

Documentation for OpenKM 8.1

Table of Contents

Table of Contents 2
OpenKM 8.1 40
Hardware and software requirements 42
Requirements 42
Checking the Java version 42
For Windows users 43
Recommendations 43
Hardware configuration 43
Browser support 44
Operating systems 44
Small questionary 45
Third-party software requirements 47
Installation 48
Installing on Ubuntu and Debian 50
Preliminaries 50
Installing OpenKM 51
Post-installation checklist 52
Troubleshooting 52
Installing on RedHat and CentOS 53
Preliminaries 53
Installing OpenKM 54
Enable remote access 55
Post-installation checklist 55
Troubleshooting 55
Installing on Windows 56
Preliminaries 56
Checking Java version 56
Install Java 56
Installing database 57
Check InnoDB is the default MySQL engine 57
Create database 57
Installing Tomcat 57
Configuring Tomcat as a service 57
Check the service 62
From terminal 62
From the Windows service panel 62
From openkmw tool 62
Configuring environment variables 63
Installing OpenKM 65
Start application 66
Check openkm.properties parameters 66
Access from other computers 67
Other standard additional setup configurations 67
Using the installer 68
Supported databases 69
Other options 70
Using Docker 71
Configuration 72
Docker Compose 73
Installing on a pristine Tomcat 76
Preliminaries 76

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Installing Tomcat 76
Copy libraries and configuration files 76
Modify configuration files 76
Installing OpenKM 77
Install third-party software 77
Start application 77
Check openkm.properties parameters 77
Database 79
Configure database 79
Configuring MySQL and MariaDB 80
Preliminaries 80
Change default engine 80
Change default charset 80
Database creation 81
Create a database and user 81
Create a database in Persian 81
Configure your openkm.properties 82
MySQL Timezone 82
Configure Tomcat data sources 83
MySQL: 83
MariaDB: 83
Configure application login 83
Run application 83
Additional information 84
Remove and create MySQL service in Windows. 84
Migrate from Windows to Linux 84
Troubleshooting 84
Error ASCII '\\0' appeared in the statement" when importing a database dump. 84
MySQL tuning 86
Sample one 86
Sample two 87
Change MySQL data directory 89
Ubuntu 89
CentOS and RHEL 90
Windows 91
MySQL 8 troubleshooting 92
Could not create a connection to the database server 92
Strong Password Encryption for Authentication 92
Old JDBC driver 94
The server time zone value is unrecognized. 95
Problem connecting to your database 96
Configuring Oracle 97
Database creation 97
Create tablespace 97
Create a database and user 97
Configure your openkm.properties 97
Configure Tomcat data sources 97
Configure application login 98
Run application 98
Configuring PostgreSQL 100
Database creation 100
Create a database and user 100
Configure your openkm.properties 100
Configure Tomcat datasources 100
Configure application login 101
Enable native configuration parameter 101
Run application 101
Additional information 101

PostgreSQL 9.6 and upper 101
Configuring SQL Server 102
Database creation 102
Create a database and user 102
Configure your openkm.properties 103
Configure Tomcat data sources 104
Configure application login 104
Run application 104
Configure integrated security 105
Troubleshooting error 105
Can't connect to SQL Server 105
The connection was lost after some time. 106
Not able to login into OpenKM 106
Not able to connect TCP/ IP to localhostSQL, port 1433 when using instance in the JDBC connection URL 107
Deadlock issues 107
SSL connection error 107
The driver could not establish a secure connection to SQL Server by using Secure Sockets Layer (SSL) encryption 107
SQL Server alter compatibility level 108
Configuring Hikari pool 109
Enable debug 109
Backup and restore 110
Linux detailed steps 111
Backup 111
Restore 111
Windows detailed steps 112
Backup 112
Restore 112
Backup scripts 113
Additional information 113
Linux backup and restore with BorgBackup 114
Installation 114
Sample script 114
Backup to a remote computer with SSH 114
This is an improved SSH version 115
Backup to mapped network 115
Configure crontab 116
Recovery 117
Linux backup and restore with the rdiff-backup tool 118
Debian & Ubuntu 118
CentOS & RedHat 118
Preliminaries 118
Sample script 118
Increase the backup period 121
Do remote backup 121
Do a remote backup to SMB or CIFS 122
Do PostgreSQL backup 122
Configure crontab 122
Restoring backup 123
Additional information 123
Linux Sample Backup with Duplicity 125
Installation 125
Command line basics 125
Preliminaries 126
Sample script 126
Increase the backup period 129
Perform the remote backup 129
Perform the remote backup to SMB or CIFS 129
Do the PostgreSQL backup 130

Do the remote backup to Amazon s3 130
Configure crontab 131
Troubleshooting deleting old backups 131
Restoring backup 131
Additional information 132
Linux backup and restore with rsync 133
Preliminaries 133
Sample script 133
Do backup with rotation 135
Do remote backup 136
Do incremental backup with backup-dir 136
Do incremental backup with link-dest 137
Do remote backup to SMB or CIFS 137
Do PostgreSQL backup 137
Configure crontab 138
Rsync and FTP 138
Restoring backup 139
Additional information 139
Linux backup with LFTP 140
Preliminaries 140
Sample script 140
Configure crontab 142
Cluster configuration 143
Proxy node 145
Master node 146
Slave nodes 147
Sharding configuration 149
Sharding and Docker 150
Instance replication 152
Different implementations 153
Sample Linux implementation 153
How to move to another server 155
Linux 155
Preliminaries 155
Stop OpenKM service 155
Backup the database 155
Restore the database 155
Copy binary files 156
Configure service in the new server 156
Install third-party tools 156
Start OpenKM service 157
Additional steps 157
Moving license 157
Final checks 157
Setup 158
Antivirus configuration 159
Linux 159
Debian / Ubuntu 159
Red Hat / CentOS 159
Windows 160
Clamd Service Setup 161
Freshclam Virus Signature Auto Update 162
Cleaning older activity log data 164
MySQL 164
Linux 164
Configure crontab 164
Windows 165

Database replication with federated tables 165
Configuring Apache HTTP Reverse-Proxy 167
Internal IP vs external IP 167
Debian and Ubuntu 168
Check the configuration 170
Red Hat and CentOS 170
Check the configuration 171
Error Permission denied: proxy: AJP: attempt to connect 171
Windows 171
Register Apache service 172
Apache SSL configuration 172
Troubleshooting 173
Additional information 174
Configuring Apache HTTPS Reverse-Proxy 175
Internal IP vs external IP 175
Debian and Ubuntu 176
Redirect all HTTP connections to HTTPS 178
Check the configuration 178
Using Let's Encrypt 178
Red Hat and CentOS 179
Check the configuration 180
Error Permission denied: proxy: AJP: attempt to connect 180
Windows 180
Troubleshooting 181
Additional information 181
Configuring Nginx HTTP Reverse-Proxy 183
Internal IP vs external IP 183
Debian and Ubuntu 184
Check the configuration 185
Red Hat and CentOS 185
Check the configuration 187
Troubleshooting 187
Configuring Apache HTTP to rewrite old OpenKM URL 189
Configuring Lucene Analyzer 190
Configure an Analyzer 191
Configuring CADViewer 192
Troubleshooting 195
Client side (browser) 195
Server side (utilities) 196
Configuring OCR engine 197
Tesseract 197
Installation 197
Check installation 198
Configuration 198
Additional information 199
Dictionary 200
Configuration 200
OCR rotate configuration 200
Configuration 200
Additional information: 200
Configuring OpenSSL 202
Create OpenSSL configuration file 202
Create Certification Authority (CA) 203
Create Apache Certification 204
Apache configuration 204
Certification revocation 205

Add CRL to Apache 206
Configuring mail 207
Change mail configuration 207
Gmail example 207
Office 365 207
Email server with old SSL configuration 208
Additional information 208
Testing configuration 208
Usage 208
Configuring Tomcat memory utilization 210
OpenJDK 11 210
Configuring Tomcat port 211
Change tomcat ports 211
Configuring Tomcat session time-out 213
Configuring Tomcat timezone 214
Configuring openkm.xml 215
Enable external openkm.xml 215
Memory sample 215
Database authentication sample 216
Ldap Authentication sample 216
Multiple providers sample 217
Configuring jBPM 219
Configuring ONLYOFFICE 220
ONLYOFFICE installation 220
Configuration 220
ONLYOFFICE access 220
WOPI validator 221
OpenKM configuration 221
Configuring Collabora Online 223
Code installation 223
Code access 223
Code configuration 224
WOPI validator 224
OpenKM configuration 224
Parameters 225
Sample URLs 225
Other sample configuration 225
Configuring safe trash purge 226
Configuring search highlighter 227
Configuring Tomcat to work behind an Internet proxy 228
Linux 228
Windows 229
Step one 229
Step two 230
Configuring VidSigner 232
Change repository home 234
Understanding the contents of repository folders 234
Move entire repository location 235
Sample1 235
Windows 235
Linux 235
Move repository folders individually 235
Sample 236

Windows 236
Linux 236
Troubleshooting 236
Windows mapped network disk is not working 236
Change default role names 237
Change the default roles 237
Edit the openkm.properties file 237
Update default roles to new names in the database 237
Enable extensions 238
LDAP configuration 240
OpenLDAP 240
Active Directory 240
Uncategorised examples shared by users 240
Tools 240
OpenLDAP example with URL base at login 241
LDAP Structure 241
Step 1 - configuration parameters 242
Step 2 - configure login 244
Another option to configure login 244
Active Directory example with referral enabled 246
LDAP Structure 246
Step 1 - configuration parameters 247
Step 2 - configure login 249
Another option to configure login 249
Active Directory example with login based on filtering users by roles 251
LDAP structure 251
Step 1 - configuration parameters 252
Step 2 - configure login 254
Another option to configure login 255
Active Directory basic configuration 257
LDAP structure 257
Step 1 - configuration parameters 257
Step 2 - configure login 259
Another option to configure login 260
Active Directory advanced configuration 261
LDAP structure 261
Step 1 - configuration parameters 261
Step 2 - configure login 263
Another option to configure login 264
Active Directory mixed configuration 266
LDAP structure 266
Step 1 - configuration parameters 266
Step 1 - configure login 268
Active Directory mixed configuration with LDAP synced 270
LDAP structure 270
Step 1 - configuration parameters 270
Step 2 - Simplified login 272
Step 2 - configure login (old configuration) 272
LDAP best practices for filtering users and roles 275
Example based on single LDAP group 275
Example based on ROLE_USER and ROLE_ADMIN group 275
ClearOS LDAP configuration 277
LDAP Structure 277

Step 1 - configuration parameters 278
Step 2 - create a Certificate 280
Step 3 - configure login 281
Another option to configure login 281
LDAP troubleshooting 283
How to do a nested group membership filter in active directly 283
No UserDetailsService registered error 283
javax.naming.PartialResultException error 283
Can't connect to LDAP server 284
Slow login or not able to login 284
Force all users to be members of ROLE_USER. 285
org.springframework.beans.factory.parsing.BeanDefinitionParsingException 286
LDAP: error code 8 - BindSimple: Transport encryption required 287
Ldap error: Caused by java.security.cert.CertificateException: No subject alternative DNS name matching
Log Configuration 290288
Logback based Log Configuration 290
Third-party application execution 290
Enable log authentication 290
LibreOffice configuration 291
Linux 291
Debian & Ubuntu 291
Windows 291
Security configuration 293
Diferences betweem simple and recursive 294
Extending security 295
Configuration for security changes which take a long time 295
Configuring Two Factor Authentication 297
Compliance 297
Prerequisite 297
OpenKM configuration 297
Scan QR code 298
Google Authenticator use 298
Configuration 299
Register parameters 299
Configure parameters 299
Register metadata group 301
Crontab task 301
Register reports 302
Configuring CIFS 303
Database Authentication 303
OpenKM server 303
OpenKM Configuration 304
Windows client 305
Active Directory Authentication 306
Active Directory 306
OpenKM server in Active Directory configuration 307
Introduce server in domain 307
Redirect the required ports 308
kerberos configuration file 308
cifs.login.config File 309
Enable Java login configuration. 310
OpenKM Configuration in Active Directory Configuration 310
Windows client in Active Directory configuration 312
Cache configuration 314
Configuring digital email signature 315
More info 315

Configuring anonymous user 316
Prometheus configuration 317
Setup validation 317
Running Prometheus with Docker 317
Troubleshooting 321
Server configuration 322
The packet for the query is too large (MySQL) 322
Not able to connect from another computer in the network 322
Error downloading document as PDF 322
IMAP SSL certificate error 322
OpenKM server has no direct Internet connection. 323
Problems with character encodings 323
PermGen errors 323
LDAP Size limit exceeded 323
Behind the IIS proxy, you get error 404 or 404.15 324
IIS gets IOException: Internal Server Error while uploading file or HTTP 413 Request Entity Too Large Error 326
Solution one 326
Solution two 326
WARN net.sf.ehcache.hibernate.strategy.AbstractReadWriteEhcacheAccessStrategy 327
RedHat and Oracle Linux issues 328
OpenKM not starting as a service automatically 328
ImageMagick 329
Client configuration 331
ZIP filenames with wrong charset 331
Microsoft Office Addin disappears after Office restart 332
Best practices for reporting issues 333
Getting the stack trace 333
Linux 333
Windows 333
Sample of stack trace capture 334
Administration guide 335
Configuration parameters 339
Edit a parameter 339
Delete a parameter 339
Backup configuration parameters 339
New configuration parameter 340
Check Configuration 340
Multitenant configuration 340
Recommended configuration parameters 341
Preview parameters 344
Linux 344
Windows 345
LDAP configuration parameters 347
Sample query 347
LDAP queries 347
Parameters 348
Sample configuration 350
User interface configuration parameters 353
AI configuration parameters 356
Workflow configuration parameters 358

Notification configuration parameters 360
Substitutions 362
Document version numeration configuration 364
Version numeration schemas 364
Migration between versions 365
Password validation configuration 366
Sample configuration 367
Switch application to read-only mode 368
Security configuration parameters 369
Extensions configuration parameters 373
Extratab extension 373
External preview extension 373
External tab document extension 373
External tab folder extension 374
External tab mail extension 374
External tab record extension 374
Performance configuration parameters 375
Text extractor queue 375
Download files and folders as ZIP file 376
Antivirus checker 377
Re-indexing the whole Lucene repository 378
Execution timeout 379
Activity log actions 380
Dashboard activity 380
Lucene 381
Other configuration properties 382
LibreOffice configuration parameters 383
CSV configuration parameters 384
Antivirus configuration parameters 385
Other configuration parameters 387
Experimental configuration parameters 394
AutoCAD configuration parameters 398
FTP configuration parameters 399
Active Mode 399
Passive Mode 400
Two factor authentication configuration parameters 401
Multi-tenant configuration parameters 402
File plan 403
Enable the File Plan 403
Mandatory 403
Optional 403
How to construct the File Plan 403
Search configuration parameters 406
MIME types 407
Add new MIME type 407
Edit MIME Type 408
Delete MIME type 408
Backup MIME types 408
New MIME types 408

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Statistics 409
Check the status of the Pending task queue 409
Check the status of the Text extraction queue 409
Check the status of the Antivirus check queue 410
Metadata 411
Edit metadata 412
Metadata XML definition 413
Basic XML skeleton 413
Metadata group definition 413
Basic example 414
Using file system DTD 415
How metadata is stored into OpenKM 415
Metadata fields 418
Metadata Checkbox field 420
Checkbox 420
Metadata Iframe field 422
Metadata Select field 424
Basic select example 426
Allow multiple choices 426
Based on SQL query 426
Based on className 427
Suggested options 427
Hierarchical relationship 428
Hierarchical relationship with SQL 429
Metadata Separator field 430
Metadata Suggestbox field 431
Based on filterQuery and valueQuery 432
Based on className 433
Metadata Text field 434
Metadata TextArea field 436
Metadata Validator element 438
Basic example 439
Regular expression 439
Metadata Option element 441
Basic select with selected option 441
Hierarchical relationship 442
Metadata Input field 443
Metadata VerticalPanel field 446
Metadata HorizontalPanel field 447
Metadata Security element 448
Basic grant example 448
Basic revoke example 449
Metadata internationalization 450
Label internationalization 450
Translations 450
Metadata definition 451
Manage users and roles 452
Manage users 453
Create a user 454

Edit a user 454
Delete a user 454
Change the user profile 455
Export the users 455
Manage user mail accounts 456
Create a new mail account 458
Edit a mail account 458
Delete a mail account 458
Browser emails in the mail server 458
Manage mail account filters 458
Create a mail account filter 460
Edit a mail account filter 460
Delete a mail account filter 461
Create a mail account rule 461
Edit a mail account rule 461
Delete a mail account rule 461
Manage roles 462
Create a role 462
Edit a role 463
Delete a role 463
Export the roles 463
Logged users 464
Real time user activity log 464
Message queue 465
Send a message 466
Edit a message 466
Delete a message 466
Delete all messages queue 466
Refresh message queue 466
Profiles 467
Create a new profile 467
Edit a profile 467
Delete a profile 467
Clone a profile 468
Profile General tab 469
General section 469
Misc section 469
Wizard section 471
Pagination section 472
Section General 473
Profile Components tab 475
Stacks section 475
Dashboard section 475
Tab section 476
Tab document section 477
Tab folder section 477
Tab mail section 478
Tab record section 478
Tab search section 479
Tab basic section 480
Tab advanced section 480
Profile Menu tab 482

Menu section 482
Menu bookmark section 482
Menu tool section 483
Menu help section 484
Menu file section 484
Menu edit section 486
Profile Notification tab 488
Events 488
Common 488
Document 488
Folder 490
Mail 490
Record 491
Profile Other tab 493
Toolbar section 493
File browser section 495
Relation types 496
Types 496
Bidirectional 496
Parent - child 496
Many-to-many 497
Create a relation 497
Edit a relation 497
Delete a relation 497
CSS 498
Create a new CSS 498
Edit a CSS 498
Delete a CSS 498
Clone a CSS 498
Download a CSS 499
Samples 499
New UI samples (KCenter) 499
Large font sample 499
Medium font sample 499
Arabic sample 499
Reports 501
Create a new report 501
Edit a report 501
Delete a report 502
Show report report parameters 502
Execute a report 502
Reports pack 503
Template reports 503
Authentication 503
Documents 503
Folders 506
Mails 507
Others 508
Subreports 508
Restrictions: 509
XML sections sample 509
Screenshots 509
Suggestions during development 511
Activity log 513
Auth actions 513
Document actions 514

Folder actions 515
Mail actions 517
Record actions 518
Property group actions 519
Node actions 519
Repository actions 520
Admin actions 520
Misc actions 530
Automation 532
Preliminar concepts. 532
Understanding Validators and Actions 534
Creating a rule 535
Editing a rule 535
Edit general parameters 535
Edit action and validation definitions 536
Deleting a rule 536
Built-in Automation Validator plugins 537
Built-in Automation Action plugins 541
Stamp 549
Create new stamp based on text 549
Create new stamp based on image 552
Edit stamp 554
Delete stamp 554
Test stamp 554
AI Prompt 555
Create AI prompt 555
Edit AI prompt 556
Delete AI prompt 556
Crontab 557
Create a new crontab job 559
Edit a crontab job 559
Execute a crontab job 559
Delete a crontab job 559
Built in crontab tasks 560
OCR templates 564
Validation Control field description 564
Data Capture field description 566
Understanding OCR Validation Control and OCR Data Capture 567
Create a new OCR template 568
Edit a OCR template 571
Edit a OCR template fields 571
Edit a OCR Validation Control 571
Edit a OCR Data Capture 572
Delete a OCR template fields 572
Delete a OCR Validation Control 572
Delete a OCR Data Capture 572
Check OCR template 572
Check recognise a document 573
Delete a OCR template 574
Enable OCR Data Capture 574
Built in OCR Validation Control plugins 575
Built in OCR Data Capture plugins 577

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Languages 579
Create a new language 579
Edit a language 579
Delete a language 579
Edit language translations 580
Export a language translations 580
Import a language translations 580
Language translations 581
How to create a new translation 584
A sample of new language base on Oromo from Ethiopia 584
Utilities 585
Admin permissions 587
Change user level access 587
Cache stats 588
Clean all cache 589
Refresh cache stats 589
Enable cache stats audit trail 589
Disable cache stats audit trail 589
Check email 591
Check email 591
Check text extraction 592
Check text extraction 592
Clean conversion cache 593
Clean conversion cache 593
Create support zip 595
Custom actions 596
Database import 599
Import 599
Database query 601
Execute JDBC query 601
Execute Hibernate query 602
Allowed sentences 602
Download tokens 603
Delete token 603
Execute JAR 604
Hibernate stats 605
Information provided by Hibernate stats 605
Enable Hibernate stats 605
Clear Hibernate stats 606
Disable Hibernate stats 606
LDAP Search 607
List indexes 609
Search indexes 610
Enable show terms 611
Disable show terms 611
LogCat 612
Purge all logs 612
Download file as ZIP 612
Browser log file 612

Orphan Files 614
Filter action 614
Password encryption 615
Plugins 616
Profiling stats 619
Enable profiling stats 620
Disable profiling stats 620
Clear profiling stats 620
Refresh profiling stats 620
List of detailed calls 620
Purge data 621
Purge 621
Filter 621
Purge trashes 622
Purge 622
Rebuild indexes 623
Rebuild 623
Repository checker 624
Check repository 624
Repository export 625
Export 625
Understanding output format 625
Case metadata and history are not checked 625
Case only metadata is checked 626
Case only history is checked 627
Case metadata and history are checked 627
Repository import 628
Import 629
Scripting 630
Execute script as part of application starting process 630
Execute script as part of application stopping process 630
Scripting in Automation 630
Shards 632
Shard actions 632
System properties 633
Workflow 635
Register workflow 635
Workflow forms definition 636
Basic XML skeleton 636
Workflow task form definition 636
Basic example 637
Using file system DTD 638
Workflow fields 639
Workflow Button field 642
Workflow Download field 644
Basic download example 644
Dynamic download example 645
Workflow Validator element 646
Basic example 647
Regular expression 648

Workflow Node element 649
Basic download example 649
Workflow Upload field 650
Basic upload example 651
Document updated 651
Upload an example with dynamic parameters. 652
Workflow Checkbox field 653
Checkbox 653
Workflow Input field 655
Workflow Select field 657
Basic select example 658
Allow multiple choice 658
Based on SQL query 659
Based on className 659
Hierarchical relationship 660
Hierarchical relationship with SQL 660
Workflow Separator field 662
Workflow Suggestbox field 663
Based on filterQuery and valueQuery (without table attribute) 664
Based on className 665
Workflow Text field 666
Workflow TextArea field 668
Workflow Option element 670
Basic select with selected option 670
Hierarchical relationship 671
Workflow VerticalPanel field 672
Workflow HorizontalPanel field 673
User guide 674
Menu bar 675
Quick search box 675
ToolBar 675
Desktop 676
Navigation 676
Browser 676
Properties 676
Search 677
Dashboard 679
Administration 681
Status 681
Quick start 682
Logging in 682
Create a taxonomy 682
Add documents 684
Edit documents 685
Search documents 687
Essential fundamentals 689
Basic concepts 689
Understanding taxonomy, keywords, categories and metadata groups. 689
Taxonomy 690
Keywords 690
Categories 691

Metadata 691
Plan your data organization 692
Good practice 693
Working with the menu bar 694
File 694
Edit 698
Tools 700
Bookmark 704
Templates 705
Help 705
Working with the toolbar 706
Working with desktop 710
Navigation 710
Browser 713
Pagination 717
Properties 718
Tabs on nodes of type document 719
Tabs on nodes of type folder 722
Tabs on nodes of type mail 724
Tabs on nodes of type record 726
Shortcuts 728
Working with search 730
Search panels 730
Search panel 730
Operators and wildcards 731
Default search 731
Advanced search 732
Control panel 734
Results 735
Working with Lucene queries 738
Term modifiers 738
Wildcard searches 738
Proximity searches 738
Range searches 739
Boolean operators 739
Working with Lucene queries 739
Term modifiers 740
Wildcard searches 740
Proximity searches 740
Range searches 740
Boolean operators 741
AND 741
+ 741
NOT 741
- 742
Grouping 742
Escaping special characters 742
AND 742
+ 742
NOT 743
- 743
Grouping 743
Escaping special characters 743
Working with metadata fields 743
Working with the dashboard 745

News 746
Workflow 746
Workflow lists 746
Workflow detail 747
User email signature 748
Activity log extension 749
Activity log 749
AutoCAD preview extension 750
Toolbar 750
Default preview 753
Digital Signature viewer extension 755
Verify a signed document 755
Document expiration extension 757
Add document expiration 757
Extra tab workspace extension 758
Configuration 758
Sample 758
External preview extension 760
Configuration 760
Sample 760
Configuration 760
Usage 761
External tab document extension 762
Configuration 762
Sample 762
Configuration 762
Usage 763
External tab folder extension 766
Configuration 766
Sample 766
Configuration 766
Usage 767
External tab mail extension 770
Configuration 770
Sample 770
Configuration 770
Usage 771
External tab record extension 774
Configuration 774
Sample 774
Configuration 774
Usage 775
External widget dashboard extension 778
Configuration 778
Sample 779
Configuration 779
Usage 779
HTML editor extension 781
Features 781
Edit HTML 782
Editor panel 782
Preview panel 783
Forum extension 784

Create new topic 784
Create reply 784
Update post 784
Delete post 784
Stamp extension 785
Document stamping 785
Workflow extension 786
Usage 786
Show workflows assigned to a node 786
Log 786
Graph 787
Using OpenKM from external applications 789
Microsoft Office Add-in 789
Digital Signature Client 789
Other applications 789
Migration guide 790
Troubleshooting 790
Changelog 792
v8.1.16 792
v8.1.15 792
v8.1.14 793
v8.1.13 793
v8.1.12 794
v8.1.11 796
v8.1.10 798
v8.1.9 798
v8.1.8 799
v8.1.7 799
v8.1.6 800
v8.1.5 801
v8.1.4 802
Updater utility 803
Upgrade stages 803
Manual mode 804
Updater utility english video 806
Updater utility spanish video 807
Tomcat update 808
Quick method 808
Linux 808
Preliminaries 808
Tomcat replacement 808
Windows 810
Preliminaries 810
Tomcat replacement 811
Manual update 814
Oracle Java to Temurin 815
Changes from version 7.1 817
Change 817
Deprecated features from version 7.1 817
Development 818
OpenKM professional portable development environment 818
Javadoc 819

Personal development environment 820
Requirements 820
How to get the personal development environment 820
How to install 821
License activation 821
How to upgrade the development environment 821
Step 1 - OpenKM upgrade 822
Step 2 - OpenKM compilation libraries upgrade 822
Step 3 - Jbpm-console upgrade 823
Configuring IDE from a clean Eclipse 824
Requirements 824
Software: 824
Hardware: 824
Download eclipse 824
Configure Java 824
Configure tomcat server 826
Maven support for Eclipse 832
Configuring Maven in your OS 833
Maven tips 833
Memory problems 833
Configure eclipse to use local Maven 834
Eclipse tips 835
Debug tomcat 835
Debug remote tomcat server 835
Eclipse configuration 835
Eclipse known troubleshooting 838
Maven raise error 206 the filename or extension is too long 838
Development tips 839
Tomcat application deployment 839
Deploying script sample 839
Profiling application 841
Edit Tomcat startup scripts 841
Authentication 841
Monitoring Remote JVM Over SSH 842
HPROF 843
Application version compatibility table 844
Creating a JAR file with Eclipse 845
Images 845
Sample JAR file 848
Creating plugins 849
Register a new plugin 850
Creating your own AI prompt plugin 851
Methods description 851
Image to text sample 852
Extract metadata sample 854
Basic summarization sample 857
Creating your own Antivirus Analyzer plugin 859
Method description 859
Example of the clamav antivirus analyzer implementation 860
Creating your own Automation Action 861
Methods description 862
Understanding env variable 863

Understanding params variable 863
Description of available ParamType values: 864
Description of available Src values: 864
Detecting recursion 865
Example 865
Creating your own Automation Validation 868
Methods description 869
Understanding env variable 870
Understanding params variable 870
Description of available ParamType values: 871
Description of available Src values: 871
Example 872
Creating your own Crontab plugin 875
Method description 875
Example of the disposition cron implementation 876
Understanding the crontab job execution context 877
Creating your own Autocomplete Form plugin 878
Method description 878
Example of the default form values implementation 879
Metadata definition 879
Metadata Extraction using AI 880
Creating your own Default Form Values 881
Method description 881
Example of the default form values implementation 882
Metadata definition 882
Creating your own Document Converter 884
Example of the Document converter implementation 885
Creating your own Form Validator plugin 887
Single field validation 887
Methods description 887
Example of the Form validator implementation 888
Full form validation 889
Methods description 890
Example of the Form validator implementation 891
Creating your own FormInterceptor plugin 894
Methods description 895
Example of the Form validator implementation 895
Creating your own NodePaginator plugin 899
Method description 900
Example 900
Creating your own NodeProperties plugin (customize search results objects) 904
Method description 905
Example of NodeProperties implementation 905
Creating your own OCR Validation Control plugin 907
Methods description 907
Example: 908
Creating your own OCR Data Capture plugin 910
Methods description 910
Example 911
Creating your own Option Select Values plugin 913
Methods description 913
Example of the Option select values implementation 914
Creating your own Principal Adapter 915

Methods description 916
Example of the Principal adapter implementation 918
Creating your own Report plugin 921
Methods description 922
Example of the Report plugin implementation 922
Creating your own Rest Plugin (extending REST API) 925
Method description 925
Example of REST plugin implementation 926
Example of REST plugin implementation what returns file 926
Objects declaration 927
String scenario 927
Complex Object scenario 928
Example of complex classes: 928
Advanced REST sample 929
Creating your own Search plugin 931
Method description 932
Example 932
Creating your own security access manager 934
Methods description 935
Example of Security access manager implementation 935
Creating your own Stack Metadata Adapter (deprecated) 938
Methods description 939
Example of the Stack Metadata implementation 940
Creating your own Suggestion Analyzer 948
Methods description 948
Example of Suggestion Analyzer 949
Creating your own Suggestbox plugin 951
Methods description 951
Example of implementation 952
Creating your own Text extractor 954
Methods description 954
Example of the Text extractor implementation 955
Creating your own Version Number Adapter 956
Methods description 957
Example of the Version number adapter implementation 958
Creating templates 959
Creating LibreOffice templates 960
Preliminaries 960
Creating the template 960
Date format 961
Dynamic tables 962
Configuring sample 962
Creating PDF template 964
Preliminaries 964
Creating the template 965
Date format 965
Configuring sample 965
Creating Text templates 966
Preliminaries 966
Creating the template 966
Date format 967

Configuring sample 967
Creating your own login module 968
PrincipalAdapter sample based on properties 970
Creating your own parameters 974
Sample 974
Creating your own password validator 975
Sample 976
Creating your own reports 978
Introducing to reports based on SQL 979
Create datasource 979
Testing SQL sample report 980
Closer look into JRXML file 983
Query section 983
Fields section 984
Remote image section 984
Using fields in report 984
Introducing to reports based on Java 985
Testing Java sample report 985
Understanding the code sample 987
Using in report 988
Introducing to reports with filtering 990
Understanding filtering 990
Params XML file 990
Using fields in report 990
Java sample 991
SQL sample 993
Params form fields 995
Basic XML skeleton 995
Basic example 995
Using file system DTD 996
Params fields 997
Params Validator element 999
Basic example 1000
Regular expression 1000
Params Input field 1002
Params Suggestbox field 1004
Based on the table, filterQuery, and valueQuery 1005
Based on filterQuery and valueQuery (without table attribute) 1006
Based on className 1007
Params Checkbox field 1008
Checkbox 1008
Params TextArea field 1009
Params Select field 1010
Basic select example 1011
Allow multiple choice 1011
Hierarchical relationship 1011
Based on SQL query 1012
Based on className 1013
Suggested options 1013
Params Separator field 1014
Params Text field 1015

Database stuff 1016
Webservices 1017
RESTful 1018
Sample usage 1018
Folder samples 1020
Document samples 1020
Search samples 1021
Security samples 1021
Metadata samples 1022
Notes samples 1022
Conversion samples 1023
Swagger 1024
How to access Swagger 1024
Playing with Swagger 1026
RestClient sample 1031
Login 1031
Get roles 1032
PHP Samples 1034
Download sdk4php 1034
Preview sample 1034
Login 1034
Document samples 1035
createDocument 1035
deleteDocument 1036
getDocumentProperties 1037
getContent 1038
getDocumentChildren 1039
moveDocument 1040
copyDocument 1041
Folder sample 1042
createFolder 1042
getFolderProperties 1043
deleteFolder 1044
renameFolder 1045
moveFolder 1046
getFolderChildren 1047
copyFolder 1048
extendedFolderCopy 1049
getContentInfo 1050
purgeFolder 1051
setStyle 1052
createMissingFolders 1053
createFolderFromTemplate 1054
PropertyGroup samples 1056
addPropertyGroup 1056
removePropertyGroup 1057
getPropertyGroups 1058
getAllPropertyGroups 1059
getPropertyGroupForm 1059
getPropertyGroupFormByNode 1060
setPropertyGroupProperties 1062
hasPropertyGroup 1063
getRegisteredPropertyGroupDefinition 1064
getPropertyGroupSuggestions 1065
registerPropertyGroupDefinition 1066
getPropertyGroupProperties 1068
getPropertyGroupPropertiesByVersion 1069
Search samples 1070
find 1070

findPaginated 1071
Integrate preview with other web applications 1074
Troubleshooting 1077
Samples 1078
Scripting samples 1079
Script - Basic samples 1081
Update repository statistics 1081
Create some folders and set property group 1081
Show number of documents, folders and size from a given path 1081
Get path by UUID 1081
Show all documents used in every running workflow 1082
Script - Compact documents with size equals to 0 1083
Script - Convert TIF to PDF 1084
Script - Count all children subfolder 1085
Script - Exporter 1086
Script - Export repository folder to server as a ZIP file 1087
Script - Folders deep 1088
Images 1089
Execution 1089
Results 1089
Script - Import data from CSV file 1091
Image 1093
Register the metadata group 1093
Repository view 1093
Execution 1094
Script results 1095
Imported metadata 1095
Script - Purge all folders into some parent 1096
Script - Purge all users trashes 1097
Script - Put documents from folder and descendants into pending text extraction queue 1098
Script - Read PDF form fields 1099
Script - Recursive character renaming 1100
Script - Recursive repository traversal 1101
Script - Recursive change security 1102
Script - Security change at low level 1104
Images 1107
Initial security status 1107
UUID of the folder 1108
Execution 1109
Log 1110
Rebuild Lucene indexes 1111
Security changed applied 1112
Script samples for automation 1114
Script for automation - Autonumeric 1115
Register metadata group 1116
Script for automation - Set unique document name based on metadata 1117
Register automation rule 1119
Crontab samples 1120
Crontab sample - Basic file system document importer 1121
Register a new plugin 1122

Images 1122
Crontab task 1122
Files in the application server 1123
Result 1123
Crontab sample - Document importer with metadata values 1125
Register a new plugin 1130
Images 1130
Register metadata definition 1130
Crontab task 1131
Files in application server 1132
Result 1132
Mail notification 1132
Imported files into the repository 1133
Contab sample - Generate HTML report 1135
Register a new plugin 1136
Crontab task - XML importer 1137
Register a new plugin 1142
Images 1142
Crontab task 1142
Register metadata definition 1143
Logfile folder 1144
Output folder 1144
Result 1144
Variables by automation event 1146
Available environment variables 1146
Document 1149
Create Document 1149
Update Document 1150
Delete Document 1151
Rename Document 1151
Purge Document 1151
Move Document 1152
Restore Version Document 1152
Download Document from UI 1153
Download Document from UI for preview 1154
Print Document 1154
Stamp Document 1155
Virus detected Document 1155
Set Node Class Document 1156
Folder 1157
Create Folder 1157
Delete Folder 1157
Purge Folder 1158
Move Folder 1158
Rename Folder 1159
Export Folder as ZIP 1159
Export Folder as JAR 1160
Mail 1160
Create mail 1160
Move Mail 1162
Download mail from UI 1162
Set Node Class Mail 1163
Mail import attachments Mail 1163
Delete Mail 1164
Purge Mail 1164
Record 1164
Create record 1165
Delete record 1165
Purge record 1166
Move Record 1166
Set Node Class Record 1167
Property Group (metadata) 1167

Add property group 1167
Set property group 1168
Remove property group 1169
Auth 1169
Create user 1169
User login 1170
User logout 1170
Task 1170
Create Task 1170
Update Task 1171
Delete Task 1171
Note 1171
Create Note 1172
Update Note 1172
Delete Note 1173
General 1173
Text extraction 1173
Export nodes to zip 1174
Vault of stuff 1175
Convert Exchange Legacy DN 1176
Download document with direct link 1178
Import document with bash script using REST 1179
Creating users from scripts 1180
API description 1181
OKMActivity 1183
Methods 1183
findByFilter 1183
countByFilter 1183
getActions 1184
OKMAuth 1186
Basics 1186
Methods 1186
login 1187
login 1187
grantUser 1188
revokeUser 1189
getGrantedUsers 1189
grantRole 1190
revokeRole 1191
getGrantedRoles 1192
getUsers 1192
getRoles 1193
getUsersByRole 1193
getRolesByUser 1194
changeSecurity 1194
overwriteSecurity 1195
createUser 1196
deleteUser 1196
updateUser 1197
createRole 1198
deleteRole 1198
updateRole 1199
assignRole 1199
removeRole 1200
getProfiles 1200
getUserProfile 1201
setUserProfile 1201
logout 1202
getUser 1203
setUserPermissions 1203
setRolePermissions 1204

getUserTenants 1205
setUserTenant 1205
isPasswordExpired 1206
resetPassword 1206
OKMBookmark 1208
Basics 1208
Methods 1208
add 1208
get 1209
remove 1209
rename 1210
getAll 1210
OKMDashboard 1212
Basics 1212
Methods 1212
getUserCheckedOutDocuments 1212
getUserLastModifiedDocuments 1213
getUserLockedDocuments 1214
getUserLockedRecords 1215
getUserLockekFolders 1216
getUserLockedMails 1217
getUserSubscribedDocuments 1218
getUserSubscribedFolders 1219
getUserSubscribedRecords 1220
getUserLastCreatedDocuments 1221
getUserLastCreatedDocumentsNotes 1222
getUserLastCreatedFolders 1223
getUserLastCreatedFoldersNotes 1224
getUserLastCreatedRecords 1225
getUserLastCreatedRecordsNotes 1226
getUserLastDownloadedDocuments 1227
getUserLastImportedMails 1228
getUserLastCreatedMailsNotes 1229
getUserLastImportedMailAttachments 1230
getUserDocumentsSize 1231
getUserSearches 1232
find 1233
getLastWeekTopDownloadedDocuments 1234
getLastMonthTopDownloadedDocuments 1235
getLastWeekTopModifiedDocuments 1236
getLastMonthTopModifiedDocuments 1237
getLastModifiedDocuments 1238
getLastCreatedDocuments 1239
getLastCheckedOutDocuments 1240
getLastCreatedFolders 1241
getLastCreatedRecords 1242
visitNode 1243
OKMDocument 1245
Basics 1245
Methods 1245
create 1245
createSimple 1246
delete 1247
getProperties 1247
getContent 1248
getContentByVersion 1249
getExtractedText 1250
getChildren 1250
rename 1251
setProperties 1252
setLanguage 1252
setTitle 1253
checkout 1254
cancelCheckout 1254

forceCancelCheckout 1255
isCheckedOut 1255
checkin 1256
checkin 1257
lock 1258
unlock 1258
forceUnlock 1259
isLocked 1260
getLockInfo 1260
purge 1261
move 1261
copy 1262
copy 1263
extendedCopy 1263
getVersionHistorySize 1265
isValid 1265
getPath 1266
getDetectedLanguages 1266
createFromTemplate 1267
updateFromTemplate 1268
setAnnotations 1269
getAnnotations 1270
getCheckedOut 1270
stamp 1271
stampText 1272
stampImage 1273
setDescription 1274
setNodeClass 1274
setDispositionStage 1275
createWizard 1275
getNumberOfPages 1276
getPageAsImage 1277
isConvertibleToPDF 1277
setIndexable 1278
mergePdf 1279
OKMFolder 1280
Basics 1280
Methods 1280
create 1280
createSimple 1281
getProperties 1281
delete 1282
purge 1282
rename 1283
move 1284
copy 1284
copy 1285
extendedCopy 1286
extendedCopy 1287
getChildren 1288
getContentInfo 1288
isValid 1289
getPath 1290
setStyle 1290
createMissingFolders 1291
addNodeClassChildren 1291
setDescription 1292
OKMImport 1293
Basics 1293
Methods 1293
createDocument 1293
createFolder 1294
createRecord 1295
createMail 1295
importFolder 1296

OKMMail 1298
Basics 1298
Methods 1298
getProperties 1298
createAttachment 1299
deleteAttachment 1299
getAttachments 1300
delete 1300
purge 1301
rename 1302
move 1302
copy 1303
copy 1303
extendedCopy 1304
extendedCopy 1305
getChildren 1306
isValid 1307
getPath 1307
sendMail 1308
setTitle 1308
sendMailWithAttachments 1309
importEml 1310
importMsg 1311
setNodeClass 1312
setDispositionStage 1312
setDescription 1313
getContent 1313
createWizard 1314
getExtractedText 1315
forwardEmail 1316
OKMNode 1317
Basics 1317
Methods 1317
getNodeByUuid 1317
getVersionHistory 1318
restoreVersion 1318
renameVersion 1319
purgeVersionHistory 1319
mayBePromotedAsRecord 1320
promoteAsRecord 1321
degradeRecord 1321
isElectronicRecordPath 1322
getElectronicRecordInPath 1322
getChildrenPaginated 1323
getChildrenByCategoryPaginated 1324
getBreadcrumb 1326
subscribe 1326
unsubscribe 1327
importZip 1327
unzip 1328
exportZip 1329
getNodesFromUuids 1329
evaluateZipDownload 1330
restore 1331
hasNodesLockedByOtherUser 1332
lock 1332
forceLock 1333
unlock 1333
forceUnlock 1334
isLocked 1335
getLockInfo 1335
setComment 1336
OKMNote 1337
Basics 1337

Methods 1337
add 1337
get 1338
delete 1338
set 1339
list 1340
getNoteHistories 1340
OKMNotification 1342
Basics 1342
Methods 1342
subscribe 1342
unsubscribe 1343
getSubscriptors 1343
notify 1344
OKMProperty 1345
Basics 1345
Methods 1345
addCategory 1345
removeCategory 1346
addKeyword 1346
removeKeyword 1347
setEncryption 1348
unsetEncryption 1348
setSigned 1349
isSigned 1350
OKMPropertyGroup 1351
Basics 1351
Methods 1351
addGroup 1351
removeGroup 1353
getGroups 1354
getAllGroups 1354
getUserGroups 1355
getProperties 1356
setProperties 1356
getPropertyGroupForm 1358
hasGroup 1359
getSuggestions 1359
registerDefinition 1361
getRegisteredDefinition 1361
getGroups 1362
getGroupProperties 1363
getProperties 1363
getSuggestBoxKeyValue 1364
getSuggestBoxKeyValuesFiltered 1365
OKMRecord 1366
Basics 1366
Methods 1366
create 1366
createSimple 1367
getProperties 1367
delete 1368
purge 1368
rename 1369
move 1370
copy 1370
copy 1371
extendedCopy 1372
isValid 1373
getPath 1373
getChildren 1374
lock 1374

unlock 1375
forceUnlock 1376
setTitle 1376
setNodeClass 1377
setDispositionStage 1377
addNodeClassChildren 1378
setDescription 1378
createWizard 1379
OKMRelation 1381
Basics 1381
Methods 1381
getRelationTypes 1381
add 1382
delete 1383
getRelations 1383
getRelationGroups 1384
addGroup 1384
addNodeToGroup 1385
deleteGroup 1386
getGroup 1386
setGroupName 1387
OKMReport 1389
Basics 1389
Methods 1389
list 1389
get 1390
execute 1390
execute 1391
OKMRepository 1393
Methods 1393
getRootFolder 1393
getTrashFolder 1393
getTrashFolderBase 1394
getTemplatesFolder 1395
getPersonalFolder 1395
getPersonalFolderBase 1396
getMailFolder 1396
getMailFolderBase 1397
getThesaurusFolder 1398
getCategoriesFolder 1398
purgeTrash 1399
getUpdateMessage 1399
getRepositoryUuid 1400
hasNode 1400
getNodePath 1401
getNodeUuid 1402
getAppVersion 1402
copyAttributes 1403
getMetadataFolder 1404
getChangeLog 1404
getAppLicense 1405
OKMSearch 1406
Basics 1406
Methods 1409
find 1409
find 1410
findPaginated 1411
findWithMetadata 1412
findWithMetadataPaginated 1413
findByQuery 1414
findByQuery 1415
findByQuery 1416
findByQueryPaginated 1417

findByQueryPaginated 1418
findByQueryPaginated 1419
findSimpleNodeBaseByQueryPaginated 1421
saveSearch 1422
updateSearch 1423
getSearch 1423
getAllSearches 1424
deleteSearch 1425
getKeywordMap 1425
getCategorizedDocuments 1426
getCategorizedFolders 1427
getCategorizedMails 1428
getCategorizedRecords 1428
getDocumentsByKeyword 1429
getFoldersByKeyword 1429
getMailsByKeyword 1430
getRecordsByKeyword 1430
getDocumentsByPropertyValue 1431
getFoldersByPropertyValue 1431
getMailsByPropertyValue 1432
getRecordsByPropertyValue 1433
findMailPaginated 1433
csvExport 1435
OKMStamp 1437
Methods 1437
calculateStampCoordinates 1437
calculateStampExpressions 1438
stampDocument 1439
stampTextCustom 1440
stampImageCustom 1441
stampImageCustom 1441
getAllStamps 1442
getStampTextByPk 1443
getStampImageByPk 1443
getStampBarcodeByPk 1444
calculateFlyImageDimensions 1445
getPersonalStamps 1445
getPersonalStampImage 1446
OKMStats 1448
Basics 1448
Methods 1448
getDocumentsByContext 1448
getDocumentsSizeByContext 1448
getFoldersByContext 1449
getMailsByContext 1449
getRecordsByContext 1450
OKMTask 1452
Basics 1452
Methods 1454
create 1454
update 1455
find 1456
delete 1456
getByProject 1457
getByType 1457
getAssigned 1458
getAssigned 1459
countAssigned 1460
getActive 1461
getActive 1462
countActive 1463

getFinished 1464
getFinished 1465
countFinished 1466
findStatus 1467
findProject 1468
findType 1468
findNote 1469
getStatuses 1469
getProjects 1470
getTypes 1471
getNotes 1471
createStatus 1472
createProject 1472
createType 1473
createNote 1474
updateStatus 1474
updateProject 1475
updateType 1476
updateNote 1476
deleteStatus 1477
deleteProject 1477
deleteType 1478
deleteNote 1478
getByNode 1479
getHistory 1479
getAssigned 1480
countAssigned 1482
getActive 1483
countActive 1484
getFinished 1485
countFinished 1487
getAssignedCount 1488
getActiveCount 1489
getNotified 1490
getFinishedCount 1491
OKMUserConfig 1493
Methods 1493
setHome 1493
getConfig 1493
OKMWorkflow 1495
Methods 1495
registerProcessDefinition 1495
deleteProcessDefinition 1495
getProcessDefinitionImage 1496
getProcessDefinitionForms 1497
getProcessDefinition 1498
runProcessDefinition 1498
sendProcessInstanceSignal 1499
endProcessInstance 1500
deleteProcessInstance 1501
findProcessInstances 1501
findAllProcessDefinitions 1502
findLatestProcessDefinitions 1503
findLastProcessDefinition 1503
getProcessInstance 1504
suspendProcessInstance 1505
resumeProcessInstance 1505
findUserTaskInstances 1506
findPooledTaskInstances 1507
findTaskInstances 1507
setTaskInstanceValues 1508
getTaskInstance 1509
setTaskInstanceActorId 1509
startTaskInstance 1510
endTaskInstance 1511

getToken 1511
sendTokenSignal 1512
Commons classes 1514
ArchiveUtils 1516
Methods 1516
createZip 1516
createJar 1516
ExecutionUtils 1518
Methods 1518
runScript 1518
runJar 1518
runCmd 1519
FileLogger 1520
Methods 1520
info 1520
warn 1520
error 1521
FileUtils 1522
Methods 1522
getFileName 1522
getFileExtension 1522
FormatUtil 1524
Methods 1524
formatSize 1524
parseSize 1524
formatDate 1525
ISO8601 1526
Methods 1526
parseBasic 1526
formatBasic 1526
GeneralUtils 1528
Methods 1528
uiNotify 1528
MailUtils 1529
Methods 1529
sendMessage 1529
sendMessage 1529
sendDocuments 1530
PathUtils 1532
Methods 1532
getParent 1532
getName 1532
getContext 1533
PDFUtils 1534
Methods 1534
merge 1534
stampImage 1535
stampText 1536
getNumberOfPages 1537
PrincipalUtils 1538
Methods 1538
getUser 1538
getTenantId 1538
StackTraceUtils 1540
Methods 1540
isCallingMe 1540
toString 1540

TemplateUtils 1542
Methods 1542
templateExists 1542
replace 1542
WorkflowUtils 1544
Methods 1544
findProcessInstancesByNode 1544
ShardUtils 1545
Methods 1545
setCurrentShard 1545
createShardDirs 1545
changeNodeShard 1546
getShardIds 1546
getShardSize 1547
getShardPaths 1547
getShards 1548
getShardFolders 1548
LegacySrv 1550
executeSQL 1550
executeHQL 1550
DbSessionManager 1552
Methods 1552
getSystemToken 1552
ConfigSrv 1553
getString 1553
getText 1553
getBoolean 1554
getInteger 1554
getLong 1555
getHtml 1555
getPassword 1556
getList 1556
Database description 1558
Node types 1558
SQL example queries 1559
Mark documents to be indexed again 1559
Node version 1559
SQL example queries 1560
Node metadata properties 1562
Node categories and keywords 1563
SQL example queries 1564
Advanced samples 1566
Custom actions in KCenter 1567
Sample based in a JSP 1567
Step 1 - Create a sample JSP file 1567
Step 2 - Register the custom action 1567
Step 3 - Enable the custom action in the profile 1568
Step 4 - Check the feature 1568
Full JSP sample 1568
Sample based in a Javascript 1570
Step 1 - Register the custom action 1570
Step 2 - Enable the custom action in the profile 1571
Step 3 - Check the feature 1571
Sample with validation 1571
Step 1 - Register the custom action 1572
Step 2 - Enable the custom action in the profile 1573
Step 3 - Check the feature 1573

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Custom UI template 1574
General description 1574
Properties file 1574
Authentication 1575
Log file 1575
Remember me 1575
Roles management 1575
Bootstrap 1576
Packaging 1576
Application port 1576
Catalog information 1576
Description 1576
Installation 1576
Step 1: Download 1576
Step 2: Role creation 1577
Step 3: SQL data needed 1577
Step 4: Metadata 1577
Step 5: Automations 1578
Step 6: Catalog documents from UI 1578
Form UI template 1579
Packaging 1579
Installation 1579
Step 1 - Register metadata in OpenKM 1579
Step 2 - Upload template file 1579
Security Provider 1581
General description 1581
Properties file 1581
Custom Principal Adapter 1581
Step 1: Download 1581
Step 2: OpenKM configuration 1582
Step 3: Install the plugin 1582
Authentication Provider 1582
Step 1: Download 1582
Step 2: Install the patch 1582
Step 3: OpenKM configuration 1583
Set unique document name based on metadata 1584
Tab UI content extension 1588
Packaging 1588
Installation 1588
Tab UI content extension using KCenter methods 1589
Step 1 - create a sample HTML file 1589
Step 2 - Configure the external tab document extension 1589

OpenKM 8.1

OpenKM Professional version is licensed under the "**EULA - OpenKM End User License Agreement**."

For more information, take a look at Licenses.

OpenKM is an Electronic Document and Record Management System (EDRMS). Much literature about document management terms like DMS, EDRMS, or CMS is usually more influenced by marketing rules than objective reasons.

In most cases, the differences between DMS, CMS, and EDRMS are subjective or irrelevant from the Document Management point of view. The term CMS has gotten some marketing success and is most commonly used. However, we prefer the term **EDRMS**, perhaps with lesser success in the marketing world but more strict and representative from our point of view.

OpenKM helps you to:

- Day-to-day data capture.
- Storage.
- Modify and share physical or digital documents within the organization.
- Apply retention periods to stored items.
- Identify the owner of each record series and document.
- Set a chain of custody and proper audit trail.
- When needed, apply legal records holds.
- I am managing records disposition.
- They preserve records throughout their life cycle.

Some of the OpenKM goals are:

- Increase **efficiency: faster** document approval, reduced manual data entry, and automated recurring tasks.
- Set **compliance** levels. The system helps the organization avoid penalties when regulators, auditors, and government authorities summon them for inspections.
- Preserve **historical content**. Follow the document's status (determined by the different lifecycle phases).
- **Consolidate** the company knowledge on a single access point.

The maximum of the OpenKM technical staff is "**Try finding the easy solution for each problem. If you cannot find it, try again.**"

Although we've tried to give our powerful application what it needs to solve problems with a minimalist solution, sometimes it's impossible. OpenKM can solve complex day-to-day scenarios, and it is not a secure application and has many features that can be used and configured in several ways.

Please take your time to read the documentation; it is time well spent. Suppose you have some doubt contact us. Usually, a problem has several ways to be resolved.

Your suggestions are always welcomed; we try to create a better application to solve real-life problems more efficiently. Your tips help us grow our knowledge of EDRMS solutions and apply them to real case scenarios.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Hardware and software requirements

This section presents information about the environment needed to run an OpenKM server. It lists the required and optional software and provides facts about their configuration.



This section aims to help you validate or define the production environment. However, the checklist is not exhaustive and needs to be completed with the user's experience in mind.



For best performance, OpenKM stores the binary content of the document in the filesystem. So, if you have a server for the database and another for the application itself, remember that the OpenKM server needs enough free disk space to store all these documents plus extra space for other files like preview cache and extracted text and Lucene indexes.

Requirements

OpenKM can be installed on any operating system (Linux, Windows, Mac OS X) since it is a multi-platform application.



A Linux host is a good choice because it handles heavy I/O applications, works well with OpenKM, and is conveniently priced.

The only requirement to run the OpenKM is to have **Java 11** installed.

Checking the Java version

1. Open a terminal:
    - On Linux or Mac OS X: **Open a terminal**.
    - On Windows: press the "Windows" key + r, type **cmd** (or command) in the Run window, and press "OK" or open the "Prompt command" from the "Start > Programs > Accessories" menu.
2. Type **java -version** and press **Enter**.

   If Java is installed correctly on your computer, the name and version of the Java virtual machine are displayed:

```
openjdk version "11.0.20.1" 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04, mixed mode, sharing
```

If Java is not installed on your computer, it will fail to display the Java version. Then, you will need to install Java (see below).

If Java is installed but not included in the PATH, it will fail to find the Java command. Then, you will need to add $JAVA_HOME/bin/to your PATH (see How do I set or change the PATH system variable?).

Installing Java

For Windows users

If the required version of Java is not installed on your computer:

1. Download it from the Microsoft website and choose the right platform for your hardware and Windows version.
2. Run the MSI file and follow the instructions displayed.

More information at Installing on Windows

For Linux users

You can download it from the Linux repositories

```
apt install openjdk-11-jre
```

This way, your OpenJDK will be upgraded whenever a new release is available.

In case you have another OpenJDK installed (like GCJ), you can select the right one this way:

```
$ sudo update-alternatives --config Java
```

More information on installingInstalling on Ubuntu and Debian.

Recommendations

Hardware configuration

OpenKM runs quite well on any computer with a standard hardware configuration. For better performance, we recommend at least 4 GB RAM (have at least >=2GB of RAM) and a fast hard disk like SATA or better.

A dual-core Intel-based CPU with 3.20 GHz should be fine for fewer users (30-50 concurrent users).

Have enough disk space for the document repository and its versions.

A Linux host is a good choice because this OS handles heavy I/O applications like this quite well, and it is not expensive.

It is a highly recommended 64-bit architecture.

We suggest having a modern server that performs well.

Although OpenKM can be installed on a server with other applications, the best scenario is a dedicated server. Remember that other applications can harm OpenKM in a shared scenario.

For more information about hardware requirements, please consult with your sales and marketing representatives. They should provide you with an estimation of the hardware requirements.

(See the questionnaire at the bottom of the page)

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

You can use your older history of documents and folders to get some of these values.

Browser support

OpenKM user interface is based on the GWT framework, which means the supported browsers are determined by GWT browser support.

Actual browsers supported:

- Mozilla Firefox
- Google Chrome and Chromium
- Microsoft Edge

Although you can use other browsers, such as Safari or Opera, we suggest using the latest version of Google Chrome.

We encourage your browser to be updated to the newest version.

We recommend Mozilla Firefox and Google Chrome because they offer the best performance, have fewer security holes, and follow the standards.

Operating systems

Linux (Known working configurations):

- Ubuntu
- Debian
- CentOS
- Red Hat Linux

Red Hat Linux is a more complex scenario because some packages aren't in the Red Hat official repositories by default.

Ubuntu is the preferred Linux choice. We run the tests against Linux distribution and build the packages for it. However, any recent Linux distribution should be usable as long as the:

- OpenJDK 11 is available.
- You can adapt the provided Linux scripts.
- Thirdparty software requirements

Windows (Known working configurations):

- For development: Windows 7, Windows 8
- For production: Windows Server 2003, 2008, 2012, 2016, 2019, 2022 and upper versions

Databases

Known working configurations:

- MySQL
- Oracle
- PostgreSQL
- SQL Server



The application internally uses Hibernate; in fact, you should get it running from any supported Hibernate dialects; for more information, read Hibernate: SQL Dialects.

JVM

Although OpenKM can run with other JDK, we recommend using OpenJDK 11 64-bitts architecture is highly recommended).

LDAP

- Open LDAP
- Microsoft Active Directory

Small questionary



The average value is particularly interesting in questions where the expected answers are numerical.

Try giving them as much detailed information as possible for a more accurate estimation using this basic set of questions:

- Estimated number of concurrent users.
- Type of users (% of readers and % of writers).
- Estimated first-year number of files.
- Estimated number of files growing per year.
- Estimated document size average.
- Number of documents in the initial import
- Are these documents uploaded by users or automatically?
- Most common document types (Microsoft Office, LibreOffice, PDF, etc.).

The other more detailed questions below are a guide to describe the document management system problem to solve and may be helpful to estimate hardware requisites:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- Do you have an initial import of documents?
    - If so, how many?
    - If so, what is the size of the data in GB?
- How many records are created in a day?
- How many documents are made in a week?
- How many documents are made in a month?
- How many documents are made in a year?
- What is the expected repository size in GB for the first year (initial import not included)?
- What is the typical document size in your repository?
- How many concurrent users are expected?
- How many of the simultaneous users are readers, and how many writers?
- Do you have some daily, weekly, or other kinds of periodic bulk uploads?
    - If so, how many documents are uploaded?
    - Can the bulk upload be done during non-working hours?
- Will you have some integration with third-party applications?
    - If so, describe the usage in terms of:
        - Will it be used to read data?
        - Will it be used to write data?
        - How many are the expected queries per minute and hour?
- What kind of documents did you manage, such as standard Office documents, PDFs, or less common documents like video and large TIFF files?
    - Describe them, especially what you know is not the most common in the companies.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Third-party software requirements

OpenKM uses third-party software for some features. Some of this software needs to be installed on the server, while others are optional.

Software	Usage	
Libreoffice	• Converts documents to PDF. • It is needed for the preview feature.	
ImageMagick **or** **GraphicsMagick**	• Transforms images. • Generates thumbnails. • It is needed for the preview feature.	Usually, it come In most cases, it
GhostScript	• Interpreter for the PostScript language and PDF. • Needed by Imagemagick.	Usually, it come
Tesseract	• OCR engine. • They are used for the text extraction feature.	It is required wh
FFmpeg	• Converts and streams audio and video	
Clamav or equivalent	• Antivirus	Required when

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Installation

OpenKM can be installed on several Operating Systems and configured to use different databases. Because it's implemented in Java, the only software requirement is compiling a Java Virtual Machine for your platform. However, OpenKM also depends on third-party applications to be fully configured. For example, it needs a working LibreOffice installation to preview documents. You may also want to extract text from your scanned images, which requires an OCR engine.

Before installing, you need to prepare the server. Depending on the Operating System, you should follow some tips. When installing the Operating System, remember that OpenKM does not need a graphical user desktop. So, in Linux systems, you can avoid installing these packages and save some memory. Another essential piece of advice is to install a minimal operating system. We only need the minimum to access the server. In the case of the Linux server, we prefer to access SSH because it is fast and secure. Please don't install Java, Tomcat, or any other software unless you have already asked us.

This is a checklist of information we should know before installing:

- How many documents do you want to store?
- How many users will access OpenKM?
- How many concurrent users?
- How much memory does the server have?
- How many CPUs does the server have?
- How much is free disk available on the server?
- Which Operating System is installed?
- Is the installed Operating System the 64-bit version?

In the case of the Linux server, there are some additional questions:

- Which distro (name and version)?
- Does the server have a GUI (X-Windows) installed?
- Is it a clean and minimal installation (only SSH), or have you installed another software?
- Is this server used for other applications?
- Is the database already installed?

In this section, you will learn how to install OpenKM on your server and how to match the most common third-party application requirements.

OpenKM has been successfully installed on the following:

OS	Description
Linux	Ubuntu Server

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	Linux Mint Debian Rocky Linux Oracle Linux Red Hat SuSE
Windows	Windows Server 2022 Windows Server 2019 Windows Server 2016 Windows Server 2012 Windows Server 2008 Windows Server 2003 Windows 11 Windows 8 Windows 7

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Installing on Ubuntu and Debian

- Ubuntu ISO can be downloaded from https://ubuntu.com/download/server. Please, always choose a Server LTS version.
- Debian ISO can be downloaded from https://www.debian.org/download.

This installation guide works for other **Ubuntu / Debian-based** distributions like **Linux Mint**.

This guide has been validated with the following:

- Ubuntu 22.04 LTS
- Ubuntu 20.04 LTS
- Debian 11
- Debian 12

To install OpenKM, the server needs access to **download.openkm.com** and **update.openkm.com; please** verify before installing.

If you use a proxy to access the Internet, please look at Using the installer.

Preliminaries

Ensure you have enough disk space to install OpenKM and to store the document repository:

```
$ df -h
```

Once verified, create the "openkm" user:

```
$ adduser openkm
```

If Java 11 is still not installed, please install the following:

```
$ apt install openjdk-11-jre
```

If Java 11 packages are unavailable in your distro repositories, you must download and install them from Eclipse Temurin.

OpenKM currently only works with Java 11, so once the installation process is completed, verify the default version is Java 11:

```
$ java -version
openjdk version "11.0.20.1" 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04)
```

```
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
```

Otherwise, set Java 8 as the default one:

```
$ update-alternatives --config java
```

If you do have any other preference for a database, we suggest installing MariaDB:

```
$ apt install mariadb-server
```

After that, check if the MariaDB service has already started:

```
$ systemctl status mariadb
```

Installing OpenKM

You can choose between installing OpenKM at "/home/openkm" or "/opt." In any case, go to the selected path and download the installer.

It's a text-based installer, so you must execute it from the console as root. This is because it needs to install package dependencies and configure OpenKM as a service:

```
$ java -jar OKMInstaller.jar
```

Remember to check Using the installer if you are using a proxy to access the Internet.



The installer will ask for an authorized user and password. If you don't already have this information, please open a support ticket at https://support.openkm.com or contact your sales agent.

When the installation process has finished, please ensure that Java 11 is the default one:

```
$ java -version
```

Otherwise, set it as default:

```
$ update-alternatives --config java
```

Once verified, start the required services:

```
$ systemctl start xvfb openkm
```

You can verify the status of these services this way:

```
$ systemctl status xvfb openkm
```

More info at:

- Install Java 11|Java 17|Java 18 on Debian 11

Post-installation checklist

To verify the installation was successful, please use this checklist:

- Verify the default Java is Java 11.
- Verify you can access OpenKM from another computer.
- Try to log in with the default okmAdmin / admin user and password.
- Upload a Word document and check if the thumbnail is generated and the preview is shown.

Troubleshooting

If you have a problem with thumbnail generation, please verify that you have the ImageMagick package installed and that the "convert" application works.

Download a sample image from Sample PNG Image and try to generate a thumbnail:

```
$ wget https://www.sample-videos.com/img/Sample-png-image-1mb.png
$ convert -thumbnail '80x80>' -background 'white' -flatten Sample-png-image-1mb.png sample-output.png
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Installing on RedHat and CentOS

- If you decide to install RedHat, consider that you should have a licensed RedHat version; otherwise, you won't be able to install updates.
- RedHat ISO can be downloaded from https://developers.redhat.com/products/rhel/download for testing, but you must register before.

This installation guide works for other **RedHat-based** distributions like **Oracle Linux** and **Rocky Linux**.

This guide has been validated with the following:

- RedHat 9.1
- RedHat 8.7
- Oracle Linux 7.6
- Rocky Linux 9.1
- CentOS Stream 9

To install OpenKM, the server needs access to **download.openkm.com** and **update.openkm.com; please** verify before installing.

If you use a proxy to access the Internet, please use Using the installer.

Preliminaries

Ensure you have enough disk space to install OpenKM and store the document repository:

```
$ df -h
```

Once verified, create the "openkm" user:

```
$ adduser openkm
```

If Java 11 is still not installed, please install the following:

```
$ yum install java-1.11.0-openjdk
```

If Java 11 packages are unavailable in your distro repositories, you must download and install them from Eclipse Temurin.

OpenKM currently only works with Java 11, so once the installation process is completed, verify the default version is Java 11:

```
# java -version
openjdk version "11.0.20.1" 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1, mixed mode, sharing)
```

Otherwise, set Java 11 as the default one:

```
$ update-alternatives --config java
```

If you do have any other preference for a database, we suggest installing MariaDB:

```
$ yum install mariadb-server
```

After that, enable and start the MariaDB service:

```
$ systemctl enable mariadb
$ systemctl start mariadb
```

Installing OpenKM

You can choose between installing OpenKM at "/home/openkm" or "/opt." In any case, go to the selected path and download the installer.

It's a text-based installer, so you must execute it from the console as root. This is because it needs to install package dependencies and configure OpenKM as a service:

```
$ java -jar OKMInstaller.jar
```

Remember to check Using the installer if you are using a proxy to access the Internet.



The installer will ask for an authorized user and password. If you don't already have this information, please open a support ticket at https://support.openkm.com or contact your sales agent.

When the installation process has finished, please ensure that Java 8 is the default one:

```
$ java -version
```

Otherwise, set it as default:

```
$ update-alternatives --config java
```

Once verified, start the required services:

```
$ systemctl start xvfb openkm
```

You can verify the status of these services this way:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
$ systemctl status xvfb openkm
```

More info at:

- How do I turn SELinux off in Red Hat Enterprise Linux?
- Install Extra Packages for Enterprise Linux (EPEL)
- Remove the yum repo using yum

Enable remote access

RedHat does not allow connection to the 8080 port by default, so you need to open this port in the firewall:

```
$ firewall-cmd --zone=public --permanent --add-port=8080/tcp
$ firewall-cmd --reload
```

Post-installation checklist

To verify the installation was successful, please use this checklist:

- Verify the default Java is Java 11.
- Verify you can access OpenKM from another computer.
- Try to log in with the default okmAdmin / admin user and password.
- Upload a Word document and check if the thumbnail is generated and the preview is shown.

Troubleshooting

If you have a problem with thumbnail generation, please verify that you have the ImageMagick package installed and that the "convert" application works.

