An unique identifier generated for each document.
- `versions` (**read only**) (format: uri) [string]

32.2.64 WritableDocumentType

- `delete_time_period` [integer]
Amount of time after which documents of this type in the trash will be deleted.
- `delete_time_unit` [string]

-
- days
 - hours
 - minutes
- documents_url (**read only**) (format: uri) [string]
 - documents_count (**read only**) [string]
 - id (**read only**) [integer]
 - label (**required**) (max length: 96) [string]
The name of the document type.
 - trash_time_period [integer]
Amount of time after which documents of this type will be moved to the trash.
 - trash_time_unit [string]
 - days
 - hours
 - minutes
 - url (**read only**) (format: uri) [string]

32.2.65 WritableDocumentTypeMetadataType

- id (**read only**) [integer]
- required [boolean]
- url (**read only**) [string]

32.2.66 WritableDocumentVersion

- checksum (**read only**) [string]
A hash/checkdigit/fingerprint generated from the document's binary data. Only identical documents will have the same checksum.
- comment [string]
An optional short text describing the document version.
- document_url (**read only**) [string]
- download_url (**read only**) [string]
- encoding (**read only**) [string]
The document version file encoding. binary 7-bit, binary 8-bit, text, base64, etc.
- file (**read only**) [string]
- mimetype (**read only**) [string]
The document version's file mimetype. MIME types are a standard way to describe the format of a file, in this case the file format of the document. Some examples: "text/plain" or "image/jpeg".
- pages_url (**read only**) [string]

-
- `timestamp` (**read only**) (format: date-time) [string]

The server date and time when the document version was processed.

- `url` (**read only**) [string]

32.2.67 WritableRole

- `groups_pk_list` [string]

Comma separated list of groups primary keys to add to, or replace in this role.

- `id` (**read only**) [integer]

- `label` (**required**) (max length: 64) [string]

- `permissions_pk_list` [string]

Comma separated list of permission primary keys to grant to this role.

32.2.68 WritableSmartLink

- `conditions_url` (**read only**) (format: uri) [string]

- `document_types_pk_list` [string]

Comma separated list of document type primary keys to which this smart link will be attached.

- `dynamic_label` (max length: 96) [string]

Enter a template to render. Use Django's default templating language (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>). The `{{ document }}` context variable is available.

- `enabled` [boolean]

- `label` (**required**) (max length: 96) [string]

- `id` (**read only**) [integer]

- `url` (**read only**) (format: uri) [string]

32.2.69 WritableTag

- `color` (**required**) (max length: 7) [string]

The RGB color values for the tag.

- `documents_pk_list` [string]

Comma separated list of document primary keys to which this tag will be attached.

- `id` (**read only**) [integer]

- `label` (**required**) (max length: 128) [string]

A short text used as the tag name.

32.2.70 WritableWorkflow

- document_types_pk_list [string]
Comma separated list of document type primary keys to which this workflow will be attached.
- label (**required**) (max length: 255) [string]
- id (**read only**) [integer]
- url (**read only**) (format: uri) [string]

32.2.71 WritableWorkflowInstanceLogEntry

- comment [string]
- datetime (**read only**) (format: date-time) [string]
- document_workflow_url (**read only**) [string]
- transition [Schema: *WorkflowTransition* (page 290)]
- transition_pk (**required**) [integer]
Primary key of the transition to be added.
- user [Schema: *User* (page 287)]

32.2.72 WritableWorkflowTransition

- destination_state_pk (**required**) [integer]
Primary key of the destination state to be added.
- id (**read only**) [integer]
- label (**required**) (max length: 255) [string]
- origin_state_pk (**required**) [integer]
Primary key of the origin state to be added.
- url (**read only**) [string]
- workflow_url (**read only**) [string]

32.3 How-to

32.3.1 Installing an HTTP client

Install Python Requests (<https://2.python-requests.org>):

```
pip install requests
```

32.3.2 Getting a list of document types

```
import requests

requests.get('http://127.0.0.1:8000/api/document_types/', auth=('username', 'password')) .json()
```

Output:

```
{'count': 1,
'next': None,
'previous': None,
'results': [{ 'delete_time_period': 30,
  'delete_time_unit': 'days',
  'documents_count': 12,
  'documents_url': 'http://127.0.0.1:8000/api/document_types/1/documents/',
  'filenames': [],
  'id': 1,
  'label': 'Default',
  'trash_time_period': None,
  'trash_time_unit': None,
  'url': 'http://127.0.0.1:8000/api/document_types/1/' } ]}
```

32.3.3 Uploading a new document

```
import requests

with open('test_document.pdf', mode='rb') as file_object:
    requests.post('http://127.0.0.1:8000/api/documents/', auth=('username', 'password')) .json()
```

Output:

```
{'description': '',
'document_type': 1,
'id': 19,
'label': 'test_document.pdf',
'language': 'eng'}
```

32.3.4 Obtaining API tokens

Use API tokens to avoid sending the username and password on every request. Obtain a token by making a POST request to /api/auth/token/obtain/?format=json

```
import requests

requests.post('http://127.0.0.1:8000/api/auth/token/obtain/?format=json', data={'username': 'username', 'password': 'password'}) .json()
```

Output:

```
{'token': '4ccbc35b5eb327aa82dc3b7c9747b578900f02bb'}
```

32.3.5 Adding API token to the request header

```
import requests

headers = {'Authorization': 'Token 4ccbc35b5eb327aa82dc3b7c9747b578900f02bb'}

requests.get('http://127.0.0.1:8000/api/document_types/', headers=headers).json()
```

Output:

```
{'description': '',
'document_type': 1,
'id': 19,
'label': 'test_document.pdf',
'language': 'eng'}
```

32.3.6 Using sessions

```
import requests

session = requests.Session()

headers = {'Authorization': 'Token 4ccbc35b5eb327aa82dc3b7c9747b578900f02bb'}

session.headers.update(headers)

session.get('http://127.0.0.1:8000/api/document_types/')
```

Output:

```
{'description': u '',
'document_type': 1,
'id': 19,
'label': 'test_document.pdf',
'language': 'eng'}
```

32.3.7 Searching documents

```
import requests

requests.get('http://127.0.0.1:8000/api/search/documents.Document/?q=pdf', auth=(  
    'username', 'password')).json()
```

Output:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "date_added": "2019-01-17T00:20:36.629753-04:00",
```

(continues on next page)

(continued from previous page)

```
"description": "",  
"document_type": {  
    "delete_time_period": 30,  
    "delete_time_unit": "days",  
    "documents_url": "http://127.0.0.1:8000/api/document_types/1/  
→documents/",  
    "documents_count": 42,  
    "id": 1,  
    "label": "Default",  
    "filenames": [  
        {  
            "filename": "text.pdf"  
        }  
    ],  
    "trash_time_period": null,  
    "trash_time_unit": null,  
    "url": "http://127.0.0.1:8000/api/document_types/1/"  
},  
}  
]  
}
```

32.3.8 Searching documents by field

```
import requests  
  
requests.get('http://127.0.0.1:8000/api/search/advanced/documents.Document/?label=pdf  
→', auth=('username', 'password')).json()
```

Output:

```
{  
    "count": 1,  
    "next": null,  
    "previous": null,  
    "results": [  
        {  
            "date_added": "2019-01-17T00:20:36.629753-04:00",  
            "description": "",  
            "document_type": {  
                "delete_time_period": 30,  
                "delete_time_unit": "days",  
                "documents_url": "http://127.0.0.1:8000/api/document_types/1/  
→documents/",  
                "documents_count": 42,  
                "id": 1,  
                "label": "Default",  
                "filenames": [  
                    {  
                        "filename": "text.pdf"  
                    }  
                ],  
                "trash_time_period": null,  
                "trash_time_unit": null,  
                "url": "http://127.0.0.1:8000/api/document_types/1/"  
            },  
        }  
    ]  
}
```

(continues on next page)

(continued from previous page)

```
        "url": "http://127.0.0.1:8000/api/document_types/1/"  
    },  
}  
]  
}
```

NOTES

neeraj76@yahoo.com

PART VIII.

INTERNALS

neeraj76@yahoo.com

PROJECT STRUCTURE

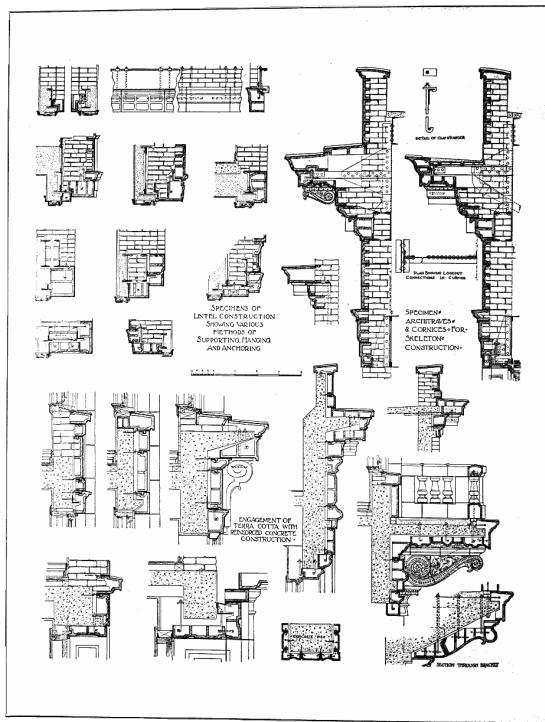


Fig. 1: Architectural terra cotta. The Northwest Terra Cotta Company, 1914.

Mayan EDMS is a coded using the Python language. All of the code is original code and copyrighted. When writing original code is not possible or not practical, other projects and libraries are used. All 3rd party code is accessed in compliance to their respective licenses. These projects and their use are:

- Django, used as the base web app framework, database ORM, and HTML templating.
- Pillow, used for generating previews of images files and for image transformations.
- PyPDF2, used for PDF introspection, specifically, page count.
- PyYAML, used for the settings system and for the parsing of arguments.
- Celery, used background tasks.
- django-activity-stream, used as the base of the event app.
- django-celery, used for dynamic scheduling of sources.
- django-colorful, used by the tags app.

-
- django-cors-headers, used to allow remote domain access to the API.
 - django-downloadview, used for the views that provide a download.
 - django-environ, used to parse the environment variables.
 - django-formtools, used as the base for the upload wizard.
 - django-mathfilters, used to provide additional template tags.
 - django-model-utils, used for the source subclasses inheritance.
 - django-mptt, used for representing binary trees in the cabinets and index apps.
 - django-pure-pagination, used for list view pagination.
 - django-qsstats-magic, used for statistics computation.
 - django-solo, used for the autoadmin app model singleton.
 - django-stronghold, used to default each view to require authentication.
 - django-widget-tweaks, used to customize the form widget rendering.
 - djangorestframework, used as the base the REST API.
 - djangorestframework-recursive, used to represent binary trees in the API.
 - drf-yasg, used for the API documentation.
 - flanker, used for the email sources.
 - flex, a requirement of drf-yasg.
 - furl, used for URL calculations.
 - fusepy, used by the mirroring app.
 - gevent, used as the worker for Gunicorn.
 - graphviz, used for the workflow previews.
 - gunicorn, used as the app server.
 - mock, used to mock services during tests.
 - node-semver, used for the NPM package version computations.
 - pathlib2, used for object oriented file path computations.
 - pycountry, used for the document language code.
 - pyocr, used by the OCR app to interface with Tesseract.
 - python-dateutil, used by the metadata app for the validators and parsers.
 - python-magic, used by the MIME type app to interface with libmagic.
 - python_gnupg, used by the signatures app to interface with GNUPG.
 - pytz, a requirements of Django to provide smart time and date objects.
 - requests, used by the workflow HTTP action.
 - sh, used to execute binaries.
 - swagger-spec-validator, a requirement of drf-yasg.
 - whitenoise, used for static media serving.

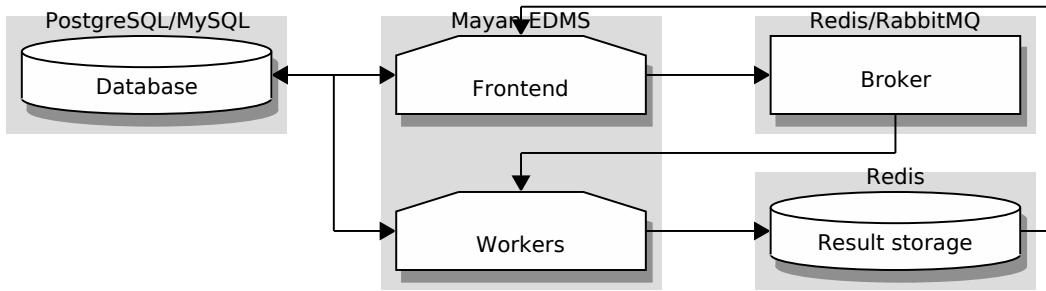


Fig. 2: Major block of functionality in Mayan EDMS's design.

How to think about Mayan EDMS when doing changes or adding new features; why things are the way they are in Mayan EDMS?

- Functionality must be as market/sector independent as possible, code for the 95% of use cases.
- Each user must be able to configure and customize it to their needs after install.
- Abstract as much as possible, each app must be an expert in just one thing, for other things they should use the API/classes/functions of other apps.
- Assume as little as possible about anything outside the project (hardware, OS, storage).
- Provide Python based abstraction so that a default install runs with a single step.
- No hard dependencies on binaries unless there is no other choice.
- Provide “drivers” or switchable backends to allow users to fine tune the installation.
- Call to binaries only when there is no other choice or the Python choices are not viable/mature/efficient.
- Each app is as independent and self contained as possible. Exceptions, the basic requirements: navigation, permissions, common, main.
- If an app is meant to be used by more than one other app, it should be as generic as possible in regard to the project and another app will bridge the functionality.
 - Example: since indexing (document_indexing) only applies to documents, the app is specialized and depends on the documents app.

33.1 Follow PEP8

Whenever possible, but don't obsess over things like line length:

```
$ flake8 --ignore=E501,E128,E122 | less
```

To perform automatic PEP8 checks, install flake8's git hook using:

```
$ flake8 --install-hook git
```

33.2 Imports

Import order should be:

- Standard Python modules
- Installed Python modules
- Core Django modules
- Installed Django modules
- Mayan EDMS modules
- Local imports

Example:

```
from __future__ import absolute_import

# Standard Python library
import base64

# 3rd party installed Python libraries
import requests

# Django core modules
from django.db.models import Q
from django.template.defaultfilters import slugify
from django.utils.translation import ugettext
from django.utils.translation import ugettext_lazy as _

# 3rd party installed Django libraries
from rest_framework import APIView

# Mayan apps
from metadata.classes import MetadataClass

# Local app imports (relative)
from .conf.settings import (
    AVAILABLE_INDEXING_FUNCTIONS,
    MAX_SUFFIX_COUNT, SLUGIFY_PATHS
)
from .exceptions import MaxSuffixCountReached
from .filesystem import (
    fs_create_index_directory, fs_create_document_link,
    fs_delete_document_link, fs_delete_index_directory,
    assemble_suffixed_filename
)
from .models import Index, IndexInstanceNode, DocumentRenameCount
```

All local app module imports are in relative form. Local app module name is to be referenced as little as possible, unless required by a specific feature, trick, restriction (e.g., Runtime modification of the module's attributes).

Incorrect:

```
# documents app views.py model
from documents.models import Document
```

Correct:

```
# documents app views.py model
from .models import Document
```

33.3 Dependencies

Mayan EDMS apps follow a hierarchical model of dependency. Apps import from their parents or siblings, never from their children. Think plugins. A parent app must never assume anything about a possible existing child app. The documents app and the Document model are the basic entities; they must never import anything else. The common and main apps are the base apps.

33.4 Variables

Naming of variables should follow a Major to Minor convention, usually including the purpose of the variable as the first piece of the name, using underscores as spaces. camelCase is not used in Mayan EDMS.

Examples:

Links:

```
link_document_page_transformation_list = ...
link_document_page_transformation_create = ...
link_document_page_transformation_edit = ...
link_document_page_transformation_delete = ...
```

Constants:

```
PERMISSION_SMART_LINK_VIEW = ...
PERMISSION_SMART_LINK_CREATE = ...
PERMISSION_SMART_LINK_DELETE = ...
PERMISSION_SMART_LINK_EDIT = ...
```

Classes:

```
class Document(models.Model):
class DocumentPage(models.Model):
class DocumentPageTransformation(models.Model):
class DocumentType(models.Model):
class DocumentTypeFilename(models.Model):
```

33.5 Strings

Quotation character used in Mayan EDMS for strings is the single quote. Double quote is used for multiple line comments or HTML markup.

33.6 Migrations

Migrations should do only one thing (example: either create a table, move data to a new table or remove an old table) to aid retrying on failure.

33.7 General

Code should appear in their modules in alphabetical order or in their order of importance, if it makes more sense for the specific application. This makes visual scanning easier on modules with a large number of imports, views or classes. Class methods that return a value should be pretended with a `get_` to differentiate from an object's properties. When a variable refers to a file it should be named as follows:

- filename: The file's name and extension only.
- filepath: The entire path to the file including the filename.
- path: A path to a directory.

Flash messages should end with a period as applicable for the language. Only exception is when the tail of the message contains an exceptions message as passed directly from the exception object.

33.7.1 Steps to deploy a development version

```
$ git clone https://gitlab.com/mayan-edms/mayan-edms.git
$ cd mayan-edms
$ git checkout development
$ virtualenv venv
$ source venv/bin/activate
$ pip install -r requirements.txt
$ ./manage.py initialsetup
$ ./manage.py runserver
```

33.7.2 Contributing changes

Follow the latest contributing guidelines outlined here: <https://gitlab.com/mayan-edms/mayan-edms/blob/master/CONTRIBUTING.md>

33.7.3 Debugging

Mayan EDMS makes extensive use of Django's new logging capabilities¹⁰.

By default debug logging for all apps is turned on. If you wish to customize how logging is managed turn off automatic logging by setting `COMMON_AUTO_LOGGING` to `False` and add the following lines to your `settings/local.py` file:

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': True,
    'formatters': {
        'verbose': {
            'format': '%(levelname)s %(asctime)s %(name)s %(process)d %(thread)d
%(message)s'
        },
        'intermediate': {
            'format': '%(name)s <% (process)d> [% (levelname)s] "% (funcName)s ()"
%(message)s"'
        }
    }
}
```

(continues on next page)

¹⁰ <https://docs.djangoproject.com/en/1.11/topics/logging/>

```

},
'simple': {
    'format': '%(levelname)s %(message)s'
},
},
'handlers': {
    'console':{
        'level':'DEBUG',
        'class':'logging.StreamHandler',
        'formatter': 'intermediate'
    }
},
'loggers': {
    'documents': {
        'handlers':['console'],
        'propagate': True,
        'level':'DEBUG',
    },
    'common': {
        'handlers':['console'],
        'propagate': True,
        'level':'DEBUG',
    },
},
}
}

```

Likewise, to see the debug output of the `tags` app, just add the following inside the `loggers` block:

```

'tags': {
    'handlers':['console'],
    'propagate': True,
    'level':'DEBUG',
},

```

Mayan EDMS apps are essentially Django app with some extra code to register navigation, permissions and other relationships.

33.8 App modules

- `__init__.py`

Should be empty if possible. No initialization code should be here, use the `ready()` method of the `MayanAppConfig` class in the `apps.py` module.

- `admin.py`

Standard Django app module to define how models are to be presented in the admin interface.

- `api_views.py`

REST API views go here. Mayan EDMS uses Django REST Framework API view classes.

- `apps.py`

Contains the `MayanAppConfig` subclass as required by Django 1.7 and up. This is a place to define the app name and translatable verbose name as well as code to be executed when the modules of the app are ready.

-
- **classes.py**
Hold python classes to be used internally or externally. Any class defined by the app that is not a model.
 - **events.py**
Define event class instances that are later committed to a log by custom code.
 - **exceptions.py**
Custom exceptions defined by the app.
 - **fields.py**
Place any custom form field classed you define here.
 - **forms.py**
Standard Django app module that hold custom form classes.
 - **handlers.py**
Contains the signal handlers, functions that will process a given signal emitted from this or other apps. Connect the handler functions to the corresponding signal in the ready() method of the MayanAppConfig subclass in apps.py
 - **html_widgets.py**
Classes to render an HTML widget. HTML widget are not the same as Django's native form widgets. Form widgets only work as part of a form field. HTML widgets are for use outside of forms, such as in a table cell.
 - **licenses.py**
This module outlines the license text of the third party content used in the app. It could be other Python libraries, JavaScript libraries, etc.
 - **links.py**
Defines the links to be used by the app. Import only from the navigation app and the local permissions.py file.
 - **literals.py**
Stores magic numbers, module choices (if static), settings defaults, and constants. Should contain all capital case variables. Must not import from any other module.
 - **managers.py**
Standard Django app module that hold custom model managers. These act as model class method to performs actions in a series of model instances or utilitarian actions on external models instances.
 - **models.py**
Standard Django app module that defines ORM persistent data schema.
 - **permissions.py**
Defines the permissions to be used to validate user access by links and views. Imports only from the permissions app. Link or view conditions such as testing for is_staff or is_superuser flag are defined in this same module.
 - **runtime.py**
Use this module when you need the same instance of a class for the entire app. This module acts as a shared memory space for the other modules of the app or other apps.
 - **search.py**
Search model definitions. Define which field of the app's models are searchable.