Download a sample image from Sample PNG Image and try to generate a thumbnail:

```
$ wget https://www.sample-videos.com/img/Sample-png-image-1mb.png
$ convert -thumbnail '80x80>' -background 'white' -flatten Sample-png-image-1mb.png sample-output.png
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Installing on Windows



This section of the documentation tries to describe how to install OpenKM manually. An OpenKM Installer tool does most things automatically; more information at Using the installer.

Preliminaries



In case to configure OpenKM in a cluster or sharding mode, please, before starting with the installation process, read the sections:

- Cluster configuration
- Sharding configuration

Check that you have enough disk size.

We recommend installing these tools:

- Notepadd++ (text editor)
- Geany (text editor).
- mTail (Windows Program that works like the Unix command "tail").

Checking Java version

```
C:\> java -version
```

If Java is correctly installed on your computer, the name and version of the Java virtual machine are displayed:

```
openjdk version "11.0.20.1"
OpenJDK Runtime Environment
OpenJDK 64-Bit Server VM
```



The Java version may differ depending on your installed one, but we recommend the latest 1.8 release.

We recommend also setting the JAVA_HOME variable:

```
setx JAVA_HOME "C:\Program Files\Java\jdk11.0.20.1" /M
```

Install Java



Do these steps only if Java 11 is not installed on your server.

If the required version of Java is not installed on your computer:

1. Download it from the Microsoft website and choose the right platform for your hardware and Windows version.
2. Run the MSI file and follow the instructions displayed.

Installing database

Install MySQL if you do not have it already installed:

Download MySQL MSI Installer from MySQL Community Server and follow the installation wizard steps.

Installing MySQL Workbench is also a good idea if you do not have MySQL client software.

Check InnoDB is the default MySQL engine

```
show engines;
```

It should show something like this:

| **InnoDB** | **DEFAULT** | Supports transactions, row-level locking, and foreign keys | YES | YES | YES |

If it does not default, modify **c:\ProgramData\MySQL\MySQL Server 8.0.34\my.ini**

and under [mysqld], add

```
default-storage-engine = innodb
```

After changing the default MySQL engine, the MySQL service must be restarted to take effect.

Create database

```
CREATE DATABASE okmdb DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_bin;
CREATE USER openkm@localhost IDENTIFIED BY 'password';
GRANT ALL ON okmdb.* TO openkm@localhost WITH GRANT OPTION;
```

Installing Tomcat

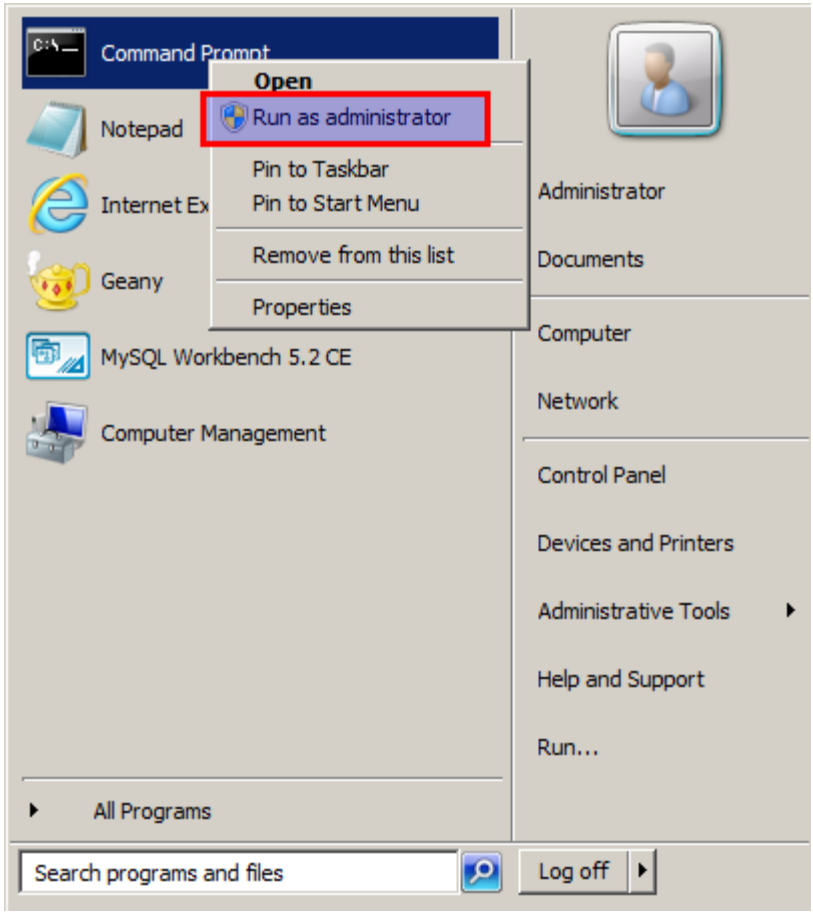
Unzip Tomcat-9.0.76.zip to c:\

The Tomcat version numeration name can be altered to be upgraded to a major version.

Configuring Tomcat as a service

Register for the service:

Open a windows terminal as administrator.



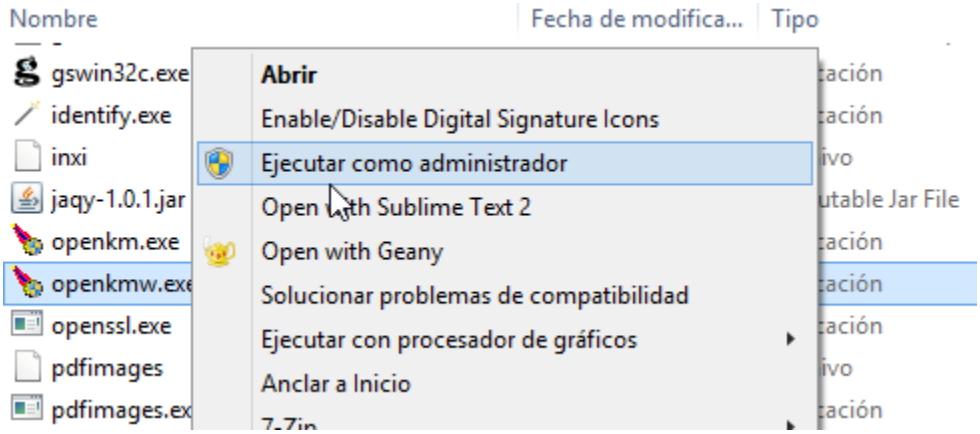

```
cd c:\tomcat-9.0.76\bin
service.bat install
```

Set the JVM parameters for the Tomcat service:

```
cd c:\tomcat-9.0.76\bin
openkm //US//OpenKM ++JvmOptions="-Xms1024m"
openkm //US//OpenKM ++JvmOptions="-Xmx4096m"
openkm //US//OpenKM ++JvmOptions="-XX:+UseG1GC"
openkm //US//OpenKM ++JvmOptions="-XX:+UseStringDeduplication"
openkm //US//OpenKM ++JvmOptions="-XX:+HeapDumpOnOutOfMemoryError"
openkm //US//OpenKM ++JvmOptions="-XX:HeapDumpPath=%CATALINA_HOME%"
openkm //US//OpenKM ++JvmOptions="-Djava.net.preferIPv4Stack=true"
openkm //US//OpenKM ++JvmOptions="-Dmail.mime.ignoreunknownencoding=true"
```

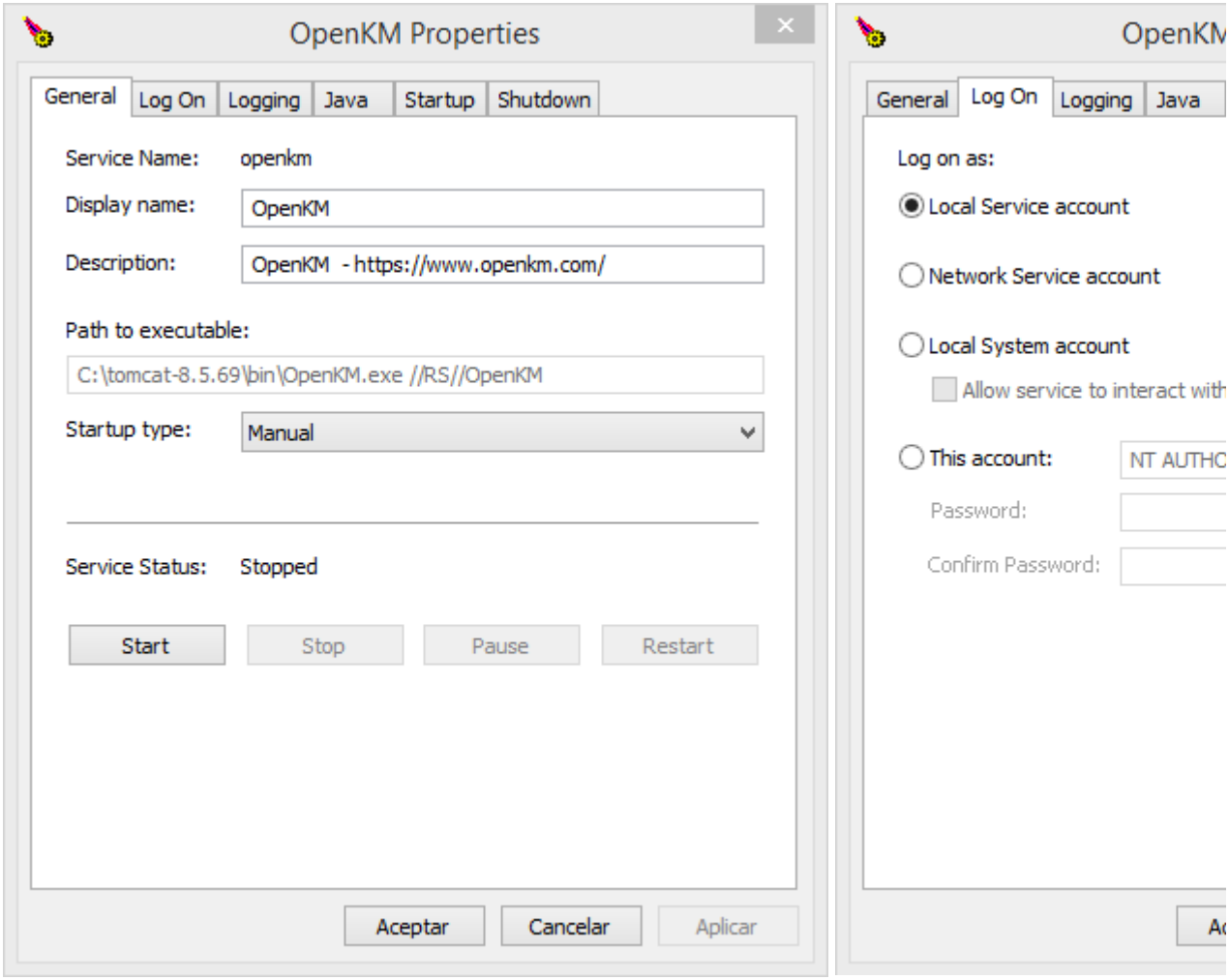
```
openkm //US//OpenKM ++JvmOptions="-Dfile.encoding=UTF-8"
openkm //US//OpenKM ++JvmOptions="-Dsun.jnu.encoding=UTF-8"
openkm //US//OpenKM ++JvmOptions="-Djava.awt.headless=true"
openkm //US//OpenKM ++JvmOptions="-Dsun.java2d.cmm=sun.java2d.cmm.kcms.KcmsServiceProvider"
```

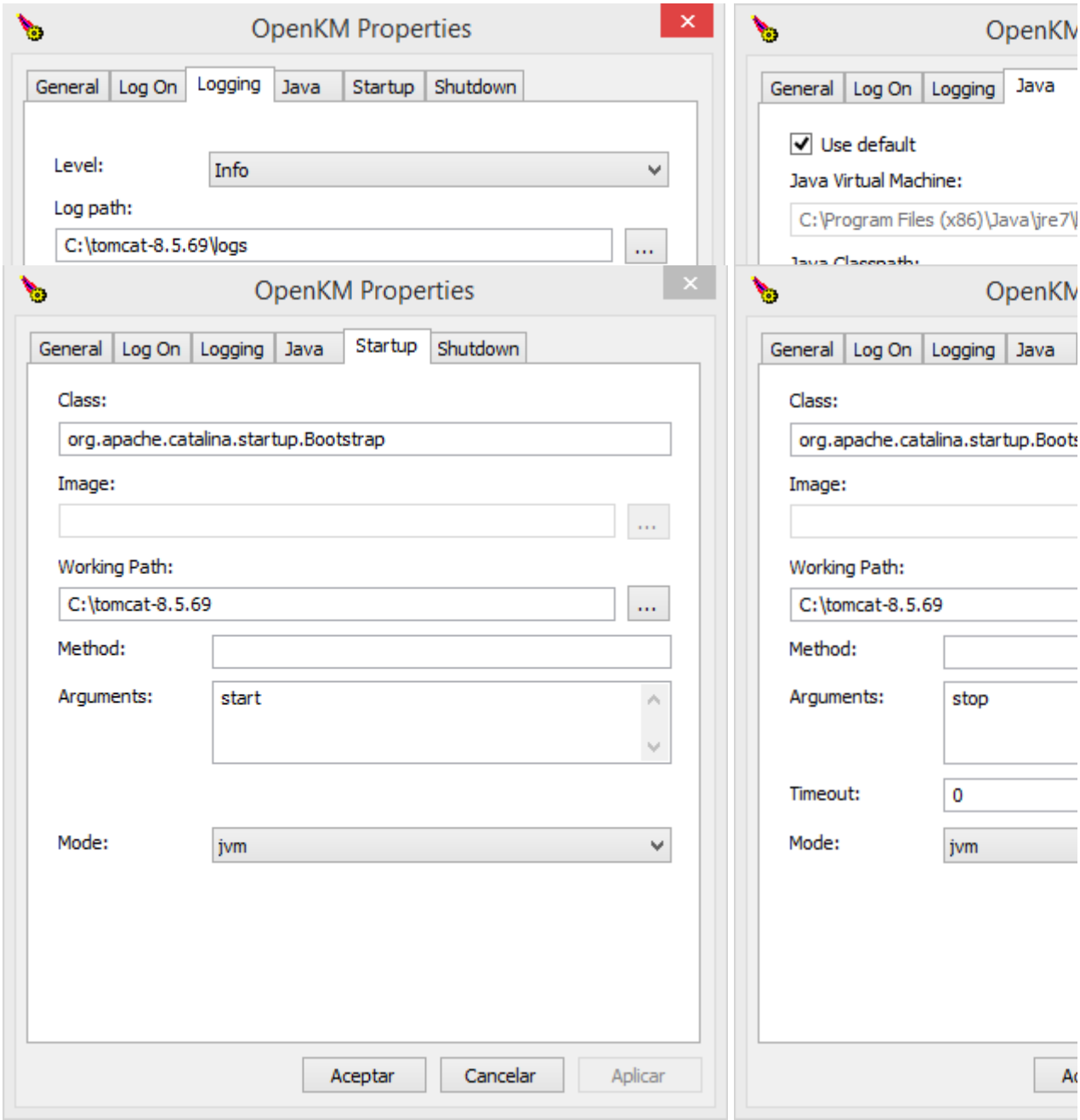
Execute the openkmw.exe as administrator:



Check the configuration parameters:

> Also, you can change "Startup type" to Automatic.


OpenKM Properties
General
Log On
Logging
Java
Startup
Shutdown
Service Name:
openkm
Display name:
OpenKM
Description:
OpenKM - https://www.openkm.com/
Path to executable:
C:\tomcat-8.5.69\bin\OpenKM.exe //RS//OpenKM
Startup type:
Manual
Service Status:
Stopped
Start
Stop
Pause
Restart
Aceptar
Cancelar
Aplicar
OpenKM
General
Log On
Logging
Java
Log on as:
Local Service account
Network Service account
Local System account
Allow service to interact with
This account:
NT AUTHO
Password:
Confirm Password:


OpenKM Properties
General
Log On
Logging
Java
Startup
Shutdown
Level:
Info
Log path:
C:\tomcat-8.5.69\logs
Use default
Java Virtual Machine:
C:\Program Files (x86)\Java\jre7\
OpenKM Properties
Class:
org.apache.catalina.startup.Bootstrap
Image:
Working Path:
C:\tomcat-8.5.69
Method:
Arguments:
start
Mode:
jvm
Aceptar
Cancelar
Aplicar
Class:
org.apache.catalina.startup.Boots
Image:
Working Path:
C:\tomcat-8.5.69
Method:
Arguments:
stop
Timeout:
0
Mode:
jvm

More information about how to configure Tomcat service on Windows at: https://tomcat.apache.org/tomcat-9.0-doc/windows-service-howto.html

Check the service

From terminal

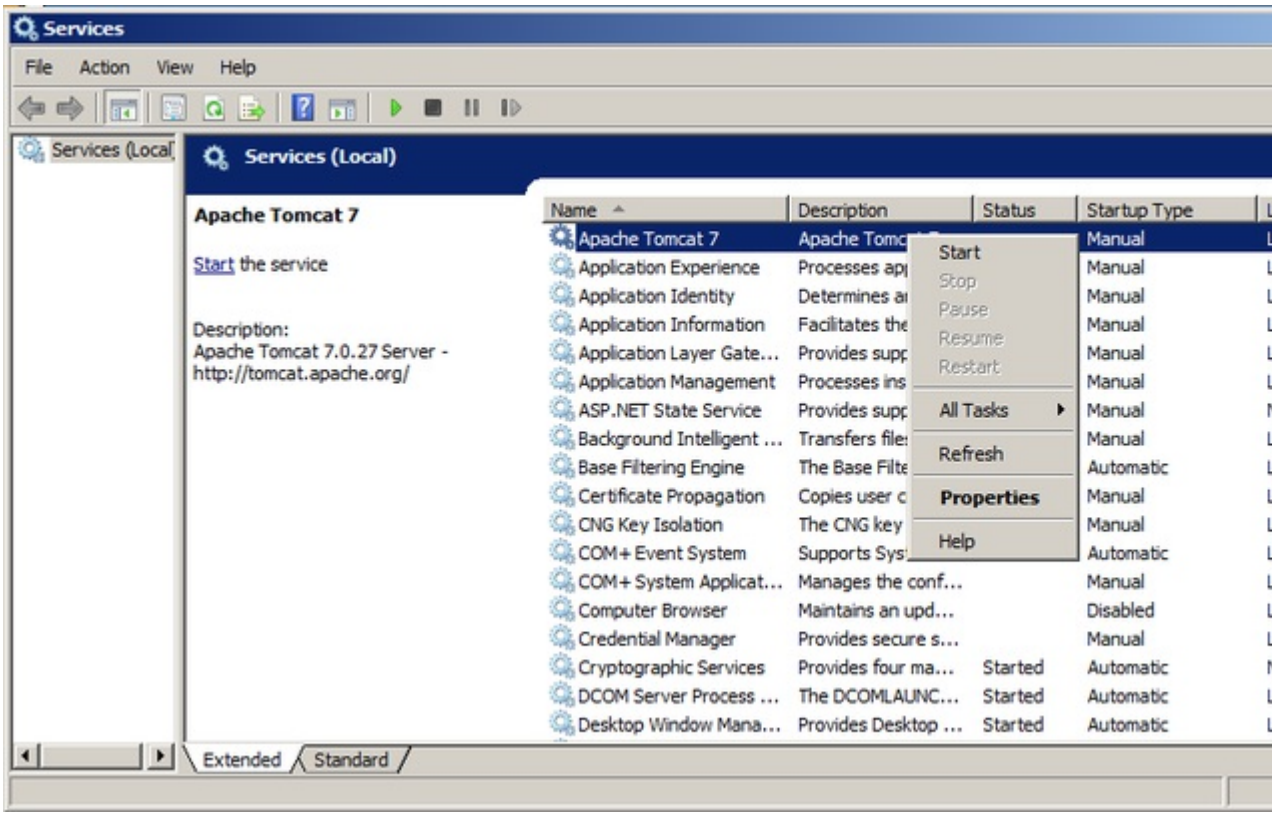
Start the service:

```
$ net start OpenKM
```

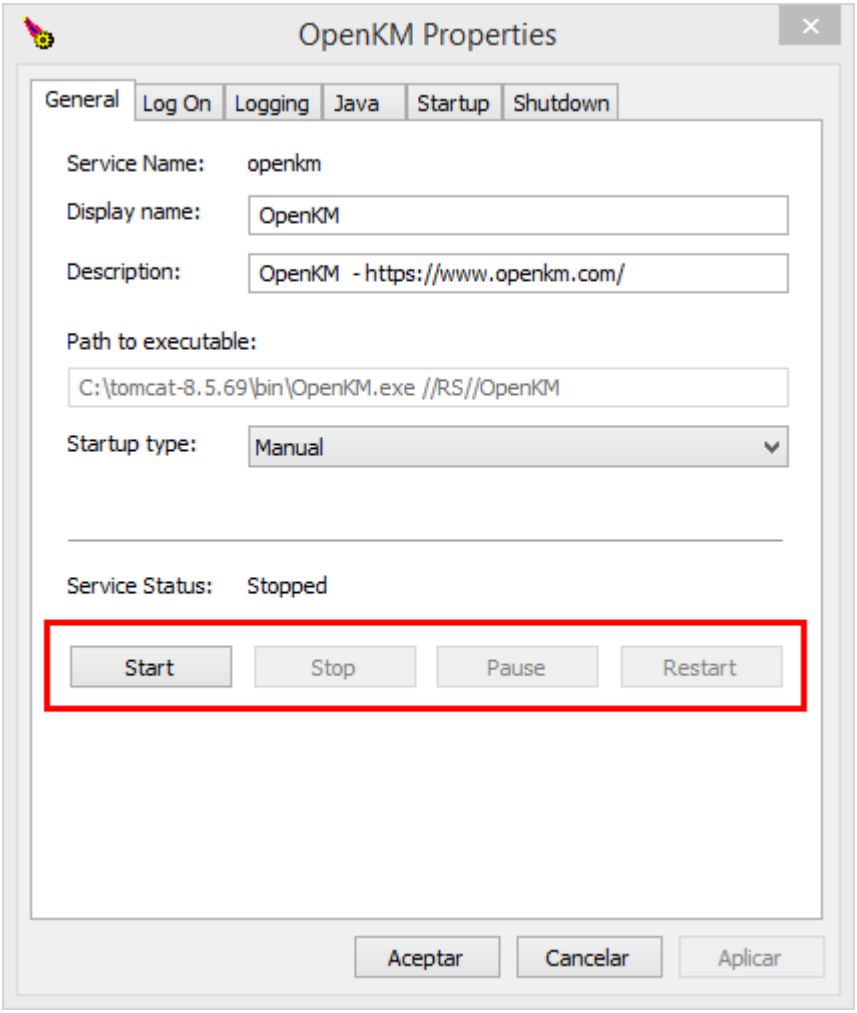
Stop the service:

```
$ net stop OpenKM
```

From the Windows service panel



From openkmw tool



Configuring environment variables

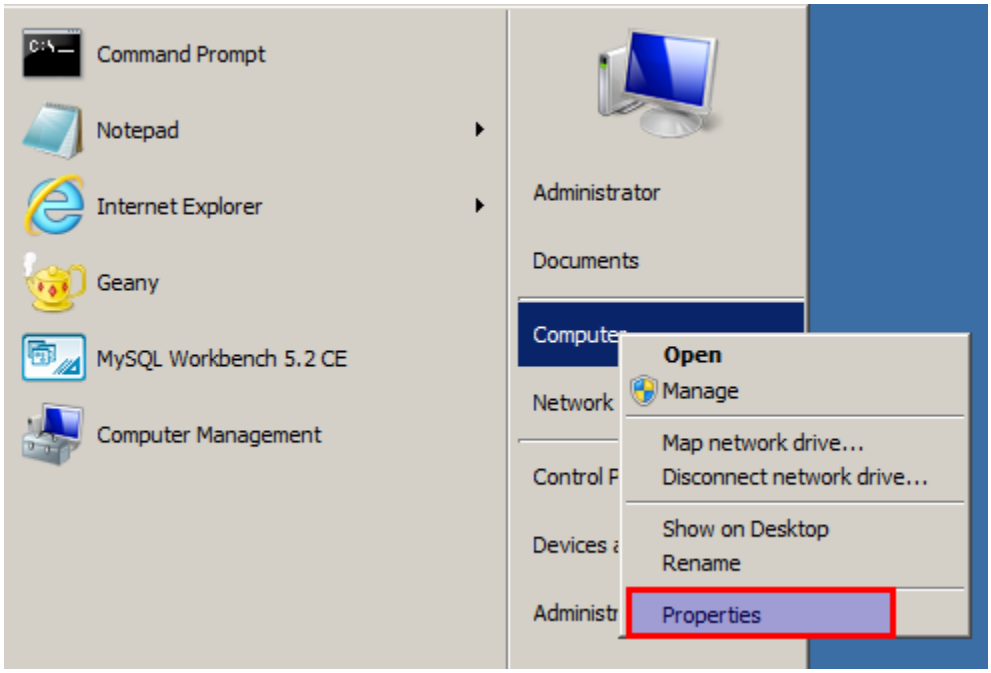
Add to the environment path variable the **bin** folder:

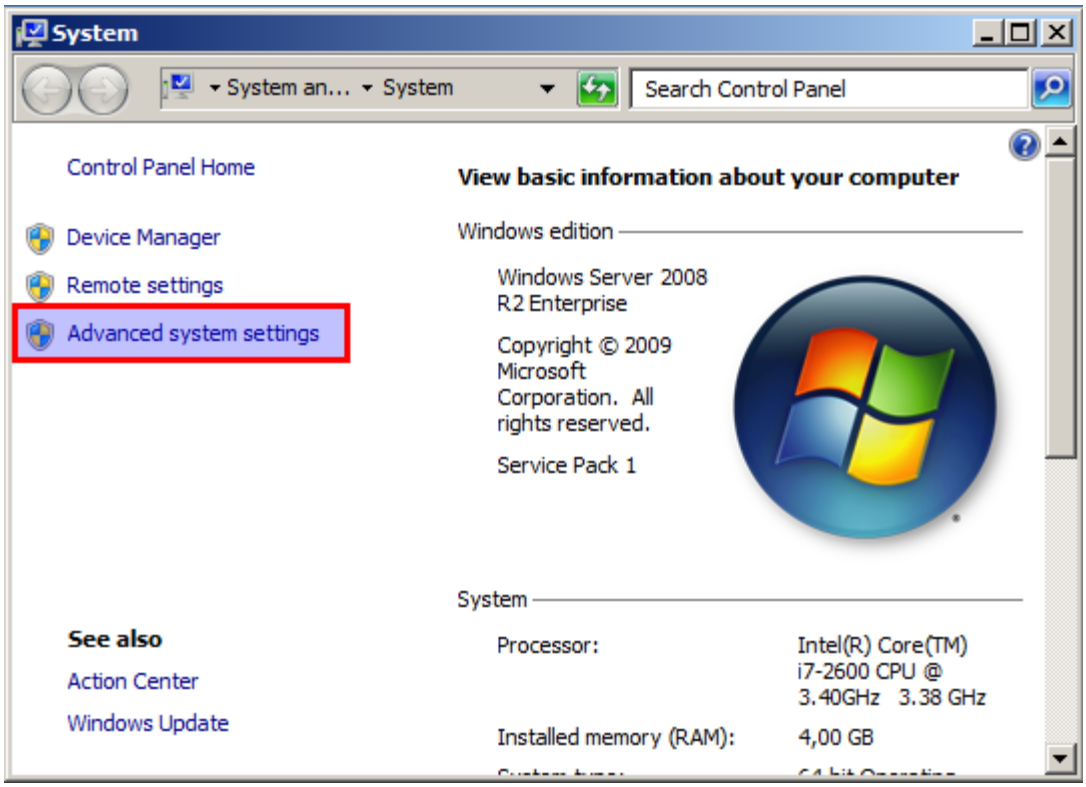
```
setx PATH "%PATH%;C:\tomcat-9.0.76\bin" /M
```

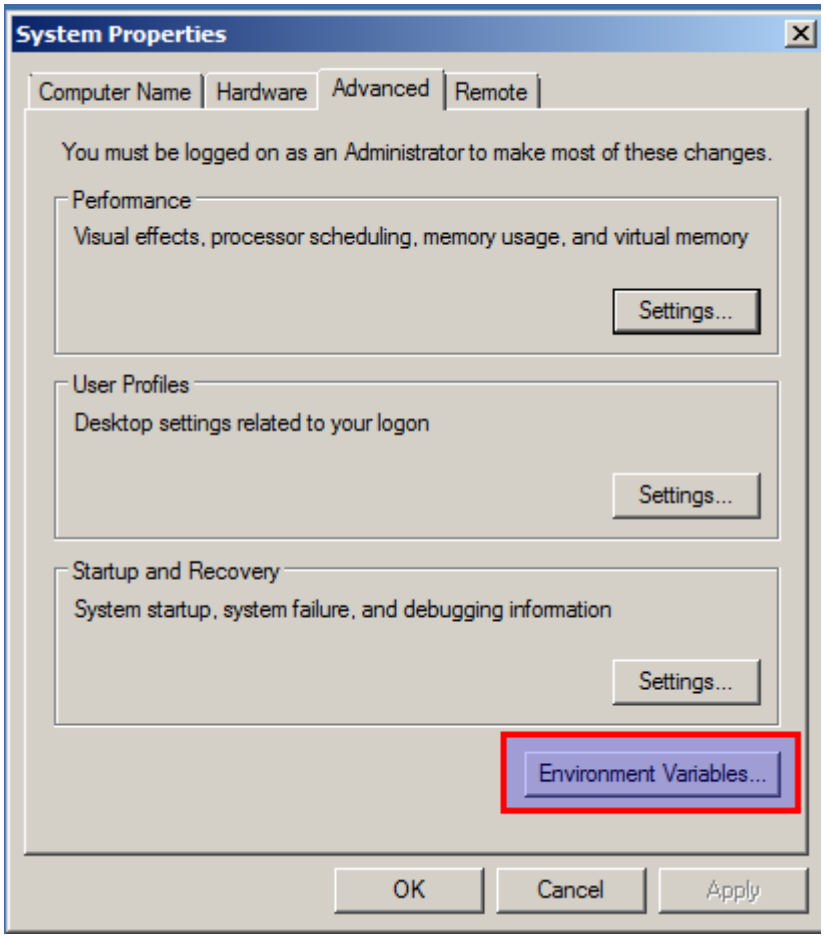
These operations can also be done from Windows environment variables:



The following screens could be quite different depending on your Windows OS version.


Command Prompt
Notepad
Internet Explorer
Geany
MySQL Workbench 5.2 CE
Computer Management
Administrator
Documents
Network
Control P
Devices a
Administr
Open
Manage
Map network drive...
Disconnect network drive...
Show on Desktop
Rename
Properties


System
System an... System
Search Control Panel
Control Panel Home
Device Manager
Remote settings
Advanced system settings
View basic information about your computer
Windows edition
Windows Server 2008 R2 Enterprise
Copyright © 2009 Microsoft Corporation. All rights reserved.
Service Pack 1
System
Processor:
Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz 3.38 GHz
Installed memory (RAM):
4,00 GB
See also
Action Center
Windows Update



Installing OpenKM

With Tomcat stopped, copy the OpenKM.war file into /home/openkm/tomcat-9.0.76/webapps

```
$ copy openkm.war c:\tomcat-9.0.76\webapps
```

Install third-party software:

Software	Required	Description
LibreOffice	Yes	We recommend using LibreOffice because it is compatible with Microsoft Office formats.
Tesseract	No	Download from https://tesseract-ocr.github.io/tessdoc/Installation.html and follow the installation wizard.
ClamAV	No	Download from http://www.clamav.net/download.html and follow the installation wizard.
ImageMagick	No	Tomcat bundle comes with an embedded convert tool.

		If you decide to install it, go to http://www.imagemagick.org/script/binary-releases.php, download the latest binary release, and follow the installation wizard. Take into consideration that 16 bits need more hardware resources than 8 bits. We suggest ImageMagick-6.9.1-7-**Q8**-x64-dll.exe rather than ImageMagick-6.9.1-7-**Q16**-x64-dll.exe (note the version numeration can be distinct due to ImageMagick version upgrade).
GhostScript	No	Tomcat bundle comes with an embedded Ghostscript tool. Go to http://ghostscript.com/download/gsdnld.html and follow the installation wizard if you decide to install it.

There is also an available **Extras.zip** file that contains all this software as a Portable edition.

Start application

Check openkm.properties parameters

```
C:\> geany c:\openkm\openkm.properties
```

Your configuration should be something like this:

```
# OpenKM Hibernate configuration values
hibernate.dialect=com.openkm.db.dialect.MySQL5InnoDBDialect
hibernate.hbm2ddl=create-only

# Initial configuration - Windows
system.libreoffice.path=C:/Program Files (x86)/LibreOffice
```

Ensure **hibernate.dialect=com.openkm.db.dialect.MySQL5InnoDBDialect** is enabled if your database is MySQL.

First-time application start:

```
$ net start OpenKM
```

Use mTail Program - or equivalent - to watch the c:\tomcat-9.0.76\log\openkm.log file.

When the application is started, you'll see it in the log file in the lines:

```
2015-07-04 18:28:10,680 [main] INFO  org.apache.coyote.http11.Http11Protocol - Starting ProtocolHandler ["http-bi
2015-07-04 18:28:10,688 [main] INFO  org.apache.coyote.ajp.AjpProtocol - Starting ProtocolHandler ["ajp-bio-127.0
2015-07-04 18:28:10,692 [main] INFO  org.apache.catalina.startup.Catalina - Server startup in 41456 ms
```

The application will be accessible at http://YOUR_IP:8080/openkm by user:**okmAdmin** with password: **admin**. (Do not forget to change it !). For more information: Manage users and roles.

Configure default extensions:

Go to Administration > Utilities > Database query and execute:

```
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('808e7a42-2e73-470c-ba23-e4c9d5c3a0f4',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('58392af6-2131-413b-b188-1851aa7b651c'
INSERT INTO OKM_PROFILE_MSC_EXTENSION (PEX_ID, PEX_EXTENSION) VALUES (1, '808e7a42-2e73-470
INSERT INTO OKM_PROFILE_MSC_EXTENSION (PEX_ID, PEX_EXTENSION) VALUES (1, '58392af6-2131-413
```

Access from other computers

Access from other computers will be denied by default due to Windows Firewall configuration. To enable access from anywhere, please execute this command:

```
netsh advfirewall firewall add rule name="OpenKM" dir=in action=allow protocol=TCP localport=8080
```

Other standard additional setup configurations

OpenKM comes with many configuration parameters. Please take a look at Recommended configuration parameters.

- Configuring Apache HTTP Reverse-Proxy
- Configuring Tomcat mail
- Configuring Tomcat memory utilization
- Configuring Tomcat ports
- Configuring Tomcat SSL/TLS
- Configuring Tomcat timezone

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Using the installer



The OpenKM installer demands credentials to use it, so in case you want to access you should need to contact the OpenKM staff.

Another way of installing OpenKM is by using the installer tool. This Java application eases installation and works on Ubuntu, Debian, RedHat, CentOS, and Windows platforms. Remember that you need a working Java installation before using this tool. We recommend OpenJDK 11. Please check these pages for more information about installing Java on your server; look for the **Checking Java version** section:

- Installing on RedHat and CentOS
- Installing on Ubuntu and Debian
- Installing on Windows



The installer downloads several packages from our servers, like Tomcat and OpenKM, so it needs to connect to http://download.openkm.com. Once the installation is completed and OpenKM runs, it should have access to http://update.openkm.com to validate the license.

Once Java 8 is installed, you can download the installer from http://download.openkm.com/.



The installer will make the installation in the folder in which it runs in, so before running, move the OKMInstaller.jar to the location where OpenKM should be installed. In the case of running on Linux, it also changes the permissions of the file to a user called "openkm," so if you haven't created such a user, please make it now:

```
$ sudo adduser openkm
```

This command should work in most common Linux distributions.

It's a text-based installer, so you need to execute it from the console as root if you are on Linux or as an administrator on Windows. This is because it needs to install package dependencies and configure OpenKM as a service:

```
$ java -jar OKMInstaller.jar
```

And this is the output:

```
root@darkman-msi /home/openkm $ java -jar OKMInstaller.jar
Running on Linux: 1.5.15 (build: 9b23d69)
Linux distro: linuxmint (Linux Mint - 20.3)
Local host: darkman-msi (127.0.1.1)
Java version: 11.0.20.1
Free disk size: 100.7 GB
RAM size: 15.5 GB
CPU cores: 12
Server IP: 192.168.1.106
Supported databases: [h2, hsqldb, mariadb, mysql, oracle, postgresql, sqlserver
Current directory: /home/openkm

Database [h2]: mariadb
Database host [localhost]:
Database name [okmdb]:
Database user [openkm]:
Database password [dy#RQil#s4G5]:
---------------------------
- Database: mariadb
- Database host: localhost
- Database name: okmdb
- Database user: openkm
- Database password: dy#RQil#s4G5
- OpenKM version: 8.1.4
- Tomcat version: 9.0.76
---------------------------
Start install process? [y/N]:
```

Values in brackets are the default ones. If you hit Enter, this is the chosen value. If you want a different one, write the string. In the end, the installed will show the entered information, and you can verify it and proceed or enter these values again.



The installer will ask for an authorized user and password. If you don't already have this information, please open a support ticket at http://support.openkm.com or contact your sales agent.

If you press "y," the installer will ask for an authorized user and password. Once the user and password are validated, the installation process will start. Depending on your Internet connection and server performance, this process may take several minutes. In the case of Linux-based systems, it will install all required dependencies like ImageMagick and LibreOffice.



If you access the Internet using a proxy, configure it when executing the application:

```
$ java -Dhttp.proxyHost=192.168.0.1 -Dhttp.proxyPort=3128 -jar OKMInstaller.jar
```

You may also need **https.proxyHost** and **https.proxyPort** or even **http.proxyUser**, **http.proxyPassword**, **https.proxyUser,** and **https.proxyPassword** parameters.

Supported databases

The installer supports these databases:

- MariaDB (mariadb)
- MySQL (MySQL)
- PostgreSQL (posgresql)
- SQL Server (sqlserver)
- Oracle (oracle)
- HSQLDB (hsqldb)
- H2 (h2)

Other options

The installer will also check if port 8080 is already being used, which is the default port used by Tomcat and should be free. If you want to disable this check, use the **-p** or **--port** parameters.

```
$ java -jar OKMInstaller.jar -p
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Using Docker

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers with everything the software needs to run, including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.

A Docker image is available at our private Docker repository at https://docker.openkm.com. It is the easiest way to get OpenKM running on your server. If you want to use this image, please submit a support ticket, and we will provide you with access to the repository.

Please refer to Docker documentation for installation instructions and to learn more about Docker.

Once Docker has been installed, create the proper **openkm.properties** file or configure it using the environment variables and execute this command to get your OpenKM instance:

```
$ docker run --name openkm -p 8080:8080 -v ${PWD}/openkm.properties docker.openkm.com/private/professional
```



Remember that you need a database in another container that the OpenKM container will use.

After that, point your browser to http://localhost:8080/. The default user and password is "okmAdmin" / "admin".

This way, you can create a container with the default embedded H2 database (not recommended for production, but just fine for testing):

```
$ docker run -d --name openkm -p 8080:8080 -v myvol:/opt/tomcat \
  -e DATABASE_DRIVER=org.h2.Driver \
  -e DATABASE_URL=jdbc:h2:/opt/tomcat/okmdb \
  -e DATABASE_DIALECT=com.openkm.db.dialect.H2Dialect \
  --restart=unless-stopped docker.openkm.com/private/professional:7.1.13
```



If you start a container that creates a new volume, as above, and the container has files in the directory to be mounted (like /opt/tomcat), the directory's contents are copied into the volume. The container then mounts and uses the book, so the volume initially has pre-populated content from the image.

This is a sample that can be used to create a container using MySQL, but the recommended way is using Docker Compose:

```
$ docker run -d --name openkm -p 8080:8080 -v myvol:/opt/tomcat \
  -e DATABASE_DRIVER="com.mysql.cj.jdbc.Driver" \
  -e DATABASE_URL="jdbc:mysql://172.17.0.1:3306/dbtest?
useUnicode=true&characterEncoding=UTF8&serverTimezone=CET&nullNamePatternMatchesAll=true" \
  -e DATABASE_USER="openkm" \
```

```
-e DATABASE_PASSWORD="openkm" \
-e DATABASE_DIALECT="com.openkm.db.dialect.MySQL5InnoDBDialect" \
--restart=unless-stopped docker.openkm.com/private/professional:7.1.13
```

The database can be created this way:

```
CREATE USER 'openkm'@'%' IDENTIFIED BY 'openkm';
CREATE DATABASE dbtest DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_bin;
GRANT ALL ON dbtest.* TO 'openkm'@'%' WITH GRANT OPTION;
```

To see the container logs:

```
$ docker logs -f openkm
```

To stop the container:

```
$ Docker stop openkm
```

To start it again:

```
$ docker start openkm
```

To access the container:

```
$ docker exec -ti openkm bash
```

For more actions, please check the Docker documentation site.

Configuration

These are the environment variables that can be used to configure the OpenKM container:

Variable	Description
DATABASE_DRIVER	The JDBC database driver connects to the database.
DATABASE_URL	The JDBC URL to connect to the database.
DATABASE_USER	The database user.
DATABASE_PASSWORD	The database user password.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

DATABASE_DIALECT	The Hibernate database dialect.

Docker Compose

To make all this configuration easy, we recommend using Docker Compose. Please, refer to Docker Compose documentation for installation instructions and more info.

This is a sample Docker Compose configuration file using environment variables, where MySQL is used as the database:

```
version: '3.2'

services:

  openkm_mysql:
    image: docker.openkm.com/private/professional:7.1.13
    container_name: openkm_mysql
    hostname: openkm
    ports:
      - 8080:8080
    volumes:
      - openkm_mysql:/opt/tomcat
    environment:
      - DATABASE_DRIVER=com.mysql.cj.jdbc.Driver
      - DATABASE_URL=jdbc:mysql://mysql:3306/okmdb?useUnicode=true&characterEncoding=UTF8&serverTimez
      - DATABASE_USER=openkm
      - DATABASE_PASSWORD=openkm
      - DATABASE_DIALECT=com.openkm.db.dialect.MySQL5InnoDBDialect
      - TZ=Europe/Madrid
    entrypoint: [ "wait-for-it.sh", "mysql:3306", "--timeout=0", "--strict", "--", "entrypoint.sh" ]
    links:
      - mysql:mysql
    networks:
      skynet:
        ipv4_address: 172.28.1.1

  mysql:
    image: mysql:8.0.18
    container_name: okmdb_mysql
    hostname: mysql
    command: --default-authentication-plugin=mysql_native_password --character-set-server=utf8 --collation-server=
    volumes:
      - okmdb_mysql:/var/lib/mysql
    environment:
      - MYSQL_DATABASE=okmdb
      - MYSQL_USER=openkm
      - MYSQL_PASSWORD=openkm
      - MYSQL_ROOT_PASSWORD=openkm
    security_opt:
      - seccomp:unconfined
    networks:
      skynet:
        ipv4_address: 172.28.1.2

volumes:
  openkm_mysql:
  okmdb_mysql:

networks:
  skynet:
    ipam:
      driver: default
```

```
      config:
        - subnet: 172.28.0.0/16
```

This is another sample Docker Compose configuration:

```
version: '3.2'

services:

  openkm_mysql:
    image: docker.openkm.com/private/professional:7.1.13
    container_name: openkm_mysql
    hostname: openkm
    ports:
      - 8080:8080
    volumes:
      - ${PWD}/openkm.properties:/opt/tomcat/openkm.properties
      - ${PWD}/repository:/opt/tomcat/repository
    environment:
      - TZ=Europe/Madrid
    entrypoint: [ "wait-for-it.sh", "mysql:3306", "--timeout=0", "--strict", "--", "entrypoint.sh" ]
    links:
      - mysql:mysql
    networks:
      skynet:
        ipv4_address: 172.28.1.1

  mysql:
    image: mysql:8.0.18
    container_name: okmdb_mysql
    hostname: mysql
    command: --default-authentication-plugin=mysql_native_password --character-set-server=utf8 --collation-server=u
    volumes:
      - ${PWD}/mysql:/var/lib/mysql
    environment:
      - MYSQL_DATABASE=okmdb
      - MYSQL_USER=openkm
      - MYSQL_PASSWORD=openkm
      - MYSQL_ROOT_PASSWORD=openkm
    security_opt:
      - seccomp:unconfined
    networks:
      skynet:
        ipv4_address: 172.28.1.2

networks:
  skynet:
    ipam:
      driver: default
      config:
        - subnet: 172.28.0.0/16
```

This is the **openkm.properties** file:

```
# OpenKM Hibernate configuration values
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://mysql:3306/okmdb?useUnicode=true&characterEncoding=UTF8&serverTimezor
spring.datasource.username=openkm
spring.datasource.password=openkm

# JPA stuff
```

```
spring.jpa.hibernate.ddl-auto=create-only
spring.jpa.properties.hibernate.dialect=com.openkm.db.dialect.MySQL5InnoDBDialect

# Cluster
cluster.enabled=false
cluster.node=master

# Logging config
spring.output.ansi.enabled=always
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Installing on a pristine Tomcat

This section will depict how to deploy OpenKM in a pristine Apache Tomcat.

Preliminaries

Take a look at the installation of the whole system according to your Operating System:

- Installing on RedHat and CentOS (OLD)
- Installing on Ubuntu and Debian (OLD)
- Installing on Windows

Installing Tomcat

Download the latest Tomcat version (Tomcat 9) from the Apache Tomcat page and extract it into the directory you want to install.

Once Tomcat has been extracted, some modifications are necessary.

Copy libraries and configuration files

Copy the following required libraries to TOMCAT_HOME/lib. These files can be found in the Tomcat already configured by the OpenKM team, which can be downloaded from http://download.openkm.com/okm/tomcat-9.0.82.zip

- unicode.ttf

Copy the following files to TOMCAT_HOME:

- openkm.properties
- logback.xml

Modify configuration files

Modify the following files from TOMCAT_HOME/conf:

setenv.sh or **setenv.bat**

Depending on your Operating System, you may have to create a file called **setenv.sh** in Linux or **setenv.bat** in Windows. This file should be placed in **TOMCAT_HOME/bin**. This is the context in the case of Linux:

```
JAVA_OPTS="$JAVA_OPTS -Xms512m -Xmx4096m -XX:+UseG1GC -XX:+UseStringDeduplication"
JAVA_OPTS="$JAVA_OPTS -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=$CATALINA_HOME"
JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv4Stack=true -Dmail.mime.ignoreunknownencoding=true"
JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8 -Djava.awt.headless=true"
JAVA_OPTS="$JAVA_OPTS -Dsun.java2d.cmm=sun.java2d.cmm.kcms.KcmsServiceProvider"
CATALINA_PID=$CATALINA_HOME/catalina.pid
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CATALINA_HOME/bin/cadviewer
```

And this is the content in the case of Windows:

```
set JAVA_OPTS=%JAVA_OPTS% -Xms512m -Xmx4096m -XX:+UseG1GC -XX:+UseStringDeduplication
set JAVA_OPTS=%JAVA_OPTS% -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=%CATALINA_HO
set JAVA_OPTS=%JAVA_OPTS% -Djava.net.preferIPv4Stack=true -Dmail.mime.ignoreunknownencoding=true
set JAVA_OPTS=%JAVA_OPTS% -Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8 -Djava.awt.headless=true
set PATH=%PATH%;%CATALINA_HOME%\bin\cadviewer
```

Installing OpenKM

With Tomcat stopped, copy the OpenKM.war file into TOMCAT_HOME \webapps

```
$ mv openkm.war $TOMCAT_HOME/webapps
```

Install third-party software

To do this, go to the section corresponding to your Operating System:

- Installing on WindowsInstalling on Ubuntu and Debian
- Installing on RedHat and CentOS
- Installing on Windows

Start application

Check openkm.properties parameters

```
$ vim TOMCAT_HOME/openkm.properties
```

Your configuration file should contain something like this:

```
# OpenKM Hibernate configuration values
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/okmdb?autoReconnect=true&useUnicode=true&characterEncodi
spring.datasource.username=openkm
spring.datasource.password=openkm

# JPA stuff
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect=com.openkm.db.dialect.MySQL5InnoDBDialect

# Logback configuration file
logback.config=logback.xml
```



This configuration works if your database is MySQL. In another case, please check the documentation for creating a support ticket.

First-time application start:

```
$ start openkm
```

Use mTail Program - or equivalent - to watch the TOMCAT_HOME\log\openkm.log file.

When the application starts, you should see these log messages:

```
2015-07-04 18:28:10,680 [main] INFO  org.apache.coyote.http11.Http11Protocol - Starting ProtocolHandler ["http-bi
2015-07-04 18:28:10,688 [main] INFO  org.apache.coyote.ajp.AjpProtocol - Starting ProtocolHandler ["ajp-bio-127.0
2015-07-04 18:28:10,692 [main] INFO  org.apache.catalina.startup.Catalina - Server startup in 41456 ms
```

The application will be accessible at http://YOUR_IP:8080/openkm using user: **okmAdmin** with password: **admin**. (Do not forget to change it !). For more information: Manage users and roles.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Database

OpenKM stores most of its data in a SQL database. Several databases are supported, but they must be configured to work correctly.

Configure database

- Configuring MySQL and MariaDB
- Configuring Oracle
- Configuring PostgreSQL
- Configuring SQL Server
- Configuring Hikari pool

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring MySQL and MariaDB

MariaDB is a community-developed split of MySQL intended to remain free under the GNU GPL. As a split of a leading Open-Source software system, it is notable for being led by the original MySQL developers. They separated it due to concerns over Oracle's acquisition.

MariaDB has supported database sequences since v10.3, but we recommend using v10.6 or newer right now. See MariaDB Server Releases.

Preliminaries

Perform all changes with the application stopped.

Check if your MySQL installation has the InnoDB engine enabled:

```
$ MySQL -h localhost -u root -p
> show engines;
```

It should show something like this:

| **InnoDB** | **DEFAULT** | Supports transactions, row-level locking, and foreign keys | YES | YES | YES |

In the case of another default engine, there are two options:

1. Use the **com.openkm.db.dialect.MySQL5InnoDBDialect** dialect and avoid changing the default MySQL Storage Engine.
2. Change default engine

Change default engine

Modify the MySQL configuration file named **my. cnf**

and under [mysqld], add

```
default-storage-engine = innodb
```

After changing the default to MySQL, the service must be restarted to take effect.

Change default charset

In Ubuntu 16.04, the default charset configured in MariaDB and MySQL is **utf8mb4,** but we recommend switching to **utf8**. To get the recommended charset, you have to modify these files:

/etc/mysql/mariadb.conf.d/50-server.cnf

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
character-set-server = utf8

collation-server = utf8_general_ci
```

/etc/mysql/mariadb.conf.d/50-client.cnf

```
default-character-set = utf8
```

/etc/MySQL/mariadb.conf.d/50-MySQL-clients.cnf

```
default-character-set = utf8
```

Once modified, a database server restart is needed.

Database creation

Create a database and user

```
DROP DATABASE IF EXISTS okmdb;
CREATE DATABASE okmdb DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_bin;
CREATE USER openkm@localhost IDENTIFIED BY '*secret*';
GRANT ALL ON okmdb.* TO openkm@localhost WITH GRANT OPTION;
```

You can check the database engine with the following:

```
$ MySQL show -h localhost -u root -p --status okmdb;
```

More information at MySQL: Case Sensitivity in String Searches

Create a database in Persian

Set Persian collation and character set in the my.ini configuration file (requires to restart the service):

```
collation_server = utf8mb4_persian_ci
character_set_server = utf8mb4
```

Set Persian collation and character set in the current mysql session (after restarting the service will be set the default values from the my.ini):

```
SET collation_server = 'utf8mb4_persian_ci';
SET character_set_server = 'utf8mb4';
```

Database:

```
CREATE DATABASE okmdb CHARACTER SET utf8mb4 COLLATE utf8mb4_persian_ci;

CREATE USER openkm@localhost IDENTIFIED BY 'k4M0j4lBi#ed';
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
GRANT ALL ON okmdb.* TO openkm@localhost WITH GRANT OPTION;
```

Configure your openkm.properties



This configuration property should be set before the database creation. Once the database has been initialized, don't modify it because it can damage your installation.

If your OpenKM installation has already been configured with another database, you can't switch to another database by simply changing this property.

Choose the Right Dialect based on your Database engine version.

MySQL dialects:

- com.openkm.db.dialect.MySQL55Dialect
- com.openkm.db.dialect.MySQL57Dialect
- com.openkm.db.dialect.MySQL8Dialect

MariaDB dialects:

- com.openkm.db.dialect.MariaDBDialect
- com.openkm.db.dialect.MariaDB53Dialect
- com.openkm.db.dialect.MariaDB10Dialect
- com.openkm.db.dialect.MariaDB102Dialect
- com.openkm.db.dialect.MariaDB103Dialect
- com.openkm.db.dialect.MariaDB106Dialect

Deprecated dialects:

- com.openkm.db.dialect.MySQL5InnoDBDialect
- com.openkm.db.dialect.MySQL57InnoDBDialect

Edit the file $. TOMCAT_HOME/openkm.properties

```
spring.jpa.hibernate.ddl-auto=create-only
spring.jpa.properties.hibernate.dialect=com.openkm.db.dialect.MySQL5InnoDBDialect
```

MySQL Timezone

If you want to set your timezone in your JDBC connection according to the timezone configured in your system, first, you can see the mappings both for MySQL and MariaDB in this way:

```
SELECT * FROM MySQL.time_zone_name
```

So, for example, you could configure your JDBC connection like this if your system timezone is UTC+1:

```
spring.data source.url=jdbc:mysql://localhost:3306/okmdb?
autoReconnect=true&useUnicode=true&characterEncoding=UTF8&nullNamePatternMatchesAll=true&serverTimezone=Euro
```

Where **Europe/Madrid** is a valid value taken from the table **time_zone_name.**

Configure Tomcat data sources

Edit the file $TOMCAT_HOME/openkm.properties and enable the resource named JDBC/OpenKMDS

MySQL:

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/okmdb?autoReconnect=true&useUnicode=true&characterEncodi
spring.datasource.username=openkm
spring.datasource.password=*secret*
```

MariaDB:

```
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://localhost:3306/okmdb?autoReconnect=true&useUnicode=true&characterEnco
spring.datasource.username=openkm
spring.datasource.password=*secret*
```



MariaDB can use these driverClassName values:

- com.MySQL.jdbc.Driver
- org.mariadb.JDBC.Driver

In recent MySQL versions, you need to add **useSSL=false** to the connection URL.



Configures the values of the attributes named:

- Username.
- Password.
- URL (change to your hosts and port).

It can only use a resource called JDBC/OpenKMDS.

Configure application login

Edit the file $. TOMCAT_HOME/openkm.properties

```
authentication.openkm.database=true
```

Run application

As a process into OpenKM startup, the application will automatically create an empty database structure.

Once the tables are created, OpenKM will automatically change the **hibernate.hbm2ddl** property from ***create-only*** *to* ***none***.

Additional information

- *InnoDB Performance Tuning Tips*
- *InnoDB - High disk write I/O on ibdata1 file and ib_logfile0*

Remove and create MySQL service in Windows.

Ensure your MySQL/bin folder is in your PATH at environment variables.

For example, in MySQL 5.7, the bin folder location is at C:\Program Files\MySQL\MySQL Server 5.7\bin

If you have MySQL Windows service, you can remove it with one of these command lines:

```
c:\> mysqld --remove MySQL57
```

or

```
c:\> sc delete MySQL57
```

Create the service:

```
c:\> "C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqld.exe" --install MYSQL57 --defaults-file="C:\ProgramData\MySQL\MySQL Server 5.7\my.ini"
```

More information at http://dev.mysql.com/doc/refman/5.7/en/windows-start-service.html

Migrate from Windows to Linux

By default table, names in Windows are lowercase and in Linux in uppercase.

When you migrate from Windows to Linux, you must convert lowercase table names to uppercase. The following script will help with it:

```
select concat('rename table ', table_name, ' to ' , upper(table_name) , ';') from information_schema.tables where tab
```

Troubleshooting

Error ASCII '\\0' appeared in the statement" when importing a database dump.

It is possible that the error is caused by the encoding of the SQL file. You should check the file format. For example, with the command:

```
> file okmdb.sql
okmdb.sql: Little-endian UTF-16 Unicode text, with very long lines, with CRLF line terminators
```

In the case of a file in UTF-16 format, you should convert it to UTF-8. For example, with the command:

```
> iconv -f utf-16 -t utf-8 okmdb.sql > okmdb-good.sql
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

MySQL tuning

Take this documentation as is, not as a definitive guide to tuning MySQL. A lot of information and knowledge should be read before changing MySQL configuration. The documentation has been created based on experience shared by OpenKM server administrator users.

Useful links should read:

- *MySQL: InnoDB Performance Tuning Tips*
- *MySQL: The MySQL Query Cache*
- *InnoDB - High disk write I/O on ibdata1 file and ib_logfile0*

Sample one

Server information

- *Virtualized Windows Server 2008*
- *CPU: Intel Xeon X3470 @2,93GHz*
- *RAM: 7GB*
- *JVM settings for Tomcat: -Xms 3072 -Xmx 3072 -XX:PermSize=256m -XX:MaxPermSize=512m*
- *The repository is 1GB*

MySQL my. cnf changes

- *innodb_buffer_pool_size=1792M*
- *innodb_additional_mem_pool_size=16M*
- *innodb_log_file_size=256M*
- *innodb_log_buffer_size=8M*

Using MySQL server InnoDB engine with the default configuration parameters in my.cnf file can cause a severe performance bottleneck in applications with high database read-write activity. MySQL should have enough RAM to cache the repository and avoid high Disk I/O activity.

The most important parameters are:

- ***innodb_buffer_pool_size***
- ***innodb_additional_mem_pool_size***
- ***innodb_log_file_size***
- ***innodb_log_buffer_size***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

According to some database experts, innodb_buffer_pool_size should occupy 70 percent of the RAM of a dedicated database server.

Before changing these values, please read MySQL documentation carefully. Before making any changes, you should ensure that the MySQL server has been stopped correctly (check the server log). Otherwise, the innodb will become corrupted, and the MySQL server won't restart.

Considerations *Repository is 1GB, so 1792M is enough RAM for* ***innodb_buffer_pool_size****. The larger the repository, the more memory you need to allocate to* ***innodb_buffer_pool_size****. Using an x86 MySQL server, you cannot allocate RAM larger than 2GB.*

Information collected from OpenKM forum posts.

Sample two

Server information

- *CPU: Intel Xeon E5606 @2.13GHz*
- *RAM: 4GB*
- *System: Windows server 2008, 64bit*
- *JAVA_OPTS=-Xms256m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m -Djava.awt.headless=true -Dfile.encoding=utf-8*
- *Database size:2.5G*
- *Repository size:40GB*
- *100,000 nodes(folders + documents)*

MySQL my.cnf changes

- *query_cache_size=0*
- *table_open_cache=2000*
- *tmp_table_size=16M*
- *thread_cache_size=9*
- *myisam_max_sort_file_size=100G*
- *myisam_sort_buffer_size=32M*
- *key_buffer_size=8M*
- *read_buffer_size=64K*
- *read_rnd_buffer_size=256K*
- *sort_buffer_size=256K*
- *innodb_additional_mem_pool_size=16M*

- *innodb_log_buffer_size=8M*
- *innodb_buffer_pool_size=1042M*

Results

Before optimization, the login time was about 45-55 seconds, with a lot of hard-disk I/O usage. After the MySQL optimization, login takes about 10 seconds.



With more RAM, getting a higher MySQL buffer and probably better performance will be possible.

Information collected OpenKM Forum post.

Change MySQL data directory

Ubuntu

Stop the service:

```
$ service MySQL stop
```

Create a new folder destination:

```
$ mkdir /new_path/mysql_data
$ cp -R -p /var/lib/MySQL /new_path/mysql_data
```



MySQL users should have enough credentials granted to the new folder /new_path/mysql_data.

Optionally modify the path of older MySQL data store folder location:

```
$ mv /var/lib/MySQL /var/lib/mysql_old
```

*Edit the **datadir** property into the **/etc/mysql/my.cnf** configuration file:*

```
$ vim /etc/mysql/my.cnf
```



Other possible config paths:

/etc/mysql/mariadb.conf.d/50-server.cnf

/etc/MySQL/MySQL.conf.d/mysqld.cnf

Change the older data store folder path with the newer:

```
data-dir = /var/lib/MySQL
```

to

```
data-dir = /new_path/mysql_data
```

*Enable AppArmor. Edit the file **/etc/AppArmor.d/usr.in.mysqld**:*

```
$ vim /etc/apparmor.d/usr.sbin.mysqld
```

Change the older data store folder path with the newer:

```
/usr/lib/MySQL/plugin/ r,
/usr/lib/MySQL/plugin/*.so* mr,
...
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
/var/lib/MySQL/ r,
/var/lib/MySQL/** rwk,
```

to

```
/new_path/mysql_data/plugin/ r,
/new_path/mysql_data/plugin/*.so* mr,
...
/new_path/mysql_data/ r,
/new_path/mysql_data/** rwk,
```

Start MySQL service:

```
$ service MySQL start
```

More information at http://www.ubuntugeek.com/how-to-change-the-mysql-data-default-directory.html.

CentOS and RHEL

Stop the service:

```
$ service mysqld stop
```

Create a new folder destination:

```
$ mkdir /new_path/mysql_data
$ cp -R -p /var/lib/MySQL /new_path/mysql_data
```

The MySQL user must have enough credentials granted to the new folder /new_path/mysql_data.

Optionally modify the path of older MySQL data store folder location:

```
$ mv /var/lib/MySQL /var/lib/mysql_old
```

*Edit the **datadir** property into the **/etc/mysql/my.cnf** configuration file:*

```
$ vim /etc/mysql/my.cnf
```

Change the older data store folder path with the newer:

```
data-dir = /var/lib/MySQL
```

to

```
data-dir = /new_path/mysql_data
```

Enable AppArmor.

```
$ yum install policycoreutils-python
$ getenforce
$ manage context -a -t mysqld_db_t "/new_path/mysql_data(/.*)?"
$ restorecon -Rv /new_path/mysql_data
```

Start MySQL service:

```
$ service mysqld start
```

More information at http://lifeofageekadmin.com/how-to-change-the-default-mysql-data-directory-on-rhel-6/

Windows

Stop the service:

```
C:\> net stop mysql56
```

Create a new folder destination:

```
C:\> robocop "C:\ProgramData\MySQL\MySQL Server 8\data" c:\new_path\mysql_data
```

Optionally modify the path of older MySQL data store folder location:

```
C:\> ren "C:\ProgramData\MySQL\MySQL Server 8\data" "C:\ProgramData\MySQL\MySQL Server 8\data_old"
```

*Edit the **datadir** property into the **C:\ProgramData\MySQL\MySQL Server 8\my.ini** configuration file:*

Change the older data store folder path with the newer:

```
datadir = C:\ProgramData\MySQL\MySQL Server 8\data
```

to

```
data-dir = c:\new_path\mysql_data
```

Start MySQL service:

```
C:\> net start mysql56
```

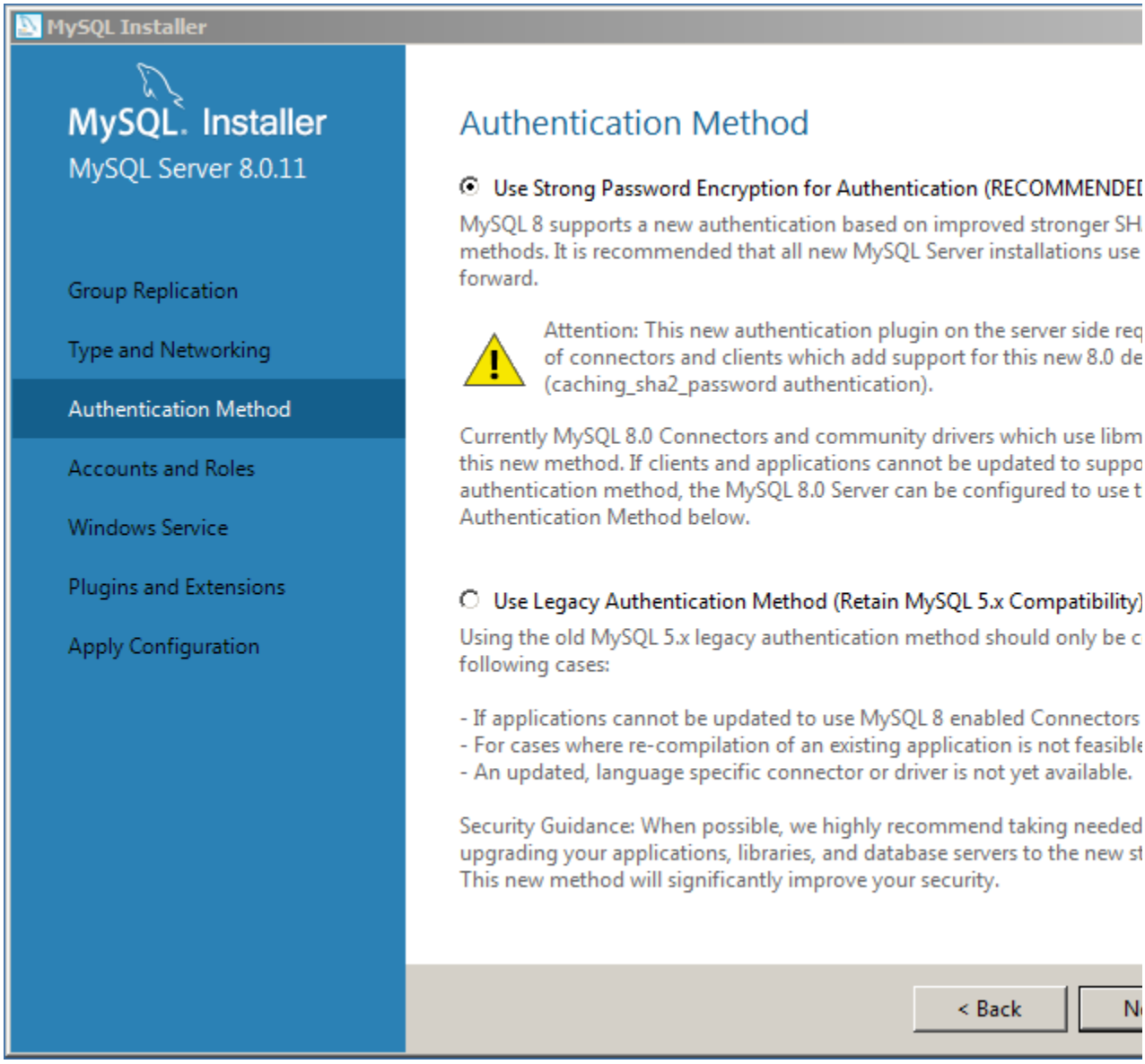
MySQL 8 troubleshooting

Could not create a connection to the database server

There might be several reasons why the application cannot establish the link to the database; look at the log file, which might give clues about what is happening.

Strong Password Encryption for Authentication

*The latest MySQL installation wizard suggests you use "**Strong Password Encryption for Authentication,**" as can be shown in the following screenshot:*



*When the option "**Strong Password Encryption for Authentication**" is enabled rather than "**Legacy Authentication Method,**"*

*you are enabling an **SSL connection by default**. That means you might read some literature and apply changes in your connection setting to your server.xml server to connect with the database. Usually, in this scenario, you will get some error like what is shown below:*

```
java.sql.SQLException: Cannot create PoolableConnectionFactory (Could not create connection to database server
 at org.apache.tomcat.dbcp.dbcp2.BasicDataSource.createPoolableConnectionFactory(BasicDataSource.java:2303
 at org.apache.tomcat.dbcp.dbcp2.BasicDataSource.createDataSource(BasicDataSource.java:2043) ~[tomcat-dbcp
 at org.apache.tomcat.dbcp.dbcp2.BasicDataSource.getConnection(BasicDataSource.java:1543) ~[tomcat-dbcp.jar:
 at org.hibernate.connection.DatasourceConnectionProvider.getConnection(DatasourceConnectionProvider.java:92
 at org.hibernate.cfg.SettingsFactory.buildSettings(SettingsFactory.java:113) ~[hibernate-core-3.6.10.Final.jar:3.6.10
 at org.hibernate.cfg.Configuration.buildSettingsInternal(Configuration.java:2863) [hibernate-core-3.6.10.Final.jar:3.6
 at org.hibernate.cfg.Configuration.buildSettings(Configuration.java:2859) [hibernate-core-3.6.10.Final.jar:3.6.10.Fin
 at org.hibernate.cfg.Configuration.buildSessionFactory(Configuration.java:1870) [hibernate-core-3.6.10.Final.jar:3.6
 at com.openkm.dao.HibernateUtil.getSessionFactory(HibernateUtil.java:367) [classes/:6.4.43-DEV]
 at com.openkm.dao.HibernateUtil.getSessionFactory(HibernateUtil.java:182) [classes/:6.4.43-DEV]
 at com.openkm.servlet.RepositoryStartupServlet.init(RepositoryStartupServlet.java:99) [classes/:6.4.43-DEV]
 at javax.servlet.GenericServlet.init(GenericServlet.java:158) [servlet-api.jar:3.1.FR]
 at org.apache.catalina.core.StandardWrapper.initServlet(StandardWrapper.java:1183) [catalina.jar:8.5.14]
 at org.apache.catalina.core.StandardWrapper.loadServlet(StandardWrapper.java:1099) [catalina.jar:8.5.14]
 at org.apache.catalina.core.StandardWrapper.load(StandardWrapper.java:989) [catalina.jar:8.5.14]
 at org.apache.catalina.core.StandardContext.loadOnStartup(StandardContext.java:4931) [catalina.jar:8.5.14]
 at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5241) [catalina.jar:8.5.14]
 at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150) [catalina.jar:8.5.14]
 at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:752) [catalina.jar:8.5.14]
 at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:728) [catalina.jar:8.5.14]
 at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:734) [catalina.jar:8.5.14]
 at org.apache.catalina.startup.HostConfig.deployWAR(HostConfig.java:952) [catalina.jar:8.5.14]
 at org.apache.catalina.startup.HostConfig$DeployWar.run(HostConfig.java:1823) [catalina.jar:8.5.14]
 at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) [na:1.8.0_171]
 at java.util.concurrent.FutureTask.run(FutureTask.java:266) [na:1.8.0_171]
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) [na:1.8.0_171]
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) [na:1.8.0_171]
 at java.lang.Thread.run(Thread.java:748) [na:1.8.0_171]
Caused by: java.sql.SQLNonTransientConnectionException: Could not create connection to database server. Attem
 at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:108) ~[mysql-connector-java-8.0.11
 at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:95) ~[mysql-connector-java-8.0.11.ja
 at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:87) ~[mysql-connector-java-8.0.11.ja
 at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:61) ~[mysql-connector-java-8.0.11.ja
 at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:71) ~[mysql-connector-java-8.0.11.ja
 at com.mysql.cj.jdbc.ConnectionImpl.connectWithRetries(ConnectionImpl.java:932) ~[mysql-connector-java-8.0.11
 at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:857) ~[mysql-connector-java-8.0.11.jar:8.0
 at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:444) ~[mysql-connector-java-8.0.11.jar:8.0.11]
 at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:230) ~[mysql-connector-java-8.0.11.jar:8.0.1
 at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:226) ~[mysql-connector-java-8.0.11.
 at org.apache.tomcat.dbcp.dbcp2.DriverConnectionFactory.createConnection(DriverConnectionFactory.java:38) ~[t
 at org.apache.tomcat.dbcp.dbcp2.PoolableConnectionFactory.makeObject(PoolableConnectionFactory.java:257) ~
 at org.apache.tomcat.dbcp.dbcp2.BasicDataSource.validateConnectionFactory(BasicDataSource.java:2313) ~[tom
 at org.apache.tomcat.dbcp.dbcp2.BasicDataSource.createPoolableConnectionFactory(BasicDataSource.java:2299
 ... 27 common frames omitted
Caused by: com.mysql.cj.exceptions.CJCommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets fror
 at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method) ~[na:1.8.0_171]
 at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62) ~[na:1.8.0_171
 at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45) ~[na:1
 at java.lang.reflect.Constructor.newInstance(Constructor.java:423) ~[na:1.8.0_171]
 at com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:59) ~[mysql-connector-java-8
 at com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:103) ~[mysql-connector-java-
 at com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:149) ~[mysql-connector-java-
 at com.mysql.cj.exceptions.ExceptionFactory.createCommunicationsException(ExceptionFactory.java:165) ~[mysq
 at com.mysql.cj.protocol.a.NativeProtocol.negotiateSSLConnection(NativeProtocol.java:355) ~[mysql-connector-jav
 at com.mysql.cj.protocol.a.NativeAuthenticationProvider.negotiateSSLConnection(NativeAuthenticationProvider.jav
 at com.mysql.cj.protocol.a.NativeAuthenticationProvider.proceedHandshakeWithPluggableAuthentication(NativeAu
```

```
at com.mysql.cj.protocol.a.NativeAuthenticationProvider.connect(NativeAuthenticationProvider.java:217) ~[mysql-c
at com.mysql.cj.protocol.a.NativeProtocol.connect(NativeProtocol.java:1411) ~[mysql-connector-java-8.0.11.jar:8.0
at com.mysql.cj.NativeSession.connect(NativeSession.java:165) ~[mysql-connector-java-8.0.11.jar:8.0.11]
at com.mysql.cj.jdbc.ConnectionImpl.connectWithRetries(ConnectionImpl.java:876) ~[mysql-connector-java-8.0.11.
... 35 common frames omitted
Caused by: javax.net.ssl.SSLHandshakeException: java.security.cert.CertificateException: java.security.cert.CertPa
at sun.security.ssl.Alerts.getSSLException(Alerts.java:192) ~[na:1.8.0_171]
at sun.security.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1964) ~[na:1.8.0_171]
at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:328) ~[na:1.8.0_171]
at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:322) ~[na:1.8.0_171]
at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1614) ~[na:1.8.0_171]
at sun.security.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:216) ~[na:1.8.0_171]
at sun.security.ssl.Handshaker.processLoop(Handshaker.java:1052) ~[na:1.8.0_171]
at sun.security.ssl.Handshaker.process_record(Handshaker.java:987) ~[na:1.8.0_171]
at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:1072) ~[na:1.8.0_171]
at sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1385) ~[na:1.8.0_171]
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1413) ~[na:1.8.0_171]
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1397) ~[na:1.8.0_171]
at com.mysql.cj.protocol.ExportControlled.performTlsHandshake(ExportControlled.java:213) ~[mysql-connector-jav
at com.mysql.cj.protocol.StandardSocketFactory.performTlsHandshake(StandardSocketFactory.java:206) ~[mysql-
at com.mysql.cj.protocol.a.NativeSocketConnection.performTlsHandshake(NativeSocketConnection.java:99) ~[mys
at com.mysql.cj.protocol.a.NativeProtocol.negotiateSSLConnection(NativeProtocol.java:350) ~[mysql-connector-jav
... 41 common frames omitted
Caused by: java.security.cert.CertificateException: java.security.cert.CertPathValidatorException: Path does not cha
at com.mysql.cj.protocol.ExportControlled$X509TrustManagerWrapper.checkServerTrusted(ExportControlled.java:
at sun.security.ssl.AbstractTrustManagerWrapper.checkServerTrusted(SSLContextImpl.java:985) ~[na:1.8.0_171]
at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1596) ~[na:1.8.0_171]
... 52 common frames omitted
Caused by: java.security.cert.CertPathValidatorException: Path does not chain with any of the trust anchors
at sun.security.provider.certpath.PKIXCertPathValidator.validate(PKIXCertPathValidator.java:154) ~[na:1.8.0_171]
at sun.security.provider.certpath.PKIXCertPathValidator.engineValidate(PKIXCertPathValidator.java:80) ~[na:1.8.0_
at java.security.cert.CertPathValidator.validate(CertPathValidator.java:292) ~[na:1.8.0_171]
at com.mysql.cj.protocol.ExportControlled$X509TrustManagerWrapper.checkServerTrusted(ExportControlled.java:
... 54 common frames omitted
```

You have two options:

- *Configure SSL for the MySQL connection.*
- *Switch from "**Strong Password Encryption for Authentication**" to "**Legacy Authentication Method**."*

> *Some URLs that might be helpful for you if you decide to go ahead with SSL configuration:*
>
> - *https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-reference-using-ssl.html*
> - *https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files-using-mysql.html*
>
> *In Windows is relatively easy to enable the "**Legacy Authentication Method**":*
>
> - *Execute the installer.*
> - *Click on reconfigure beside MySQL Server.*
> - *Follow the wizard screens and choose "**Legacy Authentication Method**."*

Old JDBC driver

If your log file is shown the error below:

```
Caused by: java.lang.NullPointerException: null
 at com.mysql.jdbc.ConnectionImpl.getServerCharset(ConnectionImpl.java:3005) ~[mysql-connector-java-5.1.38-bir
 at com.mysql.jdbc.MysqlIO.sendConnectionAttributes(MysqlIO.java:1916) ~[mysql-connector-java-5.1.38-bin.jar:5.1
 at com.mysql.jdbc.MysqlIO.proceedHandshakeWithPluggableAuthentication(MysqlIO.java:1845) ~[mysql-connecto
 at com.mysql.jdbc.MysqlIO.doHandshake(MysqlIO.java:1215) ~[mysql-connector-java-5.1.38-bin.jar:5.1.38]
 at com.mysql.jdbc.ConnectionImpl.coreConnect(ConnectionImpl.java:2255) ~[mysql-connector-java-5.1.38-bin.jar:5
 at com.mysql.jdbc.ConnectionImpl.connectWithRetries(ConnectionImpl.java:2106) ~[mysql-connector-java-5.1.38-b
 ... 41 common frames omitted
```

You are using the wrong driver version for connecting to MySQL Server 8; you should make the following changes:

- *Replace the MySQL-connector-java-5.1.38-bin.jar into your $TOMCAT_HOME/libs with the latest JDBC driver for MySQL Server 8. You can get the newest driver from https://dev.mysql.com/downloads/connector/j/.*
- *Change the MySQL connection settings set into your $TOMCAT_HOME/conf/server.xml; it should be something like this:*



*In the latest version of MySQL JDBC driver, the name of the class **com.MySQL.jdbc.The driver** has been changed to **com.MySQL.cj.JDBC.Driver**.*

```
<Resource name="jdbc/OpenKMDS" auth="Container" type="javax.sql.DataSource"
          maxTotal="100" maxIdle="30" maxWaitMillis="10000" validationQuery="select 1"
          username="openkm" password="openkm" driverClassName="com.mysql.cj.jdbc.Driver"
          url="jdbc:mysql://localhost:3306/okmdb?autoReconnect=true&amp;useUnicode=true&amp;characterEncodi
```

The server time zone value is unrecognized.

If your log file is shown the error below or similar:

```
Caused by: com.mysql.cj.exceptions.InvalidConnectionAttributeException: The server time zone value 'CEST' is un
 at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method) ~[na:1.8.0_171]
 at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62) ~[na:1.8.0_171
 at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45) ~[na:1
 at java.lang.reflect.Constructor.newInstance(Constructor.java:423) ~[na:1.8.0_171]
 at com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:59) ~[mysql-connector-java-8
 at com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:83) ~[mysql-connector-java-8
 at com.mysql.cj.util.TimeUtil.getCanonicalTimezone(TimeUtil.java:128) ~[mysql-connector-java-8.0.11.jar:8.0.11]
 at com.mysql.cj.protocol.a.NativeProtocol.configureTimezone(NativeProtocol.java:2201) ~[mysql-connector-java-8.
 at com.mysql.cj.protocol.a.NativeProtocol.initServerSession(NativeProtocol.java:2225) ~[mysql-connector-java-8.0.
 at com.mysql.cj.jdbc.ConnectionImpl.initializePropsFromServer(ConnectionImpl.java:1391) ~[mysql-connector-java
 at com.mysql.cj.jdbc.ConnectionImpl.connectWithRetries(ConnectionImpl.java:895) ~[mysql-connector-java-8.0.11.
 ... 45 common frames omitted
```

*It seems the latest MySQL JDBC driver have some issue with your OS time zone configuration; a quick workaround is to indicate the server time zone, for example, **serverTimezone=UTC**:*

```
<Resource name="jdbc/OpenKMDS" auth="Container" type="javax.sql.DataSource"
          maxTotal="100" maxIdle="30" maxWaitMillis="10000" validationQuery="select 1"
          username="openkm" password="openkm" driverClassName="com.mysql.cj.jdbc.Driver"
          url="jdbc:mysql://localhost:3306/okmdb_test?autoReconnect=true&amp;useUnicode=true&amp;characterEn
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Problem connecting to your database

If you get an error like this:

```
Caused by: com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException: Public Key Retrieval is no
```

You need to modify your file ***conf/server.xml*** *and add the following to the URL by JDBC:* ***allowPublicKeyRetrieval=true****.*

```
<Resource name="jdbc/OpenKMDS" auth="Container" type="javax.sql.DataSource"
        maxTotal="100" maxIdle="30" maxWaitMillis="10000" validationQuery="select 1"
        username="openkm" password="openkm" driverClassName="com.mysql.cj.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/okmdb_test?allowPublicKeyRetrieval=true&amp;useU
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Oracle

Database creation

Create tablespace

```
create tablespace OPENKMDATA datafile '/u01/app/oracle/oradata/XE/openkmdata.dbf' size 100M;
```

Create a database and user

```
create user OPENKM identified by *secret* default tablespace OPENKMDATA quota unlimited on OPENKMDATA;
grant all privileges to OPENKM;
grant create any context, create cluster, create dimension, create indextype, create job, create materialized view, cr
```

Configure your openkm.properties



This configuration property should be set before the database creation. Once the database has been initialized, don't modify it because it can damage your installation.

If your OpenKM installation has already been configured with another database, you can't switch to another database by changing this property.



Choose the dialect that fits better with your database version:

- *Oracle10gDialect*
- *Oracle12cDialect*
- *Oracle8iDialect*
- *Oracle9iDialect*
- *Oracle9Dialect: deprecated, use Oracle9iDialect or Oracle10gDialect*
- *OracleDialect: deprecated; use Oracle8iDialect*

Edit the file $TOMCAT_HOME/openkm.properties

```
spring.jpa.properties.hibernate.dialect=com.openkm.db.dialect.Oracle10gDialect
spring.jpa.hibernate.ddl-auto=create-only
```

Configure Tomcat data sources

Edit the file $TOMCAT_HOME/openkm.properties and enable the resource named jdbc/OpenKMDS

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE
spring.datasource.username=OPENKM
spring.datasource.password=*secret*
```

```
spring.datasource.validationQuery=select 1 from dual
```

Sample using a service name instead of SID:



The main difference between SID connections is the ":" is replaced by "/."

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/ServiceName
spring.datasource.username=OPENKM
spring.datasource.password=*secret*
spring.datasource.validationQuery=select 1 from dual
```

Another sample:

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(HOST=localhost)(PROTOCOL=TCP)(POR
spring.datasource.username=OPENKM
spring.datasource.password=*secret*
spring.datasource.validationQuery=select 1 from dual
```



Configure the values of the attributes named:

- *Username.*
- *Password.*
- *URL (change to your hosts and port).*



If you find this error starting the tomcat:

ORA-01882: timezone region not found

*Put this line in the file **$CATALINA_BASE/bin/setenv.sh**:*

```
JAVA_OPTS=-Doracle.jdbc.timezoneAsRegion=false
```

Configure application login

Edit the file $. TOMCAT_HOME/openkm.properties

```
authentication.openkm.database=true
```

Run application

As a process into OpenKM startup, the application will automatically create an empty database structure.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Once the tables are created, OpenKM will automatically change the ***hibernate.hbm2ddl*** *property from* ***create-only*** *to* ***none****.*

Configuring PostgreSQL

Database creation

Create a database and user

```
DROP DATABASE IF EXISTS okmdb;
CREATE USER openkm WITH PASSWORD '*secret*';
CREATE DATABASE okmdb WITH OWNER openkm ENCODING 'UTF8';
```

or from the command line

```
$ createuser --pwprompt openkm

$ createdb --owner=openkm --encoding=UTF8 okmdb
```

More info at:

- *PostgreSQL 9.1 Documentation*
- *Guía Ubuntu: PostgreSQL*

Configure your openkm.properties



This configuration property should be set before the database creation. Once the database has been initialized, don't modify it because it can damage your installation.

If your OpenKM installation has already been configured with another database, you can't switch to another database by changing this property.

Edit the file $. TOMCAT_HOME/openkm.properties

```
hibernate.dialect=com.openkm.db.dialect.PostgreSQL92Dialect
hibernate.hbm2ddl=create-only
```

Configure Tomcat datasources

Edit the file $TOMCAT_HOME/openkm.properties and enable the resource named jdbc/OpenKMDS

```
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/okmdb
spring.datasource.username=openkm
spring.datasource.password=*secret*
spring.datasource.validationQuery=SELECT 1
```



Configure the values of the attributes called:

- *Username.*

- *Password.*
- *URL (change to your hosts and port).*

Configure application login

Edit the file $. TOMCAT_HOME/openkm.properties

```
authentication.openkm.database=true
```

Enable native configuration parameter

Go to Administration > Configuration parameters

```
native.sql.lowercase=true
```

Run application

As a process into OpenKM startup, the application will automatically create an empty database structure.

*Once the tables are created, OpenKM will automatically change the **hibernate.hbm2ddl** property from **create-only** to **none**.*

Additional information

If you want to obtain the maximum performance from PostgreSQL, pgFouine will help in your task.

PostgreSQL 9.6 and upper

*PostgreSQL version 9.6 and upper have **changed some tables and columns of the schema**. Check that you are using the correct JDBC driver.*

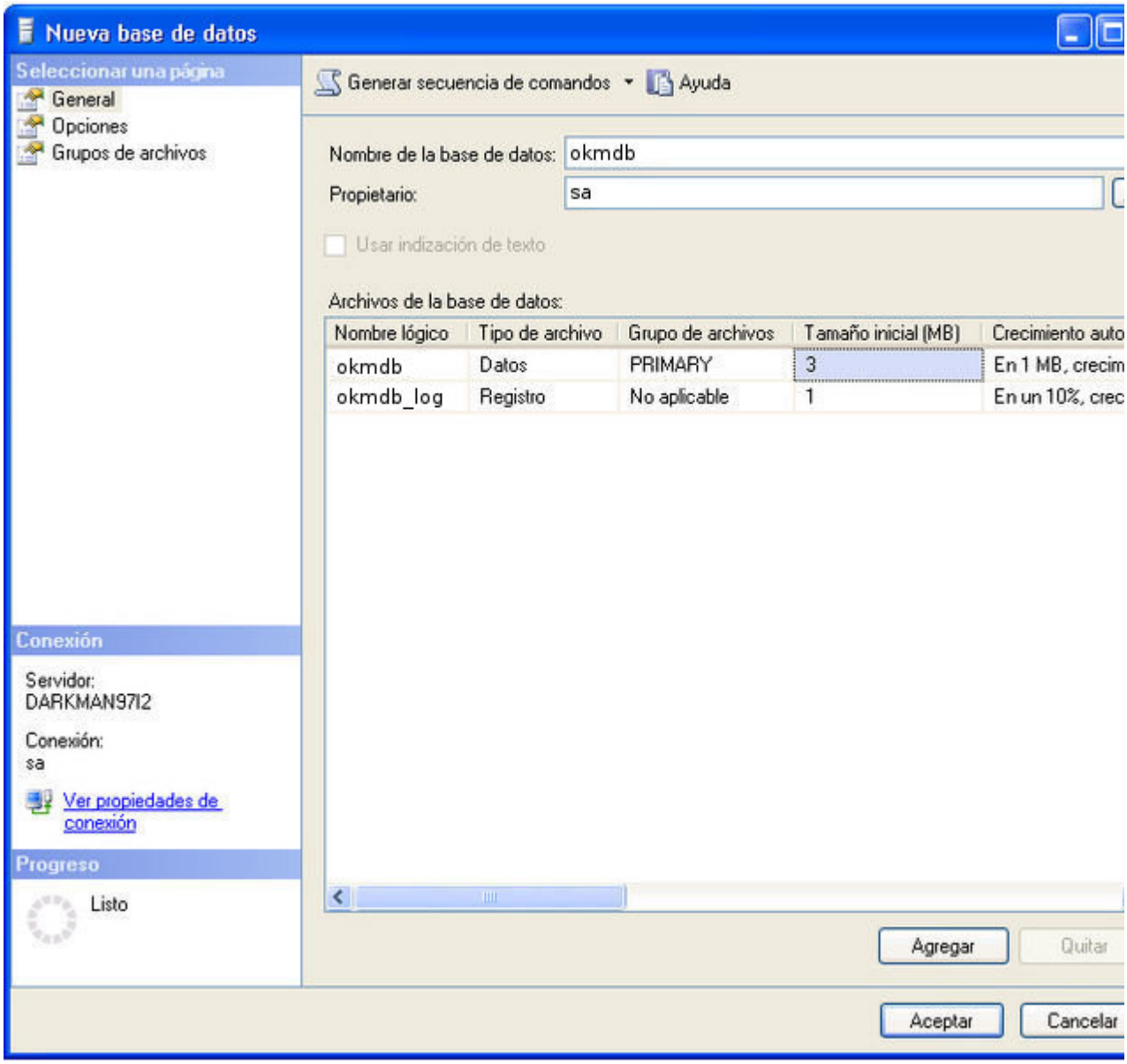
Read this related issue for more information PostgreSQL - ERROR o.h.tool.hbm2ddl.SchemaUpdate - could not complete schema update

Configuring SQL Server

Database creation

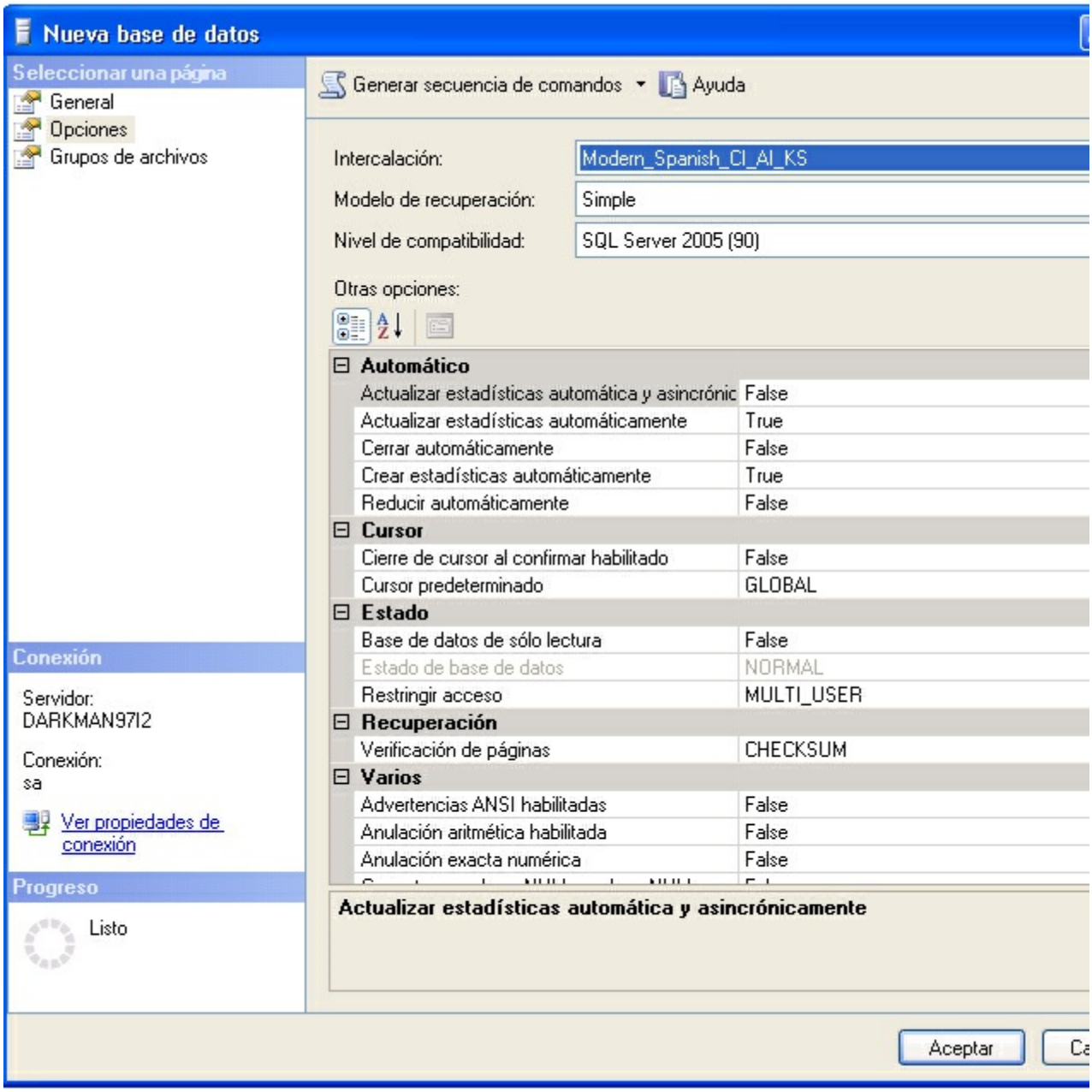
Create a database and user

Start SQL Server Management Studio Express and create a database called okmdb.



Select your appropriate database to collate.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



*We checked with **Modern_Spanish_CS_AS_KS**, where **CS indicates case-sensitive** collation and **AS indicates accent-sensitive** collation.*

*To ensure the correct OpenKM functionality, it should be selected **case-sensitive**.*

For more info, please check Collation and Unicode support.

Configure your openkm.properties

This configuration property should be set before the database creation. Once the database has been initialized, don't modify it because it can damage your installation.

If your OpenKM installation has already been configured with another database, you can't switch to another

database by changing this property.



Choose the hibernate dialect that fits better with your current SQLServer database version:

- *com.openkm.db.dialect.SQLServer2005Dialect*
- *com.openkm.db.dialect.SQLServer2008Dialect*
- *com.openkm.db.dialect.SQLServer2012Dialect*
- *com.openkm.db.dialect.SQLServer2016Dialect*
- *com.openkm.db.dialect.SQLServerDialect*



Sometimes you also could need to perform this compatibility level in the command line (ALTER DATABASE):

Alter database compatibility level

Edit the file $TOMCAT_HOME/openkm.properties

```
spring.jpa.hibernate.ddl-auto=create-only
spring.jpa.properties.hibernate.dialect=com.openkm.db.dialect.SQLServerDialect
```

Configure Tomcat data sources

Edit the file $TOMCAT_HOME/openkm.properties and enable the resources

```
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.datasource.url=jdbc:sqlserver:///localhost:1433;databaseName=okmdb
spring.datasource.username=openkm
spring.datasource.password=*secret*
spring.datasource.validationQuery=SELECT 1
```



Configure the values of the attributes named:

- *Username.*
- *Password.*
- *URL (change to your hosts and port).*

Configure application login

Edit the file $TOMCAT_HOME/openkm.xml

```
authentication.openkm.database=true
```

Run application

As a process into the OpenKM startup, the application will automatically create an empty database structure.

> *Once the tables are created, OpenKM will automatically change the **hibernate.hbm2ddl** property from **create-only** to **none**.*

Configure integrated security

You must add the "integratedSecurity=true" parameter to the JDBC connection URL to enable Integrated Security. For example:

```
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=okmdb;autoReconnect=true;integratedSecurit
spring.datasource.username=
spring.datasource.password=
```

*You must also download the driver package from https://go.microsoft.com/fwlink/?linkid=2122434 and copy the **sqljdbc_auth.dll** file to TOMCAT_HOME/bin folder.*

More info at:

- *Microsoft SQL Server documentation*

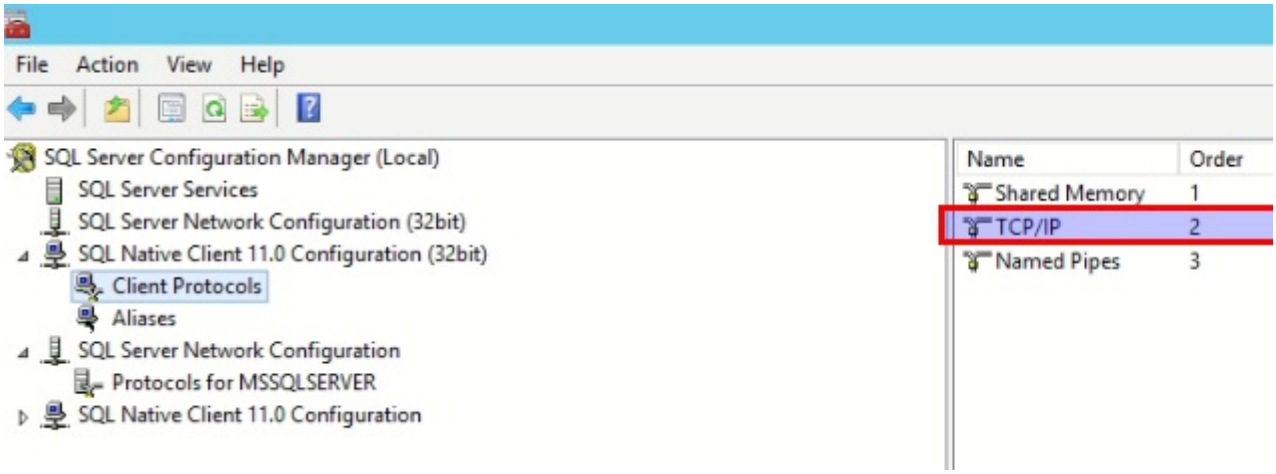
Troubleshooting error

Can't connect to SQL Server

Ensure you have TCP protocol enabled (by default is disabled)



> *In this section (first image), you can see TCP/IP protocol enabled, but it is not enabled. See the second image...*



You configure the TCP/IP protocol in this section. Here you can see if TCP/IP protocol is enabled.



The connection was lost after some time.

We have observed in some SQL Server editions what seems to be new behavior that passed some hours; the connection is closed (it might be some extra configuration parameter in the latest SQL server editions that comes enabled by default or similar).

There's a new configuration attribute name ***autoReconnect*** *that solves it.*

```
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=okmdb;autoReconnect=true
spring.datasource.username=openkm
spring.datasource.password=*secret*
spring.datasource.validationQuery=SELECT 1
```

Not able to login into OpenKM

When first-time OpenKM login fails with the correct username and password (***okmAdmin*** */* ***admin)****, you should check the database configuration.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*Check if the database is configured to be **case-sensitive**; it might be one reason you cannot log in.*



*If you have this troubleshooting, you should drop the database and create it again, selecting the collation as **case-sensitive.***

Not able to connect TCP/ IP to localhostSQL, port 1433 when using instance in the JDBC connection URL

*If **you use the database instance name** in the connection, you must use "\\" in the **spring.datasource.URL** configuration parameter. The sample below it use a **database instance named SampleSQL**.*

```
spring.datasource.url=jdbc:sqlserver://localhost\\SampleSQL;databaseName=okmdb;autoReconnect=true
```

Deadlock issues

These errors appear in the log this way:

```
o.h.engine.JDBC.spi.SqlExceptionHelper : SQL Error: 1205, SQLState: 40001
o.h.engine.JDBC.spi.SqlExceptionHelper: Transaction (Process ID 56) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.
```

Modify the database to avoid these issues:

```
ALTER DATABASE okmdb SET READ_COMMITTED_SNAPSHOT ON WITH ROLLBACK IMMEDIATE
```

Source How to Solve Database Deadlock on Microsoft SQL Server .

SSL connection error

If you see this error:

```
The driver could not establish a secure connection to SQL Server by using Secure Sockets Layer (SSL) encryption.
```

*The problem can be fixed adding the "**encrypt=false**" parameter to the JDBC URL:*

```
spring.datasource.url=jdbc:sqlserver://sqlserver:1433;databaseName=okmdb;encrypt=false;
```

This is due to a change in latest version of SQL Server Java JDBC driver.

More info at:

- *Understanding encryption support*
- *Connecting with encryption*

The driver could not establish a secure connection to SQL Server by using Secure Sockets Layer (SSL) encryption

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

That may happen when the SQL Server only allows SSL connections.

*The problem can be fixed by adding the "**encrypt=true**" and "**trustServerCertificate=true**" parameters to the JDBC URL:*

```
spring.datasource.url=jdbc:sqlserver://sqlserver:1433;databaseName=okmdb;encrypt=true;trustServerCertificate=true;
```

SQL Server alter compatibility level

To set SQL Server compatibility level 2012

```
ALTER DATABASE okmdb SET COMPATIBILITY_LEVEL = 110;
```

More information at https://learn.microsoft.com/es-es/sql/t-sql/statements/alter-database-transact-sql-compatibility-level

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Hikari pool

OpenKM use Hikari pool, the default configuration parameters are:

```
spring.datasource.hikari.minimumIdle=5
spring.datasource.hikari.maximumPoolSize=50
spring.datasource.hikari.idleTimeout=30000
spring.datasource.hikari.maxLifetime=1800000
spring.datasource.hikari.connectionTimeout=30000
spring.datasource.hikari.leak-detection-threshold=48000
```

You can modify these values setting your own in the $TOMCAT_HOME/openkm.properties file.

Enable debug

```
logging.level.com.zaxxer.hikari.pool.HikariPool=DEBUG
logging.level.com.zaxxer.hikari.HikariConfig=DEBUG
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Backup and restore

One of the most important things to have is an updated backup of your OpenKM.

If you have followed the installation instructions, you have configured OpenKM to use a production-safe database and optionally set a path to the OpenKM repository data store.

OpenKM has several configurations, but for optimal performance, we recommend the following:

- *Use a default data store that saves binary information directly to the server disk (in a non-human comprehensible format).*
- *All the other information, folders, document structure, metadata, etc., is stored in a database.*

In that case:

1. *First, backup your database (make a SQL dump)*
2. *Then, backup your data on the filesystem*

*By default, all documents are stored in $TOMCAT_HOME/repository. This default folder can be changed using the property "**repository.home**" in the file $TOMCAT_HOME/openkm.properties file. Remember from OpenKM version 8.1.x, the **Lucene indexes path location** in the file system **is not managed by "repository.home"**. For more information, take a look at the Change repository home.*

Although OpenKM supports hot backups of your data, it is recommended to stop Tomcat before starting the backup. Apply this good practice when possible.

Backup steps:

- *Stop Tomcat*
- *Back up the database*
  - *In case of MariaDB, please visit Full Backup and Restore with Mariabackup*
  - *In case of PostgreSQL, please visit PostgreSQL Backup and Restore*
  - *In case of SQL Server, please visit Create a Full Database Backup*
  - *In case of MySQL, please visit MySQL Backup and Recovery*
- *Back up the Tomcat or repository folder*
- *Start Tomcat*

Performing the backup in that order (the database first, then the filesystem) will ensure backup consistency.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Contact support before starting the new instance to prevent license validation issues. It's important not to have both instances running simultaneously to avoid license validation problems.

An alternative to traditional backup methods is to export the whole repository from OpenKM Administration > Utilities > Repository export and choose metadata and history options. But we do not suggest this way of doing backups; we suggest some incremental backup solutions for medium to large repositories.

Linux detailed steps

You can follow these steps to backup and restore if you have OpenKM configured with MySQL or MariaDB. For other databases, check their corresponding command.

Backup

```
$ systemctl stop openkm
$ mysqldump -h localhost -u root -p okmdb > /path/to/openkm/okmdb.sql
$ zip -r /path/to/backup/okm-backup.zip /path/to/openkm
$ systemctl start openkm
```

After that, you will have a ZIP archive with the whole OpenKM installation and the exported database. Please, check before destroying this installation. We always recommend restoring the backup to ensure you have all the required files.

Restore

Please note that you must create the database if it wasn't made previously. In the case of MySQL or MariaDB and Ubuntu, use this command to install it:

```
$ sudo apt install mariadb-server
```

In the case of CentOS:

```
$ sudo yum install mariadb-server
```

And execute the following sentences:

```
CREATE USER 'openkm'@'localhost' IDENTIFIED BY 's3cr3t0';
CREATE DATABASE okmdb DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_bin;
GRANT ALL ON okmdb.* TO 'openkm'@'localhost' WITH GRANT OPTION;
```

Remember that the user and database names may be different in your case. Please, take a look at the installation report or check it at $TOMCAT_HOME/openkm.properties

Once created, restore the database and files:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
$ unzip /path/to/backup/okm-backup.zip -d /path/to/restore
$ mysql -h localhost -u root -p okmdb < /path/to/restore/okmdb.sql
```

Set the permissions according to the user who starts OpenKM:

```
$ sudo chown openkm:openkm /path/to/openkm -R
```

Install the required dependencies. In the case of Ubuntu:

```
$ sudo apt install xvfb imagemagick libreoffice tesseract-ocr
```

In the case of CentOS:

```
$ sudo yum install ImageMagick tesseract libreoffice libreoffice-headless
```

Once restored, verify file permissions, start OpenKM, and make a repository check. See *Repository checker*.

Windows detailed steps

You can follow these steps to backup and restore if you have OpenKM configured with MySQL or MariaDB. For other databases, check their corresponding command.

Backup

- *Stop the OpenKM service*
- *Create a database backup*
  - *In case of MariaDB, please visit* *Full Backup and Restore with Mariabackup*
  - *In case of PostgreSQL, please visit* *PostgreSQL Backup and Restore*
  - *In case of SQL Server, please visit* *Create a Full Database Backup*
  - *In case of MySQL, please visit* *MySQL Backup and Recovery*
- *Create a backup of the Tomcat folder*
- *Start the OpenKM service again*

After that, you will have a ZIP archive with the whole OpenKM installation and the exported database. Please, check before destroying this installation. We always recommend restoring the backup to ensure you have all the required files.

Restore

Please note that you must create the database if it wasn't made previously. In the case of MySQL or MariaDB please download from:

- *https://mariadb.org/download/*
- *https://www.mysql.com/downloads/*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Once installed, execute the following sentences to create the database user and OpenKM database:

```
CREATE USER 'openkm'@'localhost' IDENTIFIED BY 's3cr3t0';
CREATE DATABASE okmdb DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_bin;
GRANT ALL ON okmdb.* TO 'openkm'@'localhost' WITH GRANT OPTION;
```

Remember that the user and database names may be different in your case. Please, take a look at the installation report or check it at $TOMCAT_HOME/openkm.properties

Once created, restore the database and files:

```
C:\> unzip okm-backup.zip -d C:\
C:\> mysql -h localhost -u root -p okmdb < okmdb.sql
```

Set the permissions according to the user who starts OpenKM.

Once restored, verify file permissions, start OpenKM, and make a repository check. See Repository checker.

Backup scripts

- *Linux backup and restore with BorgBackup* ***(recommended)***
- *Linux backup and restore with rdiff-backup tool*
- *Linux backup and restore with rsync*
- *Linux backup and restore with Duplicity*
- *Linux backup with LFTP*

Additional information

- *Easy Automated Snapshot-Style Backups with Linux and Rsync*
- *BorgBackup documentation*
- *Rdiff-backup home page*
- *Duplicity Backup Howto*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Linux backup and restore with BorgBackup

Borg is a deduplicating backup program that also supports compression and authenticated encryption. It provides an efficient and secure way to back up data. The data deduplication technique makes Borg suitable for daily backups since only changes are stored. The authenticated encryption technique makes it ideal for backups to not fully trusted targets.

A more detailed description of BorgBackup can be found at https://borgbackup.readthedocs.io/en/stable/index.html.

Installation

Borg is included in many Linux distribution repositories, but sometimes the packaged version is quite old, and it is better to install manually. This installation is as simple as:

> *In the official Borg releases website section, you will find the latest version:*
>
> *https://github.com/borgbackup/borg/releases*

```
$ wget https://github.com/borgbackup/borg/releases/download/1.1.5/borg-linux64
$ chmod +x borg-linux64
$ mv borg-linux64 /usr/local/bin/borg
```

Sample script

Backup to a remote computer with SSH

```
#!/bin/bash
#
## BEGIN CONFIG ##
HOST=$(uname -n)
OPENKM_DB="okmdb"
OPENKM_HOME="/home/openkm"
TOMCAT_HOME="$OPENKM_HOME/tomcat"
DATABASE_EXP="$OPENKM_HOME/db"
BACKUP_HOST="backup@bk-server.com"
BACKUP_DIR="/mnt/backup/$HOST"
BACKUP_REPO="${BACKUP_HOST}:${BACKUP_DIR}"
## END CONFIG ##

# Check root user
if [ $(id -u) != 0 ]; then echo "You should run this script as root"; exit; fi

echo "Backup destination ${BACKUP_REPO}"
if (ssh ${BACKUP_HOST} "[ ! -d ${BACKUP_DIR} ]"); then
  echo "Initialize backup repository: ${BACKUP_DIR}"
  ssh ${BACKUP_HOST} "mkdir -p ${BACKUP_DIR}"
  borg init -e none ${BACKUP_REPO}
fi

# Database
echo "Backup MySQL databases"
mkdir -p ${DATABASE_EXP}
echo "- Backuping $OPENKM_DB..."
mysqldump --single-transaction --add-drop-table --quick ${OPENKM_DB} > ${DATABASE_EXP}/mysql_$OPENKM
```

```
borg create -v --stats ${BACKUP_REPO}::'{now:%Y-%m-%d_%H:%M}' ${OPENKM_HOME}
borg prune -v ${BACKUP_REPO} --list --keep-daily=7 --keep-weekly=4 --keep-monthly=6
```

This is an improved SSH version

```
#!/bin/bash

# Configuration
DATABASE_EXP="/home/openkm/db"
BACKUP_HOST="backup@bk-server.com"
BACKUP_DIR="/mnt/backup/$HOST"
BACKUP_REPO="${BACKUP_HOST}:${BACKUP_DIR}"

echo "### BEGIN: $(date +"%x %X") ###"
echo "Backup destination ${BACKUP_REPO}"
TBEGIN=$(date +"%s")

# Check if the repository is already initialized
borg list ${BACKUP_REPO} > /dev/null
RESULT=$?; if [[ ${RESULT} != 0 ]]; then
  echo "Initializing backup repository..."
  borg init -e none ${BACKUP_REPO}
fi

# Database
echo "Backup MySQL databases"
mkdir -p ${DATABASE_EXP}
MYSQL_DBS=$(mysqlshow | awk '(NR > 2) && (/[a-zA-Z0-9]+[ ]+[|]/) { print $2 }');
for DB in $MYSQL_DBS; do
  if [[ $DB != "mysql" && $DB != "test" && $DB != "information_schema" && $DB != "performance_schema" ]]; then
    echo "- Backuping $DB..."
    mysqldump --single-transaction --add-drop-table --quick ${DB} > ${DATABASE_EXP}/mysql_$DB.sql
  fi
done

borg create -v --stats ${BACKUP_REPO}::'{now:%Y-%m-%d_%H:%M}' /opt /var /etc /root /home
RESULT=$?; if [[ ${RESULT} != 0 ]]; then
  echo "****************************************************"
  echo " Backup error => exit code: ${RESULT}"
  echo "****************************************************"
fi

# Prune old backups
echo "Prune old backups"
borg prune -v --list --keep-within=30d ${BACKUP_REPO}
RESULT=$?; if [[ ${RESULT} != 0 ]]; then
  echo "****************************************************"
  echo " Backup error => exit code: ${RESULT}"
  echo "****************************************************"
fi

TEND=$(date +"%s")
TDIFF=$(($TEND-$TBEGIN))
echo "### END: $(date +"%x %X") - $(($TDIFF / 60)) mins and $(($TDIFF % 60)) secs ###"
```

Backup to mapped network

```
#!/bin/bash
#
## BEGIN CONFIG ##
HOST=$(uname -n)
```

```
OPENKM_DB="okmdb"
OPENKM_HOME="/home/openkm"
TOMCAT_HOME="$OPENKM_HOME/tomcat"
DATABASE_EXP="$OPENKM_HOME/db"
BACKUP_DIR="/mnt/backup"
BACKUP_REPO="${BACKUP_DIR}/${HOST}"
## END CONFIG ##

# Check root user
if [ $(id -u) != 0 ]; then echo "You should run this script as root"; exit; fi

# Mount disk
if mount | grep "$BACKUP_DIR type" > /dev/null; then
  echo "$BACKUP_DIR already mounted";
else
  mount "$BACKUP_DIR";

  if mount | grep "$BACKUP_DIR type" > /dev/null; then
    echo "$BACKUP_DIR mounted";
  else
    echo "$BACKUP_DIR error mounting";
    exit -1;
  fi
fi

# Initialize repository
echo "Backup destination ${BACKUP_REPO}"
if ! ls -l ${BACKUP_DIR} | grep '^d' | grep "${HOST}" > /dev/null; then
  echo "Initialize backup repository: ${BACKUP_REPO}"
  mkdir ${BACKUP_REPO}
  borg init -e none ${BACKUP_REPO}
fi

# Database
echo "Backup MySQL databases"
mkdir -p ${DATABASE_EXP}
echo "- Backuping $OPENKM_DB..."
mysqldump --single-transaction --add-drop-table --quick ${OPENKM_DB} > ${DATABASE_EXP}/mysql_$OPENKM

# Data backup
echo "File system backup: ${BACKUP_REPO}"
borg create -v --stats ${BACKUP_REPO}::'{now:%Y-%m-%d_%H:%M}' ${OPENKM_HOME}
borg prune -v ${BACKUP_REPO} --list --keep-daily=7 --keep-weekly=4 --keep-monthly=6

# Umount disk
sync
umount "$BACKUP_DIR"
echo "$BACKUP_DIR umounted";
```

Configure crontab

To install the cron job, run the following:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

And add these lines according to your configuration:

```
MAILTO=nomail@openkm.com
PATH=/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/bin
@daily /root/backup.sh 2>&1 | tee /root/logs/backup.$(date +\%Y.\%m.\%d_\%H.\%M.\%S).log
```

More information at Crontab quick reference

*If you want to be notified by mail, you should install the "**postfix**" service on your server.*

Recovery

*To list all stored backups, use the **list** action parameter:*

```
$ borg list backup@bk-server.com:/mnt/backup/name
2018-02-07 Thu, 2018-02-07 10:12:31 [c878b314107c76b2d86ba75a9e5b2a216bf1f0d6b824313396c51ccda02a72e6]
2018-02-08 Thu, 2018-02-08 10:15:45 [2cf11f13bc7a2c9c0f1a0c243da52ac4b7657e7c6429cca0bf06a80467837f1e]
```

If you want to list all files in a given backup:

```
$ borg list backup@bk-server.com:/mnt/backup/name::2018-02-07_09:27
```

In case you need to access the contents of a backup, you can mount it as a local filesystem:

```
$ mkdir /tmp/backup
$ borg mount backup@bk-server.com:/mnt/backup/name::2018-02-07_09:27 /tmp/backup
```

If you do not specify a specific backup, all the backups will be mounted. The command line should be:

```
$ mkdir /tmp/backup
$ borg mount backup@bk-server.com:/mnt/backup /tmp/backup
```

In the end, you might be interested in umount the folder:

```
borg umount /tmp/backup
```

Linux backup and restore with the rdiff-backup tool

Rdiff-backup backs up one directory to another, possibly over a network. The target directory ends up being a copy of the source directory. However, extra reverse diffs are stored in a special subdirectory of that target directory, so you can still recover files lost some time ago.

The idea is to combine the best features of a mirror and an incremental backup. rdiff-backup also preserves subdirectories, hard links, dev files, permissions, uid/gid ownership, modification times, extended attributes, acls, and resource forks.

Also, rdiff-backup can operate bandwidth-efficient over a pipe, like rsync. Thus, you can securely use rdiff-backup and ssh to back a hard drive up to a remote location, and only the differences will be transmitted.

> *If you need to back up to CIFS (SMB, Samba) or Mac's HFS, please look at the rdiff-backup FAQ.*

Debian & Ubuntu

This utility is included in the software repository, so to install, just run as root:

```
$ apt-get install rdiff-backup
```

> *Rdiff-backup from Debian issues a warning message due to a deprecated method [os.popen2 is deprecated], which can be hidden following the steps in How to shut up Python deprecation warnings.*

CentOS & RedHat

In this case, the application is not included in the default software repository, so you need to make use of the EPEL repository:

```
$ rpm -Uvh http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm
$ yum update
$ yum install rdiff-backup
```

Preliminaries

In these scripts are assumed these default values:

- ***$TOMCAT_HOME** value is "**/home/openkm/tomcat-9.0.76**"*
- *OpenKM **database** is named "**okmdb**."*
- *OpenKM **database password** value is "***secret***".*
- *OpenKM start & stop **service script** location is "**/etc/init.d/tomcat**"*

Sample script

As a good practice, the root user should do the backup.

The script below does a backup to USB, and the application database is MySQL.

USB disk mount point can be defined in /etc/fstab as:

```
/dev/sdb1   /mnt/backup   ext4   defaults   0   0
```

These are the global script sections:

- *Global var configuration under **## BEGIN CONFIG ##** section.*
- *Checking user root under **# Check root** user section.*
- *Deleting older database backup **# Delete older local database backup** section.*
- *Mount disk (optional section) under **# Mount disk** section.*
- *Stop application (optional section) under **# Stop tomcat** section.*
- *Clean logs (optional section) under **# Clean logs** section.*
- *Backup the database to the filesystem under **#Backup database** section.*
- *Backup repository and database dump to the backup destination under **#Backup and purge old backups** section.*
- *Start application (optional section) **# Start tomcat** section.*
- *Show statistics (optional section) **# Status** section.*
- *Umount disk (optional section) under **# Umount disk** section.*

Configuration parameters explanation:

Parameter	*Description*
HOST	*The server hostname.*
DATABASE_PASS	*The database password.* *The database user who does the backup is always the root user.*
OPENKM_DB	*The application database name.* *Usually will be named "okmdb."*

OPENKM_HOME	*The openkm home folder.*
TOMCAT_HOME	*The tomcat home folder.* *Usually into the OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually, into the same server.*
BACKUP_DIR	*Backup destination. In the case of a remote server, the form is user@server::/path/to/backup*

*Create a backup script **/root/backup.sh***

```
$ sudo su

$ vim /root/backup.sh
```

```
#!/bin/bash
#
## BEGIN CONFIG ##
HOST=$(uname -n)
DATABASE_PASS="*secret*"
OPENKM_DB="okmdb"
OPENKM_HOME="/home/openkm"
TOMCAT_HOME="$OPENKM_HOME/tomcat-9.0.76"
DATABASE_EXP="$OPENKM_HOME/db"
BACKUP_DIR="/mnt/backup"
## END CONFIG ##

# Check root user
if [ $(id -u) != 0 ]; then echo "You should run this script as root"; exit; fi

# Delete older local database backup
echo -e "### BEGIN: $(date +"%x %X") ###\n"
rm -rf $DATABASE_EXP
mkdir -p $DATABASE_EXP

# Mount disk
if mount | grep "$BACKUP_DIR type" > /dev/null; then
  echo "$BACKUP_DIR already mounted";
else
  mount "$BACKUP_DIR";

  if mount | grep "$BACKUP_DIR type" > /dev/null; then
    echo "$BACKUP_DIR mounted";
  else
    echo "$BACKUP_DIR error mounting";
    exit -1;
  fi
```

```
fi

# Stop Tomcat
/etc/init.d/tomcat stop

# Clean logs
#echo "Clean Tomcat temporal files."
#rm -rf $TOMCAT_HOME/logs/*
#rm -rf $TOMCAT_HOME/temp/*
#rm -rf $TOMCAT_HOME/work/Catalina/localhost

# Backup database
if [ -n "$DATABASE_PASS" ]; then
  echo "* Backuping MySQL data from $OPENKM_DB..."
  mysqldump -h localhost -u root -p$DATABASE_PASS $OPENKM_DB > $DATABASE_EXP/mysql_$OPENKM_DB
  echo "-------------------------------------";
fi

# Backup and purge old backups
rdiff-backup --remove-older-than 30B $BACKUP_DIR/$HOST
rdiff-backup -v 3 --print-statistics --include $OPENKM_HOME --exclude '**' / $BACKUP_DIR/$HOST

# Start Tomcat
/etc/init.d/tomcat start
echo -e "\n### END: $(date +"%x %X") ###"

# Status
echo "==================================";
rdiff-backup --list-increment-sizes $BACKUP_DIR/$HOST
echo "**********************************";
df -h | grep "$BACKUP_DIR"
echo "==================================";

# Umount disk
sync
umount "$BACKUP_DIR"
```

Increase the backup period

Modify the ***# Backup and purge old backups*** *section increasing the parameter value:*

> *The "**30B**" value indicates that you can restore a full backup of any of the last 30 days; increasing to " **90B**" will indicate any of the previous 90 days, etc...*

```
--remove-older-than 90B
```

Do remote backup

Modify the ***# Backup and purge the old backups*** *section.*

> *The backup@server indicates remote "server" accessed by user "backup".*

```
# Backup and purge old backups
rdiff-backup --remove-older-than 30B backup@server::$BACKUP_DIR/$HOST
rdiff-backup -v 3 --print-statistics --include $FILES --exclude '**' / backup@server::$BACKUP_DIR/$HOST
```



*Also could be only modified the **## Config section**, for example:*

```
## BEGIN CONFIG ##
BACKUP_DIR="user@server::/path/to/backup"
```

*And keep the **# Backup and purge old backups** intact.*

```
# Backup and purge old backups
rdiff-backup --remove-older-than 30B $BACKUP_DIR/$HOST
```

Do a remote backup to SMB or CIFS

*Modify the **# Mount disk** section.*

```
# Mount disk
if mount | grep "$BACKUP_DIR type" > /dev/null; then
  echo "$BACKUP_DIR already mounted";
else
  echo "Mounting $BACKUP_DIR ...";
  mount -t cifs //REMOTE_SERVER/REMOTE_DIR $BACKUP_DIR -o username=REMOTE_USER,password=REM

  if mount | grep "$BACKUP_DIR type" > /dev/null; then
    echo "$BACKUP_DIR mounted";
  else
    echo "Error mounting $BACKUP_DIR.";
    exit -1;
  fi
fi
```

Do PostgreSQL backup

*Modify the **# Backup database** section*

```
# Backup de PostgreSQL
echo "* Backuping PostgreSQL data from $OPENKM_DB..."
su postgres -c "pg_dump $OPENKM_DB" > $DATABASE_EXP/pg_$OPENKM_DB.sql

# Databases optimizations
su postgres -c "vacuumdb -a -z" > /dev/null
su postgres -c "reindexdb -a -q" 2> /dev/null
```

Configure crontab

To install the cron job, run the following:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

And add these lines according to your configuration:

```
MAILTO=nomail@openkm.com
@daily /root/backup.sh | tee /root/logs/backup.$(date +\%Y.\%m.\%d_\%H.\%M.\%S).log
```

More information at Crontab quick reference

*If you want to be notified by mail, you should install the "**postfix**" service on your server.*

Restoring backup

You can list the available backups:

```
$ rdiff-backup --list-increment-sizes /path/to/backup
Sun Sep 11 05:00:19 2012        4.25 GB         4.25 GB   (current mirror)
Sun Sep  4 00:00:18 2012        13.3 MB         4.26 GB
Sun Aug 28 00:00:13 2012         674 MB         4.92 GB
Sun Aug 21 00:00:14 2012        5.50 MB         4.93 GB
Sun Aug 14 00:00:16 2012        1.75 MB         4.93 GB
Sun Aug  7 00:00:12 2012         288 KB         4.93 GB
Sun Jul 31 00:00:13 2012        43.0 KB         4.93 GB
Fri Jul 29 10:36:39 2012        5.56 KB         4.93 GB
```

Then decide to restore some of the backups, for example, the one made on Sun Aug 28 00:00:13 2012:

```
$ rdiff-backup --restore-as-of 2012-08-28 /path/to/backup /path/to/destination
```

Or restore the last backup

```
$ rdiff-backup --restore-as-of now /path/to/backup /path/to/destination
```

*The **--restore-as-of** parameter accepts several formats. See **rdiff-backup** documentation for more info.*

Sample for restoring single folder (the /home/openkm/tomcat-9.0.76/ must exist)

```
rdiff-backup --restore-as-of now /mnt/backup/ms1/home/openkm/tomcat-9.0.76/ /home/openkm/tomcat-8.5.69/
```

*Inside of **/path/to/destination,** you should see a directory **/home/openkm**, and inside it, a couple of directories:*

- ***db**: The backup of the database.*
- ***tomcat-9.0.76**: The backup of the Tomcat installation and OpenKM repository.*

Additional information

- *rdiff-backup home page*

- *rdiff-backup wiki*
- *Automated backups with rdiff-backup*
- *Using rdiff-backup for remote backups*
- *Suppress socket warnings when using rdiff-backup*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Linux Sample Backup with Duplicity

Duplicity backs directories by producing encrypted tar-format volumes and uploading them to a remote or local file server. Because duplicity uses librsync, the incremental archives are space efficient and only record the parts of files that have changed since the last backup. Because duplicity uses GnuPG to encrypt and/or sign these archives, they will be safe from spying and/or modification by the server.

Installation

Duplicity can be installed in Debian / Ubuntu as simple as:

```
$ sudo apt-get install duplicity ncftp python-boto python-paramiko
```

> *To avoid the **Import of duplicity.backends.giobackend Failed: No module named gio** warning, you need to install the **python-gobject-2** package:*
>
> ```
> $ apt-get install python-gobject-2
> ```

But it not in the CentOS / RedHat default repositories, so you need to install from another source. This script will help in this installation process:

```
#!/bin/bash

wget http://dl.fedoraproject.org/pub/epel/6/x86_64/duplicity-0.6.18-1.el6.x86_64.rpm
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/ncftp-3.2.4-1.el6.x86_64.rpm
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/librsync-0.9.7-15.el6.x86_64.rpm
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/python-GnuPGInterface-0.3.2-6.el6.noarch.rpm
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/python-boto-2.5.2-1.el6.noarch.rpm

rpm -Uvh duplicity-0.6.18-1.el6.x86_64.rpm ncftp-3.2.4-1.el6.x86_64.rpm librsync-0.9.7-15.el6.x86_64.rpm python-G
```

Command line basics

To make a backup you can do:

```
$ duplicity --no-encryption /home/openkm file:///path/to/backup
```

To verify it:

```
$ duplicity --no-encryption verify file:///path/to/backup /home/openkm
```

To list backed files:

```
$ duplicity --no-encryption list-current-files file:///path/to/backup
```

To restore a single file:

```
$ duplicity --no-encryption --file-to-restore tomcat-9.0.76/openkm.properties file:///path/to/backup RestoredBackup
```

The path to the file that is to be restored is related to the directory on which the backup set is based. So in the command above, tomcat-9.0.76/openkm.properties plus the directory on which we based our backup (/home/openkm) equals /home/openkm/tomcat-9.0.76/openkm.properties

Preliminaries

In these scripts are assumed these default values:

- ***$TOMCAT_HOME** value is "**/home/openkm/tomcat-9.0.76**"*
- *OpenKM **database** is named "**okmdb**".*
- *OpenKM **database password** value is "***secret***".*
- *OpenKM start & stop **service script** location is "**/etc/init.d/tomcat**"*

Sample script

As a good practice the backup should be done by root user.

The script below does a backup to USB and the application database is MySQL.

USB disk mount point can be defined in /etc/fstab as:

```
/dev/sdb1   /mnt/backup   ext4   defaults   0   0
```

These are the global script sections:

- *Global var configuration under **## BEGIN CONFIG ##** section.*
- *Checking user root under **# Check root** user section.*
- *Deleting older database backup **# Delete older local database backup** section.*
- *Mount disk (optional section) under **# Mount disk** section.*
- *Stop application (optional section) under **# Stop tomcat** section.*
- *Clean logs (optional section) under **# Clean logs** section.*
- *Backup the database to the file system under **#Backup database** section.*
- *Backup repository and database dump to backup destination under **#Backup and purge old backups** section.*
- *Start application (optional section) **# Start tomcat** section.*
- *Show statistics (optional section) **# Status** section.*
- *Un mount disk (optional section) under **# Umount disk** section.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuration parameters explanation:

Parameter	*Description*
HOST	*The server host name.*
DATABASE_PASS	*The database password.* *The data base user to do the backup is always root user.*
OPENKM_DB	*The application database name.* *Usually will be named "okmdb".*
OPENKM_HOME	*The openkm home folder.*
TOMCAT_HOME	*The tomcat home folder.* *Usually into OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually into the same server.*
BACKUP_DIR	*Backup destination.*

*Create a backup script **/root/backup.sh***

```
$ sudo su
$ vim /root/backup.sh
```

```
#!/bin/bash
#
## BEGIN CONFIG ##
HOST=$(uname -n)
DATABASE_PASS="*secret*"
```

```
OPENKM_DB="okmdb"
OPENKM_HOME="/home/openkm"
TOMCAT_HOME="$OPENKM_HOME/tomcat-9.0.76"
DATABASE_EXP="$OPENKM_HOME/db"
BACKUP_DIR="file:///mnt/backup"
## END CONFIG ##

# Check root user
if [ $(id -u) != 0 ]; then echo "You should run this script as root"; exit; fi

# Delete older local database backup
echo -e "### BEGIN: $(date +"%x %X") ###\n"
rm -rf $DATABASE_EXP
mkdir -p $DATABASE_EXP

# Mount disk
if mount | grep "$BACKUP_DIR type" > /dev/null; then
  echo "$BACKUP_DIR already mounted";
else
  mount "$BACKUP_DIR";

  if mount | grep "$BACKUP_DIR type" > /dev/null; then
    echo "$BACKUP_DIR mounted";
  else
    echo "$BACKUP_DIR error mounting";
    exit -1;
  fi
fi

# Stop Tomcat
/etc/init.d/tomcat stop

# Clean logs
#echo "Clean Tomcat temporal files."
#rm -rf $TOMCAT_HOME/logs/*
#rm -rf $TOMCAT_HOME/temp/*
#rm -rf $TOMCAT_HOME/work/Catalina/localhost

# Backup database
if [ -n "$DATABASE_PASS" ]; then
  echo "* Backuping MySQL data from $OPENKM_DB..."
  mysqldump -h localhost -u root -p$DATABASE_PASS $OPENKM_DB > $DATABASE_EXP/mysql_$OPENKM_DB
  echo "---------------------------------------";
fi

# Backup and purge old backups
duplicity remove-older-than 1M --force $BACKUP_DIR/$HOST

if [ $(date +%u) -eq 7 ]; then
  echo "*** Full Backup ***"
  duplicity full --no-encryption $OPENKM_HOME $BACKUP_DIR/$HOST
  RETVAL=$?
else
  echo "*** Incremental Backup ***"
  duplicity --no-encryption $OPENKM_HOME $BACKUP_DIR/$HOST
  RETVAL=$?
fi

[ $RETVAL -eq 0 ] && echo "*** SUCCESS ***"
[ $RETVAL -ne 0 ] && echo "*** FAILURE ***"

# Start Tomcat
/etc/init.d/tomcat start
echo -e "\n### END: $(date +"%x %X") ###"

# Status
echo "================================";
```

```
duplicity collection-status $BACKUP_DIR/$HOST
echo "*********************************";
df -h | grep "$BACKUP_DIR"
echo "=================================";

# Umount disk
sync
umount "$BACKUP_DIR"
```

Increase the backup period

Modify the ***# Backup and purge old backups*** *section increasing the parameter value:*

> *The "**1M**" value indicate will be removed backups older one month. Changing to "**3M**" or "**1Y**" indicate older 3 Month or 1 Year, etc..*

```
--remove-older-than 3M
```

Perform the remote backup

Modify the ***## Config section*** *section.*

```
BACKUP_DIR="ftp://user@ftp.domain.es/backup"
export FTP_PASSWORD="WhateverPasswordYouSetUp"
```

Modify the ***# Backup and purge old backups*** *section.*

```
# Backup and purge old backups
duplicity remove-older-than 1M --force $BACKUP_DIR/$HOST

if [ $(date +%u) -eq 7 ]; then
  echo "*** Full Backup ***"
  duplicity full --no-encryption $OPENKM_HOME $BACKUP_DIR/$HOST
  RETVAL=$?
else
  echo "*** Incremental Backup ***"
  duplicity --no-encryption $OPENKM_HOME $BACKUP_DIR/$HOST
  RETVAL=$?
fi

[ $RETVAL -eq 0 ] && echo "*** SUCCESS ***"
[ $RETVAL -ne 0 ] && echo "*** FAILURE ***"
```

Modify the ***# Status*** *section.*

```
duplicity collection-status $BACKUP_DIR/$HOST
unset FTP_PASSWORD
```

Perform the remote backup to SMB or CIFS

Modify the ***# Mount disk*** *section.*

```
# Mount disk
if mount | grep "$BACKUP_DIR type" > /dev/null; then
  echo "$BACKUP_DIR already mounted";
else
  echo "Mounting $BACKUP_DIR ...";
  mount -t cifs //REMOTE_SERVER/REMOTE_DIR $BACKUP_DIR -o username=REMOTE_USER,password=REM

  if mount | grep "$BACKUP_DIR type" > /dev/null; then
    echo "$BACKUP_DIR mounted";
  else
    echo "Error mounting $BACKUP_DIR.";
    exit -1;
  fi
fi
```

Do the PostgreSQL backup

Modify the ***# Backup database*** *section*

```
# Backup de PostgreSQL
echo "* Backuping PostgreSQL data from $OPENKM_DB..."
su postgres -c "pg_dump $OPENKM_DB" > $DATABASE_EXP/pg_$OPENKM_DB.sql

# Databases optimizations
su postgres -c "vacuumdb -a -z" > /dev/null
su postgres -c "reindexdb -a -q" 2> /dev/null
```

Do the remote backup to Amazon s3

Modify the ***## Config section*** *section.*

```
BACKUP_DIR="s3+http://somebucket/somedirectory"
export AWS_ACCESS_KEY_ID=""
export AWS_SECRET_ACCESS_KEY=""
export PASSPHRASE=""
```

Modify the ***# Backup and purge old backups*** *section.*

```
# Backup and purge old backups
if [ $(date +%u) -eq 7 ]; then
  echo "*** Full Backup ***"
  duplicity full --no-encryption --s3-use-new-style $OPENKM_HOME $BACKUP_DIR/$HOST
  RETVAL=$?
else
  echo "*** Incremental Backup ***"
  duplicity --no-encryption --s3-use-new-style $OPENKM_HOME $BACKUP_DIR/$HOST
  RETVAL=$?
fi

[ $RETVAL -eq 0 ] && echo "*** SUCCESS ***"
```

Modify the ***# Status*** *section.*

```
# Status
```

```
duplicity collection-status --s3-use-new-style $BACKUP_DIR/$HOST
unset AWS_ACCESS_KEY_ID
unset AWS_SECRET_ACCESS_KEY
unset PASSPHRASE
```

Configure crontab

To install the cron job, run:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

Add these lines according to your personal configuration:

```
MAILTO=nomail@openkm.com
@daily /root/backup.sh | tee /root/logs/backup.$(date +\%Y.\%m.\%d_\%H.\%M.\%S).log
```



More information at Crontab quick reference

*If you want to be notified by mail you should install "**postfix**" service in your server.*

Troubleshooting deleting old backups

You have configured Duplicity to make a full backup every 7 days, and deleting backups older than 14 days to save space. What happen if you see a message like this?

```
Tue May  1 00:00:00 2012
Wed May  2 00:00:00 2012
Which can't be deleted because newer sets depend on them
```

Duplicity uses rsync, which contains incremental changes. Those files won't be deleted because, even though the backup maybe older than 7 days, there are backups which are incremental and younger than 7 days.

So, after 2 weeks have passed, those files will be deleted, since the full backup and the incremental backups are now 14 days old, and there exists a full backup newer than the full and incremental backups.

Restoring backup

You can list the available backups:

```
$ duplicity collection-status /path/to/backup
Sun Sep 11 05:00:19 2012       4.25 GB       4.25 GB   (current mirror)
Sun Sep  4 00:00:18 2012       13.3 MB       4.26 GB
Sun Aug 28 00:00:13 2012        674 MB        4.92 GB
Sun Aug 21 00:00:14 2012       5.50 MB        4.93 GB
```

```
Sun Aug 14 00:00:16 2012    1.75 MB    4.93 GB
Sun Aug  7 00:00:12 2012    288 KB     4.93 GB
Sun Jul 31 00:00:13 2012    43.0 KB    4.93 GB
Fri Jul 29 10:36:39 2012    5.56 KB    4.93 GB
```

Then decide to restore some of the backups, for example the one made on Sun Aug 28 00:00:13 2012:

```
$ duplicity restore --restore-time 2012-08-28 /path/to/backup /path/to/destination
```

*The **--restore-time** parameter accepts several formats. See **duplicity** documentation and Duplicity Backup Howto for more info.*

*Inside of **/path/to/destination** you should see a directory **/home/openkm**, and inside it a couple of directories:*

- ***db**: The backup of the database.*
- ***tomcat-9.0.76**: The backup of the Tomcat installation and OpenKM repository into.*

Additional information

- *Using Duplicity to Backup Your Data*
- *Ubuntu/Debian - Encrypted Incremental Backups With Duplicity on Amazon S3*
- *Duplicity Backup Howto*
- *Duplicity URL Format*
- *Installation and configuration of duplicity for encrypted SFTP remote backup*
- *Playing with Duplicity backup and restore tool and Amazon S3*
- *Bash wrapper script for automated backups with duplicity*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Linux backup and restore with rsync

Rsync is a software application and network protocol for Unix, Linux and Windows systems which synchronizes files and directories from one location to another while minimizing data transfer using delta encoding when appropriate. An important feature of rsync not found in similar programs/protocols is that the mirroring takes place with only one transmission in each direction. rsync can copy or display directory contents and copy files, optionally using compression and recursion.

Preliminaries

In these scripts are assumed these default values:

- ***$TOMCAT_HOME** value is "**/home/openkm/tomcat-9.0.76**"*
- *OpenKM **database** is named "**okmdb**".*
- *OpenKM **database password** value is "***secret***".*
- *OpenKM start & stop **service script** location is "**/etc/init.d/tomcat**"*

Sample script



As a good practice the backup should be done by root user.



The script below does a backup to USB and the application database is MySQL.

USB disk mount point can be defined in /etc/fstab as:

```
/dev/sdb1   /mnt/backup   ext4   defaults   0   0
```

These are the global script sections:

- *Global var configuration under **## BEGIN CONFIG ##** section.*
- *Checking user root under **# Check root** user section.*
- *Deleting older database dump **# Delete older local database backup** section.*
- *Mount disk (optional section) under **# Mount disk** section.*
- *Stop application (optional section) under **# Stop tomcat** section.*
- *Clean logs (optional section) under **# Clean logs** section.*
- *Backup the database to the filesystem under **#Backup database** section.*
- *Backup repository and database dump to backup destination under **# Create backup** section.*
- *Start application (optional section) **# Start tomcat** section.*
- *Umount disk (optional section) under **# Umount disk** section.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuration parameters explanation:

Parameter	*Description*
HOST	*The server host name.*
DATABASE_PASS	*The database password.* *The database user to do backup is always root user.*
OPENKM_DB	*The application database name.* *Usually will be named "okmdb".*
OPENKM_HOME	*The openkm home folder.*
TOMCAT_HOME	*The tomcat home folder.* *Usually into OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually into the same server.*
BACKUP_DIR	*Backup destination.*
RSYNC_OPTS	*Rsync parameters.*

*Create a backup script **/root/backup.sh***

```
$ sudo su
$ vim /root/backup.sh
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
#!/bin/bash
#
## BEGIN CONFIG ##
HOST=$(uname -n)
DATABASE_PASS="*secret*"
OPENKM_DB="okmdb"
OPENKM_HOME="/home/openkm"
TOMCAT_HOME="$OPENKM_HOME/tomcat-9.0.76"
DATABASE_EXP="$OPENKM_HOME/db"
BACKUP_DIR="/mnt/backup"
RSYNC_OPTS="-apzhR --stats --delete --exclude=*~ --delete-excluded"
## END CONFIG ##

# Check root user
if [ $(id -u) != 0 ]; then echo "You should run this script as root"; exit; fi

# Delete older local database backup
echo -e "### BEGIN: $(date +"%x %X") ###\n"
rm -rf $DATABASE_EXP
mkdir -p $DATABASE_EXP

# Mount disk
if mount | grep "$BACKUP_DIR type" > /dev/null; then
  echo "$BACKUP_DIR already mounted";
else
  mount "$BACKUP_DIR";

  if mount | grep "$BACKUP_DIR type" > /dev/null; then
    echo "$BACKUP_DIR mounted";
  else
    echo "$BACKUP_DIR error mounting";
    exit -1;
  fi
fi

# Stop Tomcat
/etc/init.d/tomcat stop

# Clean logs
#echo "Clean Tomcat temporal files."
#rm -rf $TOMCAT_HOME/logs/*
#rm -rf $TOMCAT_HOME/temp/*
#rm -rf $TOMCAT_HOME/work/Catalina/localhost

# Backup database
if [ -n "$DATABASE_PASS" ]; then
  echo "* Backuping MySQL data from $OPENKM_DB..."
  mysqldump -h localhost -u root -p$DATABASE_PASS $OPENKM_DB > $DATABASE_EXP/mysql_$OPENKM_DB
  echo "---------------------------------------";
fi

# Create backup
rsync $RSYNC_OPTS $OPENKM_HOME $BACKUP_DIR/$HOST

# Start Tomcat
/etc/init.d/tomcat start
echo -e "\n### END: $(date +"%x %X") ###"

# Umount disk
sync
umount "$BACKUP_DIR"
```

Do backup with rotation

*Modify the **# Create backup** section.*

```
# Create backup
# Calculate snapshot
LAST_SNAPSHOT=`ls -ltr $BACKUP_DIR | tail -1 | awk {'print $8'} | cut -d . -f 2`
NEW_SNAPSHOT=$((LAST_SNAPSHOT+1))

rsync $RSYNC_OPTS --link-dest="$BACKUP_DIR/$HOST/backup.$LAST_SNAPSHOT" $FILES "$BACKUP_DIR/
```

Other configuration:

```
# Create backup
"cd $BACKUP_DIR/$HOST; rm -rf backup.3; mv backup.2 backup.3; mv backup.1 backup.2; mv backup
rsync $RSYNC_OPTS --link-dest="$BACKUP_DIR/$HOST/backup.1" "OPENKM_HOME" "$BACKUP_D
```

Do remote backup

*Modify the **## BEGIN CONFIG ##** section.*

```
BACKUP_SRV="user@server.com"
```

*Modify the **# Create backup** section.*

```
# Create backup
# Create missing backup folder ( optional )
ssh $BACKUP_SRV "mkdir -p $BACKUP_DIR"

# Calculate snapshot
NEW_SNAPSHOT=$(date "+%Y-%m-%dT%H:%M:%S")
LAST_SNAPSHOT=$(ssh $BACKUP_SRV ls -tr $BACKUP_DIR/$HOST | tail -1)

if [ $LAST_SNAPSHOT ]; then
  echo "Incremental backup $NEW_SNAPSHOT based on $LAST_SNAPSHOT... "
  rsync $RSYNC_OPTS --link-dest="$BACKUP_DIR/$HOST/$LAST_SNAPSHOT" "$OPENKM_HOME" "$BACKUP
else
  echo "Initial full backup $NEW_SNAPSHOT..."
  rsync $RSYNC_OPTS "$OPENKM_HOME" "$BACKUP_SRV:$BACKUP_DIR/$HOST/$NEW_SNAPSHOT"
fi
```

More simple configuration:

```
rsync $RSYNC_OPTS "$OPENKM_HOME" "$BACKUP_SRV:$BACKUP_DIR/$HOST"
```

Do incremental backup with backup-dir

*Modify the **## BEGIN CONFIG ##** section.*

```
RSYNC_OPTS="-ahR --partial --stats --delete --exclude=*~ --exclude=tomcat-9.0.76/temp/* --exclude=tomcat-9.0.7
```

*Modify the **# Create backup** section.*

```
# Create backup
NEW_SNAPSHOT=`date +%Y-%m-%d`
echo -e "\n### RSYNC - BEGIN : $(date +"%x %X") ###\n"
echo "* New Snapshot: $NEW_SNAPSHOT"
rsync $RSYNC_OPTS --backup --backup-dir="$BACKUP_DIR/$HOST/INCREMENTAL/$NEW_SNAPSHOT" $OPE
```

Do incremental backup with link-dest

*Modify the **## BEGIN CONFIG ##** section.*

```
RSYNC_OPTS="-ahR --partial --stats --delete --exclude=*~ --exclude=tomcat-9.0.76/temp/* --exclude=tomcat-9.0.7
```

*Modify the **# Create backup** section.*

```
# Create backup
# Calculate snapshot
LAST_SNAPSHOT=`ssh $BACKUP_SRV "cd $BACKUP_DIR && stat --format='%n' * | tail -1"`
NEW_SNAPSHOT=`date +%Y-%m-%d`
echo -e "\n### RSYNC - BEGIN : $(date +"%x %X") ###\n"
echo "* Last Snapshot: $LAST_SNAPSHOT"
echo "* New Snapshot: $NEW_SNAPSHOT"

if [ ! $LAST_SNAPSHOT ]; then
  rsync $RSYNC_OPTS $OPENKM_HOME/ "$BACKUP_DIR/$HOST/$NEW_SNAPSHOT"
else
  rsync $RSYNC_OPTS --link-dest="$BACKUP_DIR/$HOST/$LAST_SNAPSHOT" $OPENKM_HOME/ "$BACKUP_
fi
```

Do remote backup to SMB or CIFS

*Modify the **# Mount disk** section.*

```
# Mount disk
if mount | grep "$BACKUP_DIR type" > /dev/null; then
  echo "$BACKUP_DIR already mounted";
else
  echo "Mounting $BACKUP_DIR ...";
  mount -t cifs //REMOTE_SERVER/REMOTE_DIR $BACKUP_DIR -o username=REMOTE_USER,password=REM
  if mount | grep "$BACKUP_DIR type" > /dev/null; then
    echo "$BACKUP_DIR mounted";
  else
    echo "Error mounting $BACKUP_DIR.";
    exit -1;
  fi
fi
```

Do PostgreSQL backup

*Modify the **# Backup database** section*

```
# Backup de PostgreSQL
echo "* Backuping PostgreSQL data from $OPENKM_DB..."
su postgres -c "pg_dump $OPENKM_DB" > $DATABASE_EXP/pg_$OPENKM_DB.sql

# Databases optimizations
su postgres -c "vacuumdb -a -z" > /dev/null
su postgres -c "reindexdb -a -q" 2> /dev/null
```

Configure crontab

To install the cron job, run:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

And add these lines according to your personal configuration:

```
MAILTO=nomail@openkm.com
@daily /root/backup.sh | tee /root/logs/backup.$(date +\%Y.\%m.\%d_\%H.\%M.\%S).log
```

More information at Crontab quick reference

*If you want to be notified by mail you should install "**postfix**" service in your server.*

Rsync and FTP

*Rsync need SSH access to work, but you can also use it with an FTP server if you mount it locally with **curlftpfs**:*

Backup performance may degrade using this method.

Install curlftpfs and rsync:

```
$ sudo apt-get install curlftpfs rsync
```

Create mountpoin:

```
$ mkdir /path/to/mount
```

Mount remote director:

```
$ curlftpfs ftp.example.com /path/to/mount
```

Create local directory:

```
$ mkdir /local/path
```

Rsync mounted directory to local director:

```
$ rsync -r -t -v --progress --bwlimit=500 /path/to/mount/ /local/path/
```

Restoring backup

You can list the available backups:

```
$ ls -ld /path/to/backup/*
drwxr-xr-x 4 root root 4096 sep 11 05:00:19 2012 2012-09-11T05:00:19
drwxr-xr-x 4 root root 4096 sep  4 00:00:18 2012 2012-09-04T00:00:18
drwxr-xr-x 4 root root 4096 aug 28 00:00:13 2012 2012-08-28T00:00:13
drwxr-xr-x 4 root root 4096 aug 21 00:00:14 2012 2012-08-21T00:00:14
drwxr-xr-x 4 root root 4096 aug 14 00:00:16 2012 2012-08-14T00:00:16
drwxr-xr-x 4 root root 4096 aug  7 00:00:12 2012 2012-08-07T00:00:12
drwxr-xr-x 4 root root 4096 jul 31 00:00:13 2012 2012-07-31T00:00:13
drwxr-xr-x 4 root root 4096 jul 29 10:36:39 2012 2012-07-29T10:36:39
```

Then decide to restore some of the backups, for example the one made on Sun Aug 28 00:00:13 2012:

```
$ cp /path/to/backup/2012-08-28T00:00:13 /path/to/destination
```

*See **rsync** documentation for more info.*

*Inside of **/path/to/destination** you should see a directory **/home/openkm**, and inside it a couple of directories:*

- ***db**: The backup of the database.*
- ***tomcat-9.0.76**: The backup of the Tomcat installation and OpenKM repository into.*

Additional information

- *Easy Automated Snapshot-Style Backups with Linux and Rsync*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Linux backup with LFTP

LFTP is a sophisticated FTP client, and a file transfer program supporting a number of network protocols.



This configuration does not provide incremental backups.

Preliminaries

In these scripts are assumed these default values:

- ***$TOMCAT_HOME** value is "**/home/openkm/tomcat-9.0.76**"*
- *OpenKM **database** is named "**okmdb**".*
- *OpenKM **database password** value is "***secret***".*
- *OpenKM start & stop **service script** location is "**/etc/init.d/tomcat**"*

Sample script



As a good practice the backup should be done by root user.



The script below does a backup to USB and the application database is MySQL.

USB disk mount point can be defined in /etc/fstab as:

```
/dev/sdb1   /mnt/backup   ext4   defaults   0   0
```

These are the global script sections:

- *Global var configuration under **## BEGIN CONFIG ##** section.*
- *Checking user root under **# Check root** user section.*
- *Deleting older database backup **# Delete older local database backup** section.*
- *Stop application (optional section) under **# Stop tomcat** section.*
- *Clean logs (optional section) under **# Clean logs** section.*
- *Backup the database to the filesystem under **#Backup database** section.*
- *Backup repository and database dump to backup destination under **#Backup and purge old backups** section.*
- *Start application (optional section) **# Start tomcat** section.*

Configuration parameters explanation:

Parameter	*Description*

HOST	*The server host name.*
NOW	*Actual date in format 'dd-MM-yyyy'.*
DATABASE_PASS	*The database password.* *The data base user to do backup is always root user.*
OPENKM_DB	*The application database name.* *Usually will be named "okmdb".*
OPENKM_HOME	*The openkm home folder.*
TOMCAT_HOME	*The tomcat home folder.* *Usually into OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually into the same server.*
BACKUP_DIR	*FTP Backup destination.*

*Create a backup script **/root/backup.sh***

```
sudo su

vim /root/backup.sh
```

```
#!/bin/bash
#
## BEGIN CONFIG ##
HOST=$(uname -n)
NOW=$(date +"%d-%m-%Y")
```

```
DATABASE_PASS="*secret*"
OPENKM_DB="okmdb"
OPENKM_HOME="/home/openkm"
TOMCAT_HOME="$OPENKM_HOME/tomcat-9.0.76"
DATABASE_EXP="$OPENKM_HOME/db"
BACKUP_DIR="ftp://user:password@ftp.server/backup"
## END CONFIG ##

# Check root user
if [ $(id -u) != 0 ]; then echo "You should run this script as root"; exit; fi

# Delete older local database backup
echo -e "### BEGIN: $(date +"%x %X") ###\n"
rm -rf $DATABASE_EXP
mkdir -p $DATABASE_EXP

# Stop Tomcat
/etc/init.d/tomcat stop

# Clean logs
#echo "Clean Tomcat temporal files."
#rm -rf $TOMCAT_HOME/logs/*
#rm -rf $TOMCAT_HOME/temp/*
#rm -rf $TOMCAT_HOME/work/Catalina/localhost

# Backup database
if [ -n "$DATABASE_PASS" ]; then
  echo "* Backuping MySQL data from $OPENKM_DB..."
  mysqldump -h localhost -u root -p$DATABASE_PASS $OPENKM_DB > $DATABASE_EXP/mysql_$OPENKM_DB
  echo "-------------------------------------";
fi

# Backup and purge old backups
lftp $BACKUP_DIR -e "mkdir $NOW; mirror --reverse $OPENKM_HOME $NOW; exit"

# Start Tomcat
/etc/init.d/tomcat start
echo -e "\n### END: $(date +"%x %X") ###"
```

Configure crontab

To install the cron job, run:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

Add these lines according to your personal configuration:

```
MAILTO=nomail@openkm.com
@daily /root/backup.sh | tee /root/logs/backup.$(date +\%Y.\%m.\%d_\%H.\%M.\%S).log
```

More information at Crontab quick reference

*If you want to be notified by mail you should install "**postfix**" service in your server.*

Cluster configuration

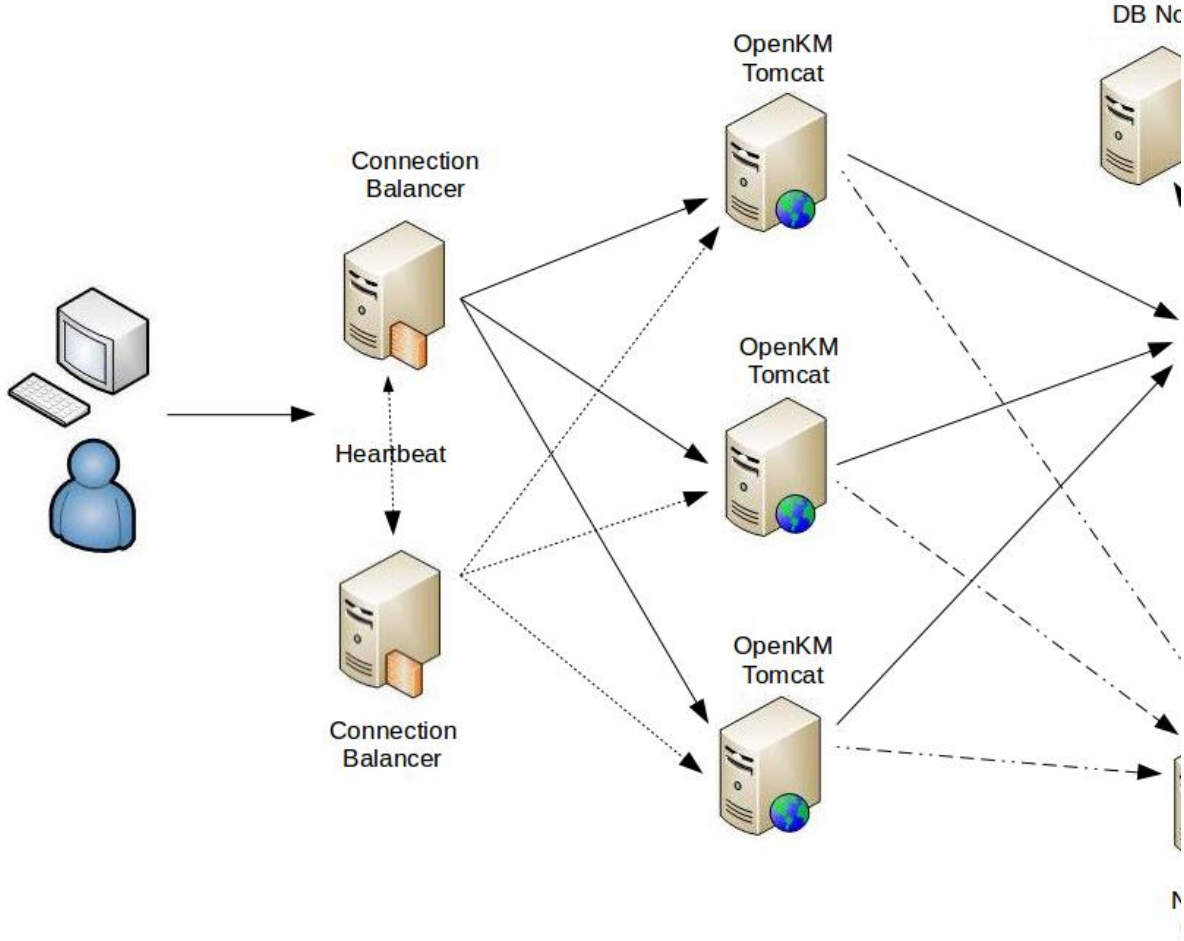
The updated cluster architecture presents enhancements over the previous one and improves specific bugs found in early versions:

- ***Single error point**: Only the master can modify the index. If the master fails, the whole system will have problems.*
- ***Latency**: Slave nodes have to refresh their local index every few minutes. So there is a latency between master and slave while the process completes.*

This kind of architecture provides server load spreading as well as high availability. Let's take a look at the proposed architecture. Several OpenKM servers and one (or several) HAProxy servers will distribute the user petitions to these OpenKM servers. HAProxy can also be configured to detect which OpenKM server is down and avoid sending petitions until it is up again.

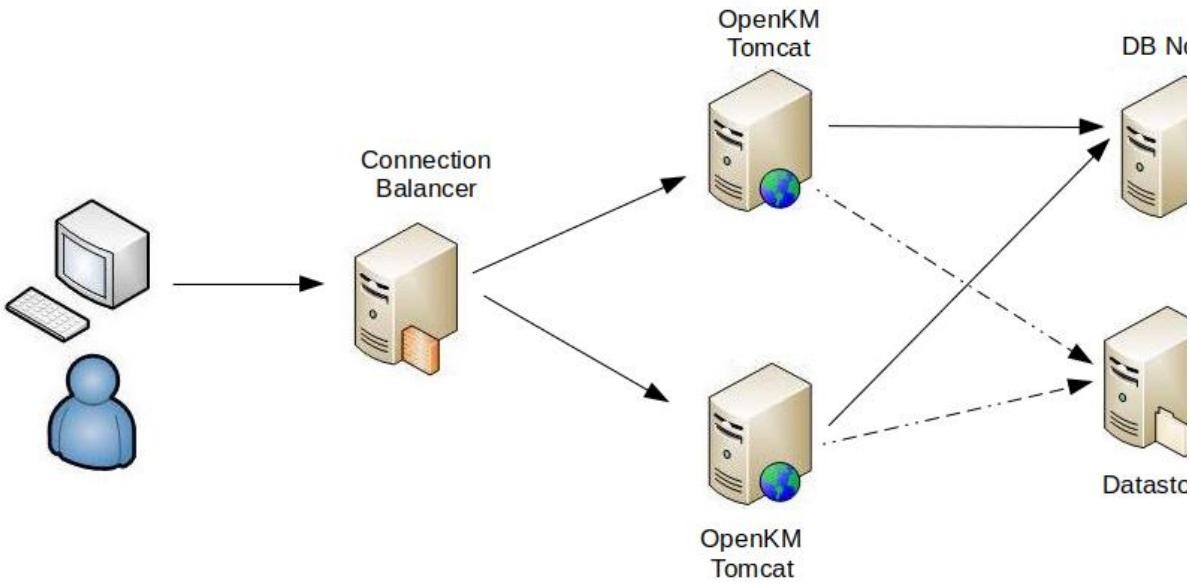
Optionally you can configure several replicated database servers (also handled by another HAProxy), which prevent the service from being interrupted in case of a database node goes down. By default, OpenKM stores the documents in disk to maximize performance, so you need these files replicated across your OpenKM servers. An option is GlusterFS, a software solution to create distributed filesystems, but NFS is quite good in most situations and easier to set up in the case of Linux servers.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Remember that this is only a proposal, and you are free to implement whatever infrastructure you want to achieve the same results. To avoid an excessive and complex configuration, we will describe a simplified architecture that will provide the same objective: the master node, the database, and the NFS server are in the same machine.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.





*Remember that every OpenKM instance should use the same database instance. Only one database for all the instances. This resource is configured in **TOMCAT_HOME/openkm.properties** file.*

If you look into the TOMCAT_HOME/repository directory, you can see these others:

- ***cache**: These are generated files used primarily on the preview. In case of being deleted, they will be re-created again.*
- ***Datastore**: These files are the binary content of the documents stored in OpenKM. You should take care of them, and a backup is always recommended.*
- ***Index**: Where Lucene stores the indexes used in document and metadata search. In case of deletion, you can re-index the repository from Administration > Utilities.*

*You can place **cache** and **datastore** folders in another server as shared resources, according to this information. On average, 80% of user actions are read documents, and only 20% write (create or update documents). The problematic one is the **index** folder because Lucene indexes can't be shared between several OpenKM instances. So, what can we do? The solution is that each OpenKM instance has its index, and they send messages between them to keep their local index updated. If a node needs to update the index, it also sends a message to the other nodes, which will process the modification locally.*

Proxy node

This is the only node that should be accessible by the users and will dispatch the petitions between the slave nodes. The most important part is the HAProxy configuration. This configuration format works on HAProxy v1.5

```
frontend LB
```

```
  bind 192.168.0.220:80
  reqadd X-Forwarded-Proto:\ http
  default_backend LB

 backend LB 192.168.0.220:80
  mode http
  stats enable
  stats hide-version
  stats uri /stats
  stats realm HAProxy\ Statistics
  stats auth admin:admin
  balance roundrobin
  option httpchk
  option httpclose
  option forwardfor
  cookie LB insert
  server node1 192.168.0.222:8080 cookie node1 check
  server node2 192.168.0.223:8080 cookie node2 check
```

According to this configuration, user petitions will be balanced using the round-robin algorithm between the two OpenKM servers. If any OpenKM servers fail, the petitions will only be forwarded to the working one. At http://proxy/stats (default user: admin, password: admin), you will be able to see the node's status and other statistics. A complete manual of HAProxy is available at http://cbonte.github.io/haproxy-dconv/configuration-1.5.html. If you use another HAProxy version, some options may have changed.

Master node

This kind of node is not very different from the slaves, and the only real difference is it can run crontab tasks among other administration actions. As we said previously, to simplify the configuration, this master node will also store the ***cache*** *and* ***datastore*** *files.*

> *In this example, we will also use this server as an NFS server, but it's widespread to have a SAN that acts as the NFS server.*

Let's create these directories:

```
$ sudo mkdir /mnt/okm_cache
$ sudo mkdir /mnt/okm_dstore
$ sudo mkdir /mnt/okm_extraction
$ sudo mkdir /mnt/okm_ocr_template
$ sudo chown openkm:openkm /mnt/okm_*
```

We are exposing these directories by NFS, so we need to install support:

```
$ sudo apt-get install nfs-kernel-server
```

Once installed, edit the ***/etc/exports*** *file and add these lines:*

```
/mnt/okm_cache *(rw,sync,no_root_squash,no_subtree_check)
/mnt/okm_dstore *(rw,sync,no_root_squash,no_subtree_check)
/mnt/okm_extraction *(rw,sync,no_root_squash,no_subtree_check)
/mnt/okm_ocr_template *(rw,sync,no_root_squash,no_subtree_check)
```

After saving it, you have to execute this command (every time you modify this file, you should execute it):

```
$ sudo exportfs -ra
```

Now let's make the symbolic links:

```
$ sudo ln -s /mnt/okm_cache TOMCAT_HOME/repository/cache

$ sudo ln -s /mnt/okm_dstore TOMCAT_HOME/repository/datastore

$ sudo ln -s /mnt/okm_extraction TOMCAT_HOME/repository/extraction

$ sudo ln -s /mnt/okm_ocr_template TOMCAT_HOME/repository/ocr_template

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/cache

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/datastore

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/extraction

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/ocr_template
```

Instead of making links, you can configure OpenKM to create these directories into other locations:

```
repository.extraction.home=/mnt/okm_extraction
repository.datastore.home=/mnt/okm_dstore
repository.cache.home=/mnt/okm_cache
repository.ocr.template.home=/mnt/okm_ocr_template
```

To enable this configuration, edit the ***TOMCAT_HOME/openkm.properties*** *and uncomment this part:*

```
# Cluster - Master
cluster.enabled=true
cluster.node=master
```

Slave nodes

These nodes can write to ***cache*** *and* ***datastore*** *folders. Here we also need to create the shared directories which be used as mount points:*

```
$ sudo mkdir /mnt/okm_cache

$ sudo mkdir /mnt/okm_dstore

$ sudo mkdir /mnt/okm_extraction

$ sudo mkdir /mnt/okm_ocr_template

$ sudo chown openkm:openkm /mnt/okm_*
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Install NFS client support:

```
$ sudo apt-get install nfs-common
```

*We want them to be mounted automatically, so edit the **/etc/fstab** file and append:*

```
# Cluster
master:/mnt/okm_cache /mnt/okm_cache nfs auto,noatime,rsize=8192,wsize=8192,timeo=14,intr
master:/mnt/okm_dstore /mnt/okm_dstore nfs auto,noatime,rsize=8192,wsize=8192,timeo=14,intr
master:/mnt/okm_extraction /mnt/okm_extraction nfs auto,noatime,rsize=8192,wsize=8192,timeo=14,intr
master:/mnt/okm_ocr_template /mnt/okm_ocr_template nfs auto,noatime,rsize=8192,wsize=8192,timeo=14,intr
```

Edit the /etc/hosts file and configure it to resolve the "master" server name to the correct IP.

Now let's mount them:

```
$ sudo mount -a
```

Let's make the symbolic links:

```
$ sudo ln -s /mnt/okm_cache TOMCAT_HOME/repository/cache

$ sudo ln -s /mnt/okm_dstore TOMCAT_HOME/repository/datastore

$ sudo ln -s /mnt/okm_extraction TOMCAT_HOME/repository/extraction

$ sudo ln -s /mnt/ocr_template TOMCAT_HOME/repository/ocr_template

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/cache

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/datastore

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/extraction

$ sudo chown -h openkm:openkm TOMCAT_HOME/repository/ocr_template
```

Instead of making links, you can configure OpenKM to create these directories into other locations:

```
repository.extraction.home=/mnt/okm_extraction
repository.datastore.home=/mnt/okm_dstore
repository.cache.home=/mnt/okm_cache
repository.ocr.template.home=/mnt/okm_ocr_template
```

*To enable this configuration, edit the **TOMCAT_HOME/openkm.properties** and uncomment this part:*

```
# Cluster - Slave
cluster.enabled=true
cluster.node=slave
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Sharding configuration

With sharding, OpenKM can distribute the repository files into different locations. This feature can help in case the main disk is full, so new documents will be placed in a another disk.

> *The best way to understand sharing it is to think that each shard is a different disk.*

All documents will be stored in the main shard, until a new shard is created and set as current shard. Shard can be managed from administration and these options are available:

- *Create a new shard*
- *Set a shard as current*

> *Deleting a shard is not possible because is dangerous and may cause data destruction.*

> *A use case of sharding is to have a fast media like SSD (which is also expensive) in the main shard, and other slower and cheaper HDD in the second shard. Documents will be moved from the fast main shard to the second slow one by a process which can identify less accessed documents. This way the fast shard will be filled with frequently used documents, which also will improve the overall system performance.*

By default, there is only one shard and a specific configuration is needed to make use of this new feature.

In order to configure sharding, use this configuration property:

```
repository.cache.home=${repository.home}/${shard.id}/${tenant.id}/cache
repository.datastore.home=${repository.home}/${shard.id}/${tenant.id}/datastore
repository.extraction.home=${repository.home}/${shard.id}/${tenant.id}/extraction
```

The key is the "shard.id" variable which will be replaced in OpenKM by the currently enabled shard. So, if you are in ***shard #1*** *and* ***tenant #2****, the datastore location will be* ***${repository.home}/shard_1/tenant_1/datastore****.*

> *Currently* ***cache****,* ***datastore*** *and* ***extraction*** *can be divided in shards.*

This is a sample ***repository*** *directory distribution:*

- *index*
- *shard_1*
    - *tenant_1*
        - *cache*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *datastore*
- *extraction*

- *shard_2*
  - *tenant_1*
    - *cache*
    - *datastore*
    - *extraction*
- *tenant_1*
  - *ocr_template*



*The **repository.cache.home**, **repository.datastore.home** and **repository.extraction.home** configuration properties should be set before uploading any documents or mails to OpenKM otherwise you will have problems with documents and mails download, preview and search.*

Sharding and Docker

In case you are using Docker to deploy OpenKM we recommend to externalize the ***openkm.properties*** *configuration property file to be able to configure sharding.*

This is a sample Docker Compose definition for version 8.1.13 (change version value for other docker builds):

```
services:

  openkm_mysql:
    image: docker.openkm.com/private/professional:8.1.13
    container_name: openkm_mysql_shard
    hostname: openkm
    ports:
      - 8080:8080
    volumes:
      - ${PWD}/openkm.properties:/opt/tomcat/openkm.properties
      - ${PWD}/repository:/opt/tomcat/repository
    environment:
      - TZ=Europe/Madrid
    entrypoint: [ "wait-for-it.sh", "mysql:3306", "--timeout=0", "--strict", "--", "entrypoint.sh" ]
    links:
      - mysql:mysql
    networks:
      skynet:
        ipv4_address: 172.28.1.1

  mysql:
    image: mysql:8.0.18
    container_name: okmdb_mysql_shard
    hostname: mysql
    command: --default-authentication-plugin=mysql_native_password --character-set-server=utf8 --collation-server=
    volumes:
      - ${PWD}/mysql:/var/lib/mysql
    environment:
      - MYSQL_DATABASE=okmdb
      - MYSQL_USER=openkm
      - MYSQL_PASSWORD=openkm
```

```
      - MYSQL_ROOT_PASSWORD=openkm
    security_opt:
      - seccomp:unconfined
    networks:
      skynet:
        ipv4_address: 172.28.1.2

volumes:
  openkm_mysql_shard:
  okmdb_mysql_shard:

networks:
  skynet:
    ipam:
      driver: default
      config:
        - subnet: 172.28.0.0/16
```

And this is the according ***openkm.properties*** *(shards with no tenants):*

```
# OpenKM Hibernate configuration values
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://mysql:3306/okmdb?useUnicode=true&characterEncoding=UTF8&serverTimezor
spring.datasource.username=openkm
spring.datasource.password=openkm

# JPA stuff
spring.jpa.hibernate.ddl-auto=create-only
spring.jpa.properties.hibernate.dialect=com.openkm.db.dialect.MySQL5InnoDBDialect

# Logging config
spring.output.ansi.enabled=always

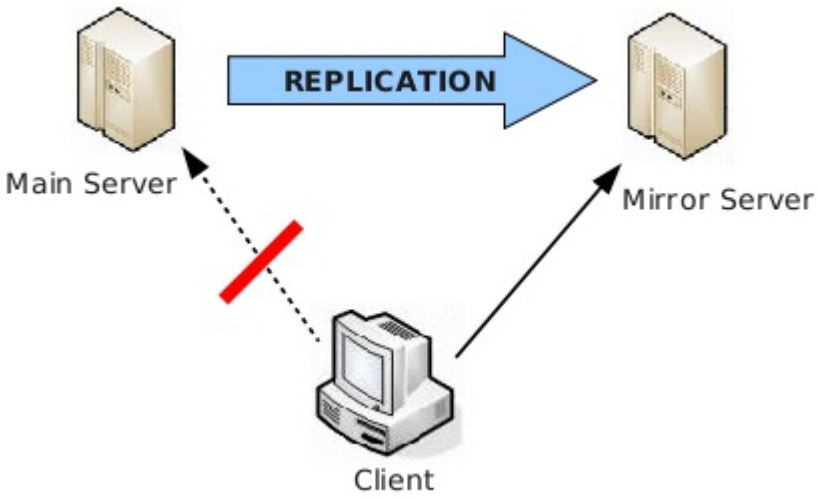
# Extra
repository.datastore.home=/opt/tomcat/repository/${shard.id}/datastore
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Instance replication

To enhance OpenKM's availability, you can have two instances of the application running on different servers. If the principal server goes down due to a hardware failure, you can switch to the mirrored server and keep working.

If you want a typical high availability configuration, you can use the Cluster configuration, but it is a complex architecture if you want to achieve a real high availability scheme. There is another option, easier to setup and also provides good protection against a server failure. This is somewhat like a simplified cluster configuration where the main instance is synchronized with the backup instance. For example, this can be done using a software solution like Rsync, which minimizes the transfer of data.



In this scheme, we assume:

- *Each node has its own document repository and database.*
- *The backup instance is supposed to be a clone of the main instance.*
- *In case the main instance is down, you should send the client request to the backup instance.*

An option is to use a DNS, which you can modify in case of server failure, or an HAProxy instance which, properly configured, can detect a problem in the main server and forward the request to the working one.

Keep in mind that in this configuration, only one server should accept client petitions.

When the main server is down, the backup one becomes the new main server, so the old main server is now the backup one (well, when it is available again).

This is the configuration of each server:

Main server

- *OpenKM is configured normally.*
- *The database is configured to send modifications to the backup instance.*
- *Rsync is configured to send datastore modifications every few minutes to the backup server.*

Mirror server

- *OpenKM is configured in read-only mode.*
- *The database is configured to accept modifications from the main instance.*
- *Datastore files are periodically updated by the Rsync tool from the main server.*

Different implementations

The provided solution is not the only one, but the simplest.

An improvement over this first approach is using an external file server (connected by NFS) so Rsync is not needed. Using a NAS solution that improves the data backup and availability. It is highly recommended if the stored information is critical for your business.

Another improvement is related to the database. Some databases can be configured in master / slave mode, which is the required mode for this server configuration. But you can also deploy the database in a dedicated server (which should also be replicated), but some providers like Oracle offer specific database cluster solutions, which should be taken into consideration.

Sample Linux implementation



The replication is done only in one direction, from the main server to the mirror server.

The mirror server must be configured to be in read-only mode so users can't add or modify documents.

The following script will propagate the repository changes from the main server to the mirror server:



On the sample, the main server 192.168.1.101.

Both OpenKMs are installed at the OpenKM user home (/home/openkm) for simplicity.

This script should be executed - by user root - from the main server, and can be scheduled to be executed every day using the Linux cron utility, for example.

The script uses rsync to minimize network load, and only modified or added documents will be transferred. The whole process can take a few minutes, depending on your repository activity.

```
#!/bin/bash
TOMCAT_HOME="/home/openkm/tomcat-9.0.76"
REMOTE_HOST="root@192.168.1.102"
MYSQL_PASS="*****"

echo -e "### BEGIN: $(date +"%x %X") ###\n"
```

```
# Stop local OpenKM
/etc/init.d/openkm stop

# Stop remote OpenKM
ssh $REMOTE_HOST '/etc/init.d/openkm stop'

# Sync OpenKM repositories
rsync -avh --stats --delete --exclude repository/.system.key --exclude logs $TOMCAT_HOME ${REMOTE_HOST}:/

# Dump and copy database
mysqldump -u root -p${MYSQL_PASS} okmdb | ssh $REMOTE_HOST "mysql -u root -p${MYSQL_PASS} okmdb"

# Start local OpenKM
/etc/init.d/openkm start

# Start remote OpenKM
ssh $REMOTE_HOST '/etc/init.d/openkm start'

echo -e "\n### END: $(date +"%x %X") ###"
```

Copy the following SQL into the remote $TOMCAT_HOME/start.sql server location.

```
update OKM_CONFIG set 'true' where CFG_KEY='system.readonly';
```



After the first synchronization, stop the remote Tomcat, update the start.sql script there, and start the Tomcat again.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

How to move to another server



Although the system should not be affected, before doing this kind of action we recommend having a backup. Refer to the Backup and restore section.

Linux



In the next documentation, will refer to the Installing on Ubuntu and Debian in some sections because are the same actions.

Preliminaries

- *Ensure you **have enough disk space** in the new server.*
- *Check if you have **enough CPU power and RAM** in the new server.*
- ***Create the user** who's the **owner** of the OpenKM files ($TOMCAT_HOME folder). This user is usually named "**openkm**" (more info at Installing on Ubuntu and Debian), but maybe in the installation was choose other one.*
- *Update the OS.*

Stop OpenKM service

It is important while moving the data between the two servers do not perform any kind of changes in the old server. That's why is a good practice to have OpenKM stopped.



*If you **do not want to stop the OpenKM service** , set OpenKM as read-only with a configuration parameter named "**system.readonly**".*

Backup the database

*In the "**openkm.properties**" file (at $TOMCAT_HOME folder) is configured the database connection. In the next sample the user is named "**openkm**" with password "**openkm**" and the connection URL is "**jdbc:mariadb://localhost:3306/okmdb?autoReconnect=true&useUnicode=true&characterEncoding=UTF8&nullNamePatternMatchesAll=true**":*

```
# Database connection
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://localhost:3306/okmdb?autoReconnect=true&useUnicode=true&characterEnco
spring.datasource.username=openkm
spring.datasource.password=openkm
```

*The command below explains how to backup the MySQL and MariaDB database. The database is usually named " **okmdb**":*

```
$ mysqldump -h localhost -u openkm -popenkm okmdb > okmdb.sql
```

Restore the database

- ***Install the same database engine*** *in the new server*.
- ***Create the database user with the same credentials*** *(more info at Installing on Ubuntu and Debian).*
- ***Copy the okmdb.sql file to the new server***.
- ***Restore the backup***.

The sample below explains how to restore the MySQL and MariaDB database. Again, the database is usually named " ***okmdb****":*

```
$ mysql -h localhost -u openkm -popenkm okmdb < okmdb.sql
```

Copy binary files

> *Take a look at the file OpenKM.cfg, the whole repository is usually in the TOMCAT_HOME folder, but depending on your configuration it may be has configured in other location. This repository folder should also be copied.*
>
> *Take a look at the parameters in Change repository home.*

- ***Copy the TOMCAT_HOME folder to the new server***. *We suggest to* ***keep the same origin folder path in destination*** *(otherwise you would need to make extra configuration steps).*
- *Set* ***grants to the owner of the TOMCAT_HOME folder in the new server*** *(usually the user named openkm).*

The sample below shows how to copy it (from the old server to the new server):

```
$ rsync -avh --progress --stats /home/openkm/tomcat-8.5.69 root@newServer:/home/openkm/tomcat-9.0.79
```

The sample below shows how to set the owner (you need to execute as root):

```
$ chown openkm:openkm -Rf /home/openkm/tomcat-8.5.69
```

Configure service in the new server

- *Copy the file* ***/etc/init.d/openkm to the new server at the same location and register as a service*** *. (more info at Installing on Ubuntu and Debian).*

Install third-party tools

> *In OpenKM Administration > Configuration parameters you will find information about most applications required. Some of them are under $TOMCAT_HOME/bin folder, so these have been just copied in previous steps as part of the TOMCAT_HOME folder.*

The minimum required packages are (more info at Installing on Ubuntu and Debian):

- *ImageMagick*
- *Ghostscript*
- *LibreOffice*

- *Tesseract OCR engine (remember the language packs required in your environment)*

Start OpenKM service



Until the license is not moved to the new server you will only be able to access the administration user interface.

- ***In case the new server has less RAM than the old server you need to check*** *the* ***$TOMCAT_HOME/bin/setenv.sh*** *file and adapt properly.*
- *Check the openkm.log file for the startup process and look for possible errors.*

Additional steps

- *If the new server has a different number of cores you need to check the value of the configuration parameters named "****text.extraction.threads****" and "****text.extraction.batch****".*
- ***Open administration*** *at http://localhost:8080/openkm/management/ and* ***execute a repository checker*** *(this action may take a lot of time until completed).*

Moving license



Coordinate with the OpenKM team to move the license from the old server to the new one.

Final checks

- *Check the preview of different document types (PDF, Microsoft Office).*
- *Check the thumbnails of different document types (PDF, Microsoft Office).*
- *Create a folder.*
- *Upload a new document.*
- *Download a previously existing document.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Setup

*This section provides information for **configuring** and **tuning** OpenKM.*

The application contains a lot of Configuration parameters and probably you will not need to know all of them. However, we encourage you to take a look at the Recommended configuration parameters section where you will find the minimum configuration required to use OpenKM.

Also you could be interested on:

- *Preview parameters*
- *User interface configuration parameters*

If you have some issue see Troubleshooting section.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Antivirus configuration

OpenKM can be integrated with antivirus software which can execute antivirus check from the command line.

The example below is based on ClamAV, an Open Source antivirus engine that detects trojans, viruses, malware & other malicious programs.

Linux

Debian / Ubuntu

Install antivirus:

```
$ sudo aptitude install clamav
```

*Go to **Administration > Configuration parameters**:*

Field / Property	Type	Description
system.antivir	**String**	/usr/bin/clamscan
background. antivirus.check	**Boolean**	true
background. antivirus.check.threads	**Integer**	20
background. antivirus.check.batch	**Integer**	2

More information about configuration parameters:

- *Antivirus configuration parameters*
- *Performance configuration parameters*

Red Hat / CentOS

ClamAV is not present by default in default repositories, so you must install it from a non-default one.

Repository valid for Centos 5.2 (for other versions, it may be different):

```
[dag]
name=Dag RPM Repository for Red Hat Enterprise Linux
baseurl=http://apt.sw.be/redhat/el$releasever/en/$basearch/dag/
gpgcheck=1
gpgkey=http://dag.wieers.com/packages/RPM-GPG-KEY.dag.txt
```

```
enabled=1
```

Install antivirus:

```
$ yum install clamd
```

Start the daemon:

```
$ /etc/init.d/clamd start
```

Update the virus database:

```
$ freshclam
```

Windows

To Install and configure ClamAV on Windows:

Note: ***GREEN Text*** *like this denotes commands run in a command console/terminal*

BLACK Text *like this indicates code changes*

- *Download the newest version from: https://www.clamav.net/downloads*
- *Unpack the ZIP archive and move it to C:\ so that it appears in your path as C:\ClamAV*
- *Copy the example conf files from C:\ClamAV\conf_examples to C:\ClamAV*
- *Open up a command terminal: Start->Run* ***cmd***
- *Change to the ClamAV directory:* ***cd C:\ClamAV***
- *Run* ***clamconf.exe****, it should inform you that you need to edit the example config file. (I included this so that you can check that the files are in the right location)*

For DEFAULT MINIMAL configuration (Probably not suitable, and you should configure the parameters for your situation), Edit freshclam.conf, clamav-milter.conf, and comment ***Example****. In addition to commenting* ***Example*** *In clamd. conf; to enable the local server, you will need to change line 80 to:* ***LocalSocket C:\CLAMAV\clamd.socket***

- *Run* ***clamconf.exe*** *again, you will get an output of all your configurations.*



In the beginning, you will get a message about being unable to open the database because it has not yet been created.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Run **mkdir database**. If you run clamconf again, you will see that the database error is gone and that you have 0 signatures*
- *Run **freshclam.exe**, this will start the initial signature download for ClamAV and may take a few minutes.*

*At this point, you can run **clamd.exe** to check that the server can run manually. (If you decide to test run at this point, you will have to kill the process for the next section)*

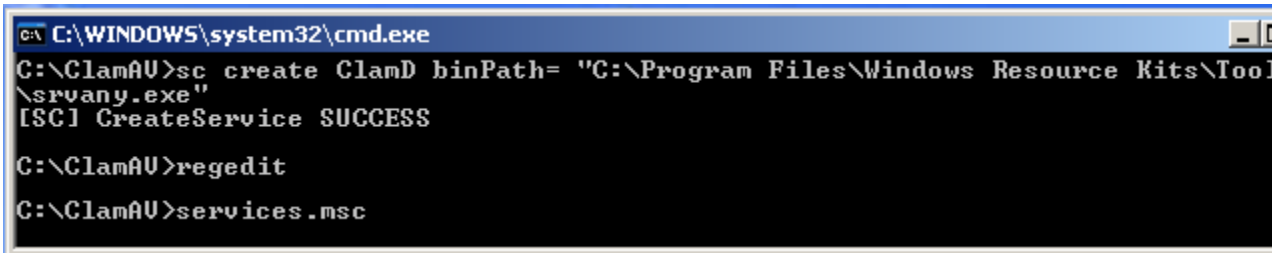
In case you need to create a freshclam.conf file, use this as a base:

```
# URL of server where database updates are to be downloaded from
# If this option is given multiple times, each will be tried in
# the order given until an update is successfully downloaded
#MAKE SYU
DatabaseDirectory "C:\Program Files\ClamAV-x64\db\"
DatabaseMirror database.clamav.net
# Number of times to try each mirror before moving to the next one
MaxAttempts 3
#LogFile C:\Program Files\ClamAV-x64\log\freshclam.log
#NotifyClamd C:\Program Files\ClamAV-x64\clamd.conf
LogFileMaxSize 20480000
LogTime true
UpdateLogFile C:\Program Files\ClamAV-x64\log\freshclam.log
```

Clamd Service Setup

Now to setup it up, so it starts automatically:

Details for the screen capture are below:



- *Run **sc create ClamD binPath= C:\Program Files\Windows Resource Kits\Tools\srvany.exe** (yes, the space is required in this command, and you may have your resource tools in a different location)*

It should tell you: [SC] CreateService SUCCESS

- *Run **Regedit**. Find: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ClamD*
  - *Right-click on ClamD and create a new Key*
  - *Name the key **Parameters***
  - *Click on the new Parameters Key in the right pane to create a new String Value*
  - *Name the String Value **Application***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Right-click and Modify the Application to:* ***C:\ClamAV\clamd.exe -c C:\ClamAV\clamd.conf***
- *Close Regedit*

- *Run* ***services.msc***
  - *Find ClamD in the list*
  - *Right click -> Properties*
  - *Change the Startup type to Automatic*
  - *Select the recovery tab and change all three failures to Restart the Service*
  - *Select the General tab and Apply*

Freshclam Virus Signature Auto Update

Almost done!

- *Navigate to: Start->All Programs->Accessories->System Tools->Scheduled Tasks*
- *Add Scheduled Task*
- *Next->Browse to C:\ClamAV\freshclam.exe*
- *Select to Perform this task daily*
- *Next*
- *Select a good time to update*
- *Next*
- *Enter the user credentials to run as*
- *Next*
- *Select open Advance options*
- *Next*
- *Alter Run to be:* ***C\ClamAV\freshclam.exe -c C:\ClamAV\freshclam.conf***
- *Select the Settings Tab*
- *Change the Stop Task to 30 minutes*
- *Apply*
- *Ok*

Right-click and run the freshclam task to ensure you have no errors.

Go to ***Administration > Configuration parameters****:*

Field / Property	*Type*	*Description*
system.antivir	***String***	***C:\ClamAV\clamscan.exe***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

background. antivirus.check	***Boolean***	*true*
background. antivirus.check.threads	***Integer***	*20*
background. antivirus.check.batch	***Integer***	*2*

More information about configuration parameters:

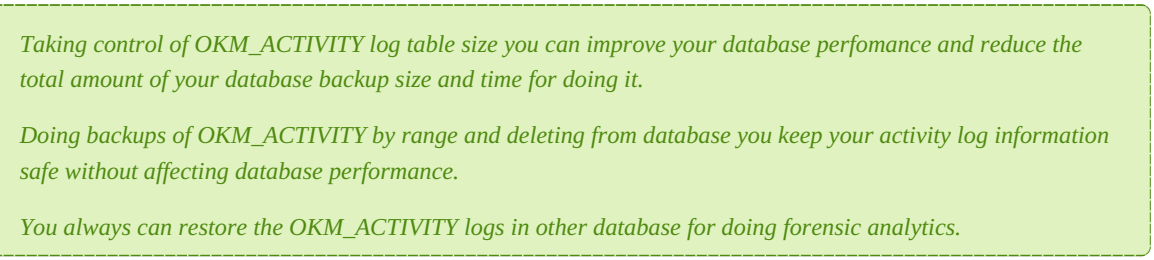
- *Antivirus configuration parameters*
- *Performance configuration parameters*

If everything is configured correctly, OpenKM will scan every uploaded document to check if it's safe.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Cleaning older activity log data

There's a table named OKM_ACTIVITY where openkm saves a log will the actions done by the users on all the nodes. In this table can be easily several milions of rows what can affect the perfomance of the database. Really you should be interested on most actual activity log data - for example last year - and keep the other data outside the database. For it is a good practice to perfom some cleaning process, exporting oldest activity log data and then deleting it from database.



Taking control of OKM_ACTIVITY log table size you can improve your database perfomance and reduce the total amount of your database backup size and time for doing it.

Doing backups of OKM_ACTIVITY by range and deleting from database you keep your activity log information safe without affecting database performance.

You always can restore the OKM_ACTIVITY logs in other database for doing forensic analytics.

MySQL

Linux

*Create a file named **clean_okm_activity.sh** at /root/ folder:*

```
#!/bin/sh
## BEGIN CONFIG ##
DATABASE_PASS="*secret*"
BACKUP_DIR="/mnt/backup/okm_activity"
## END CONFIG ##

mysqldump -h localhost -u root -p$DATABASE_PASS -w"ACT_DATE < (NOW() - INTERVAL 365 DAY)" --no-create
mysql -h localhost -u root -p$DATABASE_PASS okmdb < /root/delete.sql
```

*Make **clean_okm_activity.sh** as executable file:*

```
$chmod +x clean_okm_activity.sh
```

*Create a file named **delete.sql** at /root/ folder:*

```
DELETE FROM OKM_ACTIVITY WHERE ACT_DATE < (NOW() - INTERVAL 365 DAY);
ANALYZE TABLE OKM_ACTIVITY;
```

Configure crontab

To install the cron job, run:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

And add these lines according to your personal configuration:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
MAILTO=nomail@openkm.com
@daily /root/clean_okm_activity.sh | tee /root/logs/okm_activity.$(date +\%Y.\%m.\%d_\%H.\%M.\%S).log
```

More information at Crontab quick reference

*If you want to be notified by mail you should install "**postfix**" service in your server.*

Windows

*Create a file named **clean_okm_activity.bat** at c:\backup folder:*

```
set SQLFILE_DATE=%DATE:~6,4%.%DATE:~3,2%.%DATE:~0,2%
set SQLFILE_TIME=%TIME:~0,2%.%TIME:~3,2%
set SQLFILE=c:\backup\okm_activity-%SQLFILE_DATE%-%SQLFILE_TIME%.sql
set DATABASE_PASS=*secret*

mysqldump -h localhost -u root -p%DATABASE_PASS% -w"ACT_DATE < (NOW() - INTERVAL 365 DAY)" --no-cre
mysql -h localhost -u root -p%DATABASE_PASS% okmdb < delete.sql
```

*Create a file named **delete.sql** at c:\backup folder:*

```
DELETE FROM OKM_ACTIVITY WHERE ACT_DATE < (NOW() - INTERVAL 365 DAY);
ANALYZE TABLE OKM_ACTIVITY;
```

You can use Windows Task Scheduler for configuring periodically your task.

Database replication with federated tables

The procedure for using FEDERATED tables is very simple. Normally, you have two servers running, either both on the same host or on different hosts. (It is possible for a FEDERATED table to use another table that is managed by the same server, although there is little point in doing so.)

You should have two database servers. Production MySQL server (A) and backup MySQL server (B).

First of all must enable federated tables on production server. Stop MySQL production service service and edit the MySQL configuration file (my.cnf or my.ini depending on your OS). Go to [mysqld] section and set below:

```
[mysqld]
# To enable federated feature
federated
```

Start your production MySQL service and execute from your preferred MySQL client tool the command:

```
show create table okm_activity:
```

You will get something like:

```
CREATE TABLE `okm_activity` (
  `ACT_ID` bigint(20) NOT NULL AUTO_INCREMENT,
  `ACT_ACTION` varchar(127) COLLATE utf8_bin DEFAULT NULL,
  `ACT_DATE` datetime DEFAULT NULL,
  `ACT_IP` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `ACT_ITEM` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `ACT_PARAMS` varchar(4000) COLLATE utf8_bin DEFAULT NULL,
  `ACT_PATH` longtext COLLATE utf8_bin,
  `ACT_TENANT` bigint(20) DEFAULT NULL,
  `ACT_USER` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  PRIMARY KEY (`ACT_ID`),
  KEY `IDX_ACTIVITY_TENANT` (`ACT_TENANT`),
  KEY `IDX_ACTIVITY_DATE` (`ACT_DATE`),
  KEY `IDX_ACTIVITY_TNTUSRACT` (`ACT_TENANT`,`ACT_USER`,`ACT_ACTION`),
  KEY `IDX_ACTIVITY_TNTDATACT` (`ACT_TENANT`,`ACT_DATE`,`ACT_ACTION`)
  ) ENGINE=innodb DEFAULT CHARSET=utf8 COLLATE=utf8_bin
```

*Create the table "**okm_activity**" in your backup MySQL server.*

*Create the federated table in your production MySQL server (take a look we have **changed the table name**, the*

***ENGINE=innodb for ENGINE=federated** and we have **added the connection link**):*

```
CREATE TABLE `okm_activity_remote` (
  `ACT_ID` bigint(20) NOT NULL AUTO_INCREMENT,
  `ACT_ACTION` varchar(127) COLLATE utf8_bin DEFAULT NULL,
  `ACT_DATE` datetime DEFAULT NULL,
  `ACT_IP` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `ACT_ITEM` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `ACT_PARAMS` varchar(4000) COLLATE utf8_bin DEFAULT NULL,
  `ACT_PATH` longtext COLLATE utf8_bin,
  `ACT_TENANT` bigint(20) DEFAULT NULL,
  `ACT_USER` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  PRIMARY KEY (`ACT_ID`),
  KEY `IDX_ACTIVITY_TENANT` (`ACT_TENANT`),
  KEY `IDX_ACTIVITY_DATE` (`ACT_DATE`),
  KEY `IDX_ACTIVITY_TNTUSRACT` (`ACT_TENANT`,`ACT_USER`,`ACT_ACTION`),
  KEY `IDX_ACTIVITY_TNTDATACT` (`ACT_TENANT`,`ACT_DATE`,`ACT_ACTION`)
  ) ENGINE=FEDERATED DEFAULT CHARSET=utf8 COLLATE=utf8_bin
  CONNECTION='mysql://user:poassword@server_b:3306/okmdb/okm_activity';
```

Finally you can periodically execute an script like this one from your MySQL production server:

```
select (NOW() - INTERVAL 1 DAY) into @dump_date;
insert into okm_activity_remote select * from okm_activity where act_date < @dump_date;
delete from okm_activity where act_date < @dump_date;
analyze table okm_activity;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Apache HTTP Reverse-Proxy

Exposing OpenKM directly from Tomcat can be risky, if you need the application to be accessed from the Internet (for example https://issues.jboss.org/browse/JBAS-3861). It could be infected by PerlBot. 8080 may be blocked by a firewall, for this reason is a recommendable to expose your OpenKM installation through the standard web port 80. In the following steps we explain how to configure Apache to handle these requests and forward to Tomcat application server using the AJP13 protocol.

From the Apache documentation: The AJP13 protocol is packet-oriented. A binary format was presumably chosen over the more readable plain text for performance reasons. The web server communicates with the servlet container over TCP connections. To cut down on the expensive process of socket creation, the web server will attempt to maintain persistent TCP connections to the servlet container, and to reuse a connection for multiple request/response cycles.

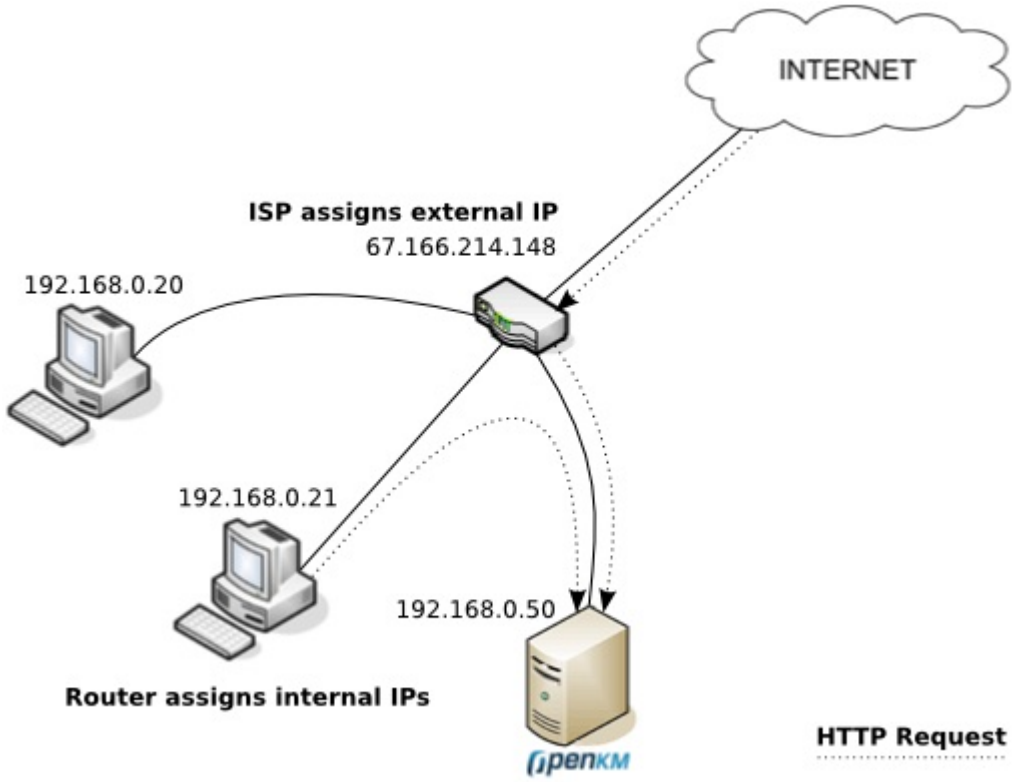
Internal IP vs external IP

Your OpenKM can be accessed from two different zones: Internet and LAN. This means that to access this server you need to use two IPs: external IP (Internet) and internal IP (LAN).

The internal IP address (also known as "local IP address") is the address that is assigned by your local network router that often begins with 192.168.x.x. These IP addresses can only be seen by other computers in your local network (LAN) and not by any computers connected in an external network such the Internet.

To reach the Internet or a computer in another network your computer is often assigned an external IP address, which can then be used to refer to the computer in your local network.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



In the above picture, there are three computers in the local network that have each been assigned their own internal IP address by the router. The ISP is connected to the router and gives the router an external IP address that allows it to communicate with the Internet. On the Internet everyone sees your external IP address, but any information coming from the router is "converted" from the external IP address to the internal IP address.

So if you want your OpenKM installation accessible from both LAN and Internet, the trick here is to configure the client computers to resolve your ***internal IP*** *(192.168.0.50) if they are inside the LAN or the* ***external IP*** *if they are on Internet. To resolve the Internet IP (67.166.214.148) your computer uses the public DNS. So, you need to configure a sort of DNS server inside the LAN or modify every client host file to resolve to the internal IP.*

Debian and Ubuntu

The first thing is to install the required Apache software. From Debian / Ubuntu you can install Apache with a single command:

```
$ sudo apt-get install apache2
```

Edit the file named ***/etc/apache2/apache2.conf*** *and configure a ServerName to prevent warnings in the Apache startup process:*

```
$ vim /etc/apache2/apache2.conf
```

```
ServerRoot "/etc/apache2"
ServerName "localhost"
```

Edit apache2.conf is optional but sometimes needed because of the default server name.

Enable the proxy module:

```
$ sudo a2enmod proxy_ajp

$ sudo a2enmod proxy_wstunnel
```

To get websocket working you also need Tomcat 8.5.95 or later. Note that ***proxy_wstunnel*** *module requires Apache 2.4.5 or later. Please refer to Apache module documentation.*

Create the configuration file ***/etc/apache2/sites-available/openkm.conf*** *with this content:*

```
$ vim /etc/apache2/sites-available/openkm.conf
```

```
<VirtualHost *:80>
  ServerName openkm.your-domain.com
  RedirectMatch ^/$ /openkm

  # WebSocket support - needs proxy-wstunnel
  ProxyPass /openkm/webSocket ws://127.0.0.1:8080/openkm/webSocket
  ProxyPassReverse /openkm/webSocket ws://127.0.0.1:8080/openkm/webSocket

  ProxyPass /openkm ajp://127.0.0.1:8009/openkm keepalive=On
  ProxyPassReverse /openkm http://openkm.your-domain.com/openkm

  ErrorLog /var/log/apache2/openkm-error.log
  CustomLog /var/log/apache2/openkm-access.log combined
</VirtualHost>
```

Change openkm.your-domain.com with your server IP or your domain value.

The VirtualHost ServerName must be other than ServerName in the main Apache configuration. Enable this site configuration:

```
$ cd /etc/apache2/sites-available/

$ sudo a2ensite openkm.conf
```

You have to enable explicitly the proxy access editing the Apache configuration file ***/etc/apache2/mods-available/proxy.conf****:*

```
$ vim /etc/apache2/mods-available/proxy.conf
```

```
<IfModule mod_proxy.c>
  #turning ProxyRequests on and allowing proxying from all may allow
  #spammers to use your proxy to send email.

  ProxyRequests Off

  <Proxy *>
    AddDefaultCharset off
    Order deny,allow
```

```
    Allow from all
    Deny from all
    #Allow from .example.com
  </Proxy>

  # Enable/disable the handling of HTTP/1.1 "Via:" headers.
  # ("Full" adds the server version; "Block" removes all outgoing Via: headers)
  # Set to one of: Off | On | Full | Block

  ProxyVia On
</IfModule>
```



For security reasons, might you be interested in removing default apache configuration:

```
$ a2dissite 000-default.conf
```

Check the configuration

Restart Apache:

```
$ sudo /etc/init.d/apache2 restart
```

Check the access your OpenKM installation from http://openkm.your-domain.com/.



Another advantage of using Apache is that you can log OpenKM access and generate web statistics.



To check the configuration files can execute the command:

```
/usr/sbin/apache2 -t
```

Red Hat and CentOS

Use the yum application manager to install Apache:

```
$ sudo yum install httpd
```

Enable it at boot:

```
$ sudo chkconfig httpd --level 2345 on
```

*Now create the file **/etc/httpd/conf.d/openkm.conf** with this content:*

```
$ vim /etc/httpd/conf.d/openkm.conf
```

```
<VirtualHost *:80>
  ServerName openkm.your-domain.com
  RedirectMatch ^/$ /openkm

  # WebSocket support
  ProxyPass /openkm/webSocket ws://127.0.0.1:8080/openkm/webSocket
  ProxyPassReverse /openkm/webSocket ws://127.0.0.1:8080/openkm/webSocket
```

```
  ProxyPass /openkm ajp://127.0.0.1:8009/openkm keepalive=On
  ProxyPassReverse /openkm http://openkm.your-domain.com/openkm

  ErrorLog /var/log/httpd/openkm-error.log
  CustomLog /var/log/httpd/openkm-access.log combined
</VirtualHost>
```



Change openkm.your-domain.com with your server IP or your domain value.

Check the configuration

Restart Apache:

```
$ sudo /etc/init.d/httpd restart
```

If this command fails, try with:

```
$ sudo systemctl restart httpd
```

Check the access your OpenKM installation from http://openkm.your-domain.com/.



Another advantage of using Apache is that you can log OpenKM access and generate web statistics.

Error Permission denied: proxy: AJP: attempt to connect

If in the log appears an error like ***Permission denied: proxy: AJP: attempt to connect*** *to solve executing the next command:*

```
$ /usr/sbin/setsebool httpd_can_network_connect 1
```

If it works then set the configuration permanent to persist across reboots:

```
$ /usr/sbin/setsebool -P httpd_can_network_connect 1
```

Windows

The Apache web server is already included in "extras" folder.

Edit the ***C:\tomcat\extras\Apache\conf\httpd.conf*** *configuration file and uncomment the* ***VirtualHost*** *section at bottom. Modify the* ***ServerName*** *to match your server name or IP****:***

```
<VirtualHost *:80>
 ServerName 192.168.10.230

 ProxyPass /openkm ajp://localhost:8009/openkm
 ProxyPassReverse /openkm http://192.168.10.230/openkm

 ErrorLog logs/openkm-error.log
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    CustomLog logs/openkm-access.log combined
</VirtualHost>
```

Change 192.168.10.230 with your server ip or your domain value.

Register Apache service

```
C:\> cd tomcat/extras/Apache/bin
C:\> httpd.exe -k install
```

Apache SSL configuration

Set openSSL configuration location:

```
C:\> set OPENSSL_CONF=C:\tomcat\extras\Apache\conf\openssl.cnf
```

Execute this command to generate the self-certificate:

```
C:\> cd c:/tomcat/extras/Apache/bin
C:\> openssl.exe req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ..\conf\server.key -out ..\conf\server.crt
```

This is a sample output:

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Alicante
Locality Name (eg, city) []:Alicante
Organization Name (eg, company) [Internet Widgits Pty Ltd]:OpenKM
Organizational Unit Name (eg, section) []:IT Manager
Common Name (e.g. server FQDN or YOUR name) []:192.168.10.230
Email Address []:noreply@openkm.com
```

Change 192.168.10.230 with your server ip or your domain value.

Un comment this configuration section:

```
# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
```

*Change default server name in **conf/extras/httpd-ssl.conf** to match your domain or IP:*

```
# General setup for the virtual host
DocumentRoot "${SRVROOT}/htdocs"
ServerName 192.168.10.230:443
ServerAdmin admin@example.com
ErrorLog "${SRVROOT}/logs/error.log"
TransferLog "${SRVROOT}/logs/access.log"
```

Troubleshooting

If you see an error message like:



Invalid command 'RewriteEngine', perhaps misspelled or defined by a module not included in the server configuration

You need to enable this Apache module:

```
$ sudo a2enmod rewrite
$ sudo a2enmod proxy_http
$ sudo a2enmod headers
```

If you see errors like these:

```
[proxy_ajp:error] [pid 27579:tid 140057021224704] (70007)The timeout specified has expired: AH01030:
ajp_ilink_receive() can't receive header
[proxy_ajp:error] [pid 27579:tid 140057021224704] [client xxx.xxx.xxx.xxx:60526] AH00992: ajp_read_header:
ajp_ilink_receive failed
[proxy_ajp:error] [pid 27579:tid 140057021224704] (70007)The timeout specified has expired: [client
xxx.xxx.xxx.xxx:60526] AH00878: read response failed from 127.0.0.1:8009 (127.0.0.1)
```

Is possible that Tomcat is not accepting AJP connections. Please, uncomment these lines at $TOMCAT_HOME/conf/server.xml

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector protocol="AJP/1.3" address="127.0.0.1"
           port="8009" redirectPort="8443" secretRequired="false" />
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*The **secretRequired** parameter is importar, and if it's not present you need to add.*

More info at Apache Tomcat 8 Configuration Reference.

Additional information

- *Apache Module mod_proxy*
- *Apache Module mod_proxy_ajp*
- *Apache Module mod_rewrite*
- *mod_rewrite, a beginner's guide*
- *Using Apache Virtual Hosts and ProxyPass Together*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Apache HTTPS Reverse-Proxy

Exposing OpenKM directly from Tomcat can be dangerous if you need the application to be accessed from Internet (for example https://issues.jboss.org/browse/JBAS-3861). As result you can be infected by PerlBot. Also this 8080 may be blocked by a firewall. For these reasons, it is a good practice to expose your OpenKM installation through the standard web port 80. In the following steps we explain how to configure Apache to handle these request and forward to Tomcat application server using the AJP13 protocol.

From the Apache documentation: The AJP13 protocol is packet-oriented. A binary format was presumably chosen over the more readable plain text for performance reasons. The web server communicates with the servlet container over TCP connections. To cut down on the expensive process of socket creation, the web server will attempt to maintain persistent TCP connections to the servlet container, and to reuse a connection for multiple request/response cycles.

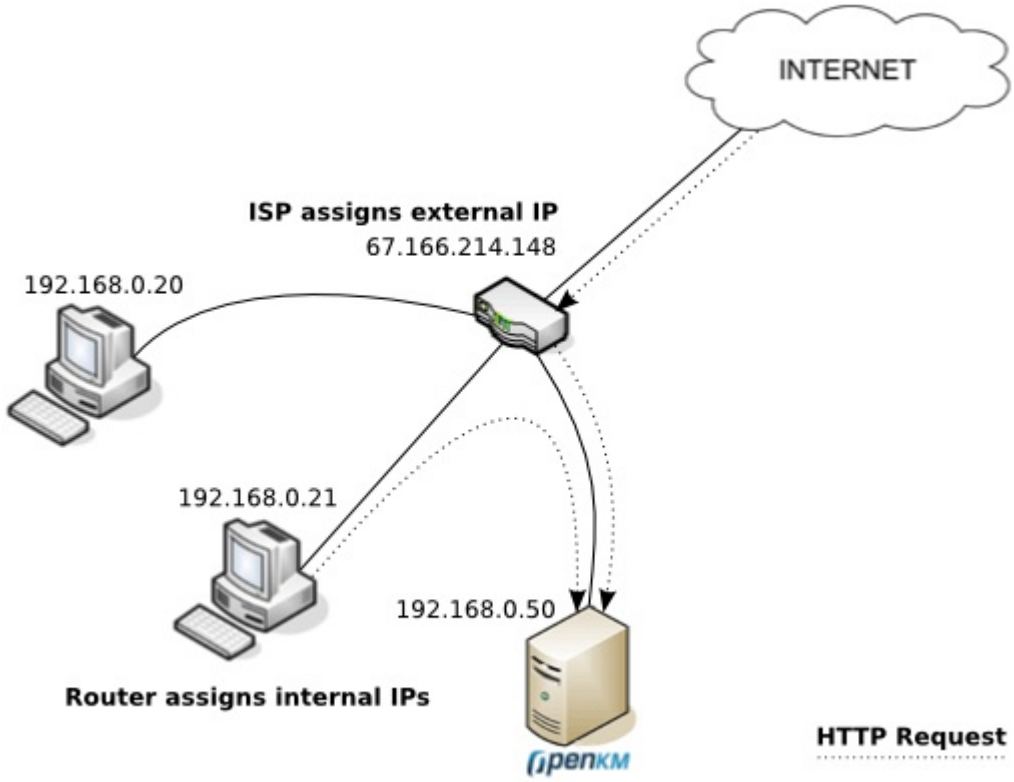
Internal IP vs external IP

Your OpenKM can be accessed from two different zones: Internet and LAN. This means that to access this server you need to use two IPs: external IP (Internet) and internal IP (LAN).

The internal IP address (also known as "local IP address") is the address that is assigned by your local network router that often begins with 192.168.x.x. These IP addresses can only be seen by other computers in your local network (LAN) and not by any computer connected in an external network such the Internet.

To reach the Internet or a computer in another network your computer is often assigned an external IP address, which can then be used to refer to the computer in your local network.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



In the above picture, there are three computers in the local network that have each been assigned their own internal IP address by the router. The ISP is connected to the router and gives the router an external IP address that allows it to communicate with the Internet. On the Internet everyone sees your external IP address, but any information coming from the router is "converted" from the external IP address to the internal IP address.

So if you want your OpenKM installation to be accessible from both LAN and Internet, the trick here is to configure the client computers to resolve your ***internal IP*** *(192.168.0.50) if they are inside the LAN or the* ***external IP*** *if they are on Internet. To resolve the Internet IP (67.166.214.148) your computer uses the public DNS. So, you need to configure a sort of DNS server inside the LAN or modify every client host file to resolve to the internal IP.*

Debian and Ubuntu

The first thing in to install the required Apache software. From Debian / Ubuntu you can install Apache with a single command:

```
$ sudo apt-get install apache2
```

Edit the file named ***/etc/apache2/apache2.conf*** *and configure a ServerName to prevent warnings in the Apache startup process:*

```
$ vim /etc/apache2/apache2.conf
```

```
ServerRoot "/etc/apache2"
ServerName "your-domain.com"
```

Edit apache2.conf is optional.

Enable the proxy module:

```
$ sudo a2enmod proxy_ajp
$ sudo a2enmod proxy_wstunnel
```

To get websocket working you also need Tomcat 7.0.61 or later. Note that ***proxy_wstunnel*** *module requires Apache 2.4.5 or later. Please refer to Apache module documentation.*

Create the Certificates

```
$ sudo mkdir /etc/apache2/ssl
$ sudo /usr/sbin/make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/ssl/apache.pem
$ sudo a2enmod ssl
```

If you need to convert .pem file to crt use the command:

```
$ openssl x509 -in cacert.pem -out cacert.crt
```

Also you could be interested on Configuring OpenSSL.

Ensure ports ***443*** *is listen in* ***/etc/apache2/ports.conf****:*

```
$ cat /etc/apache2/ports.conf | grep Listen
```

Create the configuration file ***/etc/apache2/sites-available/openkm.conf*** *with this content:*

The websocket configuration in OpenKM v8.1.10 and older was:

```
ProxyPass /openkm/frontend/webSocket wss://127.0.0.1:8080/openkm/frontend/webSocket
ProxyPassReverse /openkm/frontend/webSocket wss://127.0.0.1:8080/openkm/frontend/webSocket
```

```
$ vim /etc/apache2/sites-available/openkm.conf
```

```
<VirtualHost *:443>
  ServerName openkm.your-domain.com
  RedirectMatch ^/$ /openkm

  # WebSocket support - needs proxy-wstunnel
  ProxyPass /openkm/webSocket wss://127.0.0.1:8080/openkm/webSocket
  ProxyPassReverse /openkm/webSocket wss://127.0.0.1:8080/openkm/webSocket

  ProxyPass /openkm ajp://127.0.0.1:8009/openkm
  ProxyPassReverse /openkm https://openkm.your-domain.com/openkm
```

```
  ErrorLog /var/log/apache2/openkm.your-domain.com-error.log
  CustomLog /var/log/apache2/openkm.your-domain.com-access.log combined

  SSLEngine on
  SSLCertificateFile /etc/apache2/ssl/apache.pem
</VirtualHost>
```

Change openkm.your-domain.com with your server IP or your domain value.

The VirtualHost ServerName must be other than ServerName in the main Apache configuration. Enable this site configuration:

```
$ cd /etc/apache2/sites-available/

$ sudo a2ensite openkm.conf
```

Redirect all HTTP connections to HTTPS

This step is optional.

*Edit the configuration file **/etc/apache2/sites-available/openkm.conf** and add these content*

```
$ vim /etc/apache2/sites-available/openkm.conf
```

```
<VirtualHost *:80>
  ServerName openkm.your-domain.com
  Redirect permanent / https://openkm.your-domain.com/
</VirtualHost>
```

Check the configuration

Restart Apache:

```
$ sudo /etc/init.d/apache2 restart
```

Check the access your OpenKM installation from https://openkm.your-domain.com/.

Another advantage of using Apache is that you can log OpenKM access and generate web statistics.

To check the configuration files can execute the command:

```
$ /usr/sbin/apache2 -t
```

Using Let's Encrypt

*Let's Encrypt is a free certificate authority that provides X.509 certificates for Transport Layer Security (TLS) encryption at no charge. To be able to request and install the certificates you need the **certbot** utility. In case of using Apache you also the*

certbot-apache *package:*

```
$ apt install certbot certbot-apache
```

Once installed execute the utility with these parameters:

```
$ certbot --domain <your-domain-name> --email <your-sysadmin-email> --agree-tos --redirect --apache
```

This certificate will be valid for 90 days, so you need to renew it before expired. It's a good practice to create a cron task to update it automatically:

```
@monthly /usr/bin/certbot renew
```

Keep also on mind that both 80 and 443 ports should be accessible from the Internet to validate the domain and generate the certificate.

Red Hat and CentOS

Use the yum application manager to install Apache:

```
$ sudo yum install httpd
```

Enable it at boot:

```
$ sudo chkconfig httpd --level 2345 on
```

Create the Certificate

```
$ sudo yum install mod_ssl openssl crypto-utils
```

Generate private keys (for more information visit Apache HTTP Secure Server Configuration)

```
$ genkey your-domain.com
```

Usually you want to generate self-certificate and not sending it to Certify Authority. Is good practice to set a password on private key, but in this case each time you restart Apache service it'll be demanded.

Also you could be interested on Configuring OpenSSL.

During the process will be generated two files at:

```
ReplaceSSLCertificateFile /etc/pki/tls/certs/your-domain.com.cert
SSLCertificateKeyFile /etc/pki/tls/private/your-domain.com.key
```

Now create the file ***/etc/httpd/conf.d/openkm.conf*** *with this content:*

```
$ vim /etc/httpd/conf.d/openkm.conf
```

```
<VirtualHost *:443>
  ServerName openkm.your-domain.com
  ProxyPass / ajp://localhost:8009/openkm/
  ProxyPassReverse / http://your-domain.com/openkm/
  ErrorLog /var/log/your-domain.com-error.log
  CustomLog /var/log/your-domain.com-access.log combined

  SSLEngine on
  SSLCertificateFile /etc/pki/tls/certs/your-domain.com.cert
  SSLCertificateKeyFile /etc/pki/tls/private/your-domain.com.key
</VirtualHost>
```

Change openkm.your-domain.com with your server ip or your domain value.

Finally must modify ***SSLCertificateFile*** *and* ***SSLCertificateKeyFile*** *values in file* ***/etc/httpd/conf.d/ssl.conf****:*

```
$ vim /etc/httpd/conf.d/ssl.conf
```

```
SSLCertificateFile /etc/pki/tls/certs/your-domain.com.cert
SSLCertificateKeyFile /etc/pki/tls/private/your-domain.com.key
```

Check the configuration

Restart Apache:

```
$ sudo /etc/init.d/httpd restart
```

Check the access your OpenKM installation from https://openkm.your-domain.com/.

Another advantage of using Apache is that you can log OpenKM access and generate web statistics.

Error Permission denied: proxy: AJP: attempt to connect

If in the log appears an error like ***Permission denied: proxy: AJP: attempt to connect*** *to solve executing the next command:*

```
$ /usr/sbin/setsebool httpd_can_network_connect 1
```

If it works then set the configuration permanent to persist across reboots:

```
$ /usr/sbin/setsebool -P httpd_can_network_connect 1
```

Windows

Please, read [Configuring Apache HTTP Reverse-Proxy]

Troubleshooting

If you see an error like:



Invalid command 'RewriteEngine', perhaps misspelled or defined by a module not included in the server configuration

You need to enable this Apache module:

```
$ sudo a2enmod rewrite
$ sudo a2enmod proxy_http
$ sudo a2enmod headers
```

If you see errors like these:

```
[proxy_ajp:error] [pid 27579:tid 140057021224704] (70007)The timeout specified has expired: AH01030:
ajp_ilink_receive() can't receive header
[proxy_ajp:error] [pid 27579:tid 140057021224704] [client xxx.xxx.xxx.xxx:60526] AH00992: ajp_read_header:
ajp_ilink_receive failed
[proxy_ajp:error] [pid 27579:tid 140057021224704] (70007)The timeout specified has expired: [client
xxx.xxx.xxx.xxx:60526] AH00878: read response failed from 127.0.0.1:8009 (127.0.0.1)
```

Is possible that Tomcat is not accepting AJP connections. Please, uncomment these lines at $TOMCAT_HOME/conf/server.xml

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector protocol="AJP/1.3"
           address="127.0.0.1"
           port="8009"
           redirectPort="8443"
           secretRequired="false" />
```

*The **secretRequired** parameter is importar, and if it's not present you need to add.*

More info at Apache Tomcat 8 Configuration Reference.

Additional information

- *Apache Module mod_proxy*
- *Apache Module mod_proxy_ajp*
- *Apache Module mod_rewrite*
- *mod_rewrite, a beginner's guide*
- *Using Apache Virtual Hosts and ProxyPass Together*
- *Rewrite HTTP To HTTPS*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Redirect to SSL*
- *Configuración SSL de Tomcat*
- *How To Create a SSL Certificate on Apache for CentOS 6*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Nginx HTTP Reverse-Proxy

Exposing OpenKM directly from Tomcat can be dangerous if you need the application to be accessed from the Internet (for example https://issues.jboss.org/browse/JBAS-3861). As result it can be infected by PerlBot. Also this 8080 may be blocked by a firewall. For these reasons, is a good idea to expose your OpenKM installation through the standard web port 80. In the following steps we explain how to configure Apache to handle these requests and forward them to Tomcat application server using the AJP13 protocol.

From the Apache documentation: The AJP13 protocol is packet-oriented. A binary format was presumably chosen over the more readable plain text for reasons of performance. The web server communicates with the servlet container over TCP connections. To cut down on the expensive process of socket creation, the web server will attempt to maintain persistent TCP connections to the servlet container, and to reuse a connection for multiple request/response cycles.



This configuration is not supported for multitenant instances. In case of multitenant configuration, please use Configuring Apache HTTP Reverse-Proxy.