-
- serializers.py
Hold Django REST Framework serializers used by the api_views.py module.
 - settings.py
Define the configuration settings instances that the app will use.
 - signals.py
Any custom defined signal goes here.
 - statistics.py
Provides functions that will compute any sort of statistical information on the app's data.
 - tasks.py
Code to be execute in the background or as an out-of-process action.
 - tests/ directory
Hold test modules. There should be one test_*.py module for each aspect being tested, examples: test_api.py, test_views.py, test_parsers.py, test_permissions.py Any shared constant data used by the tests should be added to tests/literals.py
 - utils.py
Holds utilitarian code that doesn't fit on any other app module or that is used by several modules in the app. Anything used internally by the app that is not a class or a literal (should be as little as possible)
 - widgets.py
Custom form widgets go here. This should be the only place with presentation directives in the app (aside the templates).

33.9 Views

The module common.generics provides custom generic class based views to be used. The basic views used to create, edit, view and delete objects in Mayan EDMS are: SingleObjectCreateView, SingleObjectDetailView, SingleObjectEditView, and SingleObjectListView

These views handle aspects relating to view permissions, object permissions, post action redirection and template context generation.

Mayan EDMS source is controlled with Git¹¹.

The project is publicly accessible, hosted and can be cloned from **GitLab** using:

```
$ git clone https://gitlab.com/mayan-edms/mayan-edms.git
```

33.10 Git branch structure

Mayan EDMS follows a simplified model layout based on Vincent Driessen's Successful Git Branching Model¹² blog post.

¹¹ <http://git-scm.org>

¹² <http://nvie.com/posts/a-successful-git-branching-model/>

/versions/micro Working branch for the next bugfix release. Micro increment (third digit). Only bug fixes, minor features, back-ported urgent features. This branch is stable and safe for production.

/versions/minor Working branch for the next minor release (second digit). New features, occasional breakage. Not for production but should run in test environment most of the time. This is the branch you will want to try out if you want to check out new features.

/versions/major Working branch for the next major release (first digit). New features, incompatible changes to the user facing interfaces. Broken most of the time, not for production and should only be cloned by developers with experience with Mayan's development.

master Current production release (). Points to the latest version of the latest series. Production quality code.

features/ Working branches for unfinished and unmerged feature. Likely unstable, don't use in production. Once the feature is complete, it is merged into one of the versions branches and deleted.

Special branches:

releases/all Pushing code to this branch will trigger the build and release a new Docker image, Documentation and Python package.

releases/docker Pushing code to this branch will trigger the build and release of a new Docker image to Docker Hub.

releases/documentation Pushing code to this branch will trigger the build and release of new documentation.

releases/python Pushing code to this branch will trigger the build and release of a new Python package to PyPI.

nightly Pushing code to this branch will trigger the build and release of a new Docker image based on development code to the GitLab image repository only. The image will not be published to Docker Hub.

Each release is tagged separately using annotated Git tags.

When submitting patches, please place your code in its own **features/** branch prior to opening a Merge Request on GitLab¹³.

33.11 Commit messages

1. Use English as the language for the commit messages.
2. Provide a subject line composed of a tag and a short explanation:

Indexing: Add document base property reindex

3. Keep the subject line to 50 or less characters.
4. Capitalize the subject line.
5. Don't end the subject line with a period, leave like a phrase in English.
6. Use active voice in the. Say what the commit will do when applied not what you did:

Add new properties to the model.

Vs.

Added new properties to the model.

7. Limit the body of the commit to 72 characters.

¹³ <https://gitlab.com/mayan-edms/mayan-edms>

-
8. When a commit fixes, or improves an issue, add the issue number in the commit message. Either in the subject or in the body.
 9. Sign commit messages.
 10. Use explicit language even for minor commits. Don't do:

```
Fix typo
```

Use:

```
Document: Fix typo in label description
```

33.12 Source file package

This is the sequence of steps used to produce an installable package:

1. Generate the packaged version (will produce dist/mayan-edms-x.y.z.tar.gz):

```
$ make sdist
```

2. Do a test install:

```
$ cd /tmp
$ virtualenv venv
$ source venv/bin/activate
$ pip install <path of the Git repository>/dist/mayan-edms-x.y.z.tar.gz
$ mayan-edms.py initialsetup
$ mayan-edms.py runserver
```

33.13 Wheel package

1. Install the development requirements:

```
$ pip install -r requirements/development.txt
```

2. Create wheel package using the makefile:

```
$ make wheel
```

3. Do a test install:

```
$ cd /tmp
$ virtualenv venv
$ source venv/bin/activate
$ pip install <path of the Git repository>/dist/mayan_edms-x.y.z-py2-none-any.whl
$ mayan-edms.py initialsetup
$ mayan-edms.py runserver
```

33.13.1 Version numbering

Mayan EDMS uses the Semantic Versioning (<http://semver.org/>) method to choose version numbers along with Python's PEP-0440 (<https://www.python.org/dev/peps/pep-0440/>) to format them.

X.YaN # Alpha release X.YbN # Beta release X.YrcN # Release Candidate X.Y # Final release

33.13.2 Release checklist

1. Check for missing migrations:

```
make check-missing-migrations
```

2. Synchronize translations:

```
make translations-pull
```

3. Compile translations:

```
make translations-compile
```

4. Update changelog.

5. Write release notes.

6. Scan the code with the flake8 program for simple style warnings.

7. Check README.rst format with:

```
python setup.py check -r -s
```

or with:

```
make check-readme
```

8. Bump version in mayan/__init__.py and docker/version:

```
make increase-version PART=<major, minor or micro>
```

1. Update requirements version in setup.py using:

```
make generate-setup
```

2. Build source package and test:

```
make test-sdist-via-docker-ubuntu
```

3. Build wheel package and test:

```
make test-wheel-via-docker-ubuntu
```

4. Tag version:

```
git tag -a vX.Y.Z -m "Version X.Y.Z"
```

5. Generate set setup.py again to update the build number:

```
make generate-setup
```

6. Commit the new `setup.py` file.
7. Release the version using one of the two following methods: GitLab CI or manual

33.14 Release using GitLab CI

1. Switch to the `releases/all` branch and merge the latest changes:

```
git checkout releases/all  
git merge versions/next
```

2. Push code to trigger builds:

```
git push
```

3. Push tag upstream:

```
git push --tags
```

33.15 Manual release

1. Build and upload a test release:

```
make release-test-via-docker-ubuntu
```

2. Build and upload a final release:

```
make release-via-docker-ubuntu
```

The documentation is written in reStructured Text¹⁴ format, processed with Sphinx¹⁵, and resides in the `docs` directory. In order to build it, you will first need to install the documentation editing dependencies with:

```
$ pip install -r requirements/documentation.txt
```

Then, to build an HTML version of the documentation, run the following command from the `docs` directory:

```
$ make docs-serve
```

The generated documentation can be viewed by browsing to `http://127.0.0.1:8000` or by browsing to the `docs/_build/html` directory.

You can also generate the documentation in formats other than HTML. Consult the Sphinx¹⁶ documentation for more details.

Translations are handled online via the **Transifex** website: <https://www.transifex.com/projects/p/mayan-edms/>. To create a translation team for a new language or contribute to an already existing language translation, create a **Transifex** account and contact the team coordinator of the respective language in which you are interested.

¹⁴ <http://docutils.sourceforge.net/rst.html>

¹⁵ <http://sphinx.pocoo.org>

¹⁶ <http://sphinx.pocoo.org>

Feel free to open translation issues inside **Transifex** itself if you have a question about the usage or meaning of a source text string. If you open a translation issue, it will be your responsibility to close it after you get an answers that satisfies your question. Administrator will not close new issues as they have no way to determine if your question has been properly answered. However to avoid clutter, answered questions will be scanned periodically and closed if no activity is observed from the original poster in a period of time.

33.16 Docker image

33.16.1 Building the image

Clone the repository with:

```
git clone https://gitlab.com/mayan-edms/mayan-edms.git
```

Change to the directory of the cloned repository:

```
cd mayan-edms
```

Execute Docker's build command using the provided makefile:

```
make docker-build
```

Or using an apt cache to speed up the build:

```
make docker-build-with-proxy APT_PROXY=172.17.0.1:3142
```

Replace the IP address *172.17.0.1* with the IP address of the computer running the APT proxy and caching service.

NOTES

neeraj76@yahoo.com

MERCS

34.1 MERC 1: Purpose and Guidelines

MERC 1

Author Michael Price

Status Accepted

Type Process

Created 2018-02-17

Last-Modified 2018-02-17

Table of Contents

- *What is a MERC?* (page 319)
- *MERC Types* (page 320)
- *MERC submission workflow* (page 320)
 - *Pre-proposal* (page 320)
 - *Submitting the draft* (page 320)
 - *Implementation* (page 320)
- *MERC format* (page 320)
 - *MERC Metadata* (page 321)
 - *Auxiliary Files* (page 322)
- *Reporting MERC Bugs, or Submitting MERC Updates* (page 322)

34.1.1 What is a MERC?

A Mayan EDMS Request For Comment document (MERC) document is a design document providing information to the Mayan EDMS community, or describing a new feature or process for Mayan EDMS. MERCs provide concise technical specifications of features, along with rationales.

34.1.2 MERC Types

There are three kinds of MERCs:

1. A **Feature** MERC describes a new feature or implementation for Mayan EDMS. Most MERCs will be Feature MERCs.
2. An **Informational** MERC describes a Mayan EDMS design issue, or provides general guidelines or information to the Mayan EDMS community, but does not propose a new feature. Informational MERCs do not necessarily represent a community consensus or recommendation, so users and implementers are free to ignore Informational MERCs or follow their advice.
3. A **Process** MERC describes a process surrounding Mayan EDMS, or proposes a change to (or an event in) a process. Process MERCs are like Feature MERCs but apply to areas other than the Mayan EDMS framework itself. They may propose an implementation, but not to Mayan EDMS's codebase; they often require community consensus; unlike Informational MERCs, they are more than recommendations, and users are typically not free to ignore them. Examples include procedures, guidelines, changes to the decision-making process, and changes to the tools or environment used in Mayan EDMS development. Any meta-MERC is also considered a Process MERC. (So this document is a Process MERC).

34.1.3 MERC submission workflow

34.1.3.1 Pre-proposal

The MERC process begins with a new idea for Mayan EDMS. It is highly recommended that a single MERC contain a single key proposal or new idea. Small enhancements or patches usually don't need a MERC and follow Mayan EDMS's normal contribution process.

MERCs should be focused on a single topic. If in doubt, split your MERC into several well-focused ones.

Once the idea's been vetted, a draft MERC should be presented to the Mayan EDMS mailing list. This gives the author a chance to flesh out the draft MERC to make sure it's properly formatted, of high quality, and to address initial concerns about the proposal.

The Core Developers will be responsible for accepting, or rejecting, the MERC proposal.

34.1.3.2 Submitting the draft

Following the discussion on Mayan EDMS mailing list, the proposal should be sent as a merge request to the Mayan EDMS repository. The draft must be written in MERC style; if it isn't the merge request may be rejected until proper formatting rules are followed.

34.1.3.3 Implementation

Finally, once a MERC has been accepted, the implementation must be completed. In many cases some (or all) implementation will actually happen during the MERC process: Feature MERCs will often have fairly complete implementations before being reviewed. When the implementation is complete and incorporated into the main source code repository, the status will be changed to "Final".

34.1.4 MERC format

MERCs need to follow a common format and outline; this section describes that format.

MERCs must be written in reStructuredText¹⁷ (the same format as Mayan EDMS's documentation).

Each MERC should have the following parts:

1. A short descriptive title (e.g. “User document filters”), which is also reflected in the MERC’s filename (e.g. `0002-user-document-filters.rst`).
2. A preamble – a rST field list¹⁸ containing metadata about the MERC, including the MERC number and so forth. See *MERC Metadata* (page 321) below for specific details.
3. Abstract – a short (~200 word) description of the technical issue being addressed.
4. Specification – The technical specification should describe the syntax and semantics of any new feature. The specification should be detailed enough to allow implementation – that is, developers other than the author should, given the right experience, be able to independently implement the feature given only the MERC.
5. Motivation – The motivation is critical for MERCs that want to add substantial new features or materially refactor existing ones. It should clearly explain why the existing solutions are inadequate to address the problem that the MERC solves. MERC submissions without sufficient motivation may be rejected outright.
6. Rationale – The rationale fleshes out the specification by describing what motivated the design and why particular design decisions were made. It should describe alternate designs that were considered and related work. The rationale should provide evidence of consensus within the community and discuss important objections or concerns raised during discussion.
7. Backwards Compatibility – All MERCs that introduce backwards incompatibilities must include a section describing these incompatibilities and their severity. The MERC must explain how the author proposes to deal with these incompatibilities. MERC submissions without a sufficient backwards compatibility treatise may be rejected outright.
8. Reference Implementation – The reference implementation must be completed before any MERC is given status “Final”, but it need not be completed before the MERC is accepted. While there is merit to the approach of reaching consensus on the specification and rationale before writing code, the principle of “rough consensus and running code” is still useful when it comes to resolving many discussions of API details.

The final implementation must include tests and documentation, per Mayan EDMS development guide.