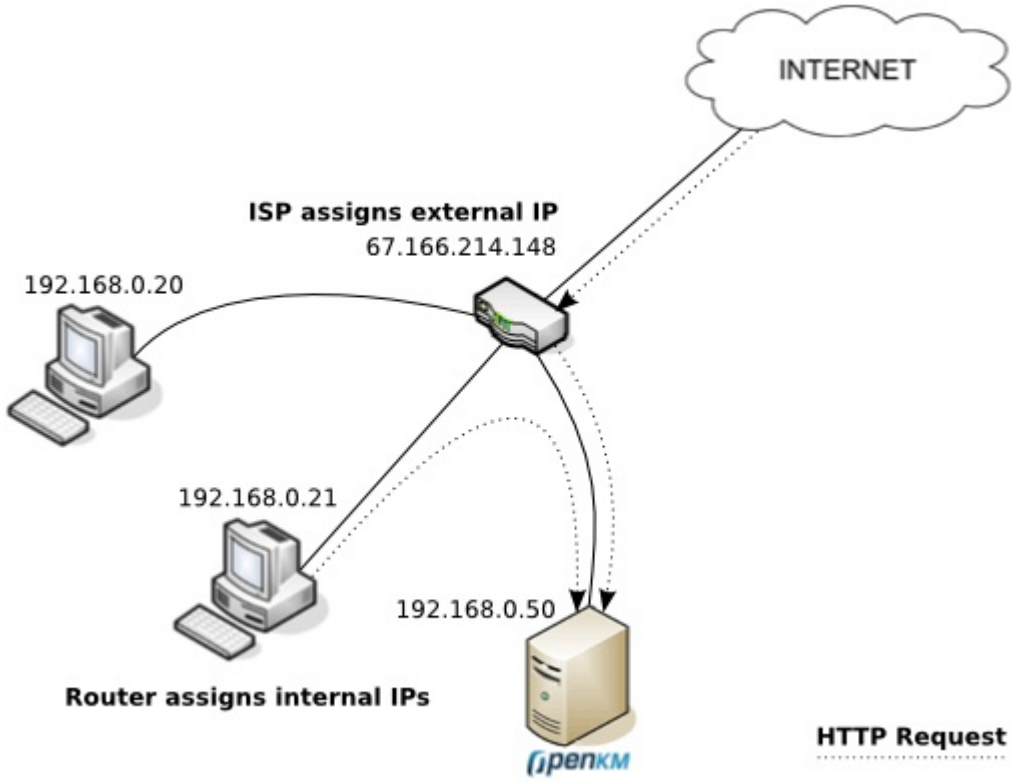
Internal IP vs external IP

Your OpenKM can be accessed from two different zones: Internet and LAN. This means that to access this server you need to use two IPs: external IP (Internet) and internal IP (LAN).

The internal IP address (also known as "local IP address") is the address that is assigned by your local network router that often begins with 192.168.x.x. These IP addresses can only be seen by other computers in your local network (LAN) and not by any computers connected in an external network such the Internet.

To reach the Internet or a computer in another network your computer is often assigned an external IP address, which can then be used to refer to the computer in your local network.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



In the above picture, there are three computers in the local network that have each been assigned their own internal IP address by the router. The ISP is connected to the router and gives the router an external IP address that allows it to communicate with the Internet. On the Internet everyone sees your external IP address, but any information coming from the router is "converted" from the external IP address to the internal IP address.

So if you want your OpenKM installation accessible from both LAN and Internet, the trick here is configure the client computers to resolve your ***internal IP*** *(192.168.0.50) if they are inside the LAN or the* ***external IP*** *if they are on the Internet. To resolve the Internet IP (67.166.214.148) your computer uses the public DNS. So, you need to configure a sort of DNS server inside the LAN or modify every client host file to resolve to the internal IP.*

Debian and Ubuntu

The first thing in to install the required Apache software. From Debian / Ubuntu you can install Apache with a single command:

```
$ sudo apt-get install nginx
```

Now create the configuration file ***/etc/nginx/sites-available/openkm*** *with this content:*

The websocket configuration in OpenKM v8.1.10 and older was:

```
location /openkm/frontend/webSocket {
    proxy_pass http://localhost:8080/openkm/frontend/webSocket;
    proxy_http_version 1.1;
```

```
    proxy_set_header Host $host;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
  }
```

```
$ vim /etc/nginx/sites-available/openkm
```

```
server {
  listen 80;
  server_name openkm.your-domain.com;
  # Avoid checking files size #
  client_max_body_size 0;

  rewrite ^/$ /openkm permanent;

  location /openkm/webSocket {
    proxy_pass http://localhost:8080/openkm/webSocket;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
  }

  location /openkm {
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Server $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://localhost:8080/openkm;
  }
}
```



Change openkm.your-domain.com with your server ip or your domain value.

Enable this site configuration:

```
$ ln -s /etc/nginx/sites-available/openkm /etc/nginx/sites-enabled/
```

Check the configuration

Restart Nginx:

```
$ sudo /etc/init.d/nginx restart
```

Check the access your OpenKM installation from http://openkm.your-domain.com/.

Another advantage of using Nginx is that you can log OpenKM access and generate web statistics.

Red Hat and CentOS

Use the yum application manager to install Nginx:

```
$ sudo yum install nginx
```

Enable it at boot:

```
$ sudo chkconfig nginx --level 2345 on
```

Depending on your CentOS version this Nginx version may be too old so it's better to install it from their repositories. To do this, create a file called ***/etc/yum.repos.d/nginx.repo*** *and include one on these configurations:*

CentOS

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
```

RedHat

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/rhel/$releasever/$basearch/
gpgcheck=0
enabled=1
```

Now create the configuration file /etc/nginx/conf.d/openkm.conf with this content:



The websocket configuration in OpenKM v8.1.10 and older was:

```
location /openkm/frontend/webSocket {
    proxy_pass http://localhost:8080/openkm/frontend/webSocket;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
```

```
$ vim /etc/nginx/conf.d/openkm.conf
```

```
server {
    listen 80;
    server_name openkm.your-domain.com;
    # Avoid checking files size #
    client_max_body_size 0;

    rewrite ^/$ /openkm/ permanent;

    location /openkm/webSocket {
        proxy_pass http://localhost:8080/openkm/webSocket;
        proxy_http_version 1.1;
```

```
        proxy_set_header Host $host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    location /openkm {
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://localhost:8080/openkm;
    }
}
```

Change openkm.your-domain.com with your server IP or your domain value.

Check the configuration

Restart Nginx:

```
$ sudo /etc/init.d/nginx restart
```

Check the access your OpenKM installation from http://openkm.your-domain.com/.

If you can't access server port from other machines, please take a look at your firewall configuration. For more info read How To Set Up a Basic Iptables Firewall on Centos 6.

Another advantage of using Nginx is that you can log OpenKM access and generate web statistics.

In case you see this kind of errors in the Nginx log:

```
connect() to 127.0.0.1:8080 failed (13: Permission denied)
```

Execute this command and try again:

```
$ setsebool -P httpd_can_network_connect on
```

These are several useful command related to SELinux:

Get all related HTTP configuration

```
$ getsebool -a | grep httpd
```

More information at SELinux Booleans.

Troubleshooting

This is a different configuration which work better with multitenant configuration:

```
location /openkm {
    # Forward hostname
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Host $host;

    # Forward original client IP
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    # Forward port
    proxy_set_header X-Forwarded-Port $server_port;

    proxy_pass http://localhost:8080/openkm;

    # Redirect http to https
    proxy_redirect http:// https://;

    # Optional: rewrite protocol
    sub_filter 'http://' 'https://';
    sub_filter_once off;
}
```

Configuring Apache HTTP to rewrite old OpenKM URL

This page contains obsolete information and will be removed in a near future.

In the case of having links in a document based on the old OpenKM version, these links by default will not work, because the format has changed.

*For example, a link in version 6.4 may have this aspect "**https://old.server.de/OpenKM/index.jsp?uuid=39448438-1f42-43fb-b33f-7b9ad5ac010c**" to match with version 7.1.x should be converted to this one "**https://new.server.de/openkm/kcenter/#/browser/uuid/39448438-1f42-43fb-b33f-7b9ad5ac010c**".*

*You can use the Apache **mod_rewrite** feature to convert the old URL to the new format and take the next configuration as a sample.*

```
<VirtualHost *:80>
  ServerName new.server.de
  RedirectMatch ^/$ /openkm

  # WebSocket support - needs proxy-wstunnel
  ProxyPass /openkm/webSocket ws://127.0.0.1:8080/openkm/webSocket
  ProxyPassReverse /openkm/webSocket ws://127.0.0.1:8080/openkm/webSocket

  ProxyPass /openkm ajp://127.0.0.1:8009/openkm keepalive=On
  ProxyPassReverse /openkm http://new.server.de/openkm

  ErrorLog /var/log/apache2/openkm-error.log
  CustomLog /var/log/apache2/openkm-access.log combined


  ### sample configuration below ###

  RewriteEngine on
  RewriteCond %{HTTP_HOST} ^old\.server\.de$ // if condition needed, not mandatory
  RewriteCond %{REQUEST_URI} ^\/OpenKM\/index\.jsp$ // if condition needed, not mandatory
  RewriteCond %{QUERY_STRING} ^uuid=([\s\S]*)$ // condition is mandatory, checks if URL comes with uuid and s
  RewriteRule ^.*$ https://new.server.de/openkm/kcenter/#/browser/uuid/%1 [NE,L]  // NE-no-escape needed for has

  ###
</VirtualHost>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Lucene Analyzer

Depending on the language used in the documents and properties, you have obtain better search results configuring a proper Lucene Analyzer.

By default, OpenKM use the ***org.apache.lucene.analysis.standard.StandardAnalyzer*** *which works fine with English and most languages, but you can get better search results configuring more specific analyzer for you language.*

Some analyzers:

- *org.apache.lucene.analysis.en.EnglishAnalyzer*
- *org.apache.lucene.analysis.es.SpanishAnalyzer*
- *org.apache.lucene.analysis.fr.FrenchAnalyzer*
- *org.apache.lucene.analysis.it.ItalianAnalyzer*
- *org.apache.lucene.analysis.de.GermanAnalyzer*
- *org.apache.lucene.analysis.el.GreekAnalyzer*
- *org.apache.lucene.analysis.hi.HindiAnalyzer*

Special analyzers:

- *com.openkm.search.lucene.analysis.AccentInsensitiveAnalyzer*

More information at Lucene documentation site and Guide to Lucene Analyzers.

If you want non accent sensitive analyzer you can try with AccentInsensitiveAnalyzer.

If you are working with oriental languages like Chinese or Japanese you have several analyzers to use. Read Lucene documentation. You can also try ik-analyzer.

If you want only a white space tokenized analyzer can try with this one WhitespaceAnalyzer.

If you have not configured the search analyzer before the first time you start OpenKM, then Lucene indexed will be created using this default analyzer.

If you want to change this configuration property after the OpenKM repository has been created you need to Rebuild Lucene Indexes.

Once the operation has been completed, the Lucene indexes will be using the new analyzer.

For more information take a look at Rebuild indexes.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configure an Analyzer

*Edit the **$TOMCAT_HOME/openkm.properties** file and add the line:*

```
spring.jpa.properties.hibernate.search.analyzer=org.apache.lucene.analysis.es.SpanishAnalyzer
```



The changes will take effect after restarting the application.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring CADViewer

CADViewer is a third-party product which can be integrated into OpenKM to provide previsualization of AutoCAD documents.

> *The required execubles should be already included in your OpenKM installation. Please, check if the **$TOMCAT_HOME/bin/cadviewer** folder exists and contains executable files like **autoxchange** and **dwglist**, among others.*

*Go to **Administration** > **Utilities** > **Database query**. At bottom right select option **JDBC**, paste into the box the SQL insert:*

```
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('8e642680-0dcd-4ded-a501-2aa3e53b1286
```

Some configuration properties are needed:

Field / Property	*Type*	*Description*
extension.cadviewer.license.key	***String***	*The CADViewer frontend key. The default license only work with local installations. Is the javascript CAD previewer.*
extension.cadviewer.autoxchange	***String***	*Path to **autoxchange** utility. Used in thumbnail generation, preview and PDF conversion.*
extension.cadviewer.linklist	***String***	*Path to **linklist** utility. Extract text and metadata from DWG documents.*
extension.cadviewer.dwglist	***String***	*Path to **dwglist** utility. Get XREFs from DWG documents. Also used in thumbnail, preview and PDF conversion.*
extension.cadviewer.sibling.xrefs	***Boolean***	*When this parameter is set to **true** the AutoCAD files's references have to be all in the same folder. If it is set to **false** the AutoCAD files's references can be stored in any place in the taxomony tree. In this case, keep on mind that you should avoid document name duplication. There is an automation called **CheckUniqueDocumentName** which can help you to maintain this restriction.*
		*Set to **true** to view the commands executed to conversion in the*

extension.cadviewer.debug	***Boolean***	*log file. Only for debug purpose.*
extension.cadviewer.remove.temp.files	***Boolean***	*Set to **false** to keep the temporary files creating during conversion. Only for debug purpose.*
extension.cadviewer.qr.url	***String***	*The URL of the QR code application: **http://openkm.your-domain.com/qrcad/***
extension.cadviewer.qr.user	***String***	*Name of the user of the QR code application.*
extension.cadviewer.qr.password	***Password***	*The password of the user of the QR code application.*
extension.cadviewer.qr.pos_x	***Integer***	*Percent X from the lower-left where to put the QR code in the AutoCAD file.*
extension.cadviewer.qr.pos_y	***Integer***	*Percent Y from the lower-left where to put the QR code in the AutoCAD file.*
extension.cadviewer.qr.width	***Integer***	*QR code width percent.*

These utilities can be downloaded from https://cadviewer.com/alldownloads/

CADViewer has several licenses:

- ***axlic.key**: This is the license for the **autoxchange** executable. This one should be placed in the same directory of the executable, typically located at **$TOMCAT_HOME/bin/cadviewer** folder.*
- ***dl_lic.key**: This is the license for the **dwglist** executable. This one should be placed in the same directory of the executable, typically located at **$TOMCAT_HOME/bin/cadviewer** folder.*
- ***dm_lic.key**: This is the license for the **dwgmerge** executable. This one should be placed in the same directory of the executable, typically located at **$TOMCAT_HOME/bin/cadviewer** folder.*
- ***ll_lic.key**: This is the license for the **linklist** executable. This one should be placed in the same directory of the executable, typically located at **$TOMCAT_HOME/bin/cadviewer** folder.*
- ***Javascript CAD previewer license**: This license key is configured using the **extension.cadviewer.license.key** configuration property.*

If you have problems with the conversion, try to execute from the command line. For example, this would make a PDF conversion:

```
$ ./autoxchange -i=test.dwg -f=pdf
```

In case of Linux, ensure you have configured the LD_LIBRARY_PATH properly. In case of Windows the program may fail because VCRUNTIME140.DLL is missing. If you see this error, download the Microsoft Visual C++ 2015 Redistributable from https://www.microsoft.com/en-us/download/details.aspx?id=52685 and install it.

There is also an script which may help because set the right libraries path:

```
$ ./cad2svg test.dwg
```

The resulting "test.svg" file can be viewed with any browser and you will see if the license was setup correctly.

*To enable this extension go to **Administration** > **Utilities** > **Database query**. At bottom right select option **JDBC**, paste into the box the SQL insert:*

```
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('38a9d032-30b9-470d-93ad-47f5e4f93f97',
```

Configuring Automations:

Name	*Event*	*At*	*Validations*	*Actions*
CAD Create	***Document create***	***post***	***Type:*** *NameMatches* ***Param 0:*** *.+\.[dD][wW][gG]*	***Type:*** *NewCadReferences* ***Order:*** *0*
CAD Update Pre	***Document update***	***pre***	***Type:*** *NameMatches* ***Param 0:*** *.+\.[dD][wW][gG]*	***Type:*** *DelCadReferences* ***Order:*** *0*
CAD Update Post	***Document update***	***post***	***Type:*** *NameMatches* ***Param 0:*** *.+\.[dD][wW][gG]*	***Type:*** *NewCadReferences* ***Order:*** *0*
CAD Delete	***Document delete***	***pre***	***Type:*** *NameMatches* ***Param 0:*** *.+\.[dD][wW][gG]*	***Type:*** *DelCadReferences* ***Order:*** *0*
CAD Folder Delete	***Folder delete***	***pre***	***Type:*** *PathContains* ***Param 0****: /okm:root*	***Type****: DelCadReferences* ***Order****: 0*

In case of using old UI you need to use an extension, but keep on mind this will be unsupported soon.

*To view external references for each AutoCAD document must set the value of the configuration key **extension.external.tab.document**. For do it, go to **Administration** > **Configuration** and set the appropiate value for this key:*

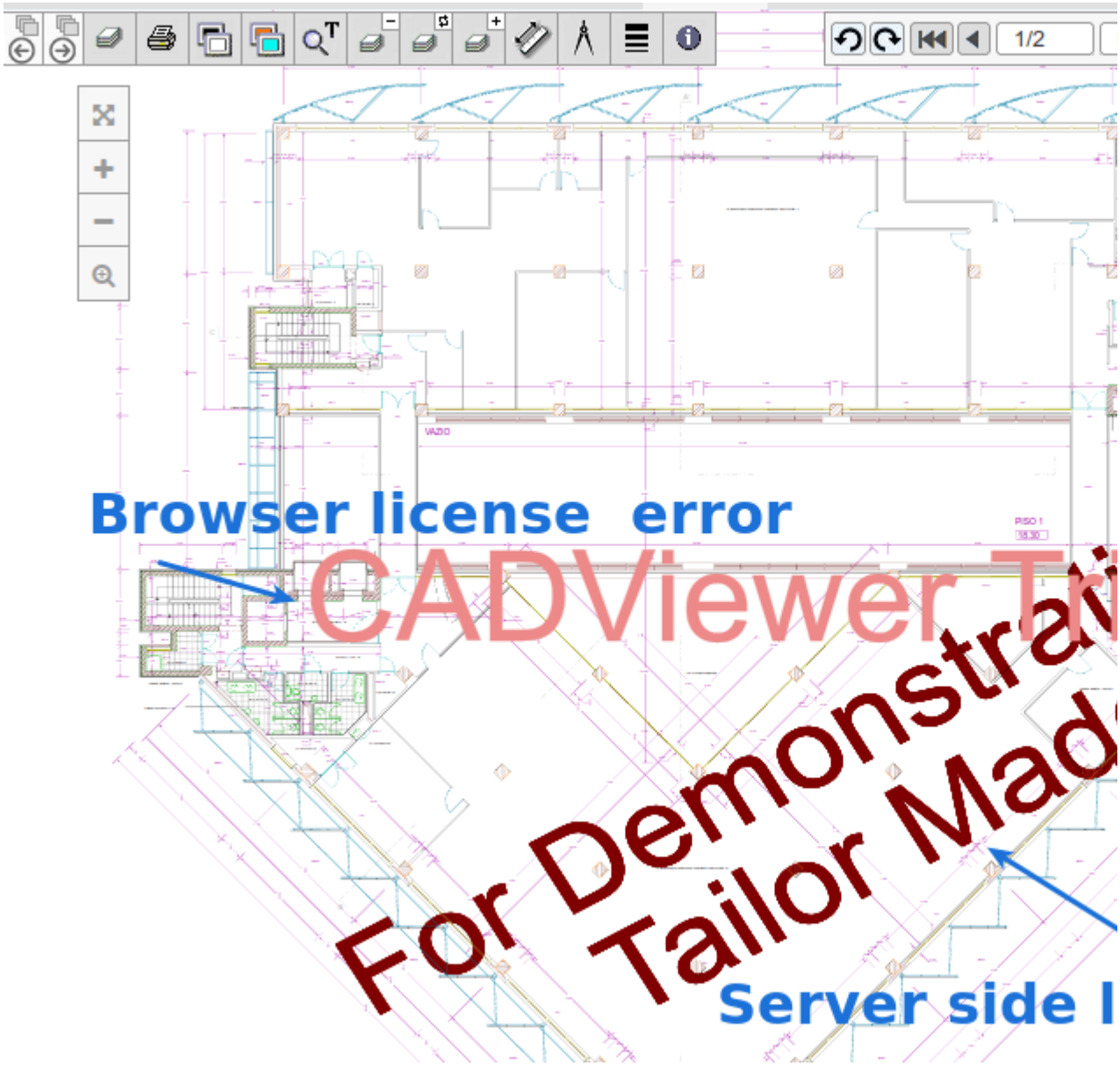
```
extension.external.tab.document = References;/openkm/frontend/XRefs
```

*You can use this SQL sentence to set the configuration property. Go to **Administration** > **Utilities** > **Database query** and execute:*

```
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('extension.external.tab
```

Troubleshooting

Mainly there are two license problems: one related to the browser client side license, and the other related to the server side conversion utilities.



Client side (browser)

These problems are related to the CADViewer Javascripr previewer. The license set in ***extension.cadviewer.license.key*** *configuration property is wrong.*

You can identity this issue by the "CADViewer Trial Version" watermark.

To solve this issue set the right key in the ***extension.cadviewer.license.key*** *configuration property and refresh the browser.*

Server side (utilities)

These problems are generated by CADViewer conversion utilities installed in the server. This license is configured in the ***axlic.key*** *server file.*

You can identify this issue by the "For Demonstration Purposes / Tailor Made Software" watermark.

To solve this issue copy the right ***axlic.key*** *license file to the OpenKM server. Once copied, please go to* ***Administration > Utilities > Clean Conversion Cache*** *and clean the* ***SVG*** *cache.*

You can verify the license this way:

```
$ ax2023 -verify
```

This parameter is only available in a recent AutoXchange version.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring OCR engine

OpenKM can work with several OCR engines, like Tesseract.

OpenKM can be integrated with any OCR engine that can be executed from command line.

Tesseract

Check the list of supported languages at https://tesseract-ocr.github.io/tessdoc/Data-Files-in-different-versions.html

More info about installation in different OS and Linux distros at https://tesseract-ocr.github.io/tessdoc/Installation.html

Tesseract is an Open Source OCR engine adopted by Goggle. It works really well. The OCR natively can read TIFF documents and has hight ratio of recognition with images 300 dpi of resolution and converted to lineart (1 bit color).

The supported image formats are:

- *TIFF*
- *PNG*
- *JPG*
- *GIF*

Installation

OS	Description

Ubuntu and Debian	`$ apt install tesseract-ocr`
	
	The default installed language is English so, depending on your locale you would like to install other language file.
	
	For example, to install Spanish language:
	`$ apt install tesseract-ocr-spa`
	
	You can check the installed languages this way:
	`$ tesseract --list-langs`
	
	More information at:
	- *https://github.com/tesseract-ocr/tesseract*

	• https://tesseract-ocr.github.io/tessdoc/Installation.html
Red Hat and CentOS	*In CentOS you can install Tesseract from the EPEL repository:* *$ yum install epel-release* *$ yum install tesseract* *The default installed language is English so, depending on your locale you would like to install other language file.* *For example, to install Spanish language:* *$ yum install tesseract-langpack-spa* *You can check the installed languages this way:* *$ tesseract --list-langs* *In RedHat we recommend contacting the RedHat support for more information.* *More information at:* • https://tesseract-ocr.github.io/tessdoc/Installation.html
Windows	*Download from https://tesseract-ocr.github.io/tessdoc/Installation.html and follow the installation wizard.*

Check installation

From command line execute (where image.jpg should be an existing image file):

```
$ tesseract image.jpg text
```

*The file **image.jpg** is an image file.*

*If all goes right a file named **text.txt** what contains the extracted text will be created.*

Configuration

*Go to **Administration** > Configuration parameters and edit the parameter named **system.ocr**:*

OS	
Linux	*/usr/bin/tesseract ${fileIn} ${fileOut}*

or

```
/usr/bin/tesseract ${fileIn} ${fileOut} -l esp
```

*The parameter **-l esp** indicate must be used the Spanish language support while executing OCR.*

To see all tesseract options execute:

```
$ tesseract
```

Windows

```
c:\Tesseract-ocr-3.0.2\tesseract.exe ${fileIn} ${fileOut}
```

or

```
c:\Tesseract-ocr-3.0.2\tesseract.exe ${fileIn} ${fileOut} -l spa
```

*The parameter **-l esp** indicate must be used the Spanish language support while executing OCR.*

To see all tesseract options execute:

```
c:\Tesseract-ocr-3.0.2> tesseract.exe
```

*Parameter **${fileIn}** will be replaced internally by image document to be processed.*

*Parameter **${fileOut}** will be replaced internally by temporal text file.*

*Can set other additional parameters like **-l spa** parameter in example.*

Each parameter is separated by a single "white space".

*Once configured, please go to **Administration > Utilities > Plugins**, choose **com.openkm.plugin.extractor.TextExtractor** from select and enable the **TesseractTextExtractor** plugin*

Additional information

- *http://code.google.com/p/tesseract-ocr/*
- *Tesseract - Summary & first experiences*.
- *Tesseract OCR Google Groups*
- *First Interactions with Tesseract OCR on Ubuntu Linux*
- *http://code.google.com/p/tesseract-ocr/wiki/ReadMe*
- *http://code.google.com/p/tesseract-ocr/wiki/FAQ*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Dictionary

You can also use an OpenOffice.org dictionary to enhance the OCR process. Can find these language specific dictionaries at OpenOffice.org Dictionary Repository.

Configuration

*Go to **Administration** > Configuration parameters and edit the parameter named **system.openoffice.dictionary**:*

OS	
Linux	/home/openkm/dictionary/en-GB.zip
Windows	c:\tomcat-8.5.69\dictionary\en-GB.zip

Dictionaries comes in .zip or oxt format. Can use both.

OCR rotate configuration

There's a configuration parameter that force document ocr on rotated document.

The feature is useful for example in upside down scanned pages, where standard OCR direction will fails.

When the feature is enabled the operation will be done on all OCR extractions, that can take undesired effects on OCR templates feature.

Configuration

*Go to **Administration** > Configuration parameters and edit the parameter named **system.ocr.rotate**:*

Property	Description
system.ocr.rotate	The parameter is a collection of degrees separated by character ";" 90;180;270;

Additional information:

- Linux OCR Software Comparison
- Tesseract OCR
- Ocropy

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Gocr*
- *Ocrad*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring OpenSSL

Create OpenSSL configuration file

> *This sample has been done with Ubuntu and Debian server, for Red Hat and CentOS the path of Apache files is not the same.*
>
> *On Ubuntu and Debian the Apache path **/etc/apache2** is **/etc/httpd** on Red Hat and CentOS.*

*Edit file **/etc/ssl/openssl.cnf***

```
$ vim /etc/ssl/openssl.cnf
```

```
RANDFILE        = $ENV::SSLDIR/.rnd

[ ca ]
default_ca      = CA_default

[ CA_default ]
dir             = /opt/ca
certs           = $dir/certs
new_certs_dir   = $dir/newcerts
crl_dir         = $dir/crl
database        = $dir/index.txt
private_key     = $dir/private/ca.key
certificate     = $dir/ca.crt
serial          = $dir/serial
crl             = $dir/crl.pem
RANDFILE        = $dir/private/.rand
default_days    = 365
default_crl_days = 30
default_md      = sha1
preserve        = no
policy          = policy_anything
name_opt        = ca_default
cert_opt        = ca_default

[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
default_bits            = 2048
default_md              = sha1
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
x509_extensions         = v3_ca
string_mask             = nombstr

[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
countryName_min         = 2
countryName_max         = 2
stateOrProvinceName     = State or Province Name (full name)
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
localityName            = Locality Name (eg, city)
0.organizationName      = Organization Name (eg, company)
organizationalUnitName  = Organizational Unit Name (eg, section)
commonName              = Common Name (eg, YOUR name)
commonName_max          = 64
emailAddress            = Email Address
emailAddress_max        = 64

[ usr_cert ]
basicConstraints        = CA:FALSE
# nsCaRevocationUrl     = https://url-to-exposed-clr-list/crl.pem

[ ssl_server ]
basicConstraints        = CA:FALSE
nsCertType              = server
keyUsage                = digitalSignature, keyEncipherment
extendedKeyUsage        = serverAuth, nsSGC, msSGC
nsComment               = "OpenSSL Certificate for SSL Web Server"

[ ssl_client ]
basicConstraints        = CA:FALSE
nsCertType              = client
keyUsage                = digitalSignature, keyEncipherment
extendedKeyUsage        = clientAuth
nsComment               = "OpenSSL Certificate for SSL Client"

[ v3_req ]
basicConstraints        = CA:FALSE
keyUsage                = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]
basicConstraints        = critical, CA:true, pathlen:0
nsCertType              = sslCA
keyUsage                = cRLSign, keyCertSign
extendedKeyUsage        = serverAuth, clientAuth
nsComment               = "OpenSSL CA Certificate"

[ crl_ext ]
basicConstraints        = CA:FALSE
keyUsage                = digitalSignature, keyEncipherment
nsComment               = "OpenSSL generated CRL"
```

Create Certification Authority (CA)

Create folder structure to hold certificates:

```
$ mkdir /opt/met-ca

$ mkdir /opt/met-ca/certs

$ mkdir /opt/met-ca/crl

$ mkdir /opt/met-ca/newcerts

$ mkdir /opt/met-ca/private

$ mkdir /opt/met-ca/requests

$ touch /opt/met-ca/index.txt

$ echo “01” > /opt/ca/serial

$ echo “01” > /opt/ca/crlnumber

$ chmod 700 /opt/ca
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Create public and private auto signed keys of our own Certification Authority (in the example is valid for 10 years). We will need to specify certificate data, password to crypt and the process will generate two files:

```
$ openssl req -config /etc/ssl/openssl.cnf -new -x509 -days 3650 -sha1 -newkey rsa:2048 -keyout /opt/ca/private/ca.key -out /opt/ca/ca.crt
```

Change key grants:

```
$ chmod 600 /opt/met-ca/private/ca.key
```

Create Apache Certification

Create a pair of keys. We will need to specify password and certification data. Will be created the file key (server.key) and certification request (server.pem).

```
$ openssl req -new -sha1 -newkey rsa:2048 -nodes -keyout server.key -out server.pem
```

CA should sign the certification request. Copy server.pem to certification request and sign.

```
$ openssl ca -config /etc/ssl/openssl.cnf -policy policy_anything -extensions ssl_server -out requests/server-signed.pem -infiles requests/server.pem
```

Apache will be waiting a PEM format, is mandatory convert to this format:

```
$ openssl x509 -in requests/server-signed.pem -out requests/server.crt
```

Apache configuration

Copy server.crt and server.key into /etc/apache2/ssl (if SSL folder not exists, create it). Change files grants (server.key should only be read by root user, server.crt should have read grant to everybody):

```
$ chmod 400 server.key
$ chmod 444 server.crt
```

If Apache is not listening SSL port, enable it configuring /etc/apache2/ports.conf:

```
Listen 80
Listen 443
```

Enable Apache SSL support:

```
$ apt-get install libapache-mod-ssl
```

Indicate Apache to load SSL modules:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
$ a2enmod ssl
```

Configure website:

Link default-ssl file to enable SSL:

```
$ ln -s /etc/apache2/sites-available/default-ssl /etc/apache2/sites-enabled/default-ssl
```

Edit /etc/apache2/sites-availables/default-ssl (enable SSL engine and set SSLCertificateFile and SSLCertificateKeyFile)

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key
```

Create user certificate:

To create certificates and sign with CA, first create the pair of keys. We will need to specify password and certification data.

Will be created a eky (user.key) and certification request (user.pem):

```
$ openssl req -new -sha1 -newkey rsa:2048 -nodes -keyout user.key -out user.pem
```

With CA is needed sign certificates:

```
$ cp /tmp/usuario.pem /opt/ca/requests
$ openssl ca -config /etc/ssl/openssl.cnf -policy policy_anything -extensions [b]ssl_client[/b] -out requests/user-signed.pem -infiles requests/user.pem
```

Convert certificate to PKCS12 to be recognized by browser

```
$ openssl pkcs12 -export -clcerts -in user-signed.pem -inkey usuario.key -out user.p12
```

The files user-signed.pem, user.pem and user.key can be deleted

```
$ wipe user-signed.pem user.key user.pem
```



*The **user.p12** should be installed to allow user browser.*

Certification revocation

If the server has been compromised the CA should be revocated. To revocate a certificate should see serial number at /opt/ca/index.txt.

```
$ openssl ca -config /etc/ssl/openssl.cnf -revoke newcerts/<serial-number>.pem
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creat a list of revocated certificates (CRL)

```
$ openssl ca -config /etc/ssl/openssl.cnf -gencrl -crlexts crl_ext -md sha1 -out crl.pem
```

To send to the browser list should be converted to DER format:

```
$ openssl crl -in crl.pem -out ca.crl -outform DER
```

Now can be sent to browser.

Add CRL to Apache

Copy CRL file to Apache SSL folder:

```
$ cp /tmp/crl.pem /etc/apache2/ssl/ca.crl
```

Modify /etc/apache2/sites-available/default-ssl file, add SSLCARevocationFile:

```
SSLCARevocationFile /etc/apache2/ssl/ca.crl
```

Restart Apache service.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring mail

The application sometimes sends emails to users, for example when the users have been subscribed to a document or folder.



By default tomcat mail is configured assuming your server has a mail service binding in your localhost.

These are the default values:

```
spring.mail.host=localhost
spring.mail.port=25
spring.mail.username=
spring.mail.password=
spring.mail.from=noreply@openkm.com
spring.mail.properties.mail.smtp.auth=false
spring.mail.properties.mail.smtp.starttls.enable=false
```

Change mail configuration

Get your mail account data (same than you will use to configure standard mail client, like Outlook or Thunderbird).

Edit the file $TOMCAT_HOME/openkm.properties:

```
vim /home/openkm/tomcat-8.5.69/openkm.properties
```

After editing the openkm.properties file should looks something like this:

```
spring.mail.host=localhost
spring.mail.port=25
spring.mail.username=
spring.mail.password=
spring.mail.from=noreply@openkm.com
spring.mail.properties.mail.smtp.auth=false
spring.mail.properties.mail.smtp.starttls.enable=false
```

Gmail example

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=<username>
spring.mail.password=<password>
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.ssl.trust=smtp.gmail.com
spring.mail.properties.mail.debug=true
```

Office 365

```
spring.mail.host=smtp.office365.com
spring.mail.username=<username>
spring.mail.password=<password>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.ssl.trust=smtp.office365.com
spring.mail.properties.mail.debug=true
```

Email server with old SSL configuration

```
spring.mail.host=mail.myserver.com
spring.mail.port=465
spring.mail.username=<username>
spring.mail.password=<password>
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.ssl.enable=true
spring.mail.properties.mail.debug=true
```

Additional information



Optionally, you can use these attributes :

- ***spring.mail.user="<username>"***



*Be careful with **spring.mail.test-connection** because when enabled, if the **mail server connections test fails openkm won't start**:*

- *spring.mail.test-connection=true*



*If you get problems during tomcat startup, can try using the attribute **spring.mail.smtp.quitwait="false"**.*



A complete list of all available parameters can be found at sun.mail.smtp package Javadoc and javax.mail package Javadoc.



*Note that depending on the SMTP server configuration only the address configured in **spring.mail.smtp.user** may be able to send messages, this mean, the user who appears in the "From" mail header.*

Testing configuration

Sometimes configuring mail can be complex and error-prone because there is a lot of configuration parameters and many differents mail servers. To ease this configuration we have developed a utility called OKMMailChecker which helps you to test the mail configuration whithout the need of restarting OpenKM, which takes time and make it unavailable for users during these tests.

Usage

*This utility can be downloaded from https://download.openkm.com and should be placed into OpenKM directory, where the **openkm.properties** is located. Indeed, this utility will read the mail configuration from this file. This utility has several*

parameters which will be used to send the test mail:

- ***-f, --from**: From used to send the test mail (required)*
- ***-t, --to**: To used to send the test mail (required)*
- ***-a, --atachment**: An optional file to be sent as attachment*

You can execute the utility this way:

```
$ java -jar OKMMailChecker.jar -f origin@openkm.com -t destination@openkm.com -a SampleFile.txt
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Tomcat memory utilization

It's also a good idea to configure the Tomcat memory utilization.

*Edit the file $TOMCAT_HOME/bin/setenv.sh and edit the parameter **JAVA_OPTS**, where you can increase the system memory managed by the JVM (Java Virtual Machine).*

OpenJDK 11

These are the default configuration file values. The Tomcat will reserve 4096MB (Xmx parameter), and the size of the heap from version 11 is dynamically assigned by JVM. For this configuration, your server needs at least >=5.5GB of RAM (It is a good practice to always get 1GB of free RAM for the OS).

```
JAVA_OPTS="$JAVA_OPTS -Xms256m -Xmx4096m -XX:+UseConcMarkSweepGC"
JAVA_OPTS="$JAVA_OPTS -Djava.awt.headless=true -Dlog4j.configuration=file://$CATALINA_HOME/conf/log4j.p
JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8 -Dmail.mime.ignoreunknownencod
CATALINA_PID=$CATALINA_HOME/catalina.pid
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CATALINA_HOME/lib/sigar
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Tomcat port

By default the tomcat binding ports are 8005, 8080 and 8009.



If you have other tomcat running on same server or other application like JBoss Application Server, probably these ports are already used. In this case you should change default ports.

Change tomcat ports



The port change must be done with the tomcat application stopped.

Edit $TOMCAT_HOME/confg/server.xml and look for all Tomcat port properties.

Service	Default port	Example of new port
Shut-down Port	8005	8105

	Change
	`<Server port="8005" shutdown="SHUTDOWN">`
	to
	`<Server port="8105" shutdown="SHUTDOWN">`	

Tomcat Connector Port	8080	8180

	Change
	`<Connector address="0.0.0.0" connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>`
	to
	`<Connector address="0.0.0.0" connectionTimeout="20000" port="8180" protocol="HTTP/1.1" redirectPort="8443"/>`	

AJP Connector Port	8009	8109
	Change	

```
<Connector address="127.0.0.1" port="8009" protocol="AJP/1.3" redirectPort="8443"/>
```

to

```
<Connector address="127.0.0.1" port="8109" protocol="AJP/1.3" redirectPort="8443"/>
```

*To shut-down the different instances execute the **shutdown.sh** script from each $TOMCAT_HOME/bin.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Tomcat session time-out

By default the session time-out is set to 30 minutes.

To change the Tomcat session time-out, edit the $TOMCAT_HOME/conf/web.xml and change the values into the session section:

```
<!-- You can set the default session timeout (in minutes) for all newly   -->
<!-- created sessions by modifying the value below.                       -->
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Tomcat timezone

Tomcat works in GMT timezone if you want to change it you should add something like at $TOMCAT_HOME/bin/setenv.sh (or $TOMCAT_HOME/bin/setenv.bat if you're on Windows):

```
CATALINA_OPTS="-Duser.timezone=Asia/Hong_Kong"
```

More information about timezone list.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring openkm.xml



By default OpenKM is not using openkm.xml from the file system, it is using the openkm.xml file what is into openkm.war

By default OpenKM comes with tree authentication providers:

- *Superuser in memory.*
- *Database.*
- *Active Directory (LDAP).*

The configuration added into ***openkm.xml*** *file is* ***loaded only*** *if all the OpenKM* ***internal security providers are disabled***.

You should disable internal authentication in the openkm.properties:

```
#Authentication
okm.authentication.database=false
okm.authentication.supervisor=false
okm.authentication.ldap=false
```

Default openkm.xml configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:security="http://www.springframework.org/schema/security"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/security
		http://www.springframework.org/schema/security/spring-security.xsd">

</beans:beans>
```

Enable external openkm.xml

To enable external openkm.xml file you must change the configuration parameter named ***okm.authentication.config*** *in the* ***openkm.properties*** *file. The values of the parameter* ***okm.authentication.config*** *might be the file system path to the openkm.xml file.*

Example:

```
okm.authentication.config=file:/home/openkm/tomcat-8.5.69/openkm.xml
```

Memory sample

Although a user is created in memory, the user must also be defined within the OpenKM database. Otherwise, the



authentication process will throw an error because the application will not be able to establish the user's tenant.

*The password must be encrypted with **bcrypt**.*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
             xmlns:security="http://www.springframework.org/schema/security"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.springframework.org/schema/beans
                                 http://www.springframework.org/schema/beans/spring-beans.xsd
                                 http://www.springframework.org/schema/security
                                 http://www.springframework.org/schema/security/spring-security.xsd">

  <!-- Security delegated to classes -->
  <security:authentication-manager alias="authenticationManager">
 <security:authentication-provider>
   <security:password-encoder hash="bcrypt"/>
     <security:user-service>
       <security:user name="okmAdmin" password="$2a$10$Z5R69bZZgX9z3tU9zjSYyuwE4iAA8Tk58xCnZ8t86Rb3
     </security:user-service>
   </security:authentication-provider>
  </security:authentication-manager>

</beans:beans>
```

Database authentication sample

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
             xmlns:security="http://www.springframework.org/schema/security"
             xmlns:task="http://www.springframework.org/schema/task"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.springframework.org/schema/beans
                                 http://www.springframework.org/schema/beans/spring-beans.xsd
                                 http://www.springframework.org/schema/security
                                 http://www.springframework.org/schema/security/spring-security.xsd
                                 http://www.springframework.org/schema/task
                                 http://www.springframework.org/schema/task/spring-task.xsd">

  <!-- Security delegated to classes -->
  <security:authentication-manager alias="authenticationManager">
    <security:authentication-provider>
      <security:password-encoder hash="bcrypt"/>
      <security:jdbc-user-service
        data-source-ref="dataSource"
        users-by-username-query="select usr_id, usr_password, 1 from OKM_USER where usr_id=? and usr_active='
        authorities-by-username-query="select ur_user, ur_role from OKM_USER_ROLE where ur_user=?"/>
    </security:authentication-provider>
  </security:authentication-manager>

</beans:beans>
```

Ldap Authentication sample

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
             xmlns:security="http://www.springframework.org/schema/security"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
                        http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://www.springframework.org/schema/security
                        http://www.springframework.org/schema/security/spring-security.xsd">

  <!-- Security delegated to classes -->
  <security:authentication-manager alias="authenticationManager">
    <security:authentication-provider ref="ldapAuthProvider" />
  </security:authentication-manager>

 <beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
  <beans:constructor-arg value="ldap://192.168.1.40"/>
  <beans:property name="userDn" value="CN=Administrator,CN=Users,DC=openkm,DC=local"/>
  <beans:property name="password" value="Darkman2017i"/>
  <beans:property name="baseEnvironmentProperties">
    <beans:map>
     <beans:entry>
       <beans:key>
       <beans:value>java.naming.referral</beans:value>
      </beans:key>
      <beans:value>follow</beans:value>
     </beans:entry>
    </beans:map>
   </beans:property>
 </beans:bean>

 <beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationPr
  <beans:constructor-arg>
   <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
    <beans:constructor-arg ref="contextSource"/>
    <beans:property name="userSearch" ref="userSearch"/>
   </beans:bean>
  </beans:constructor-arg>
  <beans:constructor-arg name="authoritiesPopulator" ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>

 <beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
  <beans:constructor-arg index="0" value="DC=openkm,DC=local" />
  <beans:constructor-arg index="1" value="(sAMAccountName={0})" />
  <beans:constructor-arg index="2" ref="contextSource" />
  <beans:property name="searchSubtree" value="true" />
 </beans:bean>

 <beans:bean id="defaultLdapAuthoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultLda
  <beans:constructor-arg ref="contextSource"/>
  <beans:constructor-arg value="DC=openkm,DC=local"/>
  <beans:property name="groupSearchFilter" value="member={0}"/>
  <beans:property name="groupRoleAttribute" value="cn"/>
  <beans:property name="searchSubtree" value="true" />
  <beans:property name="convertToUpperCase" value="false" />
  <beans:property name="rolePrefix" value="" />
 </beans:bean>

 <beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
  <beans:constructor-arg ref="userSearch"/>
  <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>

</beans:beans>
```

Multiple providers sample

Although a user is created in memory, the user must also be defined within the OpenKM database. Otherwise, the



authentication process will throw an error because the application will not be able to establish the user's tenant.

*The password must be encrypted with **bcrypt**.*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:security="http://www.springframework.org/schema/security"
        xmlns:task="http://www.springframework.org/schema/task"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/security
                http://www.springframework.org/schema/security/spring-security.xsd
                http://www.springframework.org/schema/task
                http://www.springframework.org/schema/task/spring-task.xsd">

  <!-- Security delegated to classes -->
  <security:authentication-manager alias="authenticationManager">
    <security:authentication-provider>
   <security:password-encoder hash="bcrypt"/>
      <security:user-service>
        <security:user name="okmAdmin" password="$2a$10$Z5R69bZZgX9z3tU9zjSYyuwE4iAA8Tk58xCnZ8t86Rb3
      </security:user-service>
    </security:authentication-provider>
    <security:authentication-provider>
      <security:password-encoder hash="bcrypt"/>
      <security:jdbc-user-service
        data-source-ref="dataSource"
        users-by-username-query="select usr_id, usr_password, 1 from OKM_USER where usr_id=? and usr_active='
        authorities-by-username-query="select ur_user, ur_role from OKM_USER_ROLE where ur_user=?"/>
    </security:authentication-provider>
  </security:authentication-manager>

</beans:beans>
```

Configuring jBPM

In OpenKM v7.1 the workflow engine is deployed as an independient application to ensure security and stability.

Please, take a look at JBPM Console for more info.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring ONLYOFFICE

This guide help to install and configure ONLYOFFICE with OpenKM.

> *ONLYOFFICE is WOPI compliant. Read the specification at* ***https://wopi.readthedocs.io/projects/wopirest/en/latest/index.html***

> *ONLYOFFICE documentation:* ***https://api.onlyoffice.com/editors/wopi/***

ONLYOFFICE installation

First find out the IP address of the computer, e.g. ***192.168.1.38*** *henceforth we will use this IP for this guide.*

Run the following commands to install the Docker container:

- *Download the script:*

```
$ wget http://download.onlyoffice.com/docs/docs-enterprise-install.sh
```

- *Run the script:*

```
$ bash docs-enterprise-install.sh -skiphc true -dp 8082
```

Configuration

Within the docker container edit ***/etc/onlyoffice/documentserver/default.json.*** *After that the service needs to be restarted:* ***supervisorctl restart all****.*

> *The wopi has to be enabled manually by editing this default.json and need to set "wopi" parameter to true.*

ONLYOFFICE access

Test it in your web browser: ***http://192.168.1.38:8082/***

*Discovery****:***

- *Open your web browser and run the following url:* ***http://192.168.1.38:8082/hosting/discovery***

*Capabilities****:***

- *Open your web browser and run the following url:* ***http://192.168.1.38:8082/hosting/capabilities***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

WOPI validator

This is an optional step.

This is for validating the REST endpoints of the WOPI host (OpenKM)

- *Install it:*

```
docker pull tylerbutler/wopi-validator
```

- *Run it:*

```
docker run -it --rm tylerbutler/wopi-validator -- -w http://localhost:8080/openkm/wopi/files/80ffd1eb-a6b8-4ec2-adc1-a58b23815eaf -t <openkm token> -l 36000000
```

***80ffd1eb-a6b8-4ec2-adc1-a58b23815eaf** is the uuid of a document in OpenKM.*

OpenKM configuration

Edit the following configuration parameters from OpenKM administration:

*ONLYOFFICE source IP is in a Docker network range, usually like 172.17.x.x, so this IP should be the one configured at **onlyoffice.ip** configuration parameter The authentication token used between ONLYOFFICE and OpenKM is restricted by IP, that means only the IP of ONLYOFFICE will be able to use it.*

*If unsure about ONLYOFFICE IP, look into the log file for a error message like "**Error validating token IP where {} != {}**", or enter within the docker container (**docker exec -it name_container bash**) and see the IP: **cat /etc/hosts***

```
# value samples
online.office.action.url = http://192.168.1.38:8080/openkm/wopi/files
online.office.url = http://192.168.1.38:8082/hosting/discovery
online.office.ip = 172.19.0.2
extension.extra.tab.workspace = Online Editor;/openkm/onlineOffice/onlyoffice
```

Parameters:

- ***online.office.action.url**: URL used by ONLYOFFICE Docker container to connect to OpenKM WOPI REST endpoint.*
- ***online.office.url**: URL used by OpenKM to connect to ONLYOFFICE Docker container.*
- ***online.office.ip**: Source IP used by ONLYOFFICE to authenticate to OpenKM.*
- ***extension.extra.tab.workspace:** to access from the top main menu in OpenKM user interface.*

*About how to enable the parameter named "**extension.extra.tab.workspace**" take a look at Extra tab workspace*

extension and Enable extensions sections in the documentation.

Samples URL:

- *http://localhost:8080/openkm/onlineOffice/onlyoffice?uuid=02090e71-8283-43d7-be2a-d6f141c3f8b0*
- *http://192.168.1.38:8080/openkm/onlineOffice/onlyoffice?uuid=02090e71-8283-43d7-be2a-d6f141c3f8b0*
- *http://192.168.1.38:8080/openkm/onlineOffice/onlyoffice?uuid=02090e71-8283-43d7-be2a-d6f141c3f8b0&lang=en-gb*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Collabora Online

This guide helps to install and configure CODE (Collabora Online Development Edition) with OpenKM.

*CODE is WOPI compliant. Read the specification at **https://wopi.readthedocs.io/projects/wopirest/en/latest/index.html.***

*Once you have installed Collabora Online, you must enable the **online editor extension** in the user profile. More information at Enable extensions .*

Code installation

*First, find out the IP address of the computer, e.g. **192.168.1.38** henceforth, we will use this IP for this guide.*

Run the following commands:

- *Get the CODE image from the Docker image*

```
$ docker pull collabora/code
```

- *Run the Docker container*

```
$ docker run --name collabora -p 9980:9980 -e "username=admin" -e "password=*********" -e "extra_params=--o:ssl.enable=false" collabora/code
```

- *Logs*

```
$ docker logs <docker container name>
```

Code access

For more information about docker image https://sdk.collaboraonline.com/docs/installation/CODE_Docker_image.html

*Test it, run this command or put the url in your web browser; you should get the **OK** string if everything is in order:*

```
$ curl -k http://192.168.1.38:9980
```

*Administration UI (**user** = admin, **password** =*********):*

*For 21.11.2.4.1 and before versions, open your web browser and run the following url: **http://192.168.1.38:9980/loleaflet/dist/admin/admin.html***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Open your web browser and run the following url:* ***http://192.168.1.38:9980/browser/dist/admin/admin.html***

Discovery:

- *Open your web browser and run the following url:* ***http://192.168.1.38:9980/hosting/discovery***

Capabilities:

- *Open your web browser and run the following url:* ***http://192.168.1.38:9980/hosting/capabilities***

Code configuration

- *Copy out the config file:*

```
$ docker cp <docker container name>:/etc/loolwsd/loolwsd.xml loolwsd.xml
```

- *Edit the config file with your favorite editor, then save and copy it back to the container:*

```
$ docker cp loolwsd.xml <docker container name>:/etc/loolwsd/loolwsd.xml
```

The configuration by default is enough.

WOPI validator

This is for validating the REST endpoints of the WOPI host (OpenKM)

- *Install it:*

```
docker pull tylerbutler/wopi-validator
```

- *Run it:*

```
docker run -it --rm tylerbutler/wopi-validator -- -w http://localhost:8080/openkm/wopi/files/80ffd1eb-a6b8-4ec2-adc1-a58b23815eaf -t <openkm token> -l 36000000
```

80ffd1eb-a6b8-4ec2-adc1-a58b23815eaf *is the uuid of a document in OpenKM.*

OpenKM configuration

Edit the following configuration parameters from OpenKM administration:

Collabora source IP is in a Docker network range, usually like 172.17.x.x, so this IP should be configured at ***collabora.ip*** *configuration parameter The authentication token used between collabora and OpenKM is*

restricted by IP, which means only the IP of Collabora will be able to use it.

If unsure about Collabora IP, look into the log file for an error message like " ***Error validating token IP where {} != {}****", or enter within the docker container (* ***docker exec -it name_container bash****) and see the IP:* ***cat /etc/hosts***

```
# value samples
online.office.action.url = http://192.168.1.38:8080/openkm/wopi/files
online.office.url = http://192.168.1.38:9980/hosting/discovery
online.office.ip = 172.17.0.2
```

Parameters

- ***online.office.action.url****: URL used by Collabora Docker container to connect to OpenKM WOPI REST endpoint.*
- ***online.office.url****: URL used by OpenKM to connect to Collabora Docker container.*
- ***online.office.ip****: Source IP used by Collabora to authenticate to OpenKM.*

Sample URLs

- *http://localhost:8080/openkm/onlineOffice/collabora?uuid=3101021e-d38c-47eb-8ad7-c8e51578b6d9*
- *http://192.168.1.38:8080/openkm/onlineOffice/collabora?uuid=3101021e-d38c-47eb-8ad7-c8e51578b6d9*
- *http://192.168.1.38:8080/openkm/onlineOffice/collabora?uuid=3101021e-d38c-47eb-8ad7-c8e51578b6d9&lang=en-gb*

Other sample configuration

In this scenario, Collabora also runs on Docker (172.17.0.2) and OpenKM on the host server (172.17.0.1).

- ***online.office.action.url****: http://172.17.0.1:8080/openkm/wopi/files*
- ***online.office.url****: http://172.17.0.2:9980/hosting/discovery*
- ***online.office.ip****: 172.17.0.2*

You should access OpenKM using http://172.17.0.1:8080/openkm/.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring safe trash purge

When an user delete a node (document, folder, mail or record) from OpenKM, it goes to user trash. Periodically users should empty their trash to free repository space.



When the user empties the trash, the documents are permanently lost.

That means if accidentally the trash has been cleaned, the nodes only can be restored from a previous backup. Usually restoring a full database and repository backup, which is a tedious task.



Administrator can clean all application trashes with Purge trashes utility.



You can configure a place where the documents will go after being purged from trash.

If you need to recover accidentally deleted node, then will not be necessary restore the entire application from backup.

*You can configure a directory in the server file system where an export of the trash to be placed, using the "**repository.purgatory.home**" configuration parameter. Each time before a node is definitely removed from trash, the application does an export (like Repository export feature) to the purgatory home folder, creating into sub folder structure "**YEAR/MONTH/DAY**" what can help the administrator to locate the data to be recovered. To recover the data use the Repository import feature with the **Metadata** option checked.*

Field / Property	*Type*	*Description*
repository.purgatory.home	***String***	*Set the safe trash purge destination folder.* `/mnt/backup/trash`



Is a good practice to backup the purgatory home folder, at least for some prudential time (1-2 months depending on your company policy).

Configuring search highlighter

The search highlighter feature shown a highlight text in the search view.

By default search highlighter is disabled. To enable you must add in the ***openkm.properties*** *the parameter* ***hibernate.search.highlighter***.

```
hibernate.search.highlighter=true
```

From OpenKM v7.1.23 this configuration property will be renamed to spring.jpa.properties.hibernate.search.highlighter

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Tomcat to work behind an Internet proxy



We encourage to consider do not configure Internet connection in Servers behing an Internet proxy.

It have some disadvantages like:

- *You will need to use your proxy data each time you will upgrade the application with OpenKM automatic updater tools.*
- *It's need to configure JVM parameters with your Internet proxy data for the application be able to connect to Internet.*

Is not much clear what are the advantatges for this kind of configuration on Production Servers, what are not the same scenario than end users. ***Take it as is, only a suggestion****.*

Linux



Configuration below is based on setting proxy configuration parameters in the JVM environment. There's another option what is configuring proxy at OS level, here are several option one of them is:

*Edit your "****/etc/bash.bashrc****" file as root and put these lines at the end of your "****/etc/bash.bashrc****" file:*

```
export http_proxy=http://username:password@proxyserver.net:port/
export ftp_proxy=http://username:password@proxyserver.net:port/
```

- ***Stop OpenKM****.*
- ***Edit the $TOMCAT_HOME/bin/setenv.sh*** *file and add the lines:*



The http://proxy.url.com value must be your own url host of the proxy.

*The "****user****" value must be the user for getting connection to the proxy. If your* ***proxy does not require authentication you can remove the entire line*** *where are set these parameters.*

*The "****password****" values must be the password of the user for getting connection to the proxy. If your* ***proxy does not require authentication you can remove the entire line*** *where are set these parameters.*

```
# HTTP Proxy configuration
JAVA_OPTS="$JAVA_OPTS -Dhttp.proxyHost=proxy-host-or-ip -Dhttp.proxyPort=proxy-port"
JAVA_OPTS="$JAVA_OPTS -Dhttp.proxyUser=user -Dhttp.proxyPassword=password"

# HTTPS Proxy configuration
JAVA_OPTS="$JAVA_OPTS -Dhttps.proxyHost=proxy-host-or-ip -Dhttps.proxyPort=proxy-port"
JAVA_OPTS="$JAVA_OPTS -Dhttps.proxyUser=user -Dhttps.proxyPassword=password"
```

At the end ***your setenv.sh should looking something like that*** *:*

```
JAVA_OPTS="$JAVA_OPTS -Xms1024m -Xmx4096m -XX:+UseConcMarkSweepGC -Djava.awt.headless=true -D
JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8 -Dmail.mime.ignoreunknownencoc
JAVA_OPTS="$JAVA_OPTS -Dhttp.proxyHost=192.168.100.128 -Dhttp.proxyPort=3128"
JAVA_OPTS="$JAVA_OPTS -Dhttps.proxyHost=192.168.100.128 -Dhttps.proxyPort=3128"
```

If you have issues resolving DNS you can also try with this configuration what is trying to use Google DNS:

```
JAVA_OPTS="$JAVA_OPTS -Dsun.net.spi.nameservice.nameservers=8.8.8.8"
JAVA_OPTS="$JAVA_OPTS -Dsun.net.spi.nameservice.provider.1=dns,sun"
```

- ***Start OpenKM***.
- *Go to* ***Administration*** *>* ***Tools*** *>* Scripting *and execute the script below (should be shown google html code).*

```
import java.net.URL;
import java.net.HttpURLConnection;
import java.io.BufferedReader;

URL url = new URL("http://www.google.com");
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
urlConn.setDoInput(true);
urlConn.setDoOutput(true);
urlConn.setUseCaches(false);
urlConn.setRequestMethod("GET");
BufferedReader input = new BufferedReader(new InputStreamReader(urlConn.getInputStream()));
String line;
StringBuilder sb = new StringBuilder();
while ((line = input.readLine()) != null) {
 sb.append(line);
}
input.close();
print(sb.toString().replaceAll("\n","<br/>"));
```

Windows

Configuration below is based on setting proxy configuration parameters in the JVM environment. There's another option what is configuring proxy at OS level, consider looking for this information at Microsoft technet website.

Step one

Check starting and stopping OpenKM from the command line.

- *First stop OpenKM service.*
- *Edit $T****OMCAT_HOME/bin/seten.bat*** *file and add the lines:*

The http://proxy.url.com value must be your own url host of the proxy.

*The "****user****" value must be the user for getting connection to the proxy. If your* ***proxy does not require authentication you can remove the entire line*** *where are set these parameters.*

*The "**password**" values must be the password of the user for getting connection to the proxy. If your **proxy does not require authentication you can remove the entire line** where are set these parameters.*

```
# HTTP Proxy configuration
set JAVA_OPTS=%JAVA_OPTS% -Dhttp.proxyHost=proxy-host-or-ip -Dhttp.proxyPort=proxy-port
set JAVA_OPTS=%JAVA_OPTS% -Dhttp.proxyUser=user -Dhttp.proxyPassword=password

# HTTPS Proxy configuration
set JAVA_OPTS=%JAVA_OPTS% -Dhttps.proxyHost=proxy-host-or-ip -Dhttps.proxyPort=proxy-port
set JAVA_OPTS=%JAVA_OPTS% -Dhttps.proxyUser=user -Dhttps.proxyPassword=password
```

*At the end **your setenv.bat should looking something like that**:*

```
set JAVA_OPTS=%JAVA_OPTS% -Xms1024m -Xmx4096m -XX:+UseConcMarkSweepGC -Djava.awt.headless=tr
set JAVA_OPTS=%JAVA_OPTS% -Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8 -Dmail.mime.ignoreunknown
set JAVA_OPTS=%JAVA_OPTS% -Dhttp.proxyHost=192.168.100.128 -Dhttp.proxyPort=3128
set JAVA_OPTS=%JAVA_OPTS% -Dhttps.proxyHost=192.168.100.128 -Dhttps.proxyPort=3128
```

If you have issues resolving DNS you can also try with this configuration what is trying to use Google DNS:

```
set JAVA_OPTS=%JAVA_OPTS% -Dsun.net.spi.nameservice.nameservers=8.8.8.8
set JAVA_OPTS=%JAVA_OPTS% -Dsun.net.spi.nameservice.provider.1=dns,sun
```

- ***Start OpenKM from the command line***.

```
c:\tomcat-8.5.69\bin\> catalina.bat run
```

- *Go to **Administration** > **Tools** > Scripting and execute the script below (should be shown google html code).*

```
import java.net.URL;
import java.net.HttpURLConnection;
import java.io.BufferedReader;

URL url = new URL("http://www.google.com");
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
urlConn.setDoInput(true);
urlConn.setDoOutput(true);
urlConn.setUseCaches(false);
urlConn.setRequestMethod("GET");
BufferedReader input = new BufferedReader(new InputStreamReader(urlConn.getInputStream()));
String line;
StringBuilder sb = new StringBuilder();
while ((line = input.readLine()) != null) {
 sb.append(line);
}
input.close();
print(sb.toString().replaceAll("\n","<br/>"));
```

Step two

If all has gone right in step one, now you can configure parameters into the OpenKM service.

- ***Stop OpenKM***.
- *Go to **$TOMCAT_HOME/bin** and execute the next command line:*



The command lines below does not registering the parameters for DNS resolving. If you need it, simply add them.

```
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttp.proxyHost=proxy-host-or-ip"
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttp.proxyPort=proxy-port"
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttp.proxyUser=user"
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttp.proxyPassword=password"
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttps.proxyHost=proxy-host-or-ip"
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttps.proxyPort=proxy-port"
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttps.proxyUser=user"
c:\tomcat-8.5.69\bin\> OpenKM //US//OpenKM ++JvmOptions="-Dhttps.proxyPassword=password"
```

- ***Start OpenKM from services***.
- *Go to **Administration** > **Tools** > Scripting and execute the script below (should be shown google html code).*

```
import java.net.URL;
import java.net.HttpURLConnection;
import java.io.BufferedReader;

URL url = new URL("http://www.google.com");
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
urlConn.setDoInput(true);
urlConn.setDoOutput(true);
urlConn.setUseCaches(false);
urlConn.setRequestMethod("GET");
BufferedReader input = new BufferedReader(new InputStreamReader(urlConn.getInputStream()));
String line;
StringBuilder sb = new StringBuilder();
while ((line = input.readLine()) != null) {
 sb.append(line);
}
input.close();
print(sb.toString().replaceAll("\n","<br/>"));
```

Configuring VidSigner

VidSigner is a third-party product which can be integrated into OpenKM to allow digital signature of a document.

Some configuration properties are needed:

Field / Property	*Type*	*Description*
vidsigner.user	***String***	*Name of the user given for the subscription to VidSigner.*
vidsigner.password	***Password***	*Password of the user.*
vidsigner.device.name	***String***	*Name of the device given by VidSigner where we want to send the document to be signed. Don't set if want to sign the document in any device associated with the subscription. In this case, you can find the document in the device, enabling search mode in configuration and finding by OpenKM user.*
vidsigner.notification.url	***String***	*The URL where the notification will be sent:* ***http://openkm.your-domain.com/openkm/rest/vidSigner***
vidsigner.preproduction.mode	***Boolean***	*Set to true to test vidsigner in pre production mode.*
vidsigner.sign.page	***String***	*The page where to show the signature image. If it is set to zero then the signature will be invisible.*
vidsigner.sign.posX	***String***	*The X position in millimeters where to place the signature image.*
vidsigner.sign.posY	***String***	*The Y position in millimeters where to place the signature image.*
vidsigner.sign.sizeX	***String***	*The X size in millimeters of the signature image.*
vidsigner.sign.sizeY	***String***	*The Y size in millimeters of the signature image.*
vidsigner.generate.log	***Boolean***	*Set to true to generate a detail log with all the information of the call to VidSigner service*

In addition, you have to create a user in OpenKM with the name and password given by VidSigner to your subscription.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Also, you have to create two new Automations on event ***Set metadata group*** *(at post) and associate to the first one the action* ***VidSigner Document To Sign*** *and to the second the action* ***VidSigner Document Signed***.

Finally, you have to register this group of metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
<property-group label="ViDSigner" name="okg:vidsigner">
  <checkbox label="Pending to sign" name="okp:vidsigner.pending" value="true" />
  <input label="Document GUI" name="okp:vidsigner.docgui" readonly="true" />
</property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Change repository home

*By default the repository is located at "**$TOMCAT_HOME/repository**" but you can change to other location.*

> *This action must be done with application stopped.*

Understanding the contents of repository folders

*Into "**$TOMCAT_HOME/repository**" there are three folder **tmp**, **index**, **cache** and **datastore**.*

Folder	Backup	Description
tmp	*No*	*This folder contains some temporary files to be shared beween several clusteres OpenKM instances.*
index	*No*	*The folder which contains the Lucene search engine indexes. The content of that folder can be rebuild.* *Although is not mandatory backup this folder, we recommend doing it.*
cache	*No*	*The folder which contains the application cached files. For example, for previewing documents, sometimes is needed to transform the original document to other format. The conversion consumes a lot of hardware resources, for this reason the application stores these converted files to be reused later. The content of that folder can be cleaned without major problems.* *Although is not mandatory backup this folder, we recommend doing it.*
datastore	*Yes*	*The folder which contains all binary data of the documents.* *Is **mandatory mandatory backup** this folder otherwise the binary data of the documents could be lost on a server hardware fail.*
extraction	*Yes*	*This folder contains the text extracted from those documents stored in OpenKM. A backup is not really necessary but may be a time saver.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ocr_templates	*Yes*	*This folder contains the binary data of the ocr templates.*

Move entire repository location

- *Stop the application*
- *Move the "**$TOMCAT_HOME/repository**" to other location.*
- *Edit the "**$TOMCAT_HOME/openkm.properties**" file and add the property "**repository.home**".*
- *Start the application*

Sample1

> *From OpenKM version 7.1.x and upper when the "**repository.home**" paremeter is set, the **location** in the file system **of the lucene index is not affected**. You should use the specific parameter named "**spring.jpa.properties.hibernate.search.default.indexBase**" to move the lucene index in a new location.*

Windows

```
repository.home=C:/okmrepo
```

Linux

```
repository.home=/mnt/okmrepo
```

Move repository folders individually

Table of properties:

Folder	*Property*	*Default*
index	*spring.jpa.properties.hibernate.search.backend.directory.root*	*${catalina.home}/repository/index*
tmp	*repository.temp.home*	*${repository.home}/tmp*
cache	*repository.cache.home*	*${repository.home}/${tenant.id}/ca*
datastore	*repository.datastore.home*	*${repository.home}/${tenant.id}/da*
extraction	*repository.extraction.home*	*${repository.home}/${tenant.id}/ex*

ocr template	*repository.ocr.template.home*	*${repository.home}/${tenant.id}/oc*

*These parameters can be used in combination with the parameter "**repository.home**" and "**tenant.id**".*

- *Stop the application*
- *Move the "**$TOMCAT_HOME/repository/{folder}**" to other location.*
- *Edit the "**$TOMCAT_HOME/openkm.properties**" file and add the some property.*
- *Start the application*

Sample

Windows

```
spring.jpa.properties.hibernate.search.backend.directory.root=C:/lucene/index
repository.datastore.home=D:/okm/${repository.home}/datastore
repository.cache.home=D:/okm/${repository.home}/cacherepository.temp.home=D:/okm/${repository.home}/tmp
repository.ocr.template.home=D:/okm/${repository.home}/cacherepository.temp.home=D:/okm/${repository.home}/c
```

Linux

```
spring.jpa.properties.hibernate.search.backend.directory.root=/mnt/lucene/index
repository.datastore.home=/mnt/okm/${repository.home}/datastore
repository.cache.home=/mnt/okm/${repository.home}/cacherepository.temp.home=/mnt/okm/${repository.home}/tmp
repository.ocr.template.home=/mnt/okm/${repository.home}/cacherepository.temp.home=/mnt/okm/${repository.hon
```

Troubleshooting

Windows mapped network disk is not working

We encourage do not use networking (NAS or Samba) for storing repository. Because network issues will affect the repository we consider as a dangerous configuration.

*Seems there're restrictions using network unit (mapped with "**net use**" command line) and application running from services.*

To solve it is necessary to create a link between UNC and the file system.

```
MKLINK /D C:\okm-repository \\192.168.1.36\eDocuments\okm-repository
```

In most cases will be necessary to use a standard user for running OpenKM service either the standard configured. Take a look at OpenKM windows service configuration to change the default "Local system user" to your own.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Change default role names

*By default application comes with two connection roles named "**ROLE_USER**" and "**ROLE_ADMIN**". Any OpenKM user must have assigned one of these roles to login.*

Change the default roles

Edit the openkm.properties file

```
default.user.role=NEW_ROLE_USER_NAME
default.admin.role=NEW_ROLE_ADMIN_NAME
```

Update default roles to new names in the database



*In the case of a **LdapPrincipalAdapter** configuration **only execute the scripts below** the "**-- Update security table**" description.*

- *Go to Administration > Tools > Database query*

```
-- Update security table
UPDATE OKM_NODE_ROLE_PERMISSION SET NRP_ROLE='NEW_ROLE_USER_NAME' WHERE NRP_ROLE
UPDATE OKM_NODE_ROLE_PERMISSION SET NRP_ROLE='NEW_ROLE_ADMIN_NAME' WHERE NRP_ROLE

-- Insert new roles
INSERT INTO OKM_ROLE (ROL_ID, ROL_ACTIVE) VALUES ('NEW_ROLE_USER_NAME', 'T');
INSERT INTO OKM_ROLE (ROL_ID, ROL_ACTIVE) VALUES ('NEW_ROLE_ADMIN_NAME', 'T');

-- Update user roles
UPDATE OKM_USER_ROLE SET UR_ROLE='NEW_ROLE_USER_NAME' WHERE UR_ROLE='ROLE_USER';
UPDATE OKM_USER_ROLE SET UR_ROLE='NEW_ROLE_ADMIN_NAME' WHERE UR_ROLE='ROLE_ADMIN';

-- Remove older roles
DELETE FROM OKM_ROLE WHERE ROL_ID IN ('ROLE_USER', 'ROLE_ADMIN');
```

- *Restart the OpenKM service*
- *Go to Administration > Tools > Lucene indexes > Rebuild indexes*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Enable extensions



After the extensions have been registered in OpenKM will not be visible for the users until you enable them from the Profile General tab .

By default OpenKM comes with the extensions disabled.

Types of extensions:

- *Do not require additional software configuration.*
- *Need additional software configuration.*
- *Sample extensions for development.*

Available extensions are registered in the OKM_EXTENSION table. To enable any of them, it's only necessary to insert it in this table.