34.1.4.1 MERC Metadata

Each MERC must begin with some metadata given as an rST field list¹⁹. The headers must contain the following fields:

MERC The MERC number. In an initial merge request, this can be left out or given as XXXX; the reviewer who merges the pull request will assign the MERC number.

Type Feature, Informational, or Process

Status Draft, Accepted, Rejected, Withdrawn, Final, or Superseded

Created Original creation date of the MERC (in yyyy-mm-dd format)

Last-Modified Date the MERC was last modified (in yyyy-mm-dd format)

Author The MERC’s author(s).

Implementation-Team The person/people who have committed to implementing this MERC

Requires If this MERC depends on another MERC being implemented first, this should be a link to the required MERC.

¹⁷ <http://docutils.sourceforge.net/rst.html>

¹⁸ <http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#field-lists>

¹⁹ <http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#field-lists>

Mayan EDMS-Version (optional) For Feature MERCs, the version of Mayan EDMS (e.g. 2.7.3) that this feature will be released in.

Replaces and Superseded-By (optional) These fields indicate that a MERC has been rendered obsolete. The newer MERC must have a `Replaces` header containing the number of the MERC that it rendered obsolete; the older MERC has a `Superseded-By` header pointing to the newer MERC.

Resolution (optional) For MERCs that have been decided upon, this can be a link to the final rationale for acceptance/rejection. It's also reasonable to simply update the MERC with a "Resolution" section, in which case this header can be left out.

34.1.4.2 Auxiliary Files

MERCs may include auxiliary files such as diagrams. Such files must be named `XXXX-descriptive-title.ext`, where "XXXX" is the MERC number, "descriptive-title" is a short slug indicating what the file contains, and "ext" is replaced by the actual file extension (e.g. "png").

34.1.5 Reporting MERC Bugs, or Submitting MERC Updates

How you report a bug, or submit a MERC update depends on several factors, such as the maturity of the MERC, the preferences of the MERC author, and the nature of your comments. For the early draft stages of the MERC, it's probably best to send your comments and changes directly to the MERC author. For more mature, or finished MERCs you can submit corrections as repository issues or merge requests against the git repository.

When in doubt about where to send your changes, please check first with the MERC author and/or a core developer.

MERC authors with git push privileges for the MERC repository can update the MERCs themselves.

34.2 MERC 2: Test writing

MERC 2

Author Michael Price

Status Accepted

Type Feature

Created 2018-02-22

Last-Modified 2018-04-01

Table of Contents

- *Abstract* (page 322)
- *Motivation* (page 323)
- *Specification* (page 323)

34.2.1 Abstract

This MERC proposes a standard methodology for writing tests for Mayan EDMS.

34.2.2 Motivation

Having a standard methodology for writing tests has the following advantages:

1. Scaffolding can be reduced by providing the most frequently used paradigms as methods or helper functions.
2. Reduce the probabilities of errors slipping through poorly written tests.

34.2.3 Specification

1. Tests must test each view in at least two ways:
 - A. Object creation views must be tested with and without permissions.
 - B. Object detail, list and delete views must be tested with and without object access.
2. Tests must assert the status code of the response even when the expected status is HTTP 200.
3. The actual request performed must be enclosed in a private methods so that the fail and pass tests use the same HTTP request.
4. Tests must verify that changes happened and didn't happen in the database, regardless of the return code. Even if an edit view returns an error 4XX (404-Not found, 403-Forbidden, etc), the test must ensure that the data was not indeed modified.
5. All tests must use the test user created by the BaseAPITestCase and not a super user unless absolutely required by the test.
6. Each test, must test just one thing.
7. If a test object needs to be created before the execution of a request this object must be created by a private method.

Example:

```
def _request_tag_create(self):
    return self.post(
        viewname='rest_api:tag-list', data={
            'label': TEST_TAG_LABEL, 'color': TEST_TAG_COLOR
        }
    )

def test_tag_create_view_no_permission(self):
    response = self._request_tag_create()
    self.assertEqual(response.status_code, status.HTTP_403_FORBIDDEN)
    self.assertEqual(Tag.objects.count(), 0)

def test_tag_create_view_with_permission(self):
    self.grant_permission(permission=permission_tag_create)
    response = self._request_tag_create()
    self.assertEqual(response.status_code, status.HTTP_201_CREATED)

    tag = Tag.objects.first()
    self.assertEqual(response.data['id'], tag.pk)
    self.assertEqual(response.data['label'], TEST_TAG_LABEL)
    self.assertEqual(response.data['color'], TEST_TAG_COLOR)

    self.assertEqual(Tag.objects.count(), 1)
    self.assertEqual(tag.label, TEST_TAG_LABEL)
    self.assertEqual(tag.color, TEST_TAG_COLOR)
```

34.3 MERC 3: Using JavaScript libraries

MERC 3

Author Eric Riggs

Status Accepted

Type Feature

Created 2018-03-08

Last-Modified 2018-06-04

Table of Contents

- *Abstract* (page 324)
- *Rationale* (page 324)
- *Motivation* (page 324)
- *Backwards Compatibility* (page 325)
- *Specification* (page 325)

34.3.1 Abstract

This MERC proposes a standard way to use JavaScript libraries.

34.3.2 Rationale

Mayan EDMS uses several JavaScript libraries for user interface features. Currently, the libraries are not installed using any JavaScript package manager but copied uncompressed. Installing the libraries in this manner carries some disadvantages.

34.3.3 Motivation

The inclusion of the libraries in source form is required by many licenses if the library is not installed by a package manager in distributable form. There are several disadvantages with the current approach:

1. Having the library in source form means that the entire weight of the library's size carries over the overall size of the Mayan EDMS distribution files. The justification for not doing this is the same as with the Python libraries which are not copied with the code but downloaded upon installation.
2. Upgrading the libraries means manually examining the version of the installed in the project and manually searching, downloading, compressing and adding the files to the repository.
3. The source form of the libraries includes normal and minified versions of the code and the accompanying CSS files. There is no define preference and through the project both versions of the libraries are loaded interchangeably. Using a packager manager the minified version would be used of a pipeline to minify the installed libraries should be added.

34.3.4 Backwards Compatibility

There are no backwards compatibility issues with this proposal.

34.3.5 Specification

Changes needed:

1. Python based JavaScript package manager. Alternatively a Python wrapper for a JavaScript package manager could be used.
2. Package manifest for the JavaScript libraries used.
3. Installation pipeline to install the JavaScript libraries during the installation and setup of the project.

References:

- <https://github.com/JDeuce/powser>
- <https://github.com/javrasya/version-manager>
- <https://github.com/inveniosoftware-attic/setuptools-bower>
- <https://pypi.python.org/pypi/django-bower-cache/0.5.0>
- <http://django-pipeline.readthedocs.io/en/latest/index.html>
- <https://github.com/nvbn/django-bower>

34.4 MERC 4: Support forum

MERC 4

Author Michael Price

Status Accepted

Type Process

Created 2018-02-27

Last-Modified 2018-09-04

Table of Contents

- *Abstract* (page 325)
- *Motivation* (page 326)
- *Specification* (page 326)

34.4.1 Abstract

This MERC proposes the move of the official community meeting place from Google Group to a self hosted forum platform.

34.4.2 Motivation

Google Groups is not a proper forum platform and as such is lacking many features that would increase participation. Google Groups has not had any significant update in years and the age of the platform is visible. There are no official mobile apps for Google Groups, no responsible template is not provided. There are not integration options. These factors contribute to the problems of knowledge dilution and one time participation being experienced in the Mayan EDMS community.

Other concerns for moving the community to a self hosted forum solution are that Google Groups presents a single point of failure by relying on a free product hosted by a commercial company with no continuation plan for the product. Google Groups doesn't provide archiving features and the current archive solution relies on other third party services, one of which (GMANE) has stopped working. Recent user privacy and censorship issues regarding Google, reinforce the need for a self hosted solution.

34.4.3 Specification

Platform chosen was phpBB. Factors were: written in PHP, ease of installation, compatible with several database managers, mature, extensive development history.

34.5 MERC 5: Explicit arguments

MERC 5

Author Roberto Rosario

Status Accepted

Type Feature

Created 2018-12-30

Last-Modified 2018-12-31

Table of Contents

- *Abstract* (page 326)
- *Motivation* (page 327)
- *Specification* (page 327)
- *Backwards Compatibility* (page 327)
- *Reference Implementation* (page 327)

34.5.1 Abstract

This MERC proposes the adoption of a new methodology when performing calls. It seeks to reduce the use of positional arguments in favor of keyword arguments in as many places as possible.

34.5.2 Motivation

As the project grows, legibility of code becomes more important. Keyword arguments help document the use of services, classes and functions. Refactors that affect the interface of services are also easier to find and update and fix. Positional argument can cause a call to continue working as long as the datatype of the argument remains the same. Usage of keyword arguments will automatically raise an error that will prevent such situations. Keyword argument further eliminate the relevance of position or the arguments, and the arguments can be sorted alphabetically for easier visual scanning or by semantic significance improving code readability.

34.5.3 Specification

Adoption of this MERC will require an audit of existing calls and the use of the method proposed for new calls. Every call, regardless of the type or origin of the source callable, will name each argument used. By type it is meant: classes, functions, methods. Origin means: local from the project, from the framework, third party libraries or the standard library.

34.5.4 Backwards Compatibility

No backwards compatibility issues are expected. New errors arising from the use of keyword arguments could be interpreted as existing latent issues that have not been uncovered.

34.5.5 Reference Implementation

Example:

Before:

```
from mayan.apps.common.classes import Template

Template(
    'menu_main', 'appearance/menu_main.html'
)
```

After:

```
from mayan.apps.common.classes import Template

Template(
    name='menu_main', template_name='appearance/menu_main.html'
)
```

When calls use a mixture of positional and keyword arguments, the keyword arguments can only be found after the positional arguments. Complete use of keyword arguments allow the reposition of arguments for semantic purposes.

Example:

Before:

```
from django.conf.urls import url

from .views import AboutView, HomeView, RootView
```

(continues on next page)

(continued from previous page)

```
urlpatterns = [
    url(r'^$', RootView.as_view(), name='root'),
    url(r'^home/$', HomeView.as_view(), name='home'),
    url(r'^about/$', AboutView.as_view(), name='about_view'),
]
```

After:

```
from django.conf.urls import url

from .views import AboutView, HomeView, RootView

urlpatterns = [
    url(regex=r'^$', name='root', view=RootView.as_view()),
    url(regex=r'^home/$', name='home', view=HomeView.as_view()),
    url(regex=r'^about/$', name='about_view', view=AboutView.as_view()),
]
```

Keyword arguments should also be used for callables that pass those to others down the line like Django's `reverse` function. Any change to the name of the `pk` URL parameter will raise an exception in this code alerting to any possible incompatible use.

Example:

```
def get_absolute_url(self):
    return reverse(
        viewname='documents:document_preview', kwargs={'pk': self.pk}
    )
```

This becomes even more important when multiple URL parameters are used. Since the API documentation is auto generated from the code itself, it would make sense to rename the first URL parameter from `pk` to `document_pk`. Such change will cause all addresses to view resolutions to break, forcing their update and allowing all consumers' interface usage to remain synchronized to the callable's interface.

```
url(
    regex=r'^documents/(?P<pk>[0-9]+)/versions/(?P<document_version_pk>[0-9]+)/pages/
    ↳ (?P<document_page_pk>[0-9]+)/image/$',
    name='documentpage-image', view=APIDocumentPageImageView.as_view()
),
```

34.6 MERC 6: Lower information disclose

MERC 6

Author Michael Price

Status Accepted

Type Feature

Created 2018-12-30

Last-Modified 2018-12-31

Table of Contents

- *Abstract* (page 329)
- *Motivation* (page 329)
- *Specification* (page 329)

34.6.1 Abstract

This MERC proposes the use of errors that don't disclose the existence of a resource in the event that the requester doesn't have the required credentials.

34.6.2 Motivation

When a user tries to perform an action like opening a view to a document for which the required permission is missing, a permission required or access denied error is presented. This is semantically correct, but from the stand point of security it is still failing because it is letting the user know that such document exists in the first place. This MERC proposes changing the error message for existing resource to one that doesn't divulge any information to unauthorized parties, like "Not Found".

34.6.3 Specification

Out of the 4 basic CRUD operations, Read, Update and Delete should return an HTTP 404 error instead of an HTTP 403 error. Only the Create operation will continue returning the current HTTP 403 error, unless it is creating a new resource that is related to an existing resource.

Since most view use the internal custom CRUD classes making a change to the `ObjectPermissionCheckMixin` class to raise an HTTP 404 on object access failure will fulfill the proposal of this MERC.

Adding the `object_permission_raise_404` class attribute and setting it to default to `False` will allow fulfilling the goal of this MERC, while keeping the existing functionality intact.

Example:

```
class ObjectPermissionCheckMixin(object):
    """
    If object_permission_raise_404 is True an HTTP 404 error will be raised
    instead of the normal 403.
    """
    object_permission = None
    object_permission_raise_404 = False

    def get_permission_object(self):
        return self.get_object()

    def dispatch(self, request, *args, **kwargs):
        if self.object_permission:
            try:
                AccessControlList.objects.check_access(
                    permissions=self.object_permission, user=request.user,
                    obj=self.get_permission_object(),
```

(continues on next page)

(continued from previous page)

```
        related=getattr(self, 'object_permission_related', None)
    )
except PermissionDenied:
    if self.object_permission_raise_404:
        raise Http404
    else:
        raise

return super(
    ObjectPermissionCheckMixin, self
).dispatch(request, *args, **kwargs)
```

34.7 MERC XX: Unify Roles and Groups

MERC XX

Author Michael Price

Status Draft

Type Feature

Created 2018-02-27

Last-Modified 2018-02-27

Table of Contents

- *Abstract* (page 330)
- *Rationale* (page 330)
- *Motivation* (page 331)
- *Backwards Compatibility* (page 331)
- *Specification* (page 331)

34.7.1 Abstract

This MERC proposes the merging of the Roles and Group models.