Extensions that do not require additional software configuration:

- *Go to **Administration** > **Utilities** > **Database query**. At bottom right select option **JDBC**, **paste into the box the SQL inserts** of the extensions you want to enable. Finally click at bottom right the button **execute**.*



*The **Stapling** extension is deprecated and has been superseded by **Relation**.*

*The **Online Editor** extension is available from OpenKM version 7.1.33*

```
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('54743940-0761-11e0-81e0-0800200c9a66
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('93faba30-0c34-11e0-81e0-0800200c9a66'
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('4ef6ad10-1a5b-11e0-ac64-0800200c9a66'
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('88ca0d10-39e2-11e0-9207-0800200c9a66
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('522e7720-5c54-11e0-80e3-0800200c9a66
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('fa7f4556-3249-4268-88e0-0dd78a79872a',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('808e7a42-2e73-470c-ba23-e4c9d5c3a0f4',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('c20c69a8-7d6b-4539-9e6b-6f2c11e84168',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('58392af6-2131-413b-b188-1851aa7b651c'
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('cc7cf5df-a466-42c6-9453-340aab53dc27',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('5d3b6871-20db-442e-806f-30d507dc935b'
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('38a9d032-30b9-470d-93ad-47f5e4f93f97',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('ba5bc2e4-b1de-4cdd-8e63-459fde78867a'
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('80db33ae-a702-41ff-8818-e72daf0ba068',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('494098d5-0bdc-4e10-a30b-508184a5ee1c
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('7b08995d-fa1a-4ebf-98f3-a3ef322aa4d5', '
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('ec166b87-f4c4-4420-a200-a021f62fd64b',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('ba32b8c9-d033-4442-ad59-ace275190663
```

Extensions that require additional software configuration:

- *Go to **Administration** > **Utilities** > **Database query**. At bottom right select option **JDBC**, **paste into the box the SQL inserts** of the extensions you want to enable. Finally click at bottom right the button **execute**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('8e642680-0dcd-4ded-a501-2aa3e53b1286
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('288cdc63-78e4-434b-94e0-ccbdace1da21
```

Deprecated extensions:

```
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('df5eb783-fb06-4b4b-bc89-4fdaa244e888',
INSERT INTO OKM_EXTENSION (EXT_UUID, EXT_NAME) VALUES ('87f250d5-526d-4d8a-96ee-1e5be7910bd8'
```

Sample extensions for development:

There're some sample extension for development testing purpose.

- *Go to* ***Administration*** *>* ***Utilities*** *>* ***Database query****. At bottom right select option* ***JDBC****,* ***paste into the box the SQL inserts*** *of the extensions you want to enable. Finally click at bottom right the button* ***execute****.*

```
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME) VALUES('9f84b330-d096-11df-bd3b-0800200c9a66','
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME) VALUES('44f94470-d097-11df-bd3b-0800200c9a66','
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME) VALUES('d95e01a0-d097-11df-bd3b-0800200c9a66','
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME) VALUES('d9dab640-d098-11df-bd3b-0800200c9a66','
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

LDAP configuration

The LDAP integration is divided in two sections, login process and retrieving users and roles from LDAP server. It's usually better starting with LDAP configuration parameters which are used by OpenKM to show users and roles into administration and once this section is configured go to Spring Security configuration (openkm.xml changes).

Usually you only want to retrieve a subset of users and roles present in your LDAP, to be shown in the user interface or be able to login into OpenKM. If this is your case, please take a look at LDAP best practices for filtering users and roles .

OpenLDAP

- *OpenLDAP example with URL base at login*

Active Directory

- *Active Directory basic configuration*
- *Active Directory advanced configuration*
- *Active Directory example with referral enabled*
- *Active Directory example with login based on filtering users by roles (suggested).*
- *Active Directory example with remember-me services*
- *Active Directory mixed configuration*

Uncategorised examples shared by users

- *http://pastebin.com/iQa4LhyY (more information OpenKM public forum)*

Tools

- *Apache Directory Studio.*
- *Active Directory Explorer Utility.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OpenLDAP example with URL base at login

Since version 8.1.12, the OpenLDAP integration has changed:

- *The configuration parameters and login procedure remains the same.*
- *User data and roles are now managed in OpenKM administration and stored in database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

LDAP Structure

```
dc=com
  dc=some
    ou=organization
      cn=ROLE_ADMIN
        memberUid=okmAdmin
        memberUid=user1
        memberUid=user2
      cn=ROLE_USER
        memberUid=user3
        memberUid=user4
        ...
    ou=organization
      uid=user1
        mail=user@mail.com
        cn=User Name 1
      uid=user2
        mail=user2@mail.com
        cn=User Name 3
      uid=user3
        mail=user3@mail.com
        cn=User Name 3
      uid=user4
        mail=user4@mail.com
        cn=User Name 4
```

Valid roles:

- *cn=ROLE_X,**ou=roles,dc=some,dc=com***
- *cn=ROLE_Y,ou=dept marketing,**ou=roles,dc=some,dc=com***
- *cn=ROLE_Z,ou=dept sales,**ou=roles,dc=some,dc=com***

Invalid roles:

- *cn=ROLE_INVALID,ou=dept,dc=some,dc=com (any distinguished name not included in ou=roles,dc=some,dc=com)*

Valid users:

- *uid=USER_X,**ou=organization,dc=some,dc=com***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *uid=USER_Y,ou=dept id,**ou=organization,dc=some,dc=com***
- *uid=USER_Z,ou=dept administrator,**ou=organization,dc=some,dc=com***

Invalid users:

- *uid=USER_INVALID,ou=house,dc=some,dc=com (any distinguished name not included in ou=organization,dc=some,dc=com)*

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from Administration > Tools > Cache stats. In the second step configure login, this configuration works always in real time.

Step 1 - configuration parameters

*We suggest l**ogin in OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps will be necessary to restart OpenKM service and you do not want to lose administration access.*

*The first action should be to modify **principal.adapter** parameter value and **restart OpenKM service**. Because session ID is kept in the browser you should not lose the login after the service restarted and can continue working in the administration. After this change, the users and roles list will be empty from the administration. Until success configuring the next parameters these lists will be empty.*

*Go to **Administration > Configuration parameters**:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated since version 8.1.12* *com.openkm.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*false*
principal.ldap.server	***String***	*ldap://192.168.xxx.xxx:389*
principal.ldap.security.principal	***String***	*cn=Manager,dc=some,dc=com*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.user.attribute	***String***	*uid*
principal.ldap.user.search.base	***List***	*ou=users,dc=some,dc=com*
principal.ldap.user.search.filter	***String***	*(objectClass=inetOrgPerson)*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*ou=users,dc=some,dc=com*
principal.ldap.username.search.filter	***String***	*(uid={0})*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*dc=some,dc=com*
principal.ldap.mail.search.filter	***String***	*(uid={0})*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*ou=roles,dc=some,dc=com*
principal.ldap.role.search.filter	***String***	*(objectClass=posixGroup)*
principal.ldap.roles.by.user.attribute	***String***	*cn*
principal.ldap.roles.by.user.search.base	***String***	*ou=roles,dc=some,dc=com*
principal.ldap.roles.by.user.search.filter	***String***	*(memberUid={0})*
principal.ldap.users.by.role.attribute	***String***	*memberUid*
principal.ldap.users.by.role.search.base	***String***	*ou=roles,dc=some,dc=com*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.users.by.role.search.filter	***String***	*(&(objectClass=posixGroup)(cn={0}))*

Step 2 - configure login

- ***Important*** *login (**ldap://192.168.0.13:389/dc=some,dc=com**) sets default filter base queries at **dc=some,dc=com with is concatenated by default in all filter queries** .*
- *Roles (groups) filter base is **ou=roles** (real distinguised name is ou=roles,dc=some,dc=com). Any valid roles should have it as parent. ou=roles"/> really points to ou=roles,dc=some,dc=com"/>*
- *Users filter base is **ou=organization** (real distinguised name is ou=organization,dc=some,dc=com). Any valid user should have it as parent. to*
- *User filter is uid={0}*
- *Finally take in consideration the value 1 at memberUid={1}"/>*

*Apply changes in the **openkm.properties** file.*

*The parameter "**okm.authentication.database**" is used to disable database login.*

*The parameter "**okm.authentication.ldap**" is used to enable ldap login.*

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
ldap.server=ldap://192.168.0.13
ldap.manager.distinguished.name=CN=Administrator,CN=Users,DC=openkm,DC=local
ldap.manager.password=*secret*
ldap.base=DC=openkm,DC=local
ldap.role.attribute=cn
ldap.user.search.filter=uuid={0}
ldap.role.search.filter=memberUid={1}
```



After the update of the openkm.properties file must restart the openkm service to take effect.

Another option to configure login

In some cases might be interested to set configuration in file openkm.xml please refer to [*Configuring openkm.xml*]

documentation section for more information.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:security="http://www.springframework.org/schema/security"
        xmlns:task="http://www.springframework.org/schema/task"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/security
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
                        http://www.springframework.org/schema/security/spring-security.xsd
                        http://www.springframework.org/schema/task
                        http://www.springframework.org/schema/task/spring-task.xsd">

 <security:authentication-manager alias="authenticationManager">
  <security:authentication-provider ref="ldapAuthProvider" />
 </security:authentication-manager>

 <beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
  <beans:constructor-arg value="ldap://192.168.0.13:389/dc=some,dc=com"/>
 <beans:property name="userDn" value="cn=Manager,dc=some,dc=com"/>
  <beans:property name="password" value="*******"/>
 </beans:bean>

<beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationPr
 <beans:constructor-arg>
  <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
   <beans:constructor-arg ref="contextSource"/>
   <beans:property name="userSearch" ref="userSearch"></beans:property>
  </beans:bean>
 </beans:constructor-arg>
 <beans:constructor-arg name="authoritiesPopulator" ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>

  <beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
   <beans:constructor-arg index="0" value="ou=organization" />
   <beans:constructor-arg index="1" value="uid={0}" />
   <beans:constructor-arg index="2" ref="contextSource" />
   <beans:property name="searchSubtree" value="true" />
 </beans:bean>

 <beans:bean id="defaultLdapAuthoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultLd
 <beans:constructor-arg ref="contextSource"/>
 <beans:constructor-arg value="ou=roles"/>
 <beans:property name="groupSearchFilter" value="memberUid={1}"/>
 <beans:property name="groupRoleAttribute" value="cn"/>
 <beans:property name="searchSubtree" value="true" />
 <beans:property name="convertToUpperCase" value="true" />
 <beans:property name="rolePrefix" value="" />
</beans:bean>

 <!--Needed for remember-me services -->
<beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
 <beans:constructor-arg ref="userSearch"/>
 <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator"/>
</beans:bean>

</beans:beans>
```

Active Directory example with referral enabled



Since version 8.1.12, the OpenLDAP integration has changed:

- *The configuration parameters and login procedure remains the same.*
- *User data and roles are now managed in OpenKM administration and stored in database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

This example covers the scenario where more than one Active Directory domains work together and it's needed the ***follow*** *parameter.*

LDAP Structure

```
dc=com
  dc=company
    ou=OPENKM
      cn=ROLE_ADMIN
        member=okmAdmin
        member=user1
        member=user2
      cn=ROLE_USER
        member=user3
        member=user4
      cn=ROLE_XXXX
      cn=ROLE_YYYY
        ...
    ou=organization1
      sAMAccountName=okmAdmin
        memberOf=CN=ROLE_ADMIN,OU=OPENKM,DC=company,DC=com
        mail=okmAdmin@mail.com
        cn=OpenKM Administrator
      sAMAccountName=user1
        memberOf=CN=ROLE_ADMIN,OU=OPENKM,DC=company,DC=com
        mail=user1@mail.com
        cn=User Name 1
      sAMAccountName=user2
        memberOf=CN=ROLE_ADMIN,OU=OPENKM,DC=company,DC=com
        mail=user2@mail.com
        cn=User Name 3
    ou=organization2
      sAMAccountName=user3
        memberOf=CN=ROLE_USER,OU=OPENKM,DC=company,DC=com
        mail=user3@mail.com
        cn=User Name 3
      sAMAccountName=user4
        memberOf=CN=ROLE_USER,OU=OPENKM,DC=company,DC=com
        mail=user4@mail.com
        cn=User Name 4
```

Valid groups:

- *cn=ROLE_ADMIN,ou=OPENKM,dc=company,dc=com*

- *cn=ROLE_USER,ou=OPENKM,dc=company,dc=com*
- *cn=ROLE_XXXX,ou=OPENKM,dc=company,dc=com*
- *cn=ROLE_YYYY,ou=OPENKM,dc=company,dc=com*

Valid users:

- *cn=user1,ou=organization1,dc=company,dc=com*
- *cn=user2,ou=organization1,dc=company,dc=com*
- *cn=user3,ou=organization2,dc=company,dc=com*
- *cn=user4,ou=organization2,dc=company,dc=com*

Any distinguished name includes by default ***dc=company,dc=com***

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from Administration > Tools > Cache stats. In the second step configure login, this configuration works always in real time.

Step 1 - configuration parameters

*We suggest l***ogin in OpenKM with the admin URL** *(for example http://localhost:8080/openkm/admin/index) because in the next steps will be necessary to restart OpenKM service and you do not want to lose administration access.*

The first action should be to modify ***principal.adapter*** *parameter value and* ***restart OpenKM service****. Because session ID is kept in the browser you should not lose the login after the service restarted and can continue working in the administration. After this change, the users and roles list will be empty from the administration. Until success configuring the next parameters these lists will be empty.*

Go to ***Administration > Configuration parameters****:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated since version 8.1.12* *com.openkm.plugin.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.xxx.xxx:389*
principal.ldap.security.principal	***String***	*CN=Administrator,OU=OPENKM,DC=company,DC=com*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	*follow*
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*DC=company,DC=com*
principal.ldap.user.search.filter	***String***	*(objectclass=user)*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*DC=company,DC=com*
principal.ldap.username.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*DC=company,DC=com*
principal.ldap.mail.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*DC=company,DC=com*
principal.ldap.role.search.filter	***String***	*(objectclass=group)*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*
principal.ldap.roles.by.user.search.base	***String***	*DC=company,DC=com*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.users.by.role.attribute	***String***	*member*
principal.ldap.users.by.role.search.base	***String***	*DC=company,DC=com*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectClass=group)(CN={0}))*

Step 2 - configure login

- *Parameter **follow** indicates several domains servers working together (balanced).*
- *Users defined in any Active Directory node will be able to login, because it has defined DC=company,DC=com as base filter.*
- *Any user authenticated in Active Directory can login because has not any filtering clausule.*
- *Groups read by OpenKM can be defined in any Active Directory node, because it has defined DC=company,DC=com as base filter.*

*Apply changes in the **openkm.properties** file.*

*The parameter "**okm.authentication.database**" is used to disable database login.*

*The parameter "**okm.authentication.ldap**" is used to enable ldap login.*

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
ldap.server=ldap://192.168.0.13
ldap.manager.distinguished.name=CN=Administrator,CN=Users,DC=openkm,DC=local
ldap.manager.password=*secret*
ldap.base=DC=openkm,DC=local
ldap.role.attribute=cn
ldap.user.search.filter=sAMAccountName={0}
ldap.role.search.filter=member={0}
```



After the update of the openkm.properties file must restart the openkm service to take effect.

Another option to configure login

In some cases might be interested to set configuration in file openkm.xml please refer to [*Configuring openkm.xml*]

documentation section for more information.

```
<security:authentication-manager alias="authenticationManager">
   <security:authentication-provider ref="ldapAuthProvider" />
```

```
</security:authentication-manager>

<beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
  <beans:constructor-arg value="ldap://192.168.xxx.xxx:389"/>
  <beans:property name="userDn" value="CN=Administrator,OU=OPENKM,DC=company,DC=com"/>
  <beans:property name="password" value="*****"/>
  <beans:property name="baseEnvironmentProperties">
    <beans:map>
      <beans:entry>
        <beans:key>
          <beans:value>java.naming.referral</beans:value>
        </beans:key>
        <beans:value>follow</beans:value>
      </beans:entry>
    </beans:map>
  </beans:property>
</beans:bean>

<beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationPr
  <beans:constructor-arg>
    <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
      <beans:constructor-arg ref="contextSource"/>
      <beans:property name="userSearch" ref="userSearch"/>
    </beans:bean>
  </beans:constructor-arg>
  <beans:constructor-arg name="authoritiesPopulator" ref="defaultLdapAuthoritiesPopulator"/>
</beans:bean>

<beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
  <beans:constructor-arg index="0" value="DC=company,DC=com" />
  <beans:constructor-arg index="1" value="sAMAccountName={0}" />
  <beans:constructor-arg index="2" ref="contextSource" />
  <beans:property name="searchSubtree" value="true" />
</beans:bean>

<beans:bean id="defaultLdapAuthoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultLd
<beans:constructor-arg ref="contextSource"/>
<beans:constructor-arg value="DC=openkm,DC=local"/>
<beans:property name="groupSearchFilter" value="member={0}"/>
<beans:property name="groupRoleAttribute" value="cn"/>
<beans:property name="searchSubtree" value="true" />
<beans:property name="convertToUpperCase" value="false" />
<beans:property name="rolePrefix" value="" />
</beans:bean>

<!--Needed for remember-me services -->
<beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
 <beans:constructor-arg ref="userSearch"/>
 <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator"/>
</beans:bean>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Active Directory example with login based on filtering users by roles

Since version 8.1.12, the OpenLDAP integration has changed:

- *The configuration parameters and login procedure remains the same.*
- *User data and roles are now managed in OpenKM administration and stored in database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

In this example, we'll see an Active Directory connection that allows to connect any Active Directory authenticated user member of ROLE_USER or ROLE_ADMIN.

LDAP structure

```
dc=com
  dc=company
    ou=OPENKM
      cn=ROLE_ADMIN
        member=okmAdmin
        member=user1
      cn=ROLE_USER
        member=user3
      cn=ROLE_XXXX
        member=user2
      cn=ROLE_YYYY
        member=user4
        ...
    ou=organization1
      sAMAccountName=okmAdmin
        memberOf=CN=ROLE_ADMIN,OU=OPENKM,DC=company,DC=com
        userPrincipalName=okmAdmin@mail.com
        cn=OpenKM Administrator
      sAMAccountName=user1
        memberOf=CN=ROLE_ADMIN,OU=OPENKM,DC=company,DC=com
        userPrincipalName=user1@mail.com
        cn=User Name 1
      sAMAccountName=user2
        memberOf=CN=ROLE_XXXX,OU=OPENKM,DC=company,DC=com
        userPrincipalName=user2@mail.com
        cn=User Name 3
    ou=organization2
      sAMAccountName=user3
        memberOf=CN=ROLE_USER,OU=OPENKM,DC=company,DC=com
        userPrincipalName=user3@mail.com
        cn=User Name 3
      sAMAccountName=user4
        memberOf=CN=ROLE_YYYY,OU=OPENKM,DC=company,DC=com
        userPrincipalName=user4@mail.com
        cn=User Name 4
```

Valid groups:

- *cn=ROLE_ADMIN,**ou=OPENKM,dc=company,dc=com***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *cn=ROLE_USER,**ou=OPENKM,dc=company,dc=com***
- *cn=ROLE_XXXX,**ou=OPENKM,dc=company,dc=com***
- *cn=ROLE_YYYY,**ou=OPENKM,dc=company,dc=com***

Valid users:

- *cn=user1,**ou=organization1,dc=company,dc=com***
- *cn=user2,**ou=organization1,dc=company,dc=com***

Invalid users:

- *cn=user3,**ou=organization2,dc=company,dc=com***
- *cn=user4,**ou=organization2,dc=company,dc=com***

*Any distinguished name includes by default dc=company,dc=com except users which are not member of **ROLE_ADMIN** or **ROLE_USER***

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from Administration > Tools > Cache stats . In the second step configure login, this configuration works always in real time.

Step 1 - configuration parameters

*We suggest l**ogin in OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps will be necessary to restart OpenKM service and you do not want to lose administration access.*

*The first action should be to modify **principal.adapter** parameter value and **restart OpenKM service**. Because session ID is kept in the browser you should not lose the login after the service restarted and can continue working in the administration. After this change, the users and roles list will be empty from the administration. Until success configuring the next parameters these lists will be empty.*

- *User members of ROLE_ADMIN or ROLE_USER can be created in any active directory node, because it has defined DC=company,DC=com as base filter, **principal.ldap.user.search.base=DC=company,DC=com** and user search filter as **principal.ldap.user.search.filter=(&(objectclass=user)(|(memberOf=CN=ROLE_ADMIN,OU=OpenKM,DC=company,DC=com)**.*
- *Groups can be created in any active directory node, because is has DC=company,DC=com as base filter, **principal.ldap.role.search.base=DC=company,DC=com**.*
- *All Active Directory groups will be listed, because it has not applied any filter restriction **principal.ldap.role.search.filter=(objectclass=group).***
- *Mail attribute used in this case is **userPrincipalName** (the most usual ldap attribute to get mail is **mail**)*

principal.ldap.mail.attribute=userPrincipalName.

Go to ***Administration*** *>* ***Configuration parameters****:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated since version 8.1.12* *com.openkm.plugin.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.xxx.xxx:389*
principal.ldap.security.principal	***String***	*CN=Administrator,OU=OPENKM,DC=company,DC=com*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	*follow*
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*DC=company,DC=com*
principal.ldap.user.search.filter	***String***	*(&(objectclass=user)(|* *(memberOf=CN=ROLE_ADMIN,OU=OpenKM,DC=company,* *(memberOf=CN=ROLE_USER,OU=OpenKM,DC=company,D*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*DC=company,DC=com*
principal.ldap.username.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.mail.attribute	***String***	*userPrincipalName*
principal.ldap.mail.search.base	***String***	*DC=company,DC=com*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.mail.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*DC=company,DC=com*
principal.ldap.role.search.filter	***String***	*(objectclass=group)*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*
principal.ldap.roles.by.user.search.base	***String***	*DC=company,DC=com*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.users.by.role.attribute	***String***	*member*
principal.ldap.users.by.role.search.base	***String***	*OU=OpenKM,DC=company,DC=com*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectClass=group)(cn={0}))*

Step 2 - configure login

- *Users created in any Active Directory node members of ROLE_ADMIN or ROLE_USER will be able to login, because it has defined DC=company,DC=com as base filter, and* ***as user account filter.***
- *Groups read by OpenKM can be created in any Active Directory node, because it has defined DC=company,DC=com as base filter.*

Apply changes in the ***openkm.properties*** *file.*

*The parameter "****okm.authentication.database****" is used to disable database login.*

*The parameter "****okm.authentication.ldap****" is used to enable ldap login.*

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
ldap.server=ldap://192.168.0.13
ldap.manager.distinguished.name=CN=Administrator,CN=Users,DC=openkm,DC=local
ldap.manager.password=*secret*
ldap.base=DC=openkm,DC=local
ldap.role.attribute=cn
ldap.user.search.filter=(&(sAMAccountName={0})(|(memberOf=CN=ROLE_ADMIN,OU=OpenKM,DC=company,DC
ldap.role.search.filter=member={0}
```



After the update of the openkm.properties file must restart the openkm service to take effect.

Another option to configure login

In some cases might be interested to set configuration in file openkm.xml please refer to [*Configuring openkm.xml*]

documentation section for more information.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:security="http://www.springframework.org/schema/security"
        xmlns:task="http://www.springframework.org/schema/task"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/security
                http://www.springframework.org/schema/security/spring-security.xsd
                http://www.springframework.org/schema/task
                http://www.springframework.org/schema/task/spring-task.xsd">

<security:authentication-manager alias="authenticationManager">
    <security:authentication-provider ref="ldapAuthProvider" />
 </security:authentication-manager>

<beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
 <beans:constructor-arg value="ldap://192.168.xxx.xxx"/>
 <beans:property name="userDn" value="CN=connect,OU=OpenKM,DC=company,DC=com"/>
 <beans:property name="password" value="****"/>
 <beans:property name="baseEnvironmentProperties">
   <beans:map>
     <beans:entry>
       <beans:key>
         <beans:value>java.naming.referral</beans:value>
       </beans:key>
       <beans:value>follow</beans:value>
     </beans:entry>
   </beans:map>
 </beans:property>
</beans:bean>

<beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationPro
 <beans:constructor-arg>
   <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
     <beans:constructor-arg ref="contextSource"/>
     <beans:property name="userSearch" ref="userSearch"/>
   </beans:bean>
 </beans:constructor-arg>
 <beans:constructor-arg name="authoritiesPopulator" ref="defaultLdapAuthoritiesPopulator"/>
</beans:bean>

<beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
 <beans:constructor-arg index="0" value="DC=company,DC=com" />
 <beans:constructor-arg index="1" value="(&amp;(sAMAccountName={0})(|(memberOf=CN=ROLE_ADMIN,OU=Op
```

```
  <beans:constructor-arg index="2" ref="contextSource" />
  <beans:property name="searchSubtree" value="true" />
 </beans:bean>

 <beans:bean id="defaultLdapAuthoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultLda
  <beans:constructor-arg ref="contextSource"/>
     <beans:constructor-arg value="DC=company,DC=com"/>
     <beans:property name="groupSearchFilter" value="member={0}"/>
     <beans:property name="groupRoleAttribute" value="cn"/>
     <beans:property name="searchSubtree" value="true" />
     <beans:property name="convertToUpperCase" value="false" />
     <beans:property name="rolePrefix" value="" />
 </beans:bean>

  <!--Needed for remember-me services -->
 <beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
  <beans:constructor-arg ref="userSearch"/>
  <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>

</beans:beans>
```

Active Directory basic configuration

Since version 8.1.12, the OpenLDAP integration has changed:

- *The configuration parameters and login procedure remains the same.*
- *User data and roles are now managed in OpenKM administration and stored in database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

This is the suggested configuration to be used when roles and users are both defined in the same node, otherwise refer to other example.

LDAP structure

```
dc=com
  dc=company
    cn=users
      cn=ROLE_ADMIN
        member=okmAdmin
        member=user1
      cn=ROLE_USER
        member=user3
      cn=ROLE_XXXX
        member=user2
      cn=ROLE_YYYY
        member=user4
        ...
      sAMAccountName=okmAdmin
        memberOf=cn=ROLE_ADMIN,cn=users,dc=company,dc=com
        userPrincipalName=okmAdmin@mail.com
        cn=OpenKM Administrator
      sAMAccountName=user1
        memberOf=cn=ROLE_ADMIN,cn=users,dc=company,dc=com
        userPrincipalName=user1@mail.com
        cn=User Name 1
      sAMAccountName=user2
        memberOf=cn=ROLE_USER,cn=users,dc=company,dc=com
        userPrincipalName=user2@mail.com
        cn=User Name 3
```

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from Administration > Tools > Cache stats . In the second step configure login, this configuration works always in real time.

Step 1 - configuration parameters

*We suggest l**ogin in OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps will be necessary to restart OpenKM service and you do not want to lose administration*

access.

*The first action should be to modify **principal.adapter** parameter value and **restart OpenKM service**. Because session ID is kept in the browser you should not lose the login after the service restarted and can continue working in the administration. After this change, the users and roles list will be empty from the administration. Until success configuring the next parameters these lists will be empty.*

- *User members of ROLE_ADMIN or ROLE_USER are created into users node (these should be the distinguished names **CN=ROLE_ADMIN,cn=users,DC=company,DC=com** and **CN=ROLE_USER,cn=users,DC=company,DC=com)**.*
- *Groups can be created on any Active Directory node, because DC=company,DC=com is set as base filter, **principal.ldap.role.search.base=DC=company,DC=com**.*
- *All Active Directory groups will be listed, because it has not applied any filter restriction **principal.ldap.role.search.filter=(objectclass=group).***

Go to Administration > Configuration parameters:

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated since version 8.1.12* *com.openkm.plugin.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.0.6:389*
principal.ldap.security.principal	***String***	*CN=Administrator,cn=users,dc=company,dc=local*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*cn=users,dc=company,dc=local*
principal.ldap.user.search.filter	***String***	*(objectclass=person)*
principal.ldap.username.attribute	***String***	*cn*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.username.search.base	***String***	*cn=users,dc=company,dc=local*
principal.ldap.username.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*cn=users,dc=company,dc=local*
principal.ldap.mail.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*cn=users,dc=company,dc=local*
principal.ldap.role.search.filter	***String***	*(objectclass=group)*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*
principal.ldap.roles.by.user.search.base	***String***	*DC=company,DC=com*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.users.by.role.attribute	***String***	*member*
principal.ldap.users.by.role.search.base	***String***	*cn={0},cn=users,dc=company,dc=local*
principal.ldap.users.by.role.search.filter	***String***	*(objectclass=group)*

Step 2 - configure login

Apply changes in the ***openkm.properties*** *file.*

*The parameter "****okm.authentication.database****" is used to disable database login.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*The parameter "**okm.authentication.ldap**" is used to enable ldap login.*

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
ldap.server=ldap://192.168.0.13
ldap.manager.distinguished.name=CN=Administrator,CN=Users,DC=openkm,DC=local
ldap.manager.password=*secret*
ldap.base=DC=openkm,DC=local
ldap.role.attribute=cn
ldap.user.search.filter=sAMAccountName={0}
ldap.role.search.filter=member={0}
```



After the update of the openkm.properties file must restart the openkm service to take effect.

Another option to configure login

In some cases might be interested to set configuration in file openkm.xml please refer to *Configuring openkm.xml*

documentation section for more information.

```
<security:ldap-server id="ldapServer"
  url="ldap://192.168.0.6:389/DC=ldap,dc=company,dc=local"
  manager-dn="CN=Administrator,cn=users,dc=company,dc=local"
  manager-password="password"/>

<security:authentication-manager alias="authenticationManager">
  <security:ldap-authentication-provider
    server-ref="ldapServer"
    user-search-base="cn=Users"
    user-search-filter="(sAMAccountName={0})"
    group-search-base="cn=Users"
    group-search-filter="(member={0})"
    group-role-attribute="cn"
    role-prefix="none">
  </security:ldap-authentication-provider>
</security:authentication-manager>

<beans:bean id="userDetailService" class="com.openkm.spring.UserTenantDetailsService"/>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Active Directory advanced configuration

Since version 8.1.12, the OpenLDAP integration has changed:

- *The configuration parameters and login procedure remains the same.*
- *User data and roles are now managed in OpenKM administration and stored in database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

LDAP structure

```
dc=com
  dc=company
    cn=users
      cn=ROLE_ADMIN
        member=okmAdmin
        member=user1
      cn=ROLE_USER
        member=user3
      cn=ROLE_XXXX
        member=user2
      cn=ROLE_YYYY
        member=user4
        ...
      sAMAccountName=okmAdmin
        memberOf=cn=ROLE_ADMIN,cn=users,dc=company,dc=com
        userPrincipalName=okmAdmin@mail.com
        cn=OpenKM Administrator
      sAMAccountName=user1
        memberOf=cn=ROLE_ADMIN,cn=users,dc=company,dc=com
        userPrincipalName=user1@mail.com
        cn=User Name 1
      sAMAccountName=user2
        memberOf=cn=ROLE_USER,cn=users,dc=company,dc=com
        userPrincipalName=user2@mail.com
        cn=User Name 3
```

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from Administration > Tools > Cache stats . In the second step configure login, this configuration works always in real time.

Step 1 - configuration parameters

*We suggest l**ogin in OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps will be necessary to restart OpenKM service and you do not want to lose administration access.*

*The first action should be to modify **principal.adapter** parameter value and **restart OpenKM service**. Because session ID is kept in the browser you should not lose the login after the service restarted and can continue*

working in the administration. After this change, the users and roles list will be empty from the administration. Until success configuring the next parameters these lists will be empty.

- *User members of ROLE_ADMIN or ROLE_USER are created into users node (these should be the distinguished names **CN=ROLE_ADMIN,cn=users,DC=company,DC=com** and **CN=ROLE_USER,cn=users,DC=company,DC=com)**.*
- *Groups can be created on any Active Directory node, because DC=company,DC=com is set as base filter, **principal.ldap.role.search.base=DC=company,DC=com**.*
- *All Active Directory groups will be listed, because has not applied any filter restriction **principal.ldap.role.search.filter=(objectclass=group).***

*Go to **Administration > Configuration parameters**:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated since version 8.1.12* *com.openkm.plugin.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.0.6:389*
principal.ldap.security.principal	***String***	*CN=Administrator,cn=users,dc=company,dc=local*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*dc=company,dc=local*
principal.ldap.user.search.filter	***String***	*(objectclass=person)*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*dc=company,dc=local*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.username.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*dc=company,dc=local*
principal.ldap.mail.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*dc=company,dc=local*
principal.ldap.role.search.filter	***String***	*(objectclass=group)*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*
principal.ldap.roles.by.user.search.base	***String***	*DC=company,DC=com*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.users.by.role.attribute	***String***	*member*
principal.ldap.users.by.role.search.base	***String***	*dc=company,dc=local*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectClass=group)(cn={0}))*

Step 2 - configure login

Apply changes in the ***openkm.properties*** *file.*

*The parameter "****okm.authentication.database****" is used to disable database login.*

*The parameter "****okm.authentication.ldap****" is used to enable ldap login.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
ldap.server=ldap://192.168.0.13
ldap.manager.distinguished.name=CN=Administrator,CN=Users,DC=openkm,DC=local
ldap.manager.password=*secret*
ldap.base=DC=openkm,DC=local
ldap.role.attribute=cn
ldap.user.search.filter=sAMAccountName={0}
ldap.role.search.filter=member={0}
```



After the update of the openkm.properties file must restart the openkm service to take effect.

Another option to configure login

In some cases might be interested to set configuration in file openkm.xml please refer to [*Configuring openkm.xml*]

documentation section for more information.

```
<security:authentication-manager alias="authenticationManager">
  <security:authentication-provider ref="ldapAuthProvider" />
</security:authentication-manager>

<beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
  <beans:constructor-arg value="ldap://192.168.0.6:389/dc=company,dc=local"/>
  <beans:property name="userDn" value="CN=Administrator,cn=users,dc=company,dc=local"/>
  <beans:property name="password" value="password"/>
</beans:bean>

<beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationPro
  <beans:constructor-arg>
    <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
      <beans:constructor-arg ref="contextSource"/>
      <beans:property name="userSearch" ref="userSearch"/>
    </beans:bean>
  </beans:constructor-arg>
  <beans:constructor-arg name="authoritiesPopulator" ref="defaultLdapAuthoritiesPopulator"/>
</beans:bean>

<beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
  <beans:constructor-arg index="0" value="" />
  <beans:constructor-arg index="1" value="sAMAccountName={0}" />
  <beans:constructor-arg index="2" ref="contextSource" />
  <beans:property name="searchSubtree" value="true" />
</beans:bean>

 <beans:bean id="defaultLdapAuthoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultLda
  <beans:constructor-arg ref="contextSource"/>
      <beans:constructor-arg value=""/>
      <beans:property name="groupSearchFilter" value="memberOf={1}"/>
      <beans:property name="groupRoleAttribute" value="cn"/>
      <beans:property name="searchSubtree" value="true" />
      <beans:property name="convertToUpperCase" value="false" />
      <beans:property name="rolePrefix" value="" />
 </beans:bean>

  <!--Needed for remember-me services -->
 <beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
```

```
  <beans:constructor-arg ref="userSearch"/>
  <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Active Directory mixed configuration

Since version 8.1.12, the OpenLDAP integration has changed:

- *The configuration parameters and login procedure remains the same.*
- *User data and roles are now managed in OpenKM administration and stored in database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

The objective of this kind of configuration is to authenticate against Active Directory but, on the other hand, obtain roles from the database.

LDAP structure

```
dc=com
  dc=company
    cn=users
      sAMAccountName=okmAdmin
        cn=users,dc=company,dc=com
        userPrincipalName=okmAdmin@mail.com
        cn=OpenKM Administrator
      sAMAccountName=user1
        cn=users,dc=company,dc=com
        userPrincipalName=user1@mail.com
        cn=User Name 1
      sAMAccountName=user2
        cn=users,dc=company,dc=com
        userPrincipalName=user2@mail.com
        cn=User Name 3
```

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from administration> Tools > Cache stats. In the second step, configure login; this configuration always works in real-time.

Step 1 - configuration parameters

*We suggest l**ogin into OpenKM with the admin URL** (for example, http://localhost:8080/openkm/admin/index) because the next steps will be necessary to restart OpenKM service, and you do not want to lose administration access.*

*The first action should be to modify **principal.adapter** parameter value and **restart the OpenKM service**. Because the session ID is kept in the browser, you should not lose the login after the service resumes and can continue working in the administration. After this change, the users and roles list will be empty from the administration. Until success configuring the following parameters, these lists will be empty.*



*We **suggest** using "**com.openkm.plugin.principal.DatabasePrincipalAdapter**" in the **principal. The adapter** configuration parameter value is either "**com.openkm.plugin.principal.MixedPrincipalAdapter**" value shown in the table below.*

*If you use "**DatabasePrincipalAdapter**" the restrictions are:*

- *Before the user login, you must create all the user data in the database.*
- *The password you set in the database will not be applied; the login process works only with an AD connection.*

- *User members of ROLE_ADMIN or ROLE_USER are created into the database.*

*Go to **administration> Configuration parameters**:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	 *The parameter is deprecated since version 8.1.12* *com.openkm.plugin.principal.MixedPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.0.6:389*
principal.ldap.security.principal	***String***	*CN=Administrator,cn=users,dc=company,dc=local*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*dc=company,dc=local*
principal.ldap.user.search.filter	***String***	*(objectclass=person)*
principal.ldap.username.attribute	***String***	*cn*

principal.ldap.username.search.base	***String***	*dc=company,dc=local*
principal.ldap.username.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*dc=company,dc=local*
principal.ldap.mail.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*

Step 1 - configure login



You must disable default OpenKM authentication to enable openkm.xml

You must set the location of the openkm.xml file

Apply changes in the ***openkm.properties*** *file.*

*The parameter "**okm.authentication.database**" disables database login.*

*The parameter "**okm.authentication.ldap**" disables internal ldap login.*

```
# Disable default OpenKM authentication
okm.authentication.database=false
okm.authentication.ldap=false

# Enable openkm.xml configuration file
okm.authentication.config=file:/home/openkm/tomcat-8.5.69/openkm.xml
```

Apply changes in the ***openkm.xml*** *file (refer to Configuring openkm.xml documentation section for more information).*

```
<!-- LDAP authentication -->
<security:authentication-manager alias="authenticationManager">
  <security:authentication-provider ref="ldapAuthProvider" />
</security:authentication-manager>

<beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
  <beans:constructor-arg value="ldap://192.168.0.6:389"/>
  <beans:property name="userDn" value="CN=Administrator,cn=users,dc=company,dc=local"/>
  <beans:property name="password" value="password"/>
  <beans:property name="baseEnvironmentProperties">
    <beans:map>
      <beans:entry>
        <beans:key>
          <beans:value>java.naming.referral</beans:value>
        </beans:key>
        <beans:value>follow</beans:value>
      </beans:entry>
```

```
      </beans:map>
    </beans:property>
  </beans:bean>

  <beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationP
    <beans:constructor-arg>
      <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
        <beans:constructor-arg ref="contextSource"/>
        <beans:property name="userSearch" ref="userSearch"/>
      </beans:bean>
    </beans:constructor-arg>
    <beans:constructor-arg name="authoritiesPopulator" ref="databaseAuthoritiesPopulator"/>
  </beans:bean>

  <beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
    <beans:constructor-arg index="0" value="dc=company,dc=local" />
    <beans:constructor-arg index="1" value="(sAMAccountName={0})" />
    <beans:constructor-arg index="2" ref="contextSource" />
    <beans:property name="searchSubtree" value="true" />
  </beans:bean>

  <beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService
    <beans:constructor-arg ref="userSearch"/>
    <beans:constructor-arg ref="databaseAuthoritiesPopulator"/>
  </beans:bean>

  <!-- Get roles from database -->
  <beans:bean id="databaseAuthoritiesPopulator" class="com.openkm.principal.DatabaseAuthoritiesPopulator">
    <beans:constructor-arg ref="dataSource"/>
    <beans:constructor-arg value="select ur_user, ur_role from OKM_USER_ROLE where ur_user=?"/>
  </beans:bean>
```



After the changes in the previous files must restart the openkm service to take effect.

Active Directory mixed configuration with LDAP synced

The objective of this kind of configuration is to authenticate against Active Directory but, on the other hand, obtain roles, usernames, and emails from the database. When this configuration is enabled, a cron job named "Sync LDAP users" periodically synchronizes the active directory users' data (username, email) in the OpenKM database and removes any database user that is not present in the active directory user list.

LDAP structure

```
dc=com
   dc=company
      cn=users
         sAMAccountName=okmAdmin
            cn=users,dc=company,dc=com
            userPrincipalName=okmAdmin@mail.com
            cn=OpenKM Administrator
         sAMAccountName=user1
            cn=users,dc=company,dc=com
            userPrincipalName=user1@mail.com
            cn=User Name 1
         sAMAccountName=user2
            cn=users,dc=company,dc=com
            userPrincipalName=user2@mail.com
            cn=User Name 3
```

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from Administration> Tools > Cache stats. In the second step, configure login; this configuration always works in real-time.

Step 1 - configuration parameters

*We suggest l**ogin into OpenKM with the admin URL** (for example, http://localhost:8080/openkm/admin/index) because the next steps will be necessary to restart OpenKM service, and you do not want to lose administration access.*

*The first action should be to modify **principal.adapter** parameter value and **restart the OpenKM service**. Because the session ID is kept in the browser, you should not lose the login after the service resumes and can continue working in the Administration. After this change, the users and roles list will be empty from the Administration. Until success configuring the following parameters, these lists will be empty.*

*We **suggest** using "**com.openkm.plugin.principal.DatabasePrincipalAdapter**" in the **principal.adapter** configuration parameter value, either "**com.openkm.plugin.principal.MixedLdapSyncedPrincipalAdapter**" value shown in the table below.*

*If you use "**DatabasePrincipalAdapter**" the restrictions are:*

- *Before the user login, you must create all the user data in the database.*

- *The password you set in the database will not be applied; the login process works only with an AD connection.*

- *User members of ROLE_ADMIN or ROLE_USER are created into the database.*

*Go to **Administration> Configuration parameters**:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated since version 8.1.12* *com.openkm.plugin.principal.MixedLdapSyncedPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.0.6:389*
principal.ldap.security.principal	***String***	*CN=Administrator,cn=users,dc=company,dc=local*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*dc=company,dc=local*
principal.ldap.user.search.filter	***String***	*(objectclass=person)*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*dc=company,dc=local*
principal.ldap.username.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.mail.attribute	***String***	*mail*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.mail.search.base	***String***	*dc=company,dc=local*
principal.ldap.mail.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*

Step 2 - Simplified login

We suggest the following configuration for version 8.1.12 and later.

The following configuration only works starting from version 8.1.12.

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
ldap.server=ldap://192.168.0.13
ldap.manager.distinguished.name=CN=Administrator,CN=Users,DC=openkm,DC=local
ldap.manager.password=*secret*
ldap.base=DC=openkm,DC=local
ldap.user.search.filter=(sAMAccountName={0})
```

Step 2 - configure login (old configuration)

We suggest the following configuration for version 8.1.11 and earlier.

You must disable default OpenKM authentication to enable openkm.xml

You must set the location of the openkm.xml file

*Apply changes in the **openkm.properties** file.*

*The parameter "**okm.authentication.database**" disables database login.*

*The parameter "**okm.authentication.ldap**" disables internal ldap login.*

```
# Disable default OpenKM authentication
okm.authentication.database=false
okm.authentication.ldap=false

# Enable openkm.xml configuration file
okm.authentication.config=file:/home/openkm/tomcat-8.5.69/openkm.xml
```

*Apply changes in the **openkm.xml** file (refer to Configuring openkm.xml documentation section for more information).*

*It is mandatory for the XML to contain a bean named **contextSource**, otherwise, an error like the following will appear:*

Parameter 1 of method ldapTemplate in org.springframework.boot.autoconfigure.ldap.LdapAutoConfiguration required a single bean

```
<!-- LDAP authentication -->
<security:authentication-manager alias="authenticationManager">
  <security:authentication-provider ref="ldapAuthProvider" />
</security:authentication-manager>

<beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
  <beans:constructor-arg value="ldap://192.168.0.6:389"/>
  <beans:property name="userDn" value="CN=Administrator,cn=users,dc=company,dc=local"/>
  <beans:property name="password" value="password"/>
  <beans:property name="baseEnvironmentProperties">
    <beans:map>
      <beans:entry>
        <beans:key>
         <beans:value>java.naming.referral</beans:value>
        </beans:key>
        <beans:value>follow</beans:value>
      </beans:entry>
    </beans:map>
  </beans:property>
</beans:bean>

<beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationPr
  <beans:constructor-arg>
    <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
      <beans:constructor-arg ref="contextSource"/>
      <beans:property name="userSearch" ref="userSearch"/>
    </beans:bean>
  </beans:constructor-arg>
  <beans:constructor-arg name="authoritiesPopulator" ref="databaseAuthoritiesPopulator"/>
</beans:bean>

<beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
  <beans:constructor-arg index="0" value="dc=company,dc=local" />
  <beans:constructor-arg index="1" value="(sAMAccountName={0})" />
  <beans:constructor-arg index="2" ref="contextSource" />
  <beans:property name="searchSubtree" value="true" />
</beans:bean>

<beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService
</beans:bean>

<beans:bean id="databaseAuthoritiesPopulator" class="com.openkm.principal.DatabaseAuthoritiesPopulator">
</beans:bean>
```



After the changes in the previous files must restart the openkm service to take effect.

For versions older than 8.1.12 should use the next definition for the databaseAuthoritiesPopulator and the userDetailService beans:

```
<beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService
  <beans:constructor-arg ref="userSearch"/>
  <beans:constructor-arg ref="databaseAuthoritiesPopulator"/>
</beans:bean>

<!-- Get roles from database -->
<beans:bean id="databaseAuthoritiesPopulator" class="com.openkm.principal.DatabaseAuthoritiesPopulator">
```

```
    <beans:constructor-arg ref="dataSource"/>
    <beans:constructor-arg value="select ur_user, ur_role from OKM_USER_ROLE where ur_user=?"/>
  </beans:bean>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

LDAP best practices for filtering users and roles

Usually you only want to retrieve a subset of users and roles present in your LDAP, to be shown in user interface lists or be able to login into.

Example based on single LDAP group

- *Create a group named **OpenKM (CN=OpenKM,CN=users,DC=company,DC=com)**. It will be used for filtering **users and roles** purpose.*
- *Assign roles and users are members of OpenKM group.*
- *Goal, only users and roles which are members of OpenKM group will be displayed in user interface lists.*

For filtering user interface:

Field / Property	*Type*	*Description*
principal.ldap.user.search.filter	***String***	*(&(objectclass=person) (memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*
principal.ldap.role.search.filter	***String***	*(&(objectclass=group) (memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectclass=group)(cn={0}) (memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}) (memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*

For login restriction:

Modify openkm.properties with:

```
ldap.user.search.filter=(&(sAMAccountName={0})(memberOf=CN=OpenKM,CN=users,DC=company,DC=com))>
```

In case you use openkm.xml then with:

```
<module-option name="baseFilter">(&amp;(sAMAccountName={0})(memberOf=CN=OpenKM,CN=users,DC=comp
```

Example based on ROLE_USER and ROLE_ADMIN group

- *Create group name **OpenKM (CN=OpenKM,CN=users,DC=company,DC=com)**. It will be used for filtering **roles***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

purpose.

- *Assign roles are members of OpenKM group.*
- *Goal, only roles with OpenKM role will be displayed in user interface lists.*
- *Goal, only users what are members of ROLE_USER or ROLE_ADMIN group will be displayed in user interface lists.*

For filtering user interface:

Field / Property	*Type*	*Description*
principal.ldap.user.search.filter	***String***	*(&(objectclass=person) (| (memberOf=cn=ROLE_USER,cn=users,dc=company,dc=com) (memberOf=cn=ROLE_ADMIN,cn=users,dc=com*
principal.ldap.role.search.filter	***String***	*(&(objectclass=group) (memberOf=cn=OpenKM,dc=company,dc=com))*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectClass=group)(cn={0}) (memberOf=cn=OpenKM,cn=users,DC=company,DC=com))*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}) (memberOf=cn=OpenKM,cn=users,DC=company,DC=com))*

For login restriction:

Modify openkm.properties with:

```
ldap.user.search.filter=(&(sAMAccountName={0})(objectClass=person)(|(memberOf=CN=ROLE_USER,CN=users,c
```

In case you use openkm.xml then with:

```
<module-option name="baseFilter">(&amp;(sAMAccountName={0})(objectClass=person)(|(memberOf=CN=ROLE_
```



There are some characters restricted in XML, one of these is ***&****. Should be used* ***&*** *in place you usually should use* ***&****, otherwise you will get an error when starting the application.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ClearOS LDAP configuration

Since version 8.1.12, the OpenLDAP integration has changed:

- *The configuration parameters and login procedure remains the same.*
- *User data and roles are now managed in OpenKM administration and stored in database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

Need to install the Certificate Manager and Directory Server modules.

- *Create an **okmAdmin** user in ClearOS.*
- *Create **ROLE_ADMIN, ROLE_USER** groups in ClearOS, populate.*

LDAP Structure

```
dc=local
   dc=mydomain
    ou=Accounts
     ou=Groups
      cn=ROLE_ADMIN
       member=okmAdmin
       member=user1
       member=user2
       objectClass=posixGroup
      cn=ROLE_USER
       member=user3
       member=user4
       objectClass=posixGroup
      cn=ROLE_XXXX
       objectClass=posixGroup
      cn=ROLE_YYYY
       objectClass=posixGroup
      ...

                ou=Users
      uid=okmAdmin
       memberOf=CN=ROLE_ADMIN,ou=Groups,ou=Accounts,dc=mydomain,dc=local
       mail=okmAdmin@mail.com
       cn=OpenKM Administrator
       objectClass=inetOrgPerson
      uid=user1
       memberOf=CN=ROLE_ADMIN,ou=Groups,ou=Accounts,dc=mydomain,dc=local
       mail=user1@mail.com
       cn=User Name 1
       objectClass=inetOrgPerson
      uid=user2
       memberOf=CN=ROLE_ADMIN,ou=Groups,ou=Accounts,dc=mydomain,dc=local
       mail=user2@mail.com
       cn=User Name 3
       objectClass=inetOrgPerson
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Any distinguished name include by default ***dc=mydomain,dc=local***

The OpenKM integration with LDAP has three steps. In the first step, configure OpenKM to retrieve the list of users and roles from the LDAP. This list is cached during 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clean the cache from Administration > Tools > Cache stats. In the second step create the certificate and finally in the third step configure login, this last configuration works always in real time.

Step 1 - configuration parameters

*We suggest l**ogin in OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps will be necessary to restart OpenKM service and you do not want to lose administration access.*

*The first action should be to modify **principal.adapter** parameter value and **restart OpenKM service**. Because session ID is kept in the browser you should not lose the login after the service restarted and can continue working in the administration. After this change, the users and roles list will be empty from the administration. Until success configuring the next parameters these lists will be empty.*

Go to Administration > Configuration parameters

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated since version 8.1.12* *com.openkm.plugin.principal.LdapPrincipalAdapter*
principal.database.filter.inactive.users	***Boolean***	*The parameter is deprecated since version 8.1.12* *true*
principal.hide.connection.roles	***Boolean***	*false*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldaps://ldap.mydomain.local:636*
principal.ldap.security.principal	***String***	*cn=Manager,ou=Internal,dc=mydomain,dc=local*
principal.ldap.security.credentials	***String***	*pass1234*
principal.ldap.referral	***String***	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*uid*
principal.ldap.user.search.base	***List***	*ou=Users,ou=Accounts,dc=mydomain,dc=local*
principal.ldap.user.search.filter	***String***	*(objectClass=inetOrgPerson)*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*ou=Users,ou=Accounts,dc=mydomain,dc=local*
principal.ldap.username.search.filter	***String***	*(&(objectclass=inetOrgPerson)(uid={0}))*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*dc=company,dc=local*
principal.ldap.mail.search.filter	***String***	*(&(objectclass=inetOrgPerson)(uid={0}))*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*ou=Groups,ou=Accounts,dc=mydomain,dc=local*
principal.ldap.role.search.filter	***String***	*objectClass=posixGroup*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*
principal.ldap.roles.by.user.search.base	***String***	*ou=Users,ou=Accounts,dc=mydomain,dc=local*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(uid={0}))*
principal.ldap.users.by.role.attribute	***String***	*member*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.users.by.role.search.base	**String**	ou=Groups,ou=Accounts,dc=mydomain,dc=local
principal.ldap.users.by.role.search.filter	**String**	(&(objectClass=posixGroup)(cn={0}))

Step 2 - create a Certificate

ClearOS by default only allows LDAPS.

> *You should create a certificate on ClearOS with the leftmost RDN of the Subject being the common name (FQDN) of your ClearOS server. Easy way is to only specify the common name, and when signing use a policy that is permissive.*
>
> *You can create an alias (CNAME) called ldap if you wish.*

```
$ openssl req -out ldap.csr -key private/sys-0-key.pem -new
$ openssl ca -policy policy_anything -days 3650 -out /etc/pki/CA/certs/ldap.mydomain.local.crt -infiles ldap.csr
```

Add the CA cert to the nssdb:

```
$ certutil -A -d /etc/pki/nssdb/ -n "CA certificate" -t "CT,," -a -i /etc/pki/CA/ca-cert.pem
```

The LDAP server certificate needs to be saved in the nssdb with the name of Server-Cert.

```
$ openssl pkcs12 -export -inkey /etc/pki/CA/private/sys-0-key.pem -in /etc/pki/CA/certs/ldap.mydomain.local.crt -out
/root/Server-Cert.p12 -nodes -name 'Server-Cert'
$ pk12util -i /root/Server-Cert.p12 -d /etc/pki/nssdb/
```

Edit the file/etc/openldap/slapd.conf:

```
$ vim /etc/openldap/slapd.conf
```

```
TLSCACertificatePath    /etc/pki/nssdb
TLSCertificateFile      Server-Cert
TLSVerifyClient         never
```

Allow LDAP in ClearOS to be queried by OpenKM - edit the end of slapd.conf thusly:

```
access to *
   by self write
   by peername.ip=127.0.0.1 read
   ...
   by peername.ip=<OpenKM's IP address> read
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Copy ClearOS' public CA certificate to the OpenKM server and add it to OpenKM's keystore:

```
$ keytool -import -trustcacerts -keystore /opt/openkm-6.2.3-community/java/jre/lib/security/cacerts -alias myca -file
/etc/pki/tls/certs/myca.pem
```

Step 3 - configure login

Apply changes in the ***openkm.properties*** *file.*

*The parameter "**okm.authentication.database**" is used to disable database login.*

*The parameter "**okm.authentication.ldap**" is used to enable ldap login.*

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
ldap.server=ldaps://ldap.mydomain.local
ldap.manager.distinguished.name=cn=Manager,ou=Internal,dc=mydomain,dc=local
ldap.manager.password=*secret*
ldap.base=dc=mydomain,dc=local
ldap.role.attribute=cn
ldap.user.search.filter=uid={0}
ldap.role.search.filter=member={0}
```



After the update of the openkm.properties file must restart the openkm service to take effect.

Another option to configure login

In some cases might be interested to set configuration in file openkm.xml please refer to *Configuring openkm.xml*

documentation section for more information.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:security="http://www.springframework.org/schema/security"
        xmlns:task="http://www.springframework.org/schema/task"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/security
                http://www.springframework.org/schema/security/spring-security.xsd
                http://www.springframework.org/schema/task
                http://www.springframework.org/schema/task/spring-task.xsd">


  <!-- Security configuration -->
  <security:authentication-manager alias="authenticationManager">
   <security:authentication-provider ref="ldapAuthProvider" />
  </security:authentication-manager>

  <beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
    <beans:constructor-arg value="ldaps://ldap.mydomain.local:636/dc=mydomain,dc=local"/>
```

```
    <beans:property name="userDn" value="cn=Manager,ou=Internal,dc=mydomain,dc=local"/>
     <beans:property name="password" value="*secret*"/>
  </beans:bean>

  <beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationP
 <beans:constructor-arg>
  <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
   <beans:constructor-arg ref="contextSource"/>
   <beans:property name="userSearch" ref="userSearch"></beans:property>
    </beans:bean>
 </beans:constructor-arg>
 <beans:constructor-arg name="authoritiesPopulator" ref="defaultLdapAuthoritiesPopulator"/>
  </beans:bean>

   <beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
      <beans:constructor-arg index="0" value="ou=Users,ou=Accounts" />
      <beans:constructor-arg index="1" value="uid={0}" />
      <beans:constructor-arg index="2" ref="contextSource" />
      <beans:property name="searchSubtree" value="true" />
   </beans:bean>

   <beans:bean id="defaultLdapAuthoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultL
  <beans:constructor-arg ref="contextSource"/>
   <beans:constructor-arg value="ou=groups,ou=Accounts"/>
   <beans:property name="groupSearchFilter" value="member={0}"/>
   <beans:property name="groupRoleAttribute" value="cn"/>
   <beans:property name="searchSubtree" value="true" />
   <beans:property name="convertToUpperCase" value="true" />
   <beans:property name="rolePrefix" value="" />
  </beans:bean>


  <beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
  </beans:bean>

 </beans:beans>
```

For versions older than 8.1.12 should use the next definition for the userDetailService :

```
 <!--Needed for remember-me services -->
 <beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
  <beans:constructor-arg ref="userSearch"/>
  <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>
```

LDAP troubleshooting

How to do a nested group membership filter in active directly

The sample below demonstrates how to configure AD search to retrieve users that are members of some role that is a member of ROLE_OPENKM.

Field / Property	Type	Description
principal.ldap.user.search.filter	**String**	((objectclass=user) ((memberOf:1.2.840.113556.1.4.1941:=CN=ROLE_OPENKM,OU=Ope

No UserDetailsService registered error

*If you see an exception like this "**org.springframework.context.ApplicationContextException: No UserDetailsService registered,**" you might add in your openkm.xml this line:*

```
<beans:bean id="userDetailService" class="com.openkm.spring.UserTenantDetailsService"/>
```

This issue only affects versions of OpenKM where the "remember me" feature was included. Take a look at the Changelog to check if your OpenKM is affected.

You can find more information about the "remember me" configuration in the Security configuration parameters.

javax.naming.PartialResultException error

*If you see an exception like this "**javax.naming.PartialResultException**" you probably can not use Active Directory basic configuration.*

javax.naming.PartialResultException: Unprocessed Continuation Reference(s); remaining name 'cn=users,dc=company,dc=com'

Solution 1:

Use ports 3268 or 3269 (ldaps) in the AD connection.

*Go to **Administration > Configuration parameters**:*

Field / Property	Type	Description
principal.ldap.server	**String**	ldap://192.168.xxx.xxx:3268

Information about ports 3268 and 3269 at :

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

* *https://social.technet.microsoft.com/Forums/en-US/e52b9154-b93a-4a3b-b6f2-0285f932da14/389-and-3268-port-difference?forum=winserverDS*

* *https://github.com/stevespringett/Alpine/issues/19*

Solution 2:

Enable ldap to ignore the partial results exception property.

*Go to **Administration > Configuration parameters**:*

Field / Property	*Type*	*Description*
ldap.ignore.partial.result.exception	***Boolean***	*true*

Solution 3:

Enable ldap referral property.

*Go to **Administration > Configuration parameters**:*

Field / Property	*Type*	*Description*
principal.ldap.referral	***String***	*follow*

Read these articles:

- *Referrals in the JNDI*
- *JNDI Implementor Guidelines for LDAP Service Providers*

Can't connect to LDAP server

If your LDAP server is configured under SSL, then you should use ldaps://

Solution:

*Go to **Administration > Configuration parameters**:*

Change server connection URL to ldaps.

Field / Property	*Type*	*Description*
principal.ldap.server	***String***	*ldaps://192.168.0.6:389*

Slow login or not able to login

It may be a problem with LDAP DNS names resolution.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Solution:

To prevent this issue - especially on Windows OS Family - it's a good practice to add all subdomains in the application server's host file (/etc/hosts for Linux or C:/Windows/system32/driver/etc/host for windows).

Example based with active directory (LDAP) with distinguished base name ***dc=company,dc=com,*** *and server at* ***SBSSERVER*** *domain server name.*

```
10.10.1.2 company.com
10.10.1.2 SBSSERVER SBSSERVER.company.com
10.10.1.2 Schema.Configuration.company.com
10.10.1.2 Configuration.company.com
10.10.1.2 DomainDnsZones.company.com
10.10.1.2 ForestDnsZones.company.com
```

More information is at the OpenKM forum.

If, after adding these subdomains, the login continues taking a lot of time, check with your IT Security team if they have enabled some micro-segmentation policies. That may be another reason why login takes so much time.

To check connections done by the application in the login process to the AD, can execute the following command line:

```
$ tcpdump -A tcp port 389
```

To capture only the request:

```
$ sudo tcpdump -i any port 389 > login-test-request.txt
```

Wireshark dump (pcap) and transform to human readable:

```
$ sudo tcpdump -A port 389 -w login-test.pcap
$ tshark -r login-test.pcap -V > login-test.txt
```

Force all users to be members of ROLE_USER.

In vast user AD repositories, it can not be practiced for the AD administrator to add a new group to all the users; in this case, we can force across xml login configuration that any logged user be a member of ROLE_USER. Must be added the XML tag ***<beans:property name="defaultRole" value="ROLE_USER" />****.*

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:security="http://www.springframework.org/schema/security"
        xmlns:task="http://www.springframework.org/schema/task"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/security
```

```
                        http://www.springframework.org/schema/security/spring-security.xsd
                        http://www.springframework.org/schema/task
                        http://www.springframework.org/schema/task/spring-task.xsd">

 <security:authentication-manager alias="authenticationManager">
    <security:authentication-provider ref="ldapAuthProvider" />
  </security:authentication-manager>

 <beans:bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
  <beans:constructor-arg value="ldap://192.168.xxx.xxx"/>
  <beans:property name="userDn" value="CN=connect,OU=OpenKM,DC=company,DC=com"/>
  <beans:property name="password" value="*****"/>
  <beans:property name="baseEnvironmentProperties">
    <beans:map>
      <beans:entry>
        <beans:key>
          <beans:value>java.naming.referral</beans:value>
        </beans:key>
        <beans:value>follow</beans:value>
      </beans:entry>
    </beans:map>
  </beans:property>
 </beans:bean>

 <beans:bean id="ldapAuthProvider" class="org.springframework.security.ldap.authentication.LdapAuthenticationPro
  <beans:constructor-arg>
   <beans:bean class="org.springframework.security.ldap.authentication.BindAuthenticator">
     <beans:constructor-arg ref="contextSource"/>
     <beans:property name="userSearch" ref="userSearch"/>
   </beans:bean>
  </beans:constructor-arg>
  <beans:constructor-arg name="authoritiesPopulator" ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>

 <beans:bean id="userSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
  <beans:constructor-arg index="0" value="DC=company,DC=com" />
  <beans:constructor-arg index="1" value="(&amp;(sAMAccountName={0})(|(memberOf=CN=ROLE_ADMIN,OU=Op
  <beans:constructor-arg index="2" ref="contextSource" />
  <beans:property name="searchSubtree" value="true" />
 </beans:bean>

  <beans:bean id="defaultLdapAuthoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultLd
  <beans:constructor-arg ref="contextSource"/>
     <beans:constructor-arg value="DC=company,DC=com"/>
     <beans:property name="groupSearchFilter" value="member={0}"/>
     <beans:property name="groupRoleAttribute" value="cn"/>
     <beans:property name="searchSubtree" value="true" />
     <beans:property name="convertToUpperCase" value="false" />
     <beans:property name="rolePrefix" value="" />
     <beans:property name="defaultRole" value="ROLE_USER" />
 </beans:bean>

 <!--Needed for remember-me services -->
 <beans:bean id="userDetailService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsService"
  <beans:constructor-arg ref="userSearch"/>
  <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator"/>
 </beans:bean>

 </beans:beans>
```

org.springframework.beans.factory.parsing.BeanDefinitionParsingException

*If you see an error like this "**org.springframework.beans.factory.parsing.BeanDefinitionParsingException**" you probably have*

some wrong spring version set in the openkm.xml:

org.springframework.beans.factory.parsing.BeanDefinitionParsingException: Configuration problem: You cannot use a spring-security-2.0.xsd or spring-security-3.0.xsd or spring-security-3.1.xsd schema with Spring Security 3.2. Please update your schema declarations to the 3.2 schemas.
Offending resource: URL [file:/home/openkm/tomcat-8.5.69/openkm.xml]
at
org.springframework.beans.factory.parsing.FailFastProblemReporter.fatal(FailFastProblemReporter.java:59)

Solution:

Edit the openkm.xml file and remove the version numbers:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:security="http://www.springframework.org/schema/security"
	xmlns:task="http://www.springframework.org/schema/task"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
		http://www.springframework.org/schema/security
		http://www.springframework.org/schema/security/spring-security-3.1.xsd
		http://www.springframework.org/schema/task
		http://www.springframework.org/schema/task/spring-task-3.1.xsd">
```

to

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:security="http://www.springframework.org/schema/security"
	xmlns:task="http://www.springframework.org/schema/task"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/security
		http://www.springframework.org/schema/security/spring-security.xsd
		http://www.springframework.org/schema/task
		http://www.springframework.org/schema/task/spring-task.xsd">
```

LDAP: error code 8 - BindSimple: Transport encryption required

The transport encryption required error usually happens when you are trying to connect with ldaps (with SSL encryption), and you have not configured the trust certificate into the JAVA certs storage.

*First, check that you are using "**ldaps://**" - **ends with s** - in your **openkm.xml** file for the connection; when you get this error, use it.*

*The first step is to **get the certificate from the server,** for it follows the steps below:*

```
$ openssl s_client -showcerts -connect  your.ldap.server.com:636
```

The OpenSSL windows binaries https://wiki.openssl.org/index.php/Binaries (version tested https://indy.fulgan.com/SSL/)

The output will contain several entries delimited with the following:

```
-----BEGIN CERTIFICATE-----
aklfhskfadljasdl1340234234ASDSDFSDFSDFSDFSD
....
-----END CERTIFICATE-----
```

To prevent the loss of JKS after the Java upgrade, do not create the JKS under the JDK folders.

In the sample below, it is supposed the $TOMCAT_HOME is set to /home/openkm/tomcat

Copy the last certificate entry into a file (ldapca.crt)

*Then, **add it to the java** keystore in $JRE_HOME/lib/security*

```
$ cd $TOMCAT_HOME

$ mkdir jks

$ keytool -import -alias ldapca_self_sign -keystore cacerts -storepass changeit -file ldapca.crt
```

*Finally **enable trust store** into the file $TOMCAT_HOME/bin/setenv.sh*

```
# JAVA enable trust store
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=/home/openkm/tomcat/jks/cacerts -Djavax.net.ssl.trustStore
```

Additional information:

- *https://docs.spring.io/spring-ldap/docs/1.2.0/reference/#dir-context-url*
- *https://stackoverflow.com/questions/9903560/spring-security-ssl-ldap-and-no-certificate*

Ldap error: Caused by java.security.cert.CertificateException: No subject alternative DNS name matching

In the log, maybe show some stack trace errors like:

```
Caused by: java.security.cert.CertificateException: No subject alternative DNS name matching ldap.openkm.com fo
    at sun.security.util.HostnameChecker.matchDNS(HostnameChecker.java:214)
    at sun.security.util.HostnameChecker.match(HostnameChecker.java:96)
    at sun.security.ssl.X509TrustManagerImpl.checkIdentity(X509TrustManagerImpl.java:459)
    at sun.security.ssl.X509TrustManagerImpl.checkIdentity(X509TrustManagerImpl.java:436)
    at sun.security.ssl.X509TrustManagerImpl.checkTrusted(X509TrustManagerImpl.java:200)
    at sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:124)
    at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1621)
    ... 72 more
```

Or like:

```
Caused by: org.springframework.ldap.CommunicationException: simple bind failed: ldap.openkm.com:636; nested e
        at org.springframework.ldap.support.LdapUtils.convertLdapException(LdapUtils.java:100)
        at org.springframework.ldap.core.support.AbstractContextSource.createContext(AbstractContextSource.java:2
        at org.springframework.ldap.core.support.AbstractContextSource.doGetContext(AbstractContextSource.java:1
        at org.springframework.ldap.core.support.AbstractContextSource.getReadOnlyContext(AbstractContextSource
        at org.springframework.ldap.core.LdapTemplate.executeReadOnly(LdapTemplate.java:791)
        at org.springframework.security.ldap.SpringSecurityLdapTemplate.searchForSingleEntry(SpringSecurityLdapT
        at org.springframework.security.ldap.search.FilterBasedLdapUserSearch.searchForUser(FilterBasedLdapUser
        at org.springframework.security.ldap.authentication.BindAuthenticator.authenticate(BindAuthenticator.java:90)
        at org.springframework.security.ldap.authentication.LdapAuthenticationProvider.doAuthentication(LdapAuthent
```

Modify JVM configuration parameters in setenv.sh or setenv.bat by adding the JVM configuration parameter:

```
-Dcom.sun.jndi.ldap.object.disableEndpointIdentification=true
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Log Configuration

Logback based Log Configuration

OpenKM uses logback which is a Java-based loggin utility. This is useful, for example when you need to get more information about something that is not working properly in OpenKM, for example third-party application configuration or Spring authentication. By default only log messages with INFO level are shown, but you can increase this log information by raising the log level to WARN or DEBUG. The log configuration file is ***$TOMCAT_HOME/logback.xml*** *here you can change the error level message generated by a give class or package.*



When OpenKM detect a modification in the ***logback.xml*** *file, it will reload it so you don't need to stop and start OpenKM to make these changes effectives.*

Third-party application execution

This is useful when something like a document preview fails, because this way OpenKM will log every external program execution:

```
<logger name="com.openkm.util" level="DEBUG" />
```

Enable log authentication

If you need to debug the security configuration, edit the ***logback.xml*** *file and add this line:*

```
<logger name="org.springframework.security" level="DEBUG" />
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

LibreOffice configuration

> *After changing **system.openoffice.program** parameter is required to restart the application.*

Linux

Debian & Ubuntu

Install the application:

```
$ apt-get install libreoffice
```

Install languages:

```
$ apt-get install libreoffice-l10n-en-gb
```

*Go to **Administration** > **Configuration parameters**:*

Field / Property	*Type*	*Description*
system.openoffice.program	***String***	*The path may change depending on your OS version.* */usr/bin/soffice*

Windows

Download and install one of them:

- *LibreOffice: http://www.libreoffice.org*

*Go to **Administration** > **Configuration parameters**:*

Field / Property	*Type*	*Description*
system.openoffice.path	***String***	*The path can change depending on your OS version.* *c:\Program files\OpenOffice 4*

> *Read OpenOffice LibreOffice configuration parameters to get more information about all available LibreOffice configuration parameters.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Security configuration

OpenKM can work with several security managers, depending on the selected value, the manner in which the result is evaluated is different. This application has separated managers, one for the search engine and other for the rest of the application.

Search engine managers:

- *lucene.*
- *am_more, am_window or am_limited.*

> *The fastest one is **lucene**, because it is the Lucene search engine which restricts the search results based on the node security. The others are **am_more**, **am_window** and **am_limited**, which make use of the AccessManager - security.access.manager value - to restrict the search results, these will reduce OpenKM performance (at least in search) but will match customer requirements. The performance can be improved by customizing **com.openkm.cache.parentNodePermissions** cache.*

Rest of application manager:

> *Access manager works with plugin architecture. From*

Type	*Name*	*Description*
Default	***com.openkm.plugin.access.DbSimpleAccessManager***	*Default OpenKM security manager. Analyze only the current node, not all the hierarchy.*
Recursive	***com.openkm.plugin.access.DbRecursiveAccessManager***	*Analyze all the hierarchy.* > *Analyze all the hierarchy will take more time and may cause performance issues.*
Read recursive	***com.openkm.plugin.access.DbReadRecursiveAccessManager***	*Analyze all the hierarchy (only for read access).* > *Analyze all the hierarchy will take more time and may*

		cause performance issues.
Default file plan	***com.openkm.plugin.access.DbFilePlanAccessManagerWrapper***	*Analyze based on document type and file plan restrictions.* *Only should be enabled when file plan is enabled.*

*The **Deafult** option only evaluates the selected node. With **recursive**, the security is evaluated starting with the selected node until the repository root (so, evaluating form bottom to top). The **"Read recursive"** is used to prevent showing documents in search results when any folder in the path doesn't have read access for the user who ran the query. Should be used in combination with **am_more**.*

Application by default is configured with:

Field / Property	*Type*	*Description*
security.access.manager	***String***	*com.openkm.plugin.access.DbSimpleAccessManager*
security.search.evaluation	***String***	*lucene*

Diferences betweem simple and recursive

- *okm:root*
  - *level-1a*
    - *level-2a*
      - *level-3a*
        - *alpha.doc*
        - *beta.doc*
        - *gamma.doc*
      - *level-3b*
  - *level-1b*
  - *level-1c*

***When using simple** if you revoke permission from the level-2a folder and the user tries to view the level-3a folder, OpenKM will still show the documents to this user. When the user performs a search, OpenKM will return these documents as search results.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

However, the user can't access the documents because when OpenKM tries to access the document, it will open every path location and will fail to open level-2a because the user has no rights. But these document can be accessed using the API. You need to apply revoke recursively to prevent the user from seeing these documents, so they won't be shown in the search results.

***When using recursive** if you revoke permission from the level-2a folder and the user tries to view the level-3a folder, OpenKM will report an error because Access Manager also evaluates the permission of the level-2a ancestor node. When the user performs a search, OpenKM won't show these documents in the search results. Keep in mind that this Access Manager implementation is slower because it needs to evaluate more nodes: the more depth node the more parent nodes need to be evaluated.*

Extending security

*You can use the extend security to download files, start workflows, add, remove or modify property groups (metadata) or compact history. To enable extended security evaluation you must change the value of **security.extended.mask** parameter.*



After the value is changed, refresh the browser to take effect in user view.

Values:

- *DOWNLOAD = 1024;*
- *START_WORKFLOW = 2048;*
- *COMPACT_HISTORY = 4096;*
- *PROPERTY_GROUP = 8192;*

For example, to enable download and property groups you should put the mask 9216 (1024 + 8192)

Field / Property	*Type*	*Description*
security.extended.mask	***Integer***	*9216*

Configuration for security changes which take a long time

*By default when you change the security of a node in standard view, every time you click to grant or revoke a permission the action is performed by OpenKM. This means that if you want to change the security of 10 users, 10 commands are sent to the server. This became a problem when every change affects several hundred (or even thousands) of nodes. There's an alternative security configuration parameter named **security.mode.multiple** which enables multiple changes at a time, so it's so faster.*

Field / Property	*Type*	*Description*
security.mode.multiple	***Boolean***	*true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Some security changes can take a long time to be completed because affect a lot of nodes. It's a good practice to do it as a background task. For it there's a configuration parameter named ***security.live.change.node.limit*** *used to set the limit of nodes which can be changed in "live" or "realtime" mode. By default this value is set to 100, which means that any change which affect to more than 100 nodes will be automatically done as a background task.*

Field / Property	*Type*	*Description*
security.live.change.node.limit	***Integer***	*100*

To check the progress of these pending tasks go to ***Administration > Statistics > Pending task queue***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring Two Factor Authentication

Two factor authentication (2FA) provides extra security to OpenKM login. In addition to the typical login and password, once the user is authenticated, will be asked for an extra six digits code generated in a mobile application called Google Authenticator.

Compliance

The two-factor authentication implements a two-step verification of HTOP/TOTP. Also known as one time password.

It has been tested with:

- *The Google Authenticator*
- *The Microsoft authenticator app*

Should work with other applications like:

- *OTP Auth (IOS)*
- *Step Two (IOS)*
- *FreeOTP*
- *Twilio Authy*
- *Duo Mobile de Cisco*

Prerequisite

Because two-factor authentication has been implemented using Google Authenticator, it is necessary to download and install the application from Google Play or Apple market.



https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2

Once installed and configured, this application does not event need Internet connection.

OpenKM configuration

Each user can configure his own account to use 2FA. To do it, he has to go to ***Tools*** *>* ***Preferences*** *>* ***User Configuration****.*

If 2FA was enabled by the administrator, at the bottom of the User Configuration dialog you would be able to see a new option ***2FA****. Depending on whether 2FA is enabled ypu would see:*

- *Disable button: is 2FA was already configured for this user.*
- *Enable button: is 2FA has not been configured yet.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

If you click on enable button, a new window with the following parameters will appear:

- *The current user id to be configured.*
- *A QR code which you should scan from* ***Google Authenticator*** *application.*
- *A text box where you have to write the verification code generated by* ***Google Authenticator*** *(once QR code has been scanned).*

Once you have complete these steps, please click on the accept button.

Scan QR code

To scan the QR code, open ***Google Authenticator*** *on your mobile and perfom the following steps:*

1. *Click on configure account.*
2. *Select* ***Scan code****.*
3. *Focus the mobile toward the the QR image of the screen.*
4. *Once the image is properly scanned the accound for OpenKM is added.*
5. *Now you will see a square with a number (code) which you will have to introduce when required.*



These generated codes are time based: they will expiry every 60 seconds. If you pay attention to ***Google Authenticator*** *screen, you will see how the code changes every period of time.*

Google Authenticator use

Once ***2FA*** *is configured on your OpenKM account, next time you log into OpenKM, after your login and password are verified, a new screen will appear, where it will be required to fill in, a code that it will have to be obtained from the application previously configured inside* ***Google Authenticator*** *application.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Document expiration is a notification system alert which can be used to advice users nearly document expiration and expired documents.

*This option is only available if you have enabled the **Document expiration** feature.*

Document expiration is based on:

- *Metadata group feature.*
- *Crontab feature.*
- *Report feature.*

More information at Document expiration extension.

Configuration

Download the expiration files document_expiration_8.zip.

Versions prior to 8.1.14 must use the following file document_expiration_7.zip.

Register parameters

*Execute SQL from **Administration** > **Tools** > Database query.*

```
-- NOTIFICATION PARAMETERS
DELETE FROM OKM_CONFIG WHERE CFG_KEY LIKE '%expiration.%';
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.date.pattern','string','da
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.expiration.alert.days','s
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.notification.subject','str
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.notification.template','te
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.alert.subject','string','Al
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.alert.template','text','Do
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.expired.subject','string',
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.expired.template','text',
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.expirated.reminder.day
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.expirated.reminder.sub
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('expiration.expirated.reminder.tem
```

More information at Database query.

Configure parameters

Field / Property	*Type*	*Description*
expiration.date.pattern	***String***	*Set the date pattern.*

		dd-MM-yyyy HH:mm:ss
expiration.notification.subject	***String***	*Mail subject.* *New uploaded document with expiration*
expiration.notification.template	***String***	*Mail body.* *New document uploaded ${docUrl} Expiration date:${date} *
expiration.expiration.alert.days	***Integer***	*Set the number of days before expiration when it will be send the expiration alert message.* *15* *Default document expiration set an unique value for all document. This is the default logic but can be implemented your own.*
expiration.alert.subject	***String***	*Mail subject.* *Alert documents will expire*
expiration.alert.template	***String***	*Mail body.* *Document ${docUrl} will expire at ${date} *
expiration.expirated.reminder.days	***Integer***	*Set the number of days for sending reminder notification for expired documents that have expired more than those days ago.* *15* *Default document expiration set an unique value for all document. This is the default logic but can be implemented your own.*
expiration.expirated.reminder.subject	***String***	*Mail subject.* *Reminder alert of documents expired*

expiration.expirated.reminder.template	***String***	*Mail body.* *Document ${docUrl} expired at ${date} *

These parameters are set at Administration > Configuration parameters view.

Register metadata group

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN" "http://www.openkm.com/dtd/pr
<property-groups>
  <property-group label="Expiration" name="okg:expiration">
    <input label="Expiration date" name="okp:expiration.date" type="date">
      <validator type="req"/>
    </input>
    <select label="State" name="okp:expiration.state" type="simple">
      <option label="Notify" value="notify" selected="true"/>
      <option label="Valid" value="valid"/>
      <option label="Expired" value="expired"/>
      <validator type="req"/>
    </select>
    <select label="Notify" name="okp:expiration.groupnotify" type="multiple" className="com.openkm.plugin.form.va
      <validator type="req"/>
    </select>
    <select label="Notify expiration alert" name="okp:expiration.groupalert" type="multiple" className="com.openkm
      <validator type="req"/>
    </select>
    <select label="Notify expirated" name="okp:expiration.groupexpired" type="multiple" className="com.openkm.pl
      <validator type="req"/>
    </select>
    <select label="Notify expirated reminder alert" name="okp:expiration.groupexpiratedreminder" type="multiple" cla
      <validator type="req"/>
    </select>
  </property-group>
</property-groups>
```

Crontab task

- *Copy jar files into the $TOMCAT_HOME/plugins folder.*
- *Restart OpenKM (that will register crontab tasks).*

Task	*Expression*	*Description*
InitialNotificationation	***0 */1 * * * **	*After is added metadata group to the node is sent the "initial notification" mail.* *The crontab task is executed each minute looking for new nodes with metadata group.*

AlertExpirationNotificationation	***0 30 0 * * * ***	*Sent daily mail alert for documents expire soon.*
ExpirationNotificationation	***0 0 0 * * ****	*Sent daily mail alert for documents expired today.* *This crontab task should be executed before **AlertExpirationNotificationation** task.*
AlertExpiratedReminderNotification	***0 0 1 * * ****	*Sent daily mail alert for documents expired x days ago.*

More information at Crontab.

Register reports

Report	*Description*
ValidDocuments.jrxml	*List all valid documents.*
ExpiredDocuments.jrxml	*List all expired documents.*
NextExpirationDocuments.jrxml	*List documents what will expire soon.*

More information at Reports.

See also Profiles for configuring reports in user interface menu view.

Configuring CIFS

*The aim of this configuration is to allow to see OpenKM as a remote drive using **CIFS** protocol. Bellow it will be depicted two ways of configuiring it:*

- *Authentication using Database*
- *Authentication using Active Directory*



This integration works only with SMBv1 which is disabled by default in Windows 10. If you want to enable it, open a PowerShell and follow these steps:

```
> Get-WindowsOptionalFeature -Online -FeatureName smb1protocol
> Disable-WindowsOptionalFeature -Online -FeatureName smb1protocol
> Enable-WindowsOptionalFeature -Online -FeatureName smb1protocol
```

After that, you need to restart the server. More info at [How to detect, enable and disable SMBv1, SMBv2, and SMBv3 in Windows](#).

Database Authentication

This is the simplest case. You have to configure these parts:

- *OpenKM Server*
- *OpenKM Configuration*
- *Windows Client*

OpenKM server

*If the system could not be started with root permissions the ports commented below, must be over 1024 (the default system ports are 138,137,139 and 445), so it is necessary to redirect the original ports to the new ones. To do this it is necessary to execute the following commands as **root** or using **sudo**:*

```
echo 1 > /proc/sys/net/ipv4/ip_forward
modprobe iptable_nat
iptables -F
iptables -t nat -F
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 445 -j DNAT --to-destination :1445
iptables -t nat -A PREROUTING -p tcp --dport 139 -j DNAT --to-destination :1139
iptables -t nat -A PREROUTING -p tcp --dport 137:139 -j DNAT --to-destination :1137-1139
```

```
iptables -t nat -A PREROUTING -p udp --dport 137:139 -j DNAT --to-destination :1137-1139
```



Sometimes for testing purpose, it is required to access CIFS from the local server machine. In this case, the above configuration is insufficient and it is required to execute the following commands:

```
iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 445 -j REDIRECT --to-port 1445
iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 139 -j REDIRECT --to-port 1139
iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 137 -j REDIRECT --to-port 1137
iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 138 -j REDIRECT --to-port 1138
```

OpenKM Configuration

*To configure it go to **Administration** > **Confguration** and change the following properties:*

Field / Property	*Type*	*Description*	*Value*
system.cifs.server	***Boolean***	*Property to enable or disable CIFS.*	*true*
system.cifs.authenticator.class	***String***	*Class used by CIFS to authenticate against OpenKM.*	*com.openkm.integration.cifs.authenticator.DatabaseCIFSAut* *Depending the on authentication adapter values should be us* *authenticar class:* *com.openkm.integration.cifs.authenticator.Datab* *com.openkm.integration.cifs.authenticator.LdapC*
system.cifs.debug	***Boolean***	*Property used to allow detailed log of CIFS functionality.*	*false*
system.cifs.domain.name	***String***	*Domain or workgroup where OpenKM server is installed.*	

system.cifs.hostname	***String***	*Hostname where OpenKM is installed.*	*0.0.0.0*
system.cifs.port.datagram	***Integer***	*Port number used by datagrams.*	*By default **1138***
system.cifs.port.name	***Integer***	*Port number used by CIFS.*	*By default **1137***
system.cifs.port.session	***Integer***	*Port number used by session.*	*By default **1139***
system.cifs.port.smb	***Integer***	*Port number used by SMB.*	*By default **1445***
system.cifs.shared.name	***String***	*Property to inidicate the name of the repository*	*By default the name is **repository***



After changing these properties OpenKM must be restarted.



*If it is not a fresh installation of OpenKM, it is important to reset user passwords from OpenKM administration Configuration (no mather if it is the same password) in order to generate **md4 version** needed for authentication.*

Windows client

*It is necessary to modify a **Local Security Policy**. To do it, go to **Controls Panel** → **Administrative Tools** → **Local Security Policy** → **Local Policies** → **Security Options**:*



*Option Network security: LAN Manager authentication Level and select: **Send LM & NTLM: Use NTLMv2 session security if negotiated.***

You can also do it by lauching the following command from command line:

REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "LmCompatibilityLevel" /t "REG_DWORD" /d "1" /f

Then to connect to the remote machine, go to Computer → Map Network Drive:

- *Drive: Any letter available*
- *Folder: \\server_ip\repository OR \\server_name\repository*
- *Connect using different credentials (chek it if the Windows user and password are different to OpenKM user).*

Then click on finish button.

If connect using different credentials was checked, a new window where introduce the corresponding credentials will appear.

You can also do it by lauching the following command from command line:

net use <driver letter>** **<servername>*<sharename>** **/user:<username>** **<password>** and as an example: **net use Z: \\server_ip\repository OR \\server_name\repository /user:okmAdmin admin_password***

If you get wrong password error, try to reset the pasword of the user from OpenKM administration Configuration.

Active Directory Authentication

In this case, it is considered that the system was integrated previously with Active directory. In this case it will be necessary to configure:

- *Active Directory*
- *OpenKM Server*
- *OpenKM configuration*
- *Windows client*

Active Directory

It is necessary to create an user that OpenKM, using a private key, will use to validate the credentials of the incoming CIFS request. To do it, the next steps must be followed:

1. *Create a Computer Object that will be the one corresponding to the OpenKM server.*
2. *Once created, modify its properties by enabling the property Trusted for delegation.*
    1. *Trusted for delegation. This step is important because, if not enabled, it will not be possible negotiate among client and server.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

3. *The server must authenticate against Active Directory before validating the tickets sent by the client.This authentication is performed using a file (keytab) that map the service (application server) with a domain user. This user must be used only and exclusively for this purpose because if this user is used to logon in another computer, this user will not be able to be used to logon the application server. Once created the user, its properties are modified to not allow the modification of the password and the non expiration of the account. Finally, select the Account tab and enable* ***Do not required Kerberos preauthentication option*** *in the Account Options selection. For this example, it is created a user call* ***servercifs****.*

a. *Select the Account tab and enable the Do not require Kerberos preauthentication option in the Account Options section.*

1. *Generate keytab file. This file is an encoded copy of the private key of an user in kerberos. It is used by the web application as an authentication proxy of the credentials sent by the user. The following command is executed:*

```
ktpass -princ cifs/tomcat.example.com@EXAMPLE.COM -mapuser servercifs@EXAMAPLE.COM -crypto a
    -ptype KRB5_NT_PRINCIPAL -pass * -out cifs.keytab -kvno 0
```

- *it is important to highlight :*

  1. *tomcat.example.com@EXAMPLE.COM: full name of the tomcat server and its domain.*
  2. *servercifs@EXAMAPLE.COM : user created previously.*
  3. *cifs.keytab: file where it is exported the private key that will be used by the Tomcat server.*

2. *A Service Principal Name (SPN) needs to be setup with CIFS and a server name: in example: tomcat.example.org where tomcat servlet container is running. This is used with domain user and its keytab is then used as a service credential:*

```
C:\> setspn -A cifs/tomcat.example.org servercifs

C:\> setspn -A cifs/tomcat.example.org servercifs@EXAMPLE.COM

# where tomcat.example.org is the machine where tomcat servlet container is running and
# servercifs is the user previously created.
```

> *Where tomcat.example.org is the machine where tomcat servlet container is running and user is the user previously created.*
>
> *Usually the previous command "ktpass" should create the service cifs/tomcat.example.org, when it's created will be raised an error indicating is yet present.*

OpenKM server in Active Directory configuration

The following steps must be folowed:

Introduce server in domain

The first thing to do is to include the application server inside the domain, if it was not done previously. To do it follow the guide:

http://cerowarnings.blogspot.com.es/2011/11/how-to-linux-en-dominio-windows.html

One thing to take into account is to add the following configuration to the file /etc/host due to previously known problems:

```
# ldap in /etc/hosts
192.168.1.110 example.com
192.168.1.110 Schema.Configuration.example.com
192.168.1.110 Configuration.example.com
192.168.1.110 DomainDnsZones.example.com
192.168.1.110 ForestDnsZones.example.com
```

Redirect the required ports

If the system could not be started with root permissions the ports commented bellow, must be over 1000 (the default system ports are 138,137,139,445), so it is necessary to redirect the original ports to the new ones. To do this it is necessary to execute the following commands as root or using sudo:

```
$ modprobe iptable_nat
$ iptables -F
$ iptables -t nat -F
$ iptables -P INPUT ACCEPT
$ iptables -P FORWARD ACCEPT
$ iptables -P OUTPUT ACCEPT
$ iptables -t nat -A PREROUTING -p tcp --dport 445 -j DNAT --to-destination :1445
$ iptables -t nat -A PREROUTING -p tcp --dport 139 -j DNAT --to-destination :1139
$ iptables -t nat -A PREROUTING -p tcp --dport 137:139 -j DNAT --to-destination :1137-1139
$ iptables -t nat -A PREROUTING -p udp --dport 137:139 -j DNAT --to-destination :1137-1139
```



Sometimes for testing purpose, it is required to access CIFS from the local server machine. In this case, the above configuration is insufficient and it is required to execute the following commands:

```
$ iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 445 -j REDIRECT --to-port 1445
$ iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 139 -j REDIRECT --to-port 1139
$ iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 137 -j REDIRECT --to-port 1137
$ iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 138 -j REDIRECT --to-port 1138
```

kerberos configuration file

Install kerberos client:

```
$ apt-get krb5-user krb5-config
```

Before starting the application Kerberos must be configured properly to validate tickets correctly. In Linux systems this file is located at "/etc/kbr5.conf".

In this file Domain, KDC server and allowed algorithms must be specified. The most importans algorithms are ***arcfour-hmac-md5*** *and* ***rc4-hmac*** *which Microsoft Active Directory offers.*



Preservate almost krb5.con default configuration and make only the suggested changed shown below.

```
[libdefaults]
default_realm = EXAMPLE.COM
#these are mandatory for kerberos.
permitted_enctypes = aes128-cts rc4-hmac
default_tgs_enctypes = aes128-cts rc4-hmac
default_tkt_enctypes = aes128-cts rc4-hmac
[realms]
EXAMPLE.COM = {
  kdc = SERVER.example.com
  admin_server = SERVER.example.com
 default_domain = EXAMPLE.COM
}
[domain_realm]
 example.com = EXAMPLE.COM
 .example.com = EXAMPLE.COM
```

cifs.login.config File

Create a configuration file needed to enable java login with content:

```
JLANServerCIFS{
com.sun.security.auth.module.Krb5LoginModule required
storeKey=true
useKeyTab=true
keyTab="/path/to/cifs.keytab"
principal="cifs/tomcat.example.com";
};
```

As can be observed:

- *JLANServerCIFS correspond with the property* ***system.cifs.ldap.login.entry***

- *Property principal correspind to the property:* ***system.cifs.ldap.principal***
- *Property keytab it is the secure place where the previously created* ***.keytab file*** *that contains the key to access Active directory has been copied.*

> *The file is usually found in the folder* ***JRE/lib/security.*** *However it could be placed in any place with a minimum security.*

Enable Java login configuration.

To enable java login, it is necessary to change the configuration of the Java security file that it is usually place at: JRE/lib/security/java.security and add the following line:

```
login.config.url.1= file:${java.home}/lib/security/cifs.login.config where cifs.login.config is the file created in the prevous section.
```

> *The default value* ***${java.home}/lib/security*** *could be change to any other route where the previous created file jlan.login.config has been stored.*

OpenKM Configuration in Active Directory Configuration

To configure it : go to ***Aministration*** *>* ***Confguration*** *and change the following properties :*

Field / Property	*Type*	*Description*	*Value*
system.cifs.server	***Boolean***	*Property to active or deactive CIFS*	*true*
system.cifs.authenticator.class	***String***	*Class used by CIFS to authenticate against OpenKM*	*com.openkm.integration.cifs.authenticator.LdapCIFSAuthe* *Depending the on authentication adapter values should be authenticar class:* *com.openkm.integration.cifs.authenticator.Dat* *com.openkm.integration.cifs.authenticator.Lda*
system.cifs.debug	***Boolean***	*Property used to allow detailed log of CIFS*	*false*

		functionality	
system.cifs.domain.name	***String***	*Domain or work group where OpenKM server is installed*	*EXAMPLE*
system.cifs.hostname	***String***	*Hostname where OpenKM is installed*	*0.0.0.0*
system.cifs.ldap.kdc	***String***	*Name of the Active directory server*	*server.example.com*
system.cifs.ldap.principal	***String***	*Service created previously in Active Directory*	*cifs/tomcat.example.com*
system.cifs.ldap.password	***String***	*Password of the user created for authentication purpose*	*In the example the user was servercifs*
system.cifs.ldap.realm	***String***	*Name of the domain where Active directory is installed*	*EXAMPLE.COM*
system.cifs.ldap.login.entry	***String***	*Kerberos configuration name inside configurations file*	*JLANServerCIFS*
		Port number	

system.cifs.port.datagram	***Integer***	*used by datagrams*	*By default* ***1138***
system.cifs.port.name	***Integer***	*Port number used by cifs*	*By default* ***1137***
system.cifs.port.session	***Integer***	*Port number used by session*	*By default* ***1139***
system.cifs.port.smb	***Integer***	*Port number used by smb*	*By default* ***1445***
system.cifs.shared.name	***String***	*Property to inidicate the name of the repository*	*By default the name is* ***repository***



After changing this properties, OpenKM must be restarted.

Windows client in Active Directory configuration

It is necessary to modify a ***Local Security Policy****. To do it, go to* ***Controls Panel*** *→* ***Administrative Tools*** *→* ***Local Security Policy*** *→* ***Local Policies*** *→* ***Security Options****:*



Option Network security: LAN Manager authentication Level and select: ***Send LM & NTLM: Use NTLMv2 session security if negotiated.***



You can also do it by lauching the following command from command line:

REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "LmCompatibilityLevel" /t "REG_DWORD" /d "1" /f

Then to connect to the remote machine, go to Computer → Map Network Drive:

- *Drive: Any letter available*
- *Folder: \\server_name\repository*
- *Connect using different credentials (chek it if the Windows user and passwords are different to OpenKM user).*

Then click on finish button.



If connect using different credentials was checked, a new window where introduce the corresponding credentials

will appear.

You can also do it by lauching the following command from command line:

net use <driver letter> *****<servername>********<sharename>* ***/user:<username>*** ***<password>*** *and as an example:* ***net use Z: \\server_name\repository /user:okmAdmin admin_password***

Cache configuration

OpenKM uses several caches to enhance performance. Some of these caches are directly used by OpenKM and others indirectly, for example those used by Hibernate. All these caches have default values which make them useful but these default values are not always good for every installation. The cache engine used is the very popular EHCache, which is widely used and very configurable. You can see the default values at ***$TOMCAT_HOME/webapps/openkm/WEB-INF/classes/ehcache.xml****.*



Don't modify this file because all changes will be lost when OpenKM is restarted or updated.

If you want to tweak this default configuration you have to duplicate the provided configuration file into ***$TOMCAT_HOME/ehcache.xml*** *and make the proper changes to match your requirements. This is a sample definition:*

```
<cache name="com.openkm.cache.ldapPrincipalAdapter.general"
    maxElementsInMemory="1000"
    timeToIdleSeconds="300"
    timeToLiveSeconds="0"
    overflowToDisk="true"
    eternal="false" />
```

The most important values are:

- ***timeToLive****: The maximum number of seconds an element can exist in the cache regardless of use. The element expires at this limit and will no longer be returned from the cache. The default value is 0, which means no TTL eviction takes place (infinite lifetime).*
- ***timeToIdle****: The maximum number of seconds an element can exist in the cache without being accessed. The element expires at this limit and will no longer be returned from the cache. The default value is 0, which means no TTI eviction takes place (infinite lifetime).*



Be careful with ***eternal****, because it sets whether elements are eternal. In case of true, timeouts are ignored and the element never expires.*

So, in this case the cache called "com.openkm.cache.ldapPrincipalAdapter.general" will have a maximum of 1000 elements in memory and in case of overflow will be written to disk. These elements will be removed from this cache if are unused for 5 minutes. Otherwise, will remain forever. If you want to force a refresh, you have to change the ***timeToLiveSeconds*** *to 3600 and will be refreshed every hour.*

You can check these caches status by going to ***Administration*** *>* ***Utilities*** *>* ***Cache stats****. In this part you can enable and see the cache usage stats and even reset a give one. More info at Cache stats.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring digital email signature

A digital signature attached to an email message increases security by providing information to the receiver that you are the right person and no other person who signed the contents of the email message. Your digital signature, which includes your certificate and the public key, is part of your digital ID. This digital ID identifies your unique digital mark and signals the recipient that the content hasn't been altered in transit.

Secure/Multipurpose Internet Mail Extensions (S/MIME) is a method of exchanging emails securely. With the S/MIME technology, a digital signature can be appended to emails to ensure non-repudiation and data integrity. OpenKM, since v7.1.14, can sign email messages using S/MIME.

These are the dynamic configuration properties:

Field / Property	*Type*	*Description*
mail.smime.certificate.file	***String***	*The location of the S/MIME certificate.*
mail.smime.certificate.password	***String***	*The password to access the S/MIME certificate.*

Once these configuration properties are set, the emails from OpenKM will be signed.

More info

- *Getting an SMIME certificate*
- *Wikipedia S/MIME*



Self-signed certificates are not recommended because they won't be automatically validated at the destination. You can find more information at:

- *Can I create my S/MIME certificate for email encryption?*
- *Create Self-Signed S/MIME Certificates*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring anonymous user

Behind the anonymous user, there's a standard user named "public," which autologin to the application using a unique URL.

*This **feature must be enabled from the user profile** with the option at **Properties** > **Document Properties** > "**Share Preview**."*

Follow these steps to configure:

- ***Create a user** named "**public**."*
- ***Create a role** named "**ROLE_PUBLIC**".*
- ***Assign** the role named "**ROLE_USER**" to this user.*
- ***Assign** the role named "**ROLE_PUBLIC**" to this user.*
- *Log in with the user "**public**" in the standard UI to check which nodes and options have access (**this is mandatory to create the trash folder of the user, etc.**).*

*The "**public**" user **must have READ access** to the main root nodes (okm:root, okm: templates, etc.).*

***Only the share preview option will allow only the document node type** with "**ROLE_PUBLIC**" **read access**.*

Optional steps:

- *Create a profile for the "public" user. Disable all the features you consider should not have access.*

Finally, share the autologin public URL, which will be something like http://yourhost:8080/openkm/public/

In case you want to logout the public user, please use a URL like http://yourhost:8080/openkm/kcenter/#/logout

Other usages:

- *Share specific document link http://yourhost:8080/openkm/public/uuid (where the uuid is the UUID of a node)*
- *Share specific document preview link http://yourhost:8080/openkm/public/preview/uuid (where the uuid is the UUID of a document)*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Prometheus configuration

Prometheus is a Open Source monitorization system which some interesting features for event monitoring and alerting. You can find a more detailed description at https://prometheus.io/.

This endpoint is already included in OpenKM since v8.1.10. To enable it you need to include these two lines at ***openkm.properties*** *configuration file:*

```
management.endpoints.web.exposure.include=info,health,prometheus
```

Once modified you need to restart OpenKM and the endpoint http://localhost:8080/openkm/actuator/prometheus will be available. This endpoint is protected and will be only accesible for users with the ROLE_ACTUATOR or ROLE_ADMIN roles.

Setup validation

In order to verify the access is valid, open a terminal and execute this command:

```
$ http -v --auth okmAdmin:admin http://localhost:8080/openkm/actuator/prometheus
$ curl -v --user okmAdmin:admin http://localhost:8080/openkm/actuator/prometheus
```

If the user and password matches, and the user has the right ROLE_ACTUATOR role the output will be something like:

```
# HELP tomcat_global_request_max_seconds
# TYPE tomcat_global_request_max_seconds gauge
tomcat_global_request_max_seconds{name="http-nio-0.0.0.0-8080",} 0.449
tomcat_global_request_max_seconds{name="ajp-nio-127.0.0.1-8009",} 0.0
# HELP jvm_threads_peak_threads The peak live thread count since the Java virtual machine started or peak was reset
# TYPE jvm_threads_peak_threads gauge
jvm_threads_peak_threads 128.0
# HELP system_cpu_usage The "recent cpu usage" for the whole system
# TYPE system_cpu_usage gauge
system_cpu_usage 0.04721888755502201
```

Running Prometheus with Docker

The easiest way to get Prometheus running is using the Docker image, so we need to get it:

```
$ docker pull prom/prometheus
```

Once we have the image we need to create the ***prometheus.yml*** *configuration file:*

```
global:
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
    - targets: ['127.0.0.1:9090']

  - job_name: 'spring-actuator'
    metrics_path: '/openkm/actuator/prometheus'
    scrape_interval: 5s
    static_configs:
    - targets: ['172.18.0.1:8080']
    basic_auth:
      username: 'okmAdmin'
      password: 'admin'
```

Let's create the appropiate Docker Compose file:

```
version: '3.2'

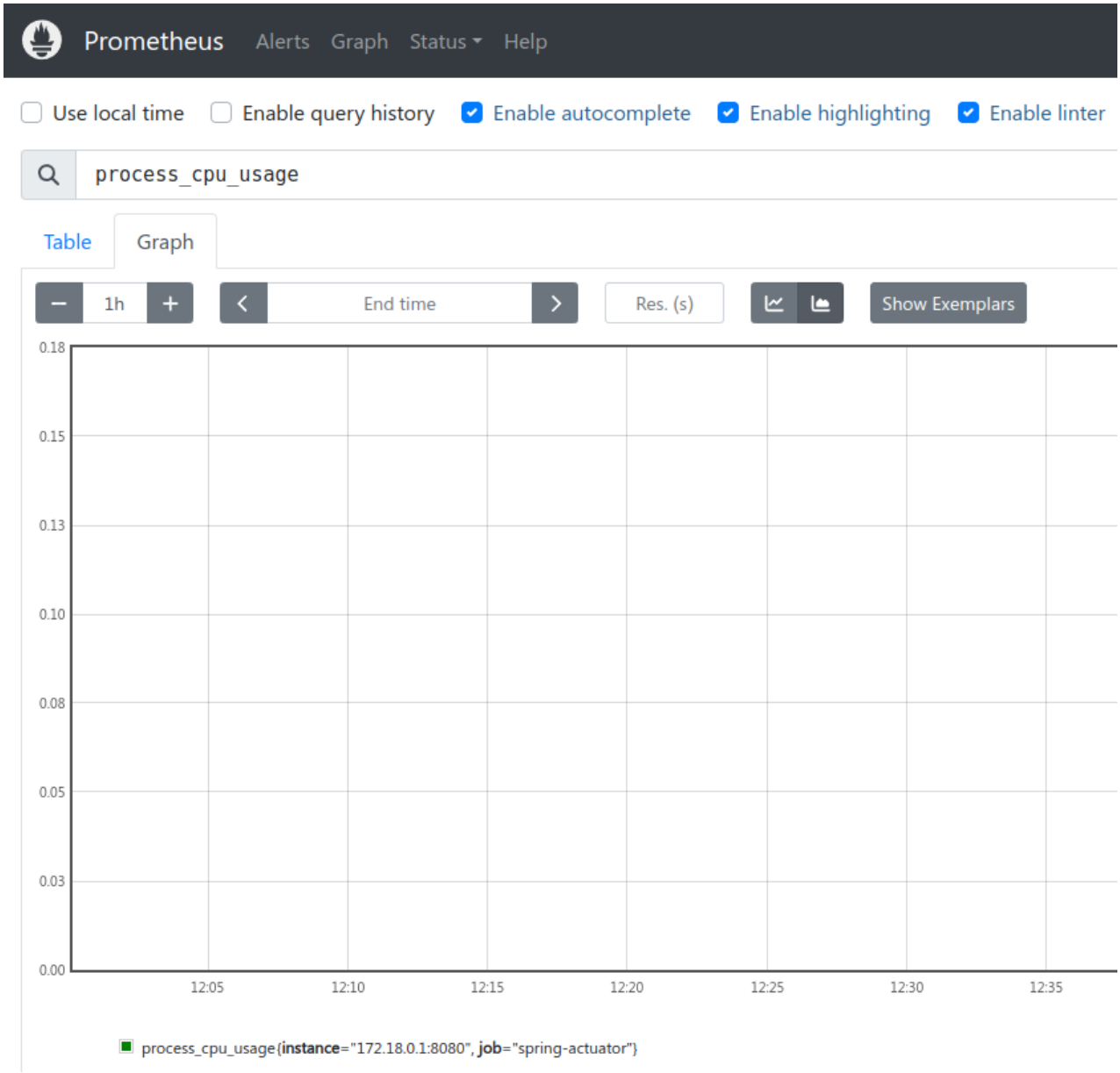
services:

  prometheus:
    image: prom/prometheus
    container_name: prometheus
    hostname: prometheus
    command:
      - "--config.file=/etc/prometheus/prometheus.yml"
    ports:
      - 9090:9090
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
```

And you can start it by:

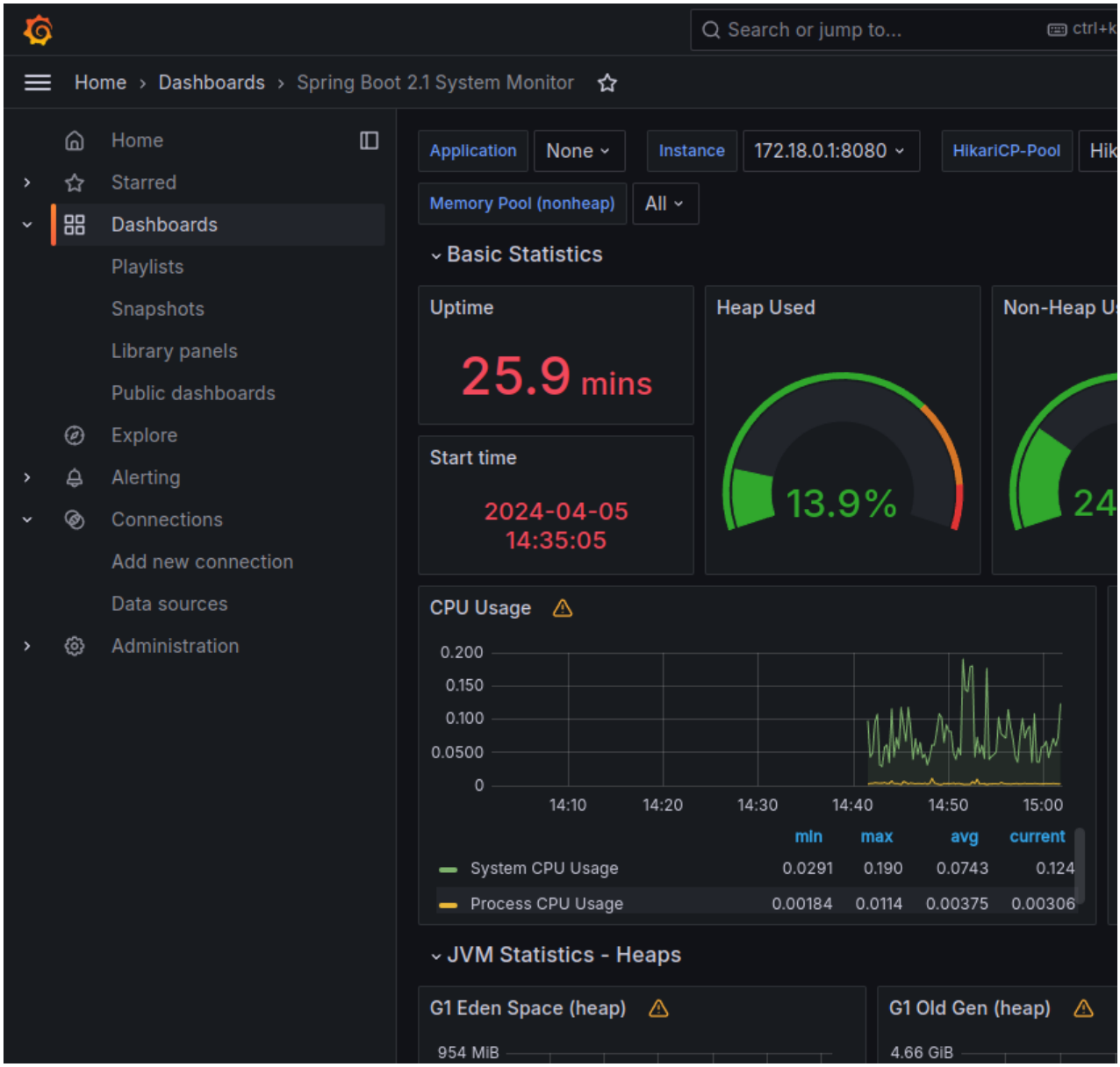
```
$ docker-compose up -d
```

Once started, Prometheus will be accesible at http://localhost:9090/ and you can see a graph of application CPU usage filtering by "process_cpu_usage":



Grafana will be available at http://localhost:3000/ and to login use "admin" as default user and password. Go to "Data sources" and add a Prometheus datasource using the URL http://172.18.0.1:9090. To create a sample dashboard, go to Dashboard and click on "Create Dashboard" button. After that click on "Import dashboard" and choose the "spring boot template" which has the id 11378. Use the previously created Prometheus datasource.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



More info at Prometheus and Grafana websites.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Troubleshooting

Here you can find some common problems and how to solve them. The context is grouped into several sections:

- *Server configuration*
- *Client configuration*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Server configuration

The packet for the query is too large (MySQL)

If you get an error like this:

> *The package for the query is too large (1708726 > 1048576).*
>
> *You can change this value on the server by setting the 'max_allowed_packet' variable.*

You need to modify your MySQL server configuration file and increase the value of the ***max_allowed_packet*** *property. Don't forget to restart MySQL after any change in this file. In Debian-based distros like Ubuntu, this configuration file is located at /etc/mysql/my.cnf.*

Not able to connect from another computer in the network

> *Ensure you have access from the other computers to the server.*
>
> *Ensure your server firewall allows remote connection to the destination port.*

*If you expose the tomcat port 8080, ensure it is binding your ip, the "**address**" parameter value should be "**0.0.0.0**" at $TOMCAT_HOME/conf/server.xml:*

```
<Connector port="8080" address="0.0.0.0" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

> *After any modification in server.xml takes, the effect must restart the tomcat.*

Error downloading document as PDF

This error usually is caused by a missing LibreOffice application installed or because of incomplete package installation.

IMAP SSL certificate error

First, obtain the certificate with the openssl command, and after this, import the certificate with the following:

```
$ keytool -import -alias mail.yourcompany.com -keystore //usr/lib/jvm/java-11-openjdk-amd64/lib/security/cacerts -file imapd.pem
```

> *The application must be restarted to refresh the SSL certificates—more information at Connecting+to+SSL+services.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OpenKM server has no direct Internet connection.

*If you need a proxy to access the Internet, edit your "**/etc/bash.bashrc**" file as root and put these lines at the end of your "**/etc/bash.bashrc**" file:*

```
export http_proxy=http://username:password@proxyserver.net:port/
export ftp_proxy=http://username:password@proxyserver.net:port/
```

You can omit the username: and password if your proxy server has no password.

Problems with character encodings

If your language has non-ASCII characters and is displayed as garbage, you have a problem with encodings.

*Check the $TOMCAT/bin/setenv.sh or $TOMCAT/bin/setenv.bat file. You should have set the JVM parameter "**-Dfile.encoding=utf-8**".*

> *The application must be restarted.*

PermGen errors

> *Since Java 8, there are no such **PermSize** and **MaxPermSize** options.*

If you see this error:

```
java.lang.OutOfMemoryError: PermGen space
```

*Edit your "**$TOMCAT_HOME/bin/setenv.sh**" (or "**$TOMCAT_HOME/bin/setenv.bat**" if using Windows) and increase the value of the JVM parameter "-XX:MaxPermSize":*

```
JAVA_OPTS="-Xms256m -Xmx2048m -XX:PermSize=128m -XX:MaxPermSize=512m"
```

See also Presenting the Permanent Generation.

LDAP Size limit exceeded

It is caused because bindDN has a size limit for each ldap query.

In OpenLDAP, The solution is to increase the limit of bindDN, for example (to change the size limit to the user used by openkm ldap connection)

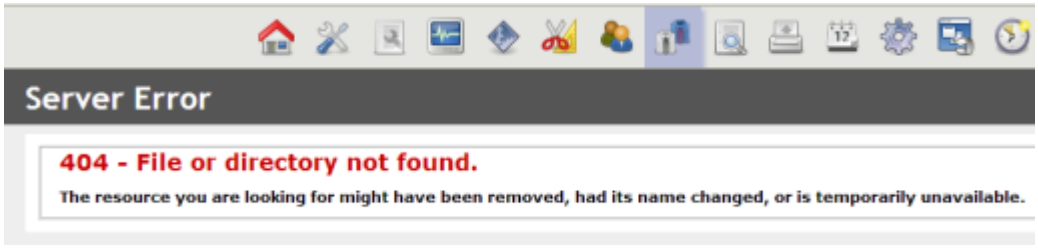
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
limits dn.exact="cn=reader,ou=accounts,ou=admins,dc=example,dc=com" size=100000
```

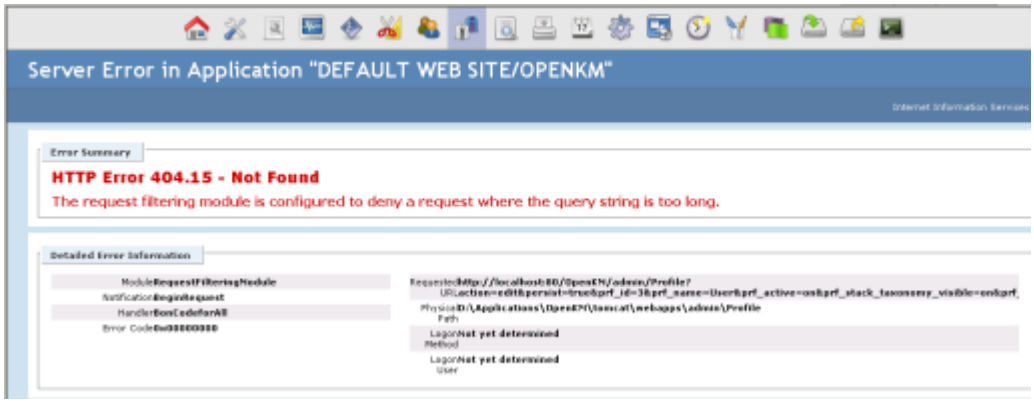
See also OpenLDAP Limits.

Behind the IIS proxy, you get error 404 or 404.15

Using IIS as a web server by default, it has a limit for the query string sent to the server. If the limit is passed, then you get some error screen on the client side:

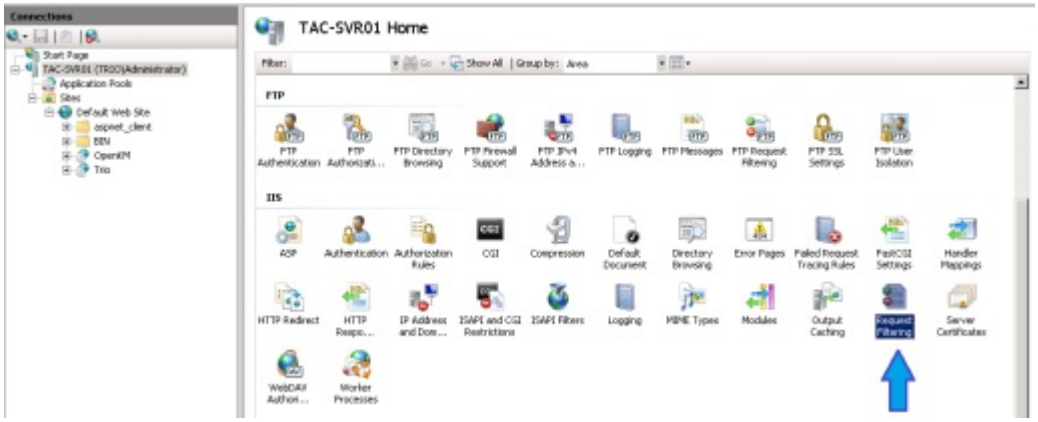


On the server side, we get a more detailed error:



To solve this issue, we need to increase the maximum size of the query string on the web server:

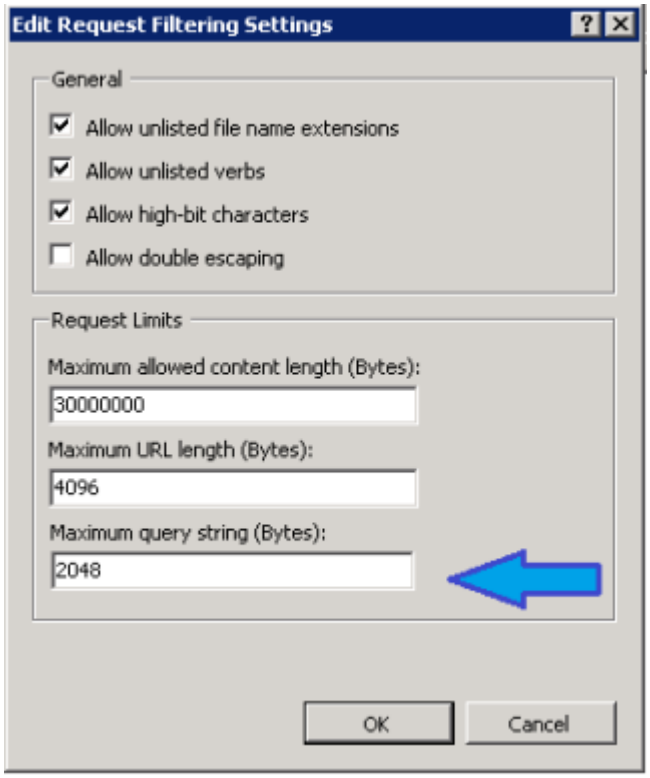
- ***Open Internet Information Services (IIS) Manager***.
- *In the* ***Connections pane****, go to the* ***connection > site > application*** *or directory for which you want to modify your request filtering settings.*
- *In the* ***Home pane, double-click Request Filtering***.



*Click **Edit Feature Settings...** in the Actions pane.*



Change the maximum query string length:



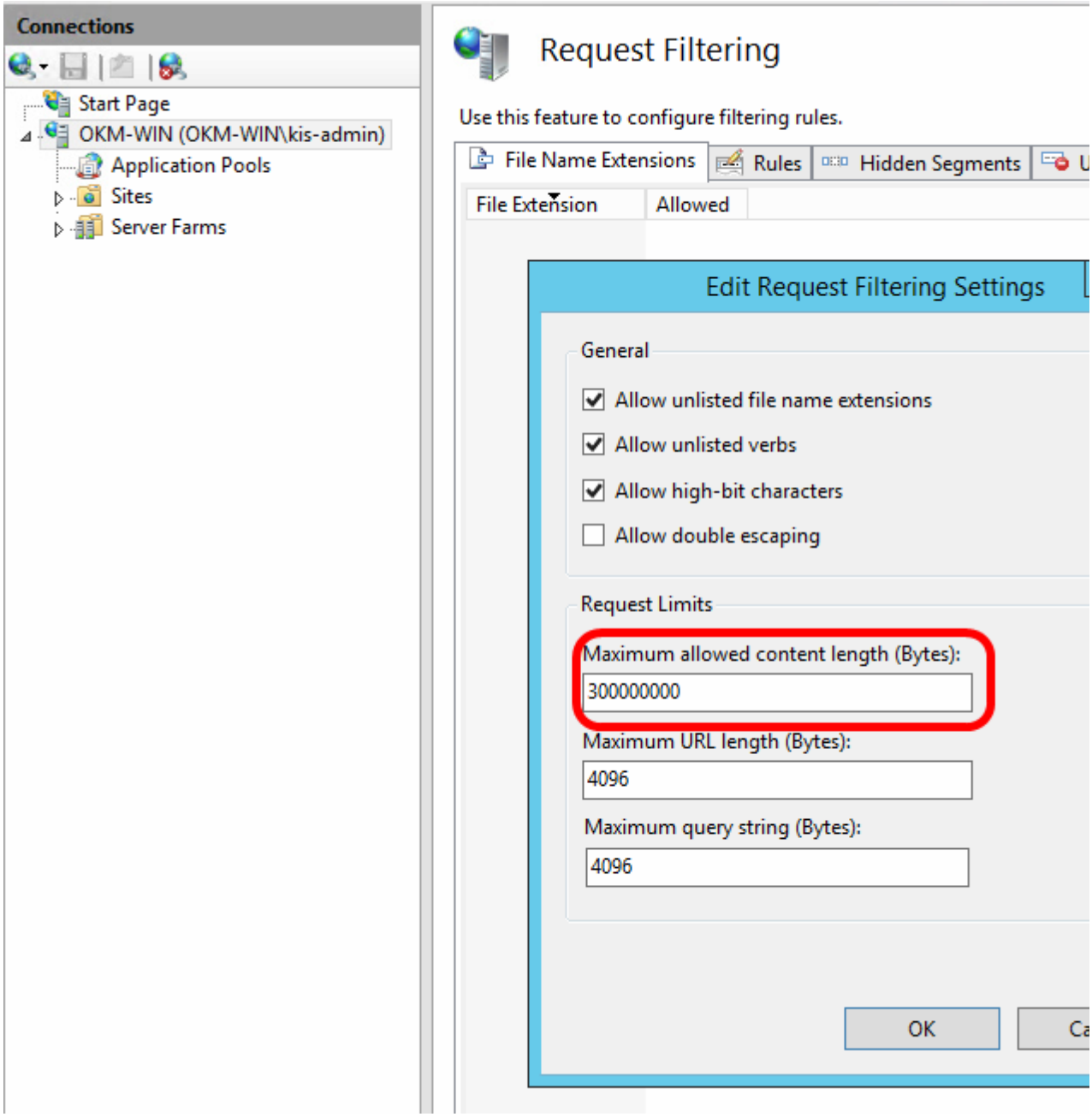
More information is at the [OpenKM forum](#).

IIS gets IOException: Internal Server Error while uploading file or HTTP 413 Request Entity Too Large Error

This error may appear when uploading a big document to OpenKM.

Solution one

Change the "Maximum allowed content lenght (Bytes)" from your IIS configuration parameters:



Solution two

- ***Go to webapps*** *folder*

- ***Edit** the **web.config** file and **add the following under <configuration> where "2097151"** is the maximum allowed in IIS:*

```
<system.web>
  <httpRuntime maxRequestLength="2097151" />
</system.web>
```

If there is no web.config, then create one from a text file and add the following :

```
<configuration>
  <system.web>
    <httpRuntime maxRequestLength="2097151" />
  </system.web>
</configuration
```

WARN net.sf.ehcache.hibernate.strategy.AbstractReadWriteEhcacheAccessStrategy

If you see this message:

```
A soft-locked cache entry was expired by the underlying Ehcache.
If this happens regularly you should consider increasing the cache timeouts and/or capacity limit
```

Are you experiencing any performance or usability problems due to this warning? If not, then it seems like you should treat this warning as a warning.

*Otherwise, it looks like the solution is to increase the size of **maxElementsInMemory** to account for the difference in what is cached versus what is being used.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

RedHat and Oracle Linux issues

OpenKM not starting as a service automatically

*If in your OS the OpenKM Service does not start automatically. This maybe because the security system (**SELinux**) blocks services that are not recognized as native.*

To enable it, we must perform the following steps:

- ***Create the openkm.service file within the path (/usr/lib/systemd/system/openkm. Service)***

```
$ nano /usr/lib/systemd/system/openkm. Service
[Unit]
Description=OpenKM daemon
[Service]
Type=simple
ExecStart=/home/openkm/tomcat-9.0.76/bin/startup.sh
RemainAfterExit=yes
[Install]
WantedBy=multi-user.target
```

***** *ExecStart contains a sample value; the correct path is where the* ***startup.sh*** *file location.*

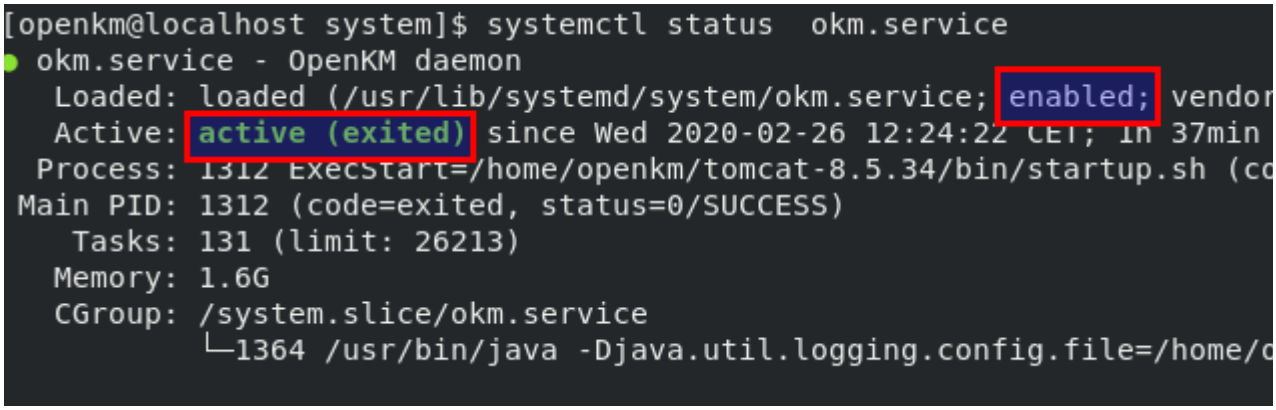
- ***Refresh, enable the Service, and activate the Service.***

```
$ systemctl daemon-reload
$ systemctl enable openkm.service
$ systemctl start openkm.service
```

With the following command, we can know the status of the Service:

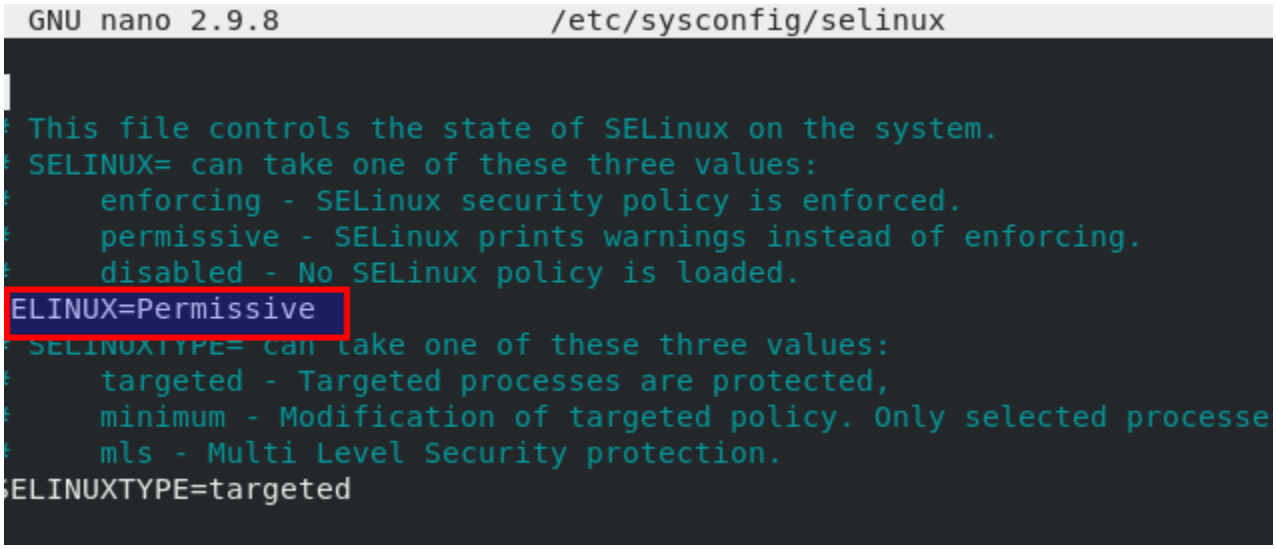
```
$ systemctl status openkm.service
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



- ***Modify the SElinux configuration file:***

```
$ nano /etc/sysconfig/selinux
```



- ***Finally, restart the server***

```
$ reboot
```

ImageMagick

If your OS does not come with the ImageMagick package in the repositories, can install a fork called GraphicsMagick from the EPEL repository:

```
$ dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
$ dnf install GraphicsMagick
```

Keep in mind that there is no "convert" utility but a general "gm" executable that can be used this way:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
$ gm convert <params>
```

More info at GraphicsMagick converts documentation.

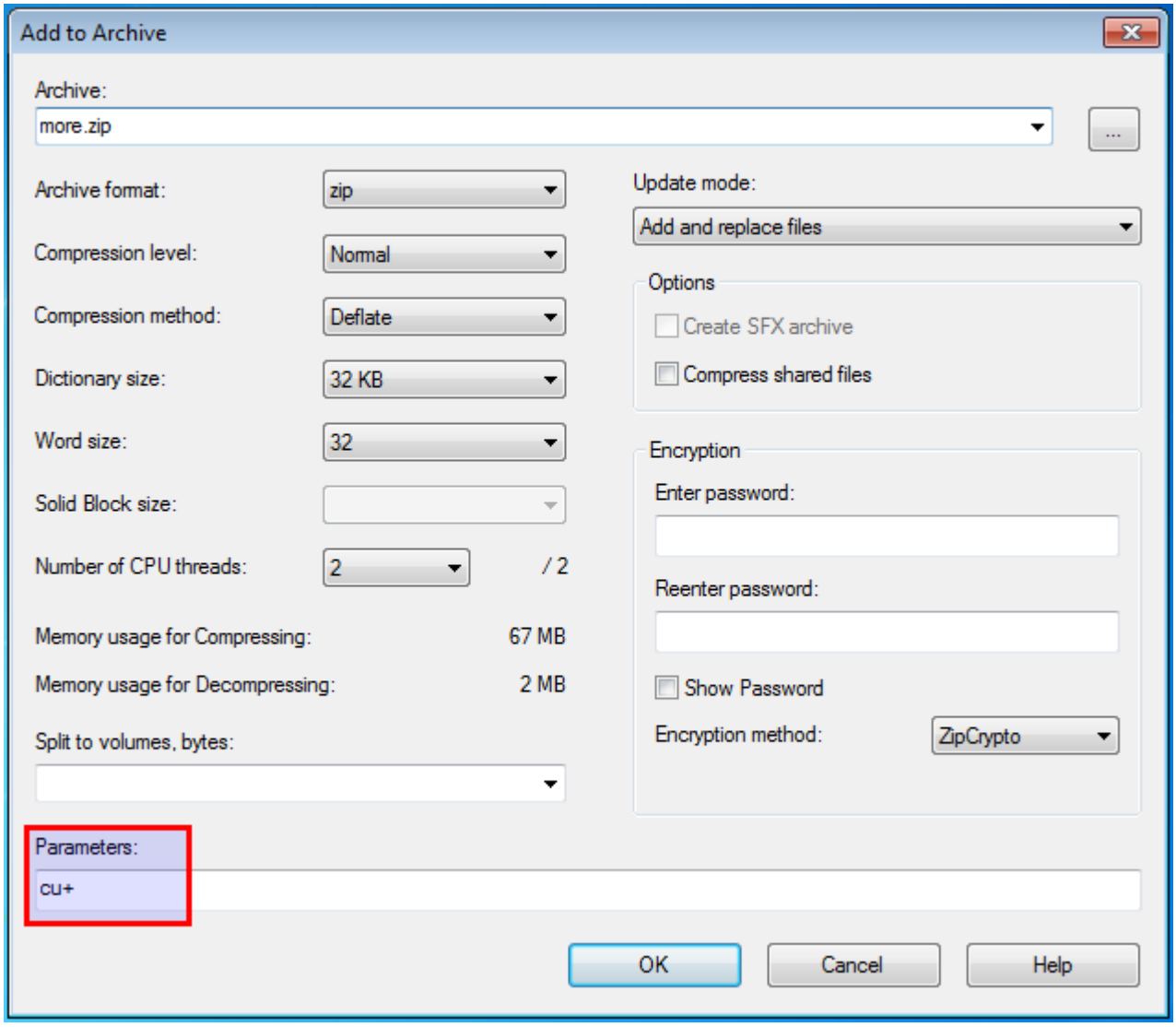
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Client configuration

ZIP filenames with wrong charset

When you download an OpenKM as ZIP maybe the included files have a wrong charset. This only happens with non-latin languages like Greek, Chinese or Arabic. This issue is corrected since OpenKM 4.1 and exported ZIP archives should be opened without problems from WinZip 14 or higher. But UNICODE support in ZIP is a little tricky and may cause problems. To ensure you won't have problems, try using 7-Zip which seems to handle these UNICODE filenames.

The other problem you may have is when you want to import a ZIP into OpenKM. In this case you can also get wrong filenames. Filenames should be encoded in UNICODE, and to ensure it you can force 7-Zip to use this encoding when creating a ZIP:



If you are on Windows you can download 7-Zip from http://www.7-zip.org/. You can also install the port of the 7-Zip compression tool for Linux:

```
$ sudo aptitude install p7zip-full p7zip
```

Microsoft Office Addin disappears after Office restart

Could be a problem with default template (normal.dot). To solve try it:

- *Open normal.dot*
- *Go to menu tools / personalize / Tool bars*
- *Open OpenKM toolbar*
- *Save normal.dot*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Best practices for reporting issues

To quickly help you in error identification and resolution, please try to follow the guidelines below:

- *Do a **detailed description** of the issue, that will help us understand the problem you have.*
- *When possible **make screenshots of all the steps you followed**. The idea is to be able to reproduce the error or at least understand better the context where has occurred.*
- *Videos are also welcome.*
- *Share the stack trace error from openkm.log file*

> *Because the support website has restrictions on document size, share with us the download link with online file-sharing tools like:*
>
> - *https://transfer.sh/*
> - *https://filepush.co/*
> - *https://filetransfer.io/*
> - *https://justbeamit.com/*

Getting the stack trace

Linux

*The **openkm.log** file is in the **$TOMCAT/logs folder**. For example, in Linux, a common location might be "**/home/openkm/tomcat-9.0.76/logs**".*

To watch how openkm.log changes in real time you can use the command line:

```
/home/openkm/tomcat-9.0.76/logs:$ tail -f openkm.log
```

To show the last 500 lines of the openkm.log file you can use the command line:

```
/home/openkm/tomcat-9.0.76/logs:$ tail -n 500 openkm.log
```

Windows

*The **openkm.log** file is in the **$TOMCAT/logs folder**. For example, in Linux a common location might be "**c:\tomcat-9.0.76\logs**".*

To watch how openkm.log changes in real time you can a windows tools like mtail.

> *You have a copy of mtail tools into **c:\tomcat-9.0.76\extras***

Sample of stack trace capture



When you capture the stack trace error it is a good practice to include some extra lines before and after the error happens.

```
2019-08-04 12:05:00,020 [Task Scheduler 18] [] INFO  c.openkm.plugin.cron.BaseCronPlugin - *** Process Pending
2019-08-04 12:05:00,032 [Task Scheduler 19] [] INFO  c.openkm.plugin.cron.BaseCronPlugin - *** Text Extractor W
2019-08-04 12:09:36,471 [http-nio-0.0.0.0-8080-exec-6] [okmAdmin] INFO  com.openkm.util.LocalEntityResolver - r
2019-08-04 12:09:36,603 [http-nio-0.0.0.0-8080-exec-6] [okmAdmin] ERROR com.openkm.util.FormUtils - Element
2019-08-04 12:09:36,612 [http-nio-0.0.0.0-8080-exec-6] [okmAdmin] ERROR c.o.s.admin.PropertyGroupsServlet - L
com.openkm.core.ParseException: Element type "hpanel" must be declared.
 at com.openkm.util.FormUtils.parsePropertyGroupsDefinition(FormUtils.java:308)
 at com.openkm.module.db.DbRepositoryModule.registerPropertyGroups(DbRepositoryModule.java:855)
 at com.openkm.module.db.DbRepositoryModule.registerPropertyGroups(DbRepositoryModule.java:844)
 at com.openkm.servlet.admin.PropertyGroupsServlet.doPost(PropertyGroupsServlet.java:220)
 at javax.servlet.http.HttpServlet.service(HttpServlet.java:661)
 at javax.servlet.http.HttpServlet.service(HttpServlet.java:742)
 at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:231)
 at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
 at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
 at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193)
 at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
 at com.openkm.servlet.UserLoggingFilter.doFilter(UserLoggingFilter.java:33)
 at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193)
 at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
 at org.springframework.boot.web.filter.ApplicationContextHeaderFilter.doFilterInternal(ApplicationContextHeaderFil
 at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:107)
 at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193)
 at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
 at org.springframework.boot.actuate.trace.WebRequestTraceFilter.doFilterInternal(WebRequestTraceFilter.java:11
 at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:107)
 at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193)
 at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
 at java.lang.Thread.run(Thread.java:748)
2019-08-04 12:10:00,000 [Task Scheduler 18] [] INFO  c.openkm.plugin.cron.BaseCronPlugin - *** Process Pending
2019-08-04 12:10:00,000 [Task Scheduler 6] [] INFO  c.openkm.plugin.cron.BaseCronPlugin - *** Antivirus Checker
2019-08-04 12:10:00,000 [Task Scheduler 25] [] INFO  c.openkm.plugin.cron.BaseCronPlugin - *** Text Extractor W
2019-08-04 12:10:00,014 [Task Scheduler 6] [] INFO  c.openkm.plugin.cron.BaseCronPlugin - *** Antivirus Checker
```

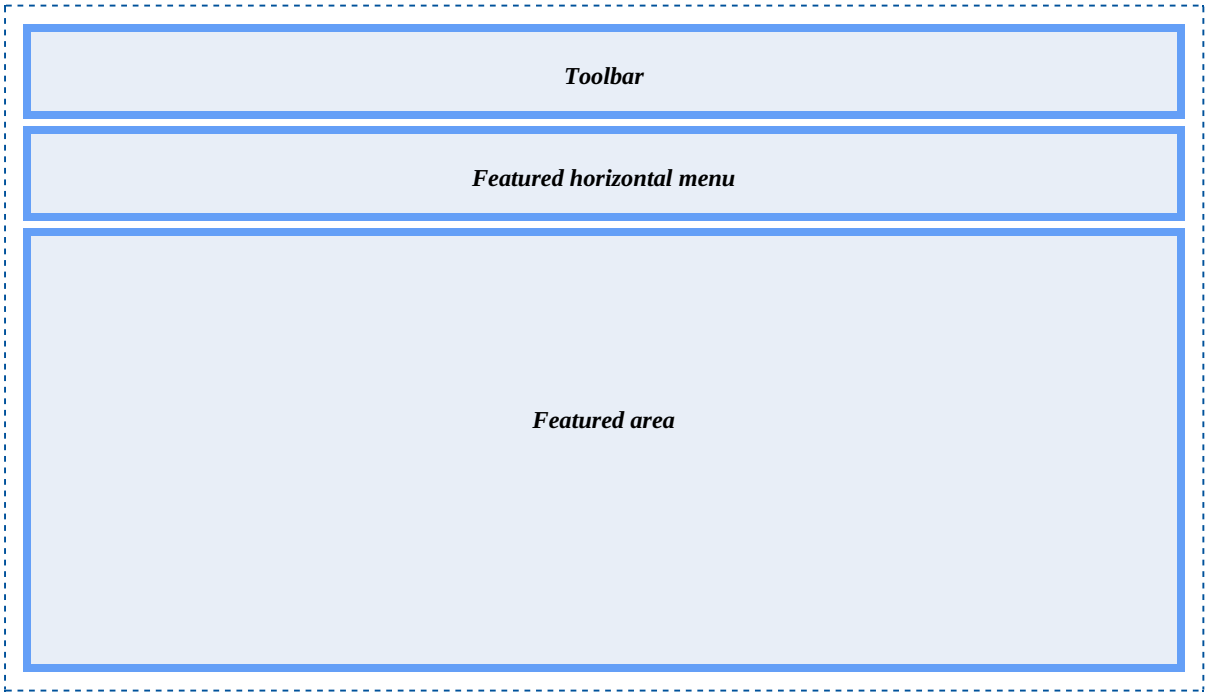
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Administration guide

OpenKM provides easy access to the application configuration using the "Administration Tab" located in the administration system area. Relevant and useful information can be accessed to assist the administrator in controlling and configuring the OpenKM repository.

> *Only users with **ROLE_ADMIN** can access the administration area. A new tab, **Administration**, will appear to these users at the top right corner of the system toolbar.*

Administration layout:



OpenKM administration provides access to:

Icon	*Name*	*Description*
	***Home***	*Administration landing page. Here is shown:* • *Application version.* • *Installation UUID.*
	***Configuration parameters***	*Almost all configuration parameters are set here.* *Changes made here affect the entire application.*

	***Mime types***	*All registered document MIME types that OpenKM understands. You can modify, remove, or add a new one.*
	***Statistics***	*Shows information about **the repository size**. Can also check the evolution of the **Pending task queue** and **Pending extraction queue**.*
	***Metadata***	*Creates, modifies, or removes metadata definitions.*
	***Authentication***	*Creates, modifies, or removes users, roles, and tenants.*  *When the application is configured to connect to LDAP, these values can only be changed on the LDAP server because OpenKM connects to LDAP only in read-only mode.*
	***Profile***	*A profile is a specific set of configurations for the user interface.*
	***Relations***	*Creates, modifies, or removes node relation type.*
	***User interface styles definition***	*Modifies the CSS used by the user interface.*
	***Reports***	*Adds, removes, or executes application reports.*
	***Activity log***	*User actions are registered in the OpenKM activity log.*  *All activity log data is stored in **OKM_ACTIVITY** table.*
	***Automation***	*Sets automatic tasks based on application events like DOCUMENT_CREATION, among others.*
		Sets text or image stamp definitions.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	***Stamp***	*This option is hidden when the extension is not enabled. Read about it at Enable extensions.*
	***Crontab***	*Adds or removes scheduled crontab tasks. The application uses these tasks to perform actions like indexing documents, antivirus analysis, among others.*
	***OCR templates***	*Sets OCR template definitions to automate the document's data capture.*
	***Languages***	*Add, modify, or remove languages and manage translations.*
	***Utilities***	*A collection of advanced utilities:* • *Admin permissions.* • *Cache Stats.* • *Check e-mail.* • *Check text extraction.* • *Clean conversion cache.* • *Create a support zip.* • *Database import.* • *Database query.* • *Download tokens.* • *Execute JAR.* • *Hibernate stats.* • *LDAP search.* • *List indexes.* • *LogCat.* • *Missing text extractions.* • *Orphan files.* • *Plugins* • *Profile stats.* • *Purge data.* • *Purge trash.* • *Rebuild indexes.* • *Repository checker.*

		• *Repository export.* • *Repository import.* • *Scripting.* • *Shards.* • *System properties.* • *Workflow tenant.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuration parameters

OpenKM is preconfigured with several parameters. However, if you plan to use it in production, you should look at Recommended configuration parameters. To access Configuration parameters, go to ***Administration > Configuration parameters****.*

Configuration parameters are shown in a table with three columns:

Property Key	Type	Value

The property key is the unique identifier for each configuration property.

There are several configuration types:

- *String.*
- *Integer (only* ***numeric*** *values are allowed).*
- *Boolean (only* ***true*** *or* ***false*** *values allowed).*
- *List (each broken line is considered as a new list element).*
- *Text.*
- *HTML (HTML tags allowed).*
- *File.*

To create your parameters, read Creating your own parameters.

Edit a parameter

- *Click the* ***edit icon****.*
- *View and* ***modify the value****.*
- *Click the* ***Edit button*** *to confirm the change.*

Delete a parameter

> *You should not delete parameters if you're not sure about the implications of this action.*

- *Click* ***remove icon***  .
- *In* ***Delete confirmation view*** *click* ***Delete button****.*

Backup configuration parameters

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Click at the top right of the **Configuration parameters export icon**  .*
- *The browser will download an SQL file with the configuration.*

New configuration parameter

- *Go to the right-up corner to see the action buttons in the table.*
- *Click the **New configuration parameter button**.*

Check Configuration

This action will check if some conflictive parameters are well configured (pdf2swf, convert tool, OCR engine or LibreOffice path).