34.7.2 Rationale

Mayan EDMS uses Groups as units of users that are meant to mirror an organization's actual user hierarchy. Roles are used as permission units.

Separation of concerns is a concept Mayan EDMS executes very successfully but when it comes to the Roles/Groups relationship that execution causes overheads without providing advantages in the day to day operations.

In reality there is almost a 1 to 1 correlation between Roles and Groups. Other permissions systems already use Groups as permission units without disadvantages. An example of this is LDAP and its commercial counterpart Active Directory.

34.7.3 Motivation

Merging the Role and Group model will reduce some complexity when initially setting up Mayan EDMS. The merge allows removing a Mayan EDMS model in favor of using a native Django model for the same task.

Merging the Role and Group models will also provide a speed boost in every permission check and queryset filtering. These checks are nested in nature. Since the access checks are performed for every view and for every link in the view the performance gain should be substantial.

34.7.4 Backwards Compatibility

To avoid loss of role configuration a data migration will be needed to convert existing roles to groups.

34.7.5 Specification

Changes needed:

1. Data migration to convert existing roles to groups.
2. Prefix or append an identifier to the migrated roles.
3. Intermediate model to map permissions to a group. This will substitute the Role model's permissions many to many field.
4. Update the `AccessControlList` models roles field to point to the group models.
5. Update the role checks in the `check_access` and `restrict_queryset` `AccessControlList` model manager methods.

neeraj76@yahoo.com

PART IX.
APPENDIX

GLOSSARY

- Debian** Debian is a free operating system (OS). Debian systems currently use the Linux kernel or the FreeBSD kernel. Homepage at: <https://www.debian.org/>
- Django** Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It is designed to reduce the work of creating a Web app. It's free and open source. Homepage at: <https://www.djangoproject.com/>
- Docker** Docker is a container technology. Containers are a standard unit of software that packages up code and all its dependencies. Independent containers can run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines (VMs). For more information visit <https://www.docker.com/>
- GPG** GnuPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880 (also known as PGP). GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories. GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications. A wealth of frontend applications and libraries are available. GnuPG also provides support for S/MIME and Secure Shell (ssh). <https://gnupg.org/>
- IMAP** IMAP stands for the Internet Message Access Protocol (IMAP). It is an Internet standard protocol used by email clients to retrieve email messages from a mail server over a TCP/IP connection. IMAP is defined by RFC 3501. Unlike POP3, email messages in IMAP are synchronized and remain in the client as well as in the server.
- Linux** Linux is a piece of software started by Linus Torvalds in 1991 and now also supported by thousands of programmers worldwide. Linux is a kernel, the center piece of an operating system. It takes care of running programs and managing devices. It needs to be paired with system and user programs to become a complete operating system.
- NFS** NFS stands for Network File System. It is a protocol that allows users remote access to files over a network as if they were local. The current version of NFS is NFSv4.
- POP3** The Post Office Protocol (POP) is an application-layer Internet standard protocol used by e-mail clients to retrieve e-mail from a mail server. Unlike IMAP, messages in POP3 are downloaded and subsequently deleted from the server. POP version 3 (POP3) is the version in common use.
- PostgreSQL** PostgreSQL is an open source object-relational database system that uses and extends the SQL language. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 30 years of active development on the core platform.
- PyPI** The Python Package Index (PyPI) is a repository of software for the Python programming language. Package authors use PyPI to distribute their software.
- Python** Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991. Homepage: <https://www.python.org/>
- Redis** Redis stands for Remove Dictionary Server, is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams.
- Samba** Samba is the standard Windows interoperability suite of programs for Linux and Unix. Started in 1992, Samba provides file and print services for all clients using the SMB/CIFS protocol, such as all versions of DOS and Windows, OS/2, Linux and many others. Homepage: <https://www.samba.org/>

Supervisord Supervisor is a Process Control System. Composed of a client and a server system, it allows its users to monitor and control a number of processes on UNIX-like operating systems. It shares some of the same goals of programs like launchd, daemontools, and runit. Homepage: <https://supervisord.org/>

Ubuntu Ubuntu is a popular Linux distribution. It is based on Debian and comes preconfigured for ease of use and integration between the programs it includes. Homepage: <http://www.ubuntu.com/>

virtualenv virtualenv is a tool to create isolated Python environments. It allows for the parallel use of different Python projects, each with their own libraries and Python interpreter. Homepage: <https://www.virtualenv.org>

neeraj76@yahoo.com

INDEX

A

Access control, 107
ACLs, 111
API, 243
Authentication, 22
Automation, 182

B

Backup, 229
Blocking, 61
BPM, 175
Brave, 12
Business processes, 175
Business processes management, 175
Button, 15

C

cabinets, 81
Categorization
 automatic, 158
 manual, 79
Check in, 61
Checkouts, 61
checksum, 148
Chrome, 12
Collaboration, 56
Comments, 57
cryptographic signatures, 150

D

Debian, 334
Debian, 196
Deduplication, 154
Deleting documents, 71
Deletion policies, 95
Deployments, 11
Django, 334
Django, 196
Docker, 334
Docker, 193, 229
Document data, 115

Document data levels, 116
Document metadata, 130
Document parsing, 117
document signatures, 150
Document types, 29
Dropdown, 13

E

emails
 system, 226
 users, 66
Events, 135

F

Features, 11
file metadata, 126
Firefox, 12

G

GNUPG, 150
GPG, 334
GPG, 150
Groups, 104
Gunicorn, 199

I

IMAP, 334
IMAP, 33
Indexing, 159, 165
Installation, 191
Internet Explorer, 12
Introduction, 9

K

keys, 150

L

LibreOffice, 196
Linux, 334
Linux, 196
Login, 22

Logout, 22

M

Mailing, 66
Mailing profiles, 66
Main menu, 13
Mayan EDMS Request For Comment, 318
MERCs, 318
Message of the day, 75
Messages, 75
Metadata, 130
Mirroring, 165
Modular design, 12

N

NFS, 334
NFS, 165

O

OCR, 122
Open standards, 12
Opera, 12
Optical Character Recognition, 122

P

Parsing, 117
Password reset, 22
Permissions, 107, 111
POP3, 334
POP3, 33
Poppler, 196
PostgreSQL, 334
PostgreSQL, 196, 229
preparestatic, 198
PyPI, 334
Python, 334
Python, 196

R

RabbitMQ, 199
Redis, 334
Redis, 196
Restore, 229
Retention policies, 95
Roles, 107, 111

S

Samba, 334
Samba, 165
Search, 45
Settings, 202
Sidebar, 14
signatures, 150
Smart links, 169

Source code, 12
Sources, 33
Staging folders, 33
States, 175
Supervisord, 335
Supervisord, 196

T

tagging, 86
tags, 86
Transitions, 175

U

Ubuntu, 335
uniqueness, 145
Unique identifiers, 145
User accounts, 100
User groups, 104
user interface, 13
user management, 99
Users, 100
UUID, 145

V

Validation, 143
Versioning, 91
virtualenv, 335
virtualenv, 196

W

Watch folder, 33
Webform, 33
Workflows, 175