> *Although a parameter may pass this check, it that not mean it is 100% correct. Please take it as a light check which cannot cover all cases but is an excellent approximation to discard most problems.*

- *Click at the top right on the **Check icon**  .*
- *It will show the **Configuration check view**. If some parameters are misconfigured or wrong, they will be shown in red.*

Multitenant configuration

> *In OpenKM the multitenant mode is enabled by default.*

- *Click on the top menu  **Configuration button** and then click on the submenu **Tenant Configuration link**.*
- *In tenant configuration, you can create your configuration parameters. These will have preference over the default values. For example, have another **application.url** value for each tenant.*

Recommended configuration parameters

There are some recommended changes in the configuration:

Field / Property	*Type*	*Description*
application.url	***String***	*This parameter is used in mail notifications and the copy-to-clipboard feature among others.* *By default OpenKM sets the access URL to* ***http://localhost:8080/openkm/index****, but obviously you don't want to access your OpenKM installation only from localhost.*
notification.message.body	***HTML***	*This parameter is used in mail notifications body.* *Message: ${notificationMessage}* *User: ${userId}* *<#list documentList as doc>Document: ${doc.path}*
notification.message.subject	***Text***	*This parameter is used in mail notifications subject. Help users identify that these emails are coming from OpenKM.* *OpenKM - NOTIFICATION*
subscription.message.enabled	***Boolean***	*Whether the suscription is evaluated and the message is sent or not. By default is enabled, but in case of massive document import we recommend to disable it to avoid performance issues.*
subscription.message.body	***HTML***	*This parameter is used in mail user subscription notification body.* *Date: ${eventDate?string("yyyy-MM-dd HH:mm:ss")}* *Document: ${documentPath}* *User: ${userId}* *Event: ${eventType}* *Comment: ${subscriptionComment}*
subscription.message.subject	***Text***	*This parameter is used in mail subscription notification subject. Help users identify that these emails are coming from OpenKM.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		OpenKM - ${eventType} - ${documentPath}
text.extraction.concurrent	***Boolean***	*Enable or disable concurrent text extraction concurrent threads.*
text.extraction.batch	***Integer***	*It indicates the number of documents being processed on each indexing cycle.* *When parameter "**text.extraction.concurrent" is enabled**, is a good practice to set multiple of "**text.extraction.threads**" value.*
text.extraction.threads	***Integer***	*Number of concurrent threads that will be used for text extraction.* *It's a good practice to set this number to be less or equal than CPU cores you want to use for text extraction.*
text.extraction.min.size	***Integer***	*Specify the minimum number of characters required to consider a text extraction successful.*
mail.extraction.concurrent	***Boolean***	*Enable or disable concurrent mail extraction concurrent threads.*
mail.extraction.batch	***Integer***	*It indicates the number of mails being processed on each indexing cycle.* *When parameter "**mail.extraction.concurrent" is enabled**, is a good practice to set multiple of "**mail.extraction.threads**" value.*
mail.extraction.threads	***Integer***	*Number of concurrent threads that will be used for mail extraction.* *It's a good practice to set this number to be less or equal than CPU cores you want to use for mail extraction.*
system.imagemagick.convert	***String***	*Path to ImageMagick convert tool.*
system.ocr	***String***	*Path to OCR text extractor engine.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

system.openoffice.program	***String***	*Path to the LibreOffice executable.*
logo.login	***File***	*Login page logo.* *It automatically tries to resize image to 316px × 74px (suggested background #F1F3F5).*
logo.mobile	***File***	*Mobile login page logo.* *It automatically tries to resize image to 161px × 38px (suggested background #F1F3F5).*
logo.report	***File***	*Default logo used in Jasper Reports.* *It automatically tries to resize image to 150px × 35px (suggested background #FFF000).*
logo.tiny	***File***	*Small logo at top right corner in frontend UI.* *It automatically tries to resize image to 51px × 13px (suggested background transparent).*
logo.favicon	***File***	*Small icon in left side of browser url.* *The uploaded file must be a valid icon (16px x 16px).*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Preview parameters

In this section you can find the most common configuration parameters related to preview. Some of these commands are included in OpenKM by default, like ***pdf2swf*** *or* ***ffmpeg*** *(binaries compiles for Linux and Windows) but others need to be installed by the system administrator.*

Linux

Field / Property	*Type*	*Description*
system.imagemagick.convert	***String***	*Path to ImageMagick convert tool.* `/usr/bin/convert`
system.ocr	***String***	*Path to OCR text extractor engine.* `/usr/bin/tesseract ${fileIn} ${fileOut}`
system.ffmpeg	***String***	*Path to FFmpeg audio and video conversion tool.* `/home/openkm/tomcat-7.0.27/bin/ffmpeg`
system.html2pdf	***String***	*Path to html2pdf conversion tool. You can download this utility from http://phantomjs.org.* `/home/openkm/phantomjs /home/openkm/rasterize.js ${urlIn} ${fileOut} A4` *Other tools:*

Tool	*Description*
cutycapt	*Sometimes we have detected better results using http://cutycapt.sourceforge.net/, which seems more configurable. This utility requires an X server, so if you are running from a non-graphical environment you may need to install a virtual X server like xvfb:* `/usr/bin/xvfb-run /usr/bin/cutycapt --url=${urlIn} --out=${fileOut}`

		wkhtmltopdf `/usr/bin/wkhtmltopdf ${urlIn} ${fileOut}`
system.pdfimages	***String***	*Path to pdfimages extractor tool.* `/usr/bin/pdfimages -j -f ${firstPage} -l ${lastPage} ${fileIn} ${imageRoot}`
system.openoffice.program	***String***	*Path to the LibreOffice executable.* `/usr/bin/soffice`
system.ghostscript	***String***	*Path tot the ghosscript conversion tool.* `/usr/bin/gs`

Windows

Field / Property	*Type*	*Description*
system.imagemagick.convert	***String***	*Path to image magick convert tool.* `C:\openkm\bin\convert.exe`
system.ocr	***String***	*Path to ocr text extractor engine.* `C:\openkm\tools\Tesseract-OCR\tesseractexe ${fileIn} ${fileOut}`
system.ffmpeg	***String***	*Path to ffmpeg audio and video converter tool.* `C:\openkm\bin\ffmpeg.exe`
system.html2pdf	***String***	*Path to html2pdf convert tool. You can download this utility from http://phantomjs.org.* `C:\openkm\tools\phantomjs C:\openkm\tools\rasterize.js ${urlIn} ${fileOut} A4`
system.pdfimages	***String***	*Path to pdfimages extractor tool.* `C:\openkm\tools\pdfimages -j -f ${firstPage} -l ${lastPage} ${fileIn} ${imageRoot}`

system.openoffice.program	***String***	*Path to the LibreOffice executable.* *C:\Program Files\LibreOffice\program\soffice.exe*
system.ghostscript	***String***	*Path to the ghosscript conversion tool.* *C:\openkm\bin\gswin32c.exe*

LDAP configuration parameters

The OpenKM LDAP integration is based on several LDAP searches to retrieve data.

LDAP queries are composed of three elements:

1. *Attribute or object to retrieve.*
2. *Node base to scope the request.*
3. *Filtering options.*

Sample query

The query to retrieve mail attribute from an object of type person with sAMAccountName attribute value with {0} - this parameter indicates the userId - from node cn=users,dc=company,dc=local and descendants.

principal.ldap.mail.attribute	*mail*
principal.ldap.mail.search.base	*cn=users,dc=company,dc=local*
principal.ldap.mail.search.filter	*(&(objectclass=person)(sAMAccountName={0}))*

LDAP queries

- ***principal.ldap.user.**** *is used to retrieve the user list.*
- ***principal.ldap.username.**** *are used to retrieve the user name.*
- ***principal.ldap.mail.**** *are used to retrieve user mail addresses.*
- ***principal.ldap.role.**** *are used to retrieve the role list.*
- ***principal.ldap.roles.by.user.**** *are used to retrieve a user's roles.*
- ***principal.ldap.users.by.role.**** *are used to retrieve users of a role.*

For retrieving, data are injected arguments into option filters. For example, for getting the user mail, the application use parameter {0} to set argument value - userId - into filtering options.

Queries that use parameters to filtering:

- ***principal.ldap.mail.search.filter**=(sAMAccountName=**{0}**). Where {0} is the attribute value retrieved from the principal.ldap.user.attribute.*
- ***principal.ldap.roles.by.user.search.filter**=(&(objectclass=user)(sAMAccountName={0})). Where **{0}** is the attribute value retrieved from the principal.ldap.user.attribute.*
- ***principal.ldap.username.search.filter**=(sAMAccountName={0}). Where **{0}** is the attribute value retrieved from the principal.ldap.user.attribute.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- ***principal.ldap.users.by.role.search.filter****=(&(objectClass=group)(cn={0})). Where* ***{0}*** *is the attribute value retrieved from the principal.ldap.role.attribute.*

Parameters

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated from version 8.1.12* *Class used to connect with LDAP.*
system.login.lowercase	***String***	*By default is empty. In case connecting to AD (Microsoft Active Directory) must be set to "**true**," that force all users to be logged in with a lowercase user Id. The reason is OpenKM is case-sensitive, and Microsoft Active Directory is not.*
principal.ldap.server	***String***	*LDAP server.*
principal.ldap.security.principal	***String***	*LDAP user distinguished name (dn).*
principal.ldap.security.credentials	***String***	*LDAP user password.*
		This property specifies how OpenKM handles the referrals sent by AD in the search results. In almost most cases, this value is empty. ***Note for Active Directory (AD) users:*** *AD servers cannot handle referrals automatically, which causes a PartialResultException to be thrown whenever a referral is encountered in a search. To avoid this, set the ignorePartialResultException property to true. There is currently no way of manually handling these referrals in the form of ReferralException, i.e. either you get the exception (and your results are lost), or all referrals are ignored (if the server is unable to handle them properly. There is no simple way to get notified that a PartialResultException has been ignored (other than in the log).* *More information at Spring LdapTemplate.* *Values might be:*

principal.ldap.referral	***String***	*"**ignore**"* *If principal.ldap.referral=ignore, the following exception will be thrown when a referral is encountered:* *javax.naming.PartialResultException: Unprocessed Continuation Reference(s); remaining name 'dc=company,dc=com'* *"**follow**"* *If principal.ldap.referral=follow, OpenKM will automatically follow the referral. To be successful, make sure OpenKM can access the referred server. The errors about ReferralExceptions should not be shown.* *"**throw**"* *If principal.ldap.referral=throw, the following exception will be thrown:* *com.sun.jndi.ldap.LdapReferralException: Continuation Reference; remaining name 'dc=company,dc=com'* *More information at Referrals in the JNDI.*
principal.ldap.users.from.roles	***Boolean***	*When "**true**" tries to retrieve all users from the roles list.*
principal.ldap.user.attribute	***String***	*User attribute.*
principal.ldap.user.search.base	***List***	*List of nodes base.*
principal.ldap.user.search.filter	***String***	*Filter options.*
principal.ldap.username.attribute	***String***	*Username attribute*
principal.ldap.username.search.base	***String***	*Node base.*

principal.ldap.username.search.filter	***String***	*Filter options.*
principal.ldap.mail.attribute	***String***	*Mail attribute.*
principal.ldap.mail.search.base	***String***	*Node base.*
principal.ldap.mail.search.filter	***String***	*Filter options.*
principal.ldap.role.attribute	***String***	*Role attribute.*
principal.ldap.role.search.base	***List***	*List of nodes base.*
principal.ldap.role.search.filter	***String***	*Filter options.*
principal.ldap.roles.by.user.attribute	***String***	*User attribute.*
principal.ldap.roles.by.user.search.base	***String***	*Node Base.*
principal.ldap.roles.by.user.search.filter	***String***	*Filter options.*
principal.ldap.users.by.role.attribute	***String***	*Role attribute.*
principal.ldap.users.by.role.search.base	***String***	*Node base.*
principal.ldap.users.by.role.search.filter	***String***	*Filter options.*

Sample configuration

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter is deprecated from version 8.1.12* *com.openkm.plugin.principal.LdapPrincipalAdapter*

system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.1.20:389*
principal.ldap.security.principal	***String***	*CN=Administrator,OU=OpenKM,DC=company,DC=com*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*DC=company,DC=com*
principal.ldap.user.search.filter	***String***	*objectclass=person*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*DC=company,DC=com*
principal.ldap.username.search.filter	***String***	*(sAMAccountName={0}*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*DC=company,DC=com*
principal.ldap.mail.search.filter	***String***	*(sAMAccountName={0}*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*DC=company,DC=com*
principal.ldap.role.search.filter	***String***	*objectclass=group*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

principal.ldap.roles.by.user.search.base	***String***	*DC=company,DC=com*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectclass=person)(sAMAccountName={0}))*
principal.ldap.users.by.role.attribute	***String***	*member*
principal.ldap.users.by.role.search.base	***String***	*DC=company,DC=com*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectClass=group)(cn={0}))*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

User interface configuration parameters

Except for the favicon, the uploaded images are automatically resized to match the correct size. Most image formats, like JPG, PNG, etc., are allowed.

Field / Property	Type	Description
default.lang	**String**	*Forces the user interface language to a specific language. By default, this parameter is empty. The value is based on the international language code.* `en-GB`
logo.favicon	**String**	*Website icon.*
logo. login	**String**	*Login page logo.* ℹ *Automatically, it tries to resize the image to 316px × 74px (suggested background #F1F3F5).*
logo. mobile	**String**	*Mobile logo.* ℹ *Automatically, it tries to resize the image to 161px × 38px (suggested background #F1F3F5).*
logo. report	**String**	*Reports logo.* ℹ *Automatically, it tries to resize the image to 150px × 35px (suggested background #FFF000).*
logo.tiny	**String**	*A small logo is used at the top right of the OpenKM user interface.*
text.title	**String**	*Browser title. This text is set between* `${text.title}` *HTML tags.*
thumbnail.size.lightbox	**String**	*Maximum size - width * height - of the lightbox thumbnail.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		640x640>
thumbnail.size.properties	***String***	*Maximum size - width x height - of the thumbnail in the properties view.* *280x280>*
thumbnail.size.search	***String***	*Maximum size - width x height - of the thumbnail in search view.* *180x180*
download.zip.live.max.files	***Integer***	*Sets the maximum number of files allowed to download in live mode. Once this number is exceeded, the application will notify the user that the ZIP creation will be executed in background mode. And after it is finished, it will notify the user - by mail - with a download URL. Value **0** indicates an **unlimited number of files**.* *Downloading a couple of selected files and folders can take time. The application on the server side creates a temp folder where all data is packed as a ZIP file and then forwarded to the end user interface.* *100*
download.zip.live.max.file.size	***String***	*Sets the maximum size of all files allowed to download in live mode. Exceeding this size, the application will notify the user that the ZIP creation will be executed in background mode. After it is finished, it will notify the user via email with a download URL. Value **0** indicates **unlimited size**. Indicates size units with KB, MB, GB, or TB; otherwise, the value will be considered bytes.* *Downloading a couple of selected files and folders can take time. The application on the server side creates a temp folder where all data is packed as a ZIP file and then forwarded to the end user interface.* *100MB*
min.search.characters	***Integer***	*Sets the minimum characters required to execute a search. With fewer characters, the search button will not be enabled.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		3

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AI configuration parameters

*The following parameters must be set in the **openkm.properties** file, and once configured, the OpenKM service must be restarted for the changes to take effect.*

You should only activate one AI model.

*It is possible to **integrate with other AI services providers**, such as the **Azure AI platform** or an **LLM service** deployed on a server (cloud, intranet, etc.).*

Property	Type	Description
chatgpt.model	**String**	*Name of the model* gpt-4o-mini
chatgpt.api.key	**String**	*The API key*
mistral.model.vision	**String**	*Name of the vision model* pixtral-large-latest
mistral.model	**String**	*Name of the model* mistral-large-latest
mistral.api.key	**String**	*The API key*
deepseek.model	**String**	*Name of the model* deepseek-chat
deepseek.api.key	**String**	*The API key*
gemini.model	**String**	*Name of the model* gemini-1.5-flash
gemini.api.key	**String**	*The API key*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow configuration parameters

Explain for what these parameteres are:

Field / Property	*Type*	*Description*
workflow.adapter	***String***	*OpenKM can be used with several workflow engines. By default, we offer an integration with jBPM but by implementing an adapter you can use other workflow engines. To do this, you need to implement the* ***com.openkm.workflow.adapter.WorkflowAdapter*** *interface.* *Possible values* • *com.openkm.workflow.adapter.DummyWorkflowAdapter* • *com.openkm.workflow.adapter.JbpmRemoteWorkflowAdapter* *com.openkm.workflow.adapter.DummyWorkflowAdapter* *When DummyWorkflowAdapter is selected the workflow icon in administration will not being shown.* *After changing this value you should restart OpenKM application to take effect.*
workflow.adapter.login	***String***	*Login credentials used to connect to the external workflow engine. Used with* ***com.openkm.workflow.adapter.JbpmRemoteWorkflowAdapter****.* *admin*
workflow.adapter.password	***String***	*Password used to connect to the external workflow engine. Used with* ***com.openkm.workflow.adapter.JbpmRemoteWorkflowAdapter****.* *admin*
workflow.adapter.url	***String***	*The URL used to connect to the external workflow engine. Used with* ***com.openkm.workflow.adapter.JbpmRemoteWorkflowAdapter****.* *http://localhost:8080/jbpm-console*
workflow.run.config.form	***String***	*The form name in case the workflow needs parameters at the start. By default this form is called "run_config".*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		run_config
workflow.start.task.auto.run	***String***	*If OpenKM should run the task defined in the start node. This start dummy task is usually used to assign the "initiator" swimlane. By default the value is set to "true". You don't need to modify this value.* *true*
workflow.node.deletion.check	***Boolean***	*Enable or disable the deletion of a node which is used in a workflow. By default is set to "true".* *true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Notification configuration parameters

When a user is subscribed to a document, any change in the document will be notified by email. These email messages are created using some defined templates, and these default templates to create messages can be adapted to your style.

Field / Property	Type	Description
send.mail.from.user	**Boolean**	*By default, the application sends mail notifications setting the from mail field value will use the default from mail address for all the emails sent by the application (all address).* *When the mail server have restricted from changing the default from th the general@mail.com account to send a mail with from mail field valu configuration is set to prevent spam), then this parameter must be desig* *true*
notification. message.subject	**Text**	*The subject of the message sent when a user has chosen to notify something other.* *OpenKM - NOTIFICATION*
notification.message.body	**HTML**	*Body of the message sent when a user has chosen to notify something other.* *ReplaceMessage: ${notificationMessage}* *User: ${userId}* *<#list documentList as doc>Document: ${doc.path}* *</#list>*
subscription. message.subject	**Text**	*Subject of the message sent automatically caused by some event. That happens bec* *OpenKM - ${eventType} - ${documentPath}*
subscription.message.body	**HTML**	*Body of the message sent automatically caused by some event. That happens becaus* *Date: ${eventDate?string("yyyy-MM-dd HH:mm:ss")}* *Document: ${documentPath}* *User: ${userId}* *Event: ${eventType}* *Comment: ${subscriptionComment}*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

task.message.subject	***Text***	*The message's subject is sent when there's some change in the task assigned to a us* *OpenKM - Task Manager Notification: ${message}*
task.message.body	***HTML***	*Body of the message sent when there's some change on the task assigned to a user.* ***Task:*** *${tasksubject}* ***Start:*** *${taskstart!}* ***End:*** *${taskend!}* ***Project:*** *${taskproject!}* ***Status:*** *${taskstatus!}* ***Note:*** *${tasknote!}*
download.message.body	***HTML***	*Body of the message sent when a user shares some document with others for downl* *Your archive is ready. Please download from ${url}*
download.message.subject	***String***	*The message's subject is sent when a user shares some document with others for do* *OpenKM - DOWNLOAD*
download. token.validity	***Integer***	*Range of days while the download url will be available.* *7*
template.message.body	***String***	*Set the basic template for all notifications.* * <table style="height: 100%; width: 100%;" border="0" cellspacing="0" cel <tbody> <tr> <td style="background: #e5e5e5;" valign="top"> <table style="margin: 20px auto 0 auto; width: 600px;" border="0" width=" <tbody> <tr style="background: #e5e5e5;"> <td valign="top"> <table width="100%" cellspacing="0" cellpadding="0"> <tbody> <tr> <td> <table width="100%" cellspacing="0" cellpadding="0"> <tbody> <tr> <td bgcolor="#018ca9" width="100%" height="6px"> </td> </tr> </tbody> </table> </td>*

```
</tr>
<tr>
<td>
<div style="border-top: solid 1px #88888b; min-height: 5px; max-height: 1
</td>
</tr>
<tr>
<td style="padding: 0 10px 10px 10px;">
<div>
<p>${body}</p>
</div>
</td>
</tr>
</tbody>
</table>
</div>
</td>
</tr>
<tr>
<td style="background: #88888b;">
<table style="width: 100%;" border="0" cellspacing="0" cellpadding="0">
<tbody>
<tr>
<td height="4"> </td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td style="background: #222222; padding: 10px 20px;">
<table style="width: 100%;" border="0" cellspacing="5" cellpadding="0">
<tbody>
<tr>
<td>
<h3 style="color: #f6a81a; font: 18px verdana, arial, sans-serif; margin: 0
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</div>
```

notification.url	***String***	*KCenter user interface:* *http://localhost:8080/openkm/kcenter/#*

Substitutions

*For **notification.message.body** property:*

- ***${documentUrl}** shows the node URL.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- ***${documentPath}*** *shows the node path.*
- ***${documentName}*** *shows the node name.*
- ***${userName}*** *display the user name who caused the event.*
- ***${userId}*** *shows the user id who caused the action.*
- ***${notificationMessage}*** *show the notification message*

For ***task. news.body*** *property:*

- ***${tasksubject}*** *shows the subject of the study.*
- ***${taskstart}*** *shows when the job started.*
- ***${taskend}*** *showed when the task ended.*
- ***${taskproject}*** *the task project.*
- ***${taskstatus}*** *indicate the status of the task.*
- ***${taskknote}*** *show the notes.*



FreeMarker does not allow you to render null values (it will throw an exception). If you expect a value to be null, you can also use the ***!*** *operator:* ***${content.title!}****. This is equivalent to* ***${content.title!}****.*



More information at:

- *java.text.MessageFormat*

Document version numeration configuration

We support different numeration schemas. By default OpenKM uses ***com.openkm.vernum.MajorMinorVersionNumerationAdapter*** *which is similar to Jackrabbit implementation.*



More information about how to Creating your own Version Number Adapter.

Field / Property	*Type*	*Description*
version.numeration.adapter	***String***	*Numeration schema.* *Available schemas :* • *com.openkm.plugin.vernum.PlainVersionNumerationAdapt* • *com.openkm.plugin.vernum.PlainZeroVersionNumerationA* • *com.openkm.plugin.vernum.MajorMinorVersionNumeratio* • *com.openkm.plugin.vernum.BranchVersionNumerationAda* • *com.openkm.plugin.vernum.MajorMinorReleaseVersionNu* • *com.openkm.plugin.vernum.VersionLetterAdapter* *com.openkm.vernum.MajorMinorVersionNumerationAdapter*
version.numeration.format	***String***	*Numeration number format. Used in* ***PlainVersionNumerationAdapter****.* *%d*

Some Numeration Adapters like ***PlainVersionNumerationAdapter*** *may have additional configuration parameters like* ***version.numeration.format*** *which is used to define the numeration number format. By default this is "%d" but you can change to "%02d" to create versions like "01", "02", etc.*



Restarting application *is needed after changing the value of* ***version.numeration.adapter.***

Also note that if already there are documents in the repository, changing the Numeration Adapter may cause problems because they are not compatible between them.

Version numeration schemas

Numeration adapter	*First version*	*Next versions*
com.openkm.plugin.vernum.PlainVersionNumerationAdapter	*1*	*2, 3, 4, etc.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

com.openkm.plugin.vernum.PlainZeroVersionNumerationAdapter	*0*	*1, 2, 3, etc.*
com.openkm.plugin.vernum.MajorMinorVersionNumerationAdapter	*1.0*	*1.1, 1.2, 1.3, etc.*
com.openkm.plugin.vernum.BranchVersionNumerationAdapter	*1*	*2, 3, 2.1, 2.2, etc.*
com.openkm.plugin.vernum.MajorMinorReleaseVersionNumerationAdapter	*0.0.0*	*0.0.1, 0.0.2, 0.1.0, 0.1.1, 1.0.0*
com.openkm.plugin.vernum.VersionLetterAdapter	*A*	*B, C*

Migration between versions



***We recommend you set the correct numeration at the beginning of using the application** . Otherwise will be needed a migration of existing numeration and this operation is not always easy.*

*You can change the default **Numeration Adapter** using the "**version.numeration.adapter**" configuration parameter. You need to restart OpenKM every time you change this parameter. Also note that if already there are documents in the repository, changing the **Numeration Adapter** may cause problems when they are not compatible between them.*

*If you change the version numeration adapter you also need to update the existing numeration, for example the migration from **PlainVersionNumerationAdapter** to **MajorMinorVersionNumerationAdapter**.*

```
-- HQL queries
UPDATE NodeDocumentVersion ndv SET ndv.name = '01' WHERE ndv.name='1.0';
UPDATE NodeDocumentVersion ndv SET ndv.name = '02' WHERE ndv.name='1.1';
UPDATE NodeDocumentVersion ndv SET ndv.name = '03' WHERE ndv.name='1.2';
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Password validation configuration

Validation password enables the application to set a couple of rules that should be accomplished by passwords. By default, application makes no password checking (***validator.password*** *property is set by default to* ***com.openkm.validator.password.NoPasswordValidator*** *).*

To enable validator password must change to ***com.openkm.validator.password.CompletePasswordValidator*** *or Creating your own password validator.*

Field / Property	*Type*	*Description*
validator.password	***String***	*Description* *Available:* • *com.openkm.validator.password.NoPasswordValidator* • *com.openkm.validator.password.CompletePasswordValidator* *com.openkm.validator.password.NoPasswordValidator*
validator.password.min.length	***String***	*Minimum length allowed.* *Numeric value greater than 0.*
validator.password.max.length	***String***	*Maximum length allowed.* *Numeric value greater than 0.*
validator.password.min.lowercase	***String***	*Minimum number of lower-case characters.* *Numeric value greater than 0.*
validator.password.min.uppercase	***String***	*Minimum number of upper-case characters.* *Numeric value greater than 0.*
validator.password.min.digits	***String***	*Minimum number of digits.* *Numeric value greater than 0.*
validator.password.mini.special	***String***	*Minimum number of special characters.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		Numeric value greater than 0.

Sample configuration

- *Must have at least 8 characters.*
- *At least must have 1 upper-case character.*
- *At least must have 1 digit.*

Field / Property	*Type*	*Description*
validator.password	***String***	*com.openkm.validator.password.CompletePasswordValidator*
validator.password.min.length	***String***	*8*
validator.password.max.length	***String***	
validator.password.min.lowercase	***String***	
validator.password.min.uppercase	***String***	*1*
validator.password.min.digits	***String***	*1*
validator.password.mini.special	***String***	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Switch application to read-only mode

It can force the whole repository to be in read-only mode. This is useful when doing maintenance tasks, or while migrating and you wish to get the system alive for your users but want to ensure nobody can change anything.

Field / Property	*Type*	*Description*
system.readonly	*Boolean*	*Switch whole repository to be read-only. By default is set to false.* *false*



*If you **restart the application**, this parameter will be **automatically set to true**.*



*For security and performance reasons, when you execute **Lucene rebuild indexes**, the system will **switch automatically to read-only mode** until the process is finished. Do not change manually, wait until be completed.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Security configuration parameters

Field / Property	Type	Description
user.admin	**String**	*Sets the default super user id. By default super user id is okmAdmin.* *okmAdmin*
user.assign.document.creation	**Boolean**	*By default when a user creates a node, he is added to the node with full permissions. You can disable this behavior by setting this parameter to false. By default the parameters is set to true.* *true*
user.password.expiration	**Integer**	*Indicates the number of days to force user change their password, the 0 value indicates the feature is disabled.* *0*
user.password.reset	**Boolean**	*Sometimes the user loses or forgets their password. When this option is enabled, used in landing page have got the option to restart their password and be notified with a newer one by mail. By default this option is set to false.* *false*
system.antivir	**String**	*Path to antivirus.* *An antivirus can decrease perfomance, taking several seconds to analyze each document and consuming a lot of CPU resources.*
security.access.manager	**String**	*Sets security access manager determinate the way on how the security is evaluated. Default value is "simple".* *Values:* • *simple.* • *recursive.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		• *read_recursive.* *simple* *Takes a look at Security configuration*
security.search.evaluation	***String***	*Sets the security search manager and determinates the way how the security is evaluated by search engine.* *Values:* • *lucene.* • *am_more, am_window or am_limited.* *Take a look at Security configuration*
security.extended.mask	***Integer***	*Security evaluation can be extended to downloaded files, starts workflows, adds, removes or modifies property groups (metadata) or compact history. Default value is empty.* *Values:* • *DOWNLOAD = 1024;* • *START_WORKFLOW = 2048;* • *COMPACT_HISTORY = 4096;* • *PROPERTY_GROUP = 8192;* *For example to enable download and property groups you should put the mask 9216 (1024+8192)* *9216* *Take a look at Security configuration*
security.live.change.node.limit	***Integer***	*When security changes affect more than the values set in this property the security changes are applied as a background task.* *100* *Take a look at Security configuration*

system.login.lowercase	***Boolean***	*By default is empty. In case connecting to AD (Microsoft Active Directory) must be set to "**true**", that force all users to be logged with lowercase user Id. The reason is OpenKM is case sensitive and Microsoft Active Directory not.*
principal.adapter	***String***	*You must restart OpenKM service after you change this parameter.* *OpenKM can handle user access using the Spring Security framework. OpenKM needs an available method for reading users and roles, so when users are stored in a database (as is the default), the class **DatabasePrincipalAdapter** does this job.* *com.openkm.plugin.principal.DatabasePrincipalAdapter* *If you configure OpenKM to authenticate against an LDAP server, you need to configure another principal adapter like **LdapPrincipalAdapter**.* *More information at:* • *LDAP configuration.* • *Creating your own login module.*
principal.identifier.validation	***String***	*Sets a regular expression to validate on the creation of user names and roles names.* ***^[a-zA-Z0-9_]+$*** *Regular expression sample for Arabic:* ***^[\u0600-\u06FF]+$*** *This parameter only can be used in combination with "**principal.adapter**" parameter value "**com.openkm.plugin.principal.DatabasePrincipalAdapter**".*
browser.password.autocompletion	***Boolean***	*By default it is true (allowed autocompletion). This paremeter allows to disable the autocompletion of the password in the login frame. This is a security messure in not trusted clients.*

default.security.recursive.role	***String***	*Set the role what is used to identify what users are able to set recursive security.* *ROLE_SECURITY_RECURSIVE*
secure.file.delete	***Boolean***	*Enable safe file delete from the file system. The process ensures the document might be recovered later from the file system.* *false*
security.login.failed.attempts	***Integer***	*Number of login failure attemps before the user will be locked. Value 0 indicates unlimited failure attemps and the user will never be locked.* *0*
default.task.manager.admin.role	***String***	*Set the role that is used to identify what users can manage all the OpenKM task.* *ROLE_TASK_MANAGER_ADMIN*
default.version.admin.role	***String***	*Set the role that is used to identify the users who can change the version of a node.* *ROLE_VERSION_ADMIN*
default.user.hidden.role	***String***	*Set the role that is used to identify the users hidden in the user list.* *ROLE_USER_HIDDEN*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Extensions configuration parameters

Extratab extension

The ExtraTab allows for easy addition of a new tab with a name and a specific URL.

Field / Property	Type	Description
extension.extra.tab.workspace	**List**	*Title and URL pair. You can add a single line (where the character ; is the separator between title and URL) or several lines.* `OpenKM;http://www.openkm.com` *If **the URL does not start with extension.extra.tab.workspacep** is considered as **OpenKM relative path** and will be merged with your OpenKM server URL like **YOUR_OPENKM_URL/url***

External preview extension

The External Preview feature helps to preview documents based on an external URL request.

Field / Property	Type	Description

extension.external.preview	*List*	*Each row couple of values is a preview content declaration, the format is:*
		mime-type;url
		`application/illustrator;http://localhost:8180/OpenKM/test.jsp`
		`application/epub+zip;http://localhost:8180/OpenKM/epub_previewer.jsp`
		`application/x-mspublisher;http://localhost:8180/OpenKM/mspreviewer_previewer.jsp`

External tab document extension

The External tab document feature shows extra tabs in tab document view based on an external URL request.

Field / Property	Type	Description
extension.external.tab.document	**List**	*The format for each line is **title;URL**, where the character ";" is used as the separator between the title of the tab and the URL of the contents.* `OpenKM;http;//www.openkm.com`

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

External tab folder extension

The External Tab Folder feature helps you to show the tab content in the properties tab based on an external URL request when a node folder is selected.

Field / Property	Type	Description
extension.external.tab.folder	**List**	*Each row is a couple of values with a tab content declaration, the format is:* **name;url** *name1;http://localhost:8180/OpenKM/test.jsp* *name2;http://www.openkm.com*

External tab mail extension

The External Tab Mail feature helps you to show tab content in the properties tab based on an external URL request when a node mail is selected.

Field / Property	Type	Description
extension.external.tab.mail	**List**	*Each row is a couple of values with a tab content declaration, the format is:* **name;url** *name1;http://localhost:8180/OpenKM/test.jsp* *name2;http://www.openkm.com*

External tab record extension

The External Tab Record feature helps you to show tab content in the properties tab based on an external URL request when a node record is selected.

Field / Property	Type	Description
extension.external.tab.record	**List**	*Each row is a couple of values with a tab content declaration, the format is:* **name;url** *name1;http://localhost:8180/OpenKM/test.jsp* *name2;http://www.openkm.com*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Performance configuration parameters

Text extractor queue

There are several configuration parameters to adapt the document text extraction to your needs. Text extraction is one of the processes that can use a lot of hardware resources (memory and CPUs). An aggressive text extraction policy can decrease the performance of the application and affect the end user's feelings about the user interface.

When documents are uploaded are automatically added into ***Text extraction queue***. *Can see the Text extraction queue at* ***Administration > Statistics > Text extraction queue***.

*There's a crontab task named "**Text extractor worker**" at **Administration > Cron tab** that controls the indexing cycle*



When a document is uploaded, it will not be able to search by content until it is processed by the Text extraction queue.



In case the application needs to index a lot of files per day, or you imported a considerable amount of files, consider creating two crontab tasks to change these parameters between day (less aggressive) and night (more aggressive).



Some mime-types like PDF or images (which need an OCR engine) can use a lot of CPU (top 100%).

Field / Property	*Type*	*Description*
text.extraction.batch	***Boolean***	*Indicate the number of documents to be processed on each indexing cycle.*  *When parameter "**text.extraction.concurrent" is enabled**, it is a good practice to be a multiple of "**text.extraction.threads**" value.* *20*
text.extraction.concurrent	***Boolean***	*Enable or disable the concurrent text extraction concurrent threads feature.* *True*
text.extraction.threads	***Integer***	*The number of concurrent threads will be used for the text extraction feature.*  *This number should be less than or equal to the number of hardware cores.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		2

*There's a crontab task named "**Mailextractor worker**" at **Administration** > **Cron tab**, which controls the indexing cycle*

> *When a mail is uploaded, it will not be able to search by content until it is processed by the Mail extraction queue. Furthermore, attachment won't appear until it is processed.*

Field / Property	*Type*	*Description*
mail.extraction.batch	***Boolean***	*Indicate the number of emails to be processed on each indexing cycle.* *When parameter "**mail.extraction.concurrent" is enabled**, it is a good practice to be a multiple of "**mail.extraction.threads**" value.* *20*
mail.extraction.concurrent	***Boolean***	*Enable or disable the concurrent mail extraction concurrent threads feature.* *True*
mail.extraction.threads	***Integer***	*The number of concurrent threads will be used for the mail extraction feature.* *This number should be less than or equal to the number of hardware cores.* *2*

Download files and folders as ZIP file

*It is not a good practice to download as ZIP files high volume of files. In most cases, take in consideration the application export tool at **Administration** > **Utilities** > **Repository export**.*

Downloading files and folders as a ZIP file can take a lot of time and hardware resources. First, the application will prepare a temporary folder on the server that will be used to create the ZIP file, which will finally be forwarded to the user interface. To prevent blocking the user interface waiting for process completion, there are two parameters that will notify the user has exceeded some limits (number of files or size) and the process will go into background mode.

Field / Property	*Type*	*Description*
		Set the maximum number of files allowed to download in live mode.

download.zip.live.max.files	***Integer***	*100*
download.zip.live.max.file.size	***String***	*Set the maximum size of all files allowed to download in live mode.*

For more information about these parameters, read User interface configuration parameters.

Antivirus checker

There are several configuration parameters to adapt the antivirus checker to your needs. Antivirus checker is one of the processes that can use a lot of hardware resources (memory and CPUs). An aggressive antivirus checker policy can decrease the performance of the application and affect the end user's feelings about the user interface. The Antivirus checker can work in live mode (as the uploading process antivirus checks the document and if a virus is detected, a warning error is immediately raised and the document is not incorporated into the repository) or in background mode.



Antivirus checker in live mode can dramatically decrease the user interface and the end user's experience. Take into consideration that to analyse some documents, the antivirus can check them in 3 seconds or more, which will be added to the uploading time process.



In case the application needs to analyse a lot of files per day, or you imported a considerable amount of files, consider creating two crontab tasks to change these parameters between day (less aggressive) and night (more aggressive).

Field / Property	*Type*	*Description*
system.antivir	***String***	*Set the antivirus path.*
background.antivirus.check	***Boolean***	*Enable or disable the background antivirus checker. By default is set to* ***false****.* *There's a* ***crontab*** *task (**Administration > Crontab**)named **Antivirus Checker Worker** that takes control of the background antivirus checker.* *When the background check is disabled, the antivirus is processed while uploading any document. That can add extra time to the uploading process (in some cases, more than 3 seconds is the time needed for an antivirus to check a document). When you do a huge import of files, the total time to complete the operation can be multiplied by x3 or more with this option disabled, because as part of. Our suggestion is to set it*

		to enabled. *true*
background.antivirus.check.threads	***Integer***	*The number of concurrent threads will be used for antivirus.* *This number should be less than or equal to the number of hardware cores.* *2*
background.antivirus.check.batch	***Integer***	*Indicate the number of documents to be processed on each indexing cycle.* *When parameter "**background.antivirus.check" is enabled**, it is a good practice to be a multiple of "**background.antivirus.check.threads**" value.* *20*

For more information, read Antivirus configuration parameters.

Re-indexing the whole Lucene repository

*The application can use two different ways of rebuilding Lucene indexes: sequential or parallel. By default, sequential re-indexing is enabled, but you can select the mode using the "**hibernate.indexer.rebuild**" configuration parameter.*

Field / Property	*Type*	*Description*
hibernate.indexer.rebuild	***String***	*By default, set to "**flush**". Possible values are "**mass**", "**flush**," and "**delayed**". Take a look at the table below to see the different value details.*

*You can use the "**hibernate.indexer.batch.size.load.objects**" configuration parameter to indicate to Hibernate how many objects should be handled every time. To avoid **OutOfMemory** problems, the repository needs to be re-indexed in batches. If the value of this property is too low, the performance will be bad, but if it is too high, you can have **OutOfMemory** problems.*

Mode	*Description*
Mass Indexer	*This mode will reindex the repository in parallel, using several threads. This may be faster, but it is also problematic in terms of resources. The repository will be in read-only mode until completed* *You have several configuration properties to tune its performance:*

- ***hibernate.indexer.batch.size.load.objects****: batch size used to load the entities.*

30

- ***hibernate.indexer.threads.load.objects****: number of threads used to load the entities.*

4

Flush to Indexes

This mode will rebuild the index sequentially in batches. It's less resource hungry but may be slower. The repository will be in read-only mode until completed

You have several configuration properties to tune its performance:

- ***hibernate.indexer.batch.size.load.objects****: batch size used to load the entities.*

30

Delayed Indexer

There's a cron task named "Lucene Delayed Indexer" which processes the queue of nodes to be indexed in the background.

This queue is filled up when you go to Administration > Tools > Rebuild indexes and choose "Lucene indexes". When the delayed indexer is enabled, the nodes are included in the batch queue to be processed with this background process.

This mode will rebuild the index sequentially in batches. It's less resource hungry but may be slower, but you have more control because you can choose which nodes need to be re-indexed, and the repository won't be in read-only mode while working.

You have several configuration properties to tune its performance:

- ***hibernate.indexer.delayed.max.objects****: how many nodes will be processed in a batch.*

1000

- ***hibernate.indexer.batch.size.load.objects****: number of nodes to read from the database at the same time.*

4

Execution timeout

The Application executes external applications to process documents, for example, extracts the text with an OCR engine, analyzes with an Antivirus software, transforms documents to other formats, among other actions. Sometimes these processes

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*can take a lot of time or for some reason are not finished correctly, and the process keeps on the OS consuming resources. To prevent it, the parameter "**system.execution.timeout**" sets the maximum allowed time of execution on external applications.*

Field / Property	Type	Description
system.execution.timeout	**Integer**	*Set the maximum allowed time of execution for external applications.* *The units are minutes. By default, the configuration is set to 5 minutes.* *5*

Activity log actions

OpenKM can log a lot of information related to the activity of the users, but sometimes these actions don't need to be logged and fill your activity log table.



The table where the activity log is stored - OKM_ACTIVITY - may grow quickly, storing millions of records.

*There is a configuration property named "**activity.log.actions**" where you can set which actions to log. By default, this is set to the most common or interesting actions. You can use regular expressions to define these actions. Read Java Regex Tutorial for more info about Java regular expressions.*

Field / Property	Type	Description
activity.log.actions	**List**	*LOGIN* *LOGOUT* *CREATE_.** *DELETE_.** *PURGE_.** *MOVE_.** *COPY_.** *CHECKOUT_DOCUMENT* *CHECKIN_DOCUMENT* *GET_DOCUMENT_CONTENT.** *MISC_TEXT_EXTRACTION_FAILURE*



For a complete list of action see Activity log.

Dashboard activity

The OpenKM dashboard displays a global view of application activity. For this, the application needs to process a huge amount of information, which can decrease the overall performance of the application (for example, for getting the last documents uploaded into the system, OpenKM might execute database slow queries). For this, we suggest that huge repositories disable dashboard activity.

Field / Property	*Type*	*Description*
dashboard.user.activity	***Boolean***	*Set to false for disabling the feature.*

Lucene

The Lucene search engine can be disabled or configured in several ways. Depending on it, the behavior of the application and the performance overall might change.



*These configuration parameters are going into the "**openkm.properties**" configuration file.*

Field / Property	*Type*	*Description*
spring.jpa.properties.hibernate.search.enabled	***String***	*Enable or disable the Lucene search engine.* *Allowed values:* • *true* • *false* *true*
spring.jpa.properties.hibernate.search.automatic_indexing.synchronization.strategy	***String***	*Set the Lucene worker execution mode. By default is set to "write-sync" which means that every entity modification won't end until the Lucene index has been properly updated. In case of "async", these*

		entity modifications will be faster because the Lucene part is added to a queue and processed in a background thread. *Allowed values:* • *sync* • *async* *sync*

Other configuration properties

Field / Property	*Type*	*Description*
background.parent.node.purge	***Boolean***	*Performs node purge as a background task so the user can keep working without needing to wait until the process has ended.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

LibreOffice configuration parameters

As part of the application startup, OpenKM executes an LibreOffice service. That's used internally for conversion purposes, for example to converting doc files to pdf.



The LibreOffice service can use a lot of hardware resources (CPU top 100%).

Field / Property	*Type*	*Description*
system.libreoffice.program	***String***	*path to the LibreOffice executable.*
system.libreoffice.dictionary	***String***	*The dictionary used by search engine tokenizer.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

CSV configuration parameters

The CSV ("Comma Separated Value") file format is often used to exchange data between disparate applications. The file format, as it is used in Microsoft Excel, has become a pseudo standard throughout the industry, even among non-Microsoft platforms.

Some application features import or export data in CSV (Comma Separated Value) like results in search view users in administration among others. The format of the output CSV file can be configured with several parameters to your needs.

Field / Property	*Type*	*Description*
csv.format.delimiter	***String***	*Set the delimiter used between columns. By default is used character "**;**".*
csv.format.ignore.empty.lines	***Boolean***	*Set if empty lines must be processed or not. By default the value is **false**.*
csv.format.quote.character	***String***	*Set the quote character used to enclose column values. By default is set to " character.*
csv.format.skip_header	***Boolean***	*Set if first row contains column names and must not be processed as values or not. By default is set to **true**.*
csv.format.comment.indicator	***String***	*Set character used to indicate the beginning of a comment. By default is set to #.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Antivirus configuration parameters

Antivirus is one of the processes that can use a lot of hardware resources (memory and CPU's). An aggressive antivirus policy can decrease the performance of the application and affect end users feeling with user interface.

Antivirus analyzer can work in two mode (depending the value of ***background.antivirus.check*** *parameter):*

- *Live mode.*
- *Batch mode.*

> *Antivirus checker in live mode can dramatically decrease the user interface end user feeling. Take in consideration that to analyse some documents the antivirus can take up to 3 seconds or more what will be added to the uploading time process.*

Field / Property	*Type*	*Description*
system.antivir	***String***	*Set the antivirus path.* */usr/bin/clamscan*
background.antivirus.check	***Boolean***	*Enable or disable background antivirus checker. By default is set to* ***false****.* *There's a* ***crontab*** *task (* ***Administration > Crontab*** *) named* ***Antivirus Checker Worker*** *that take control of background antivirus checker.* *For more information Built in crontab tasks.* *When background check is disabled the antivirus is processed while uploading any document. That can add extra time to the uploading process be complete (in some cases more than 3 seconds is the time needed for the antivirus to check a document). For example when you do a huge import of files the total time to complete the operation can be multiplied by x 3 or more with this option disabled, Our suggestion is set it enabled.*
background.antivirus.check.threads	***Integer***	*Number of concurrent threads will be used for antivirus.* *This number should be less or equal the number of*

		hardware cores.
background.antivirus.check.batch	***Integer***	*Indicate the number of documents be processed on each indexing cycle.* *When parameter "**background.antivirus.check" is enabled**, is a good practice to be multiple of **"background.antivirus.check.threads"** value.*

Other configuration parameters

Property	Type	Description
spring.jpa.properties.hibernate.search.analyzer	**String**	*This parameter is used to change the default Apache Lucene S* *org.apache.lucene.analysis.standard.StandardAnalyze* *More information at Configuring Lucene Analyzer.* *This parameter must be set on $TOMCAT_HOME*
spring.jpa.properties.hibernate.search	**Boolean**	*This parameter can be used to disable Hibernate Search comp* *able to search document or folders.* *This parameter must be set on $TOMCAT_HOME*
system.ocr.rotate	**String**	*This parameter forces OCR to process on the document rotate* *The feature is useful for example in upside down s* *OCR direction will fails.* *The parameter is a collection of degrees separated by charact* *90;180;270;* *More information at Configuring OCR engine.*
system.ocr.spell.check	**Boolean**	*When is enabled the text extracted by OCR is validated by spe* *The application must have an enabled dictionary,* *"**system.openoffice.dictionary**" at LibreOffice cor*
system.document.name.mismatch.check	**Boolean**	*This parameter is used to enable or disable the name check wh* *is uploaded.*
max.search.results	**Integer**	*This parameter sets the limits of the search results. For examp*

		never get more than 500 results of any query. `500` *The reason why the number of results is limited is a huge number of results, specially when the quer methods.*
system.keyword.lowercase	***Boolean***	*This parameter is used to force all keyword to lowercase.* *The application is case sensitive, that means a use "keyone" and "KeyOne" and for the application th To prevent this problem is always good idea force*

tomcat.connector.uri.encoding	***String***	*By default Tomcat Connector encoding is set to "**ISO-8859-1**" parameter.*
		*When you set the "**URIEncoding**" parameter in T need to set this parmeter to "**UTF-8**".*
		`URIEncoding="UTF-8"`
		`ISO-8859-1`

mail.session.debug	***Boolean***	*Enable mail session debug. It's useful to detect problems when mails.*
mobile.theme	***String***	*There're five mobile themes ("a", "b", "c", "d", "e").* *More information about themes at jQuery mobile* `b`
plugin.debug	***Boolean***	*When enabled show debug information about application plug*
principal.hide.connection.roles	***Boolean***	*Hide "**ROLE_USER**" and "**ROLE_ADMIN**" roles from avail*

		*By default the hidden roles are set by "**default.us** "**default.admin.role**" parameters value. Usually " "**ROLE_ADMIN**".*
principal.database.filter.inactive.users	***Boolean***	*Deprecated from version 8.1.12* *Hide inactive users for available users list.*
restrict.file.name	***String***	*Restrict file name extension that are not allowed.* *Each file name extension is separated by characte* **~;*.bak*
restrict.file.mime	***Boolean***	*Restrict file name extension based on existing ones at **Admini***
restrict.duplicated.documents	***Boolean***	*Is not allowed to upload more than one time the same docume* *Internally the verification is based on the docume*
native.sql.optimizations	***Boolean***	*For best performance, some database operations are done wit*
repository.stats.optimization	***Boolean***	*When is enabled the **Administration** > Statistics results need l calculated.*
repository.content.checksum	***Boolean***	*When is enabled is calculated the document checksum for eac*
system.pdf.force.ocr	***Boolean***	*Force to processing all PDF files with the OCR engine.*
stack.metadata.hidden.properties	***List***	*Filter metadata fields at metadata stack view on frontend UI.*
show.records.in.tree	***Boolean***	*When is enabled show record nodes at tree stack view on fron*
system.catdoc.xls2csv	***String***	*Native XLS to CSV converter tool. Used in **NativeMsExcelTe***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		*Ubuntu based distros is included in the **catdoc** package.* `/usr/bin/xls2csv ${fileIn}`
system.maintenance	***Boolean***	*When is enabled the application goes to "maintence mode", th okmAdmin user will be able to login.*
system.strings	***String***	*Configure Strings tool.* `/usr/bin/strings ${fileIn}`
system.zbar	***String***	*Configure bar code extractor tool.* `/usr/bin/zbarimg`
system.profiling	***Boolean***	*The Profiling stats feature gives low level value information to application performance problems.* *You also can enable profiling from **Administratio***
system.ftp.port	***Integer***	*FTP port.* *On linux only root user is able to binding ports lik good idea binding ftp at another port.*
system.ftp.server	***Boolean***	*When is enabled the application enables the FTP server featur*
user.keywords.cache	***Boolean***	*When enabled the keyword of the nodes are cached.* *The feature provide best perfomance with less dat*
user.item.cache	***Boolean***	*When enabled the application cached the number of nodes ow* *The feature provide best perfomances with less da the nodes owned by each user.*

max.search.clauses	***Integer***	*Set the maximum number of allowed clauses in a search.*
max.search.results	***Integer***	*Set the limit of the results returned by a search to the repositor*
search.suggestion.minimal.results	***Integer***	*The minimal search results to provide suggestions. By default*
search.suggestion.returned.words	***Integer***	*The number of suggested words. By default is 3.*
checkin.comment.mandatory	***Boolean***	*Force to set comment when uploading new document (create*
stack.metadata.adapter	***String***	*Set the metadata stack implementation.* `com.openkm.plugin.stack.metadata.GenericStackMeta` *More information at Creating your own Stack Me*

system.diffpdf	***String***	*Configure DiffPDF tool.*
		Windows
		You need the command line version from https://www.qtrac.eu
		`C:\openkm\extra\diffpdf\diffpdf.exe -qr ${fileOut} ${fileIn`
		Linux
		`/home/openkm/tomcat-8.5.63/bin/diffpdf ${fileIn1} ${file`
		More info at Diffpdf raises ConversionException error while l
		libpoppler-qt4.so.4
		Bash script sample
		`#!/bin/bash`
		`#echo "p1: $1";`
		`#echo "p2: $3";`
		`#echo "p2: $3";`
		`#echo "library path: $LD_LIBRARY_PATH";`
		`export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/c`
		`/opt/openkm/tomcat/bin/diffpdf $1 $2 $3`
		`#unset LD_LIBRARY_PATH`

version.append.download	***Boolean***	*Append the document version in the document file name when OpenKM.*
webservices.visible.properties	***List***	*List of configuration properties what are granted to view by us from webservices.*
rss.news	***Boolean***	*Enable or disable new in the login page.*
rss.news.box.width	***Integer***	*Width of the news box in the login page.* *300*
rss.news.max.size	***Integer***	*Maximum number of news retrieved from the news server.* *10* *News are cached from OpenKM server and refres means any change on this parameter will take effe OpenKM service.*
rss.news.visible	***Integer***	*Number of visible news at the same time.* *1*
trash.retention.days	***Integer***	*It indicates the maximum number of days nodes in the trash w purge will be executed. Default value is **0 indicates unlimited**.* *There's a Crontab task named "Purge Old Trash* *30*
file.plan	***Boolean***	*Enable or disable the file plan feature.* *false*
decimal.pattern	***String***	*Set the decimal pattern.* *.*

browser.natural.sort	***Boolean***	*Enable or disable natural sort.* *Natural sort is done at database engine level.* *For MySQL and MariaDB download and install r database.* *For another database engine, you need to create y "**okmNaturalSort(String, Integer, String)**".*
min.free.disk.percent	***Integer***	*This is the minimum percentage used to alert about low free di* *20*
search.by.attachment.parent	***Boolean***	*Enable or disable the search of documents in emails (attachm* *false*
property.group.element.default.access	***Boolean***	*When is set to true all the metadata fields are shown to the use metadata fields are shown.* *Sample scenarios:* • *When value is set to **true,** all the **metadata fields are s** scenario have sense using **revokeUser** and **revokeRol*** • *When value is set to **false,** all the **metadata fields are** scenario have sense using **grantUser** and **grantRole**.* *Additional information at Metadata Security elem* *true*

Experimental configuration parameters

Property	*Type*	*Description*
unit.testing.folder	***String***	*Folder used by testing process.* */okm:root/okmTesting*
unit.testing.user	***String***	*User used for unit testing.* *okmAdmin*
unit.testing.password	***String***	*Password of the user used for unit testing.* *admin*
unit.testing.url.base	***String***	*The unit test URL.* *http://localhost:8080/OpenKM/*
extension.digicert.base.path	***String***	*OpenKM base folder:* */okm:root/invoices*
extension.digicert.keystore.file	***String***	*Path of the keystore file.* */home/openkm/certificate.p12*
extension.digicert.keystore.password	***String***	*The password of the keystore.*
extension.digicert.keystore.type	***String***	*The keystore type (JKS or PKCS12)* *JKS*
extension.digicert.private.key.alias	***String***	*The alias of the certificate.* *b138bef79c0104378101b77b71922c61_9463f621-75bc-4*
extension.digicert.signature.algoritm	***String***	*The algorithm of the certificate.* *rsa*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

extension.digicert.type	***String***	*Values can be "**month**" or "**quarterly**".* *month*
extension.digicert.upload.base.path	***String***	*OpenKM path where the documents are upload.* */okm:root/invoices/upload*
extension.digicert.validate.properties	***String***	*Metadata group fields.* *okp:digicert.num.recepcion* *okp:digicert.num.factura* *okp:digicert.fecha.factura* *okp:digicert.nif.emisor* *okp:digicert.iva.general* *okp:digicert.iva.reducido* *okp:digicert.iva.superreducido* *okp:digicert.base.general* *okp:digicert.base.reducido* *okp:digicert.base.superreducido* *okp:digicert.base.siniva* *okp:digicert.total* *okp:digicert.year* *okp:digicert.period*
rss.news	***Boolean***	*Enable or disable new in the login page.*
rss.news.box.width	***Integer***	*Width of the news box in the login page.* *300*
rss.news.max.size	***Integer***	*Maximum number of news retrieved from the news server.* *10* *News are cached from OpenKM server and refreshed any change on this parameter will take effect tomorro service.*

rss.news.visible	***Integer***	*Number of visible news at the same time.* *1*
check.node.deletion.integrity	***Boolean***	*Enable or disable node deletion integrity check. For example, if th any of their children can be deleted, in case of folders or records. should only disable it in case of deletion of massive nodes.*
okm.content.security.policy	***String***	*By default embedding the application in an iframe is only allowed required because used in the web UI. This is the default value:* *frame-ancestors 'self'* *The next sample shows how to enable other domains:* *frame-ancestors 'self' http://example.com* *The configuration parameter must be set in the **openk** the OpenKM service to take effect.* *More information about content security policy https: US/docs/Web/HTTP/Headers/Content-Security-Policy*
okm.cors.configuration.policy	***String***	*By default, CORS is enabled to all the origins, methods, and head* *[{"path":"/rest/**", "origins":["*"], "methods":["*"], "headers":[["*"], "methods":["*"], "headers":["*"]}]* *The next sample shows how to only enable access to the REST AP others.* *[{"path":"/rest/**", "origins": ["http://domain.com","https://domain.com","http://domain2. "methods":["*"], "headers":["*"]}, {"path":"/**", "origins":["http ["*"], "headers":["*"]}]* *The configuration parameter must be set in the **openk** the OpenKM service to take effect.* *The String value is a list of objects in a json format. T relevant because internally the first match will be app "/**" should only be applied always at the end.* *More information about CORS at https://developer.m US/docs/Web/HTTP/CORS*

ldap.ignore.partial.result.exception	***Boolean***	*When AD integration by default the warnings of PartialResultExc* *When true these warnings are not shown in the log.* *true*
okm.authentication.external.authentication	***String***	*Enable the OpenKM external authentication. This feature allows t* *authenticated by third-party authentication providers like OAUTH* *okm.authentication.external.authentication=http://localhost*
rest.client.call.timeout	***String***	*Set the maximum request timeout in seconds when the OpenKM ap* *to third-party application. For example a jbpm-console request.* *15*
optimized.thumbnail.generation	***Boolean***	*Uses a new improved way to generate thumbnails. Useful when us* *true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AutoCAD configuration parameters

These parameters are used in AutoCAD integration.

Property	*Type*	
extension.cadviewer.autoxchange	***String***	*Tailormade AutoXchange AX2020 tool. This tool is used to convert a given* `/home/openkm/tomcat-9.0.76/bin/ax2020`
extension.cadviewer.linklist	***String***	*Tailormade LinkList tool. This tool is used to extract text, annotations from* `/home/openkm/tomcat-9.0.76/bin/linklist`
extension.cadviewer.dwglist	***String***	*Configure DWGList tool. This tool is used to extract XRefs from a given DW* `/home/openkm/tomcat-9.0.76/bin/dwglist`
extension.cadviewer.sibling.xrefs	***Boolean***	*When this parameter is set to* ***true*** *the AutoCAD files's references have to be keep on mind that you should avoid document name duplication. There is ar*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

FTP configuration parameters

Field / Property	Type	Description
system.ftp.server	**Boolean**	*Enable FTP service. By default is disabled. You have to restart OpenKM after changing this parameter.* *false*
system.ftp.port	**Integer**	*The port where the FTP service is listening. By default is 2121 although the default FTP port is 21 because it's a reserved port and OpenKM should be executed as an administrator to be able of using this port. You have to restart OpenKM after changing this parameter.* *2121*
system.ftp.data.port.range	**String**	*Port range used in the server for data transmission. You should also open these ports in your firewall. By default the range is 60000 to 60500 and they have to be separated by a colon. You have to restart OpenKM after changing this parameter.* *60000:60500*
system.ftp.external.address	**String**	*Set the IP Address to use when FTP connection are proxied. Empty by default.*
system.ftp.debug	**Integer**	*Set the FTP debug flags. The greater the number, the greater the debug info show.* *0*



If you change any of these configuration parameters it won't take effect until OpenKM have been restarted.

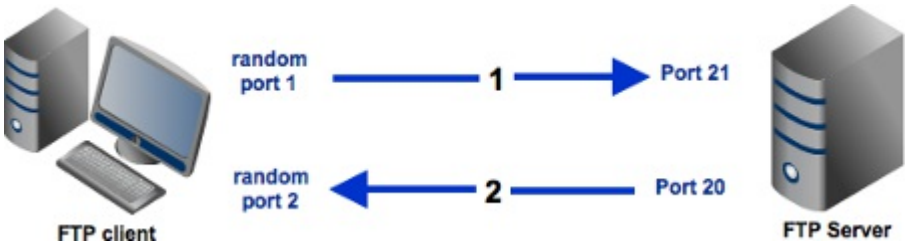


*CIFS's FTP works better in **Passive Mode**. So, it might not work properly with file browsers such as Nautilus or Nemo.*

Active Mode

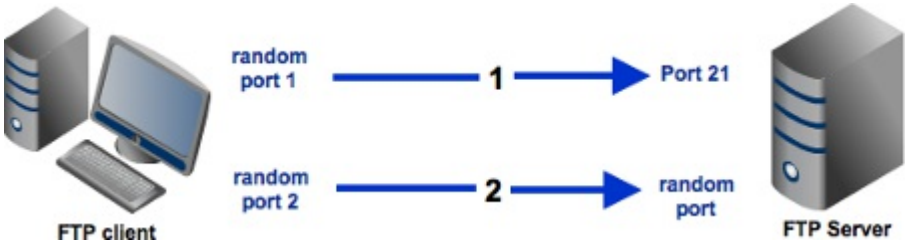
When set to active mode, the server will try to establish a connection to a client-side port chosen at random. Since this port is

not likely to be one of the predetermined ports, the firewall will block any attempt to connect to it, resulting in no connection being established.



Passive Mode

A passive configuration will not have this problem since the client will be the one initiating the connection. Of course, it's possible for the server side to have a firewall too. However, since the server is expected to receive a greater number of connection requests compared to a client, then it would be but logical for the server admin to adapt to the situation and open up a selection of ports to satisfy passive mode configurations.



More info at:

- *What is the difference between active and passive FTP?*
- *Active FTP vs. Passive FTP, a Definitive Explanation*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Two factor authentication configuration parameters

In this section you can find the most common configuration parameters related to two factor authentication.

Field / Property	*Type*	*Description*
two.factor.authentication	***Boolean***	*Enable or disable two factor authentication functionality. False by default.*
two.factor.authentication.domain	***String***	*Domain name used to configure Google Authenticator. The defaul value is "openkm.com".*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Multi-tenant configuration parameters

You can have some configuration parameters which are global for all the tenants, and some which are related to a given tenant. How can you do that? If you go to ***Administration > Configuration*** *you are able to see a option in the top right corner called "Tenant configuration". If you click in this option you will have access to these specific tenant configuration properties. By default it is empty, so you only can see an empty list.*

By clicking in the green plus icon you can create a new configuration property. For example, we are going to configure a different login image for each tenant. To do so follow these steps:

- *Go to the desired tenat (Menu -> tenant)*
- *Create a* ***File*** *property called* ***logo.login*** *and choose an image.*
- *Create a* ***String*** *property called* ***tenant.domain*** *and write the domain used by this tenant.*

To get this working, this tenant should be accessed by a given domain. For example, let's imagine you have a tenant called "Floppy" and you have the server configured to access the OpenKM instance using this domain floppy.okm-instance.com, so in this case the "tenant.domain" value should be "floppy.okm-instance.com". To create different alias you have to create the subdomain in your DNS server and make it to resolve to the real server IP. When OpenKM detects it is being accessed by this domain, it will show the configured ***logo.login*** *image.*

The same works for:

- ***logo.favicon***
- ***logo.mobile***
- ***logo.report***
- ***logo.tiny***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

File plan

A file plan is a comprehensive outline that includes the records series, file organization, active file locations, file transfer instructions, file retention, and disposition instructions, and other specific instructions that guide the effective management of records, including vital records.

File plans specify how records are to be organized once they have been created or received, provide a "roadmap" to the records created and maintained by an organizational unit, and facilitate dispositioning of the documents.

Enable the File Plan

Mandatory

Change the following OpenKM configuration parameters:

Field / Property	*Type*	*Description*
security.access.manager	***String***	*com.openkm.plugin.access.DbFilePlanAccessManagerWrapper*
file.plan	***Boolean***	***true***

After you update the values, you must restart the OpenKM service.

Optional

In the profiles, we suggest the following changes:

- *In the "**Components tab,**" disable the "**Security checkbox**" from all the tabs (Tab document, Tab Folder, Tab mail, Tab record).*
- *In the "**Components tab,**" enable the "**Bussiness activity**" checkbox.*
- *In the "**Menu tab,**" enable the "**File plan**" checkbox.*
- *In the "**Other tab,**" enable the "**File plan**" checkbox.*

How to construct the File Plan

After inventorying all of the records maintained by your office, you are ready to create your office file plan. A file plan describes every type of record in the organization, where they should be stored, the rules applied to them, the retention schedule and timeline, the manner of their disposition, and the person or persons responsible for their management.

A file plan usually consists of the following:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Description of the kinds of documents identified to be recorded.*
- *The classification of the records into a wide range of categories.*
- *Details about the location of the stored records.*
- *Description of the retention schedule and period.*
- *Making the distinction and assigning responsibility for certain types of documents.*

In the file plan, every document must have the following:

Field	*Type*	*Sample*
Name	***String***	*Records common*
Code	***Number***	*101*
Definition	***Document type***	• *Document* • *Mail* • *Record* • *Electronic record (record)*
Section	***Section***	*100- Common section*
Property group	***Metadata***	*okg:technology* *okg:consulting*
Security access Manager	***Plugin***	*Default access manager*
Security by user	***Users***	*okmAdmin*
Security by roles	***Roles***	*ROL_USER*
Catalog manager	***Plugin***	*Business catalog manager*
Record manager	***Plugin***	*Default record manager*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Disposition manager	***Plugin***	*EmptyFilePlanDispositionManager*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Search configuration parameters

All of these parameters must be set on the $TOMCAT_HOME/openkm.properties file.

After updating the openkm.properties file the OpenKM service must be restarted to take effect.

Property	*Type*	*Description*
spring.jpa.properties.hibernate.search	***Boolean***	*This parameter can be used to disable Hibernate Search completely. This means you won't be able to search documents or folders. By default, the search engine is enabled.* *true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

MIME types

The application supports several documents MIME type. Every MIME type has one or several file extensions which are used to match the uploaded document MIME type.

> *Based on the MIME type associated to the uploaded document, the application chooses the associated text extractor. Text extractor works only for a few MIME types that can easilly be extended.*
>
> *For example, if you upload a "sample.doc", it's detected MIME type is "application/msword" because its extension is ".doc". In this case, the text extractor class is "com.openkm.extractor.MsWordTextExtractor".*

The detected document MIME type is also used to display an icon which represents the type of document. You can change this icon if you like.

There is list of common documents MIME types already registered in OpenKM.

> *If you upload a document which MIME type is not in the list, it will be detected as "application/octet-stream".*

Add new MIME type

- *Click **new MIME Type** icon*  .
- *In Create mime type view **fill the required fields**:*

Field	*Value*
Name	*The MIME type name* *application/epub+zip*
Description	*General purpose description.* *EPUB*
Extensions	*Associated document extensions.* *In case multiple extensions, each one should be separated by character ","* *epub*
Image	*Image file.*

	Image size should be 16x16 pixels.
Search	*When is checked it means it will be shown in the application user interface at search view MIME type list.*
Live Edit	*When is checked it means that this mime type will be edited using Live Edit when enabled.*

- *Click **Create button**.*

Edit MIME Type

- *Click **Edit icon**  .*
- *In **Edit MIME type view** make changes in fields.*
- *Click **Edit button**.*

Delete MIME type

- *Click **Delete icon**  .*
- *In **Delete MIME type view** click **Delete button**.*

Backup MIME types

- *Click at top right the **Export Mime type icon**  .*
- *After a few seconds the application will download an SQL file.*

New MIME types

- *Go to the right up corner to see the action buttons in the table.*
- *Click  New **MIME types button**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Statistics

In the statictics view, a graphical information about repository size and it's distribution on the distinct root nodes (taxonomy, personal, trash, etc.) is shown. Also is shown the basic server information like memory and disk usage.

To understand how your repository is growing and making hard disk estimations, these pie charts are interesting:

- *Documents by context.*
- *Documents size by context.*
- *Folders by context.*
- *Mails by context.*
- *Record by context.*

Some pie charts describe the distribution by context. These contexts are:

- *Taxonomy (the /okm:root context)*
- *Template (the /okm:template context, where you can place your document templates)*
- *Personal (the /okm:personal context, where each OpenKM user can stores their personal - not shared - documents)*
- *Mail (the /okm:mail context, where the user mails are stored by default)*
- *Trash (the /okm:trash context, which stores the deleted document, folders, etc.)*



If the trash context is too large the trash bin should be emptied. Once you are sure these documents and folders can be really deleted, you should empty your trash or individually purge the documents. Remember that each user has its own trash and every user is responsible of purging their own one. There's a feature ***Administration*** *>* ***Utilities*** *>* ***Purge trashes*** *that helps on it.*

Check the status of the Pending task queue

Pending tasks are only used in some application sections where an user action can takes longer than usual.

- *Click at top right the menu option* ***Pending task queue****.*
- *In* ***Pending task queue view*** *is shown the status of the tasks. This is a list of the task to be processed or in progress.*



For example, if you want to change the permissions in a folder with lots of documents and / or a very depth structure this actions will be promoted to pending task. Meanwhile the user can do more thinks while this one is executed in background. If the number of document to process is huge, this task may take hours to complete.

Check the status of the Text extraction queue

When upload a document to the application, its content is not extracted inmediatelly but enqueued for this.

- *Click at top right the menu option* ***Pending text extractor queue****.*

- *In **Pending text extractor queue view** is shown the status of the text extractor process. This is a list of the documents to be processed or in progress.*

More information at Performance configuration parameters.



Sometimes you might be interested on process all the documents again into the text extraction queue (you have introduced new text extractor, etc ...), you can find more information at Database description about how doing it.

Check the status of the Antivirus check queue

When upload a document to the application, the antivirus analysis can be done just in time or enqueued for this.



This options is only shown when the application has been configured to antivirus analyze on batch queue.

- *Click at top right the menu option **antivirus check queue**.*
- *In **Antivirus check queue view** is shown the status of the antivirus analyze process. This is a list of the documents to be processed or in progress.*

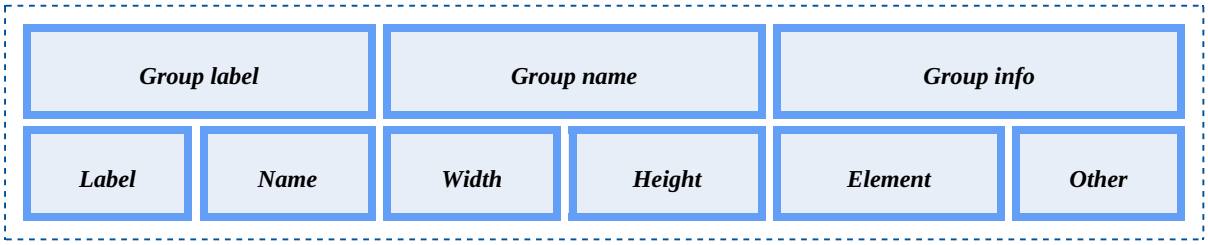
More information at Performance configuration parameters.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata

In the Metadata view are shown the registered metadata groups. In this view, it can create, modify or remove metadata definitions.

Property groups are shown in a table structure. First row (three columns) corresponds to the general Property group definition. The next rows are fields definition in Property group.



Description:

Property	*Description*
Group label	*Group label is shown in the user interface at Property group tab.*
Group name	*Unique group name identifier.* *Two property groups cannot have the same name.* *Please, use only letters, numbers and underscore: "0-9a-zA-Z_".*
Group info	*Shows information about:* • *Visibility (property group is hidden from user interface).* • *Read only (property group can not be modified from user interface).*
Field label	*Field label is shown in the user interface in the Property group tab,*
Field name	*Unique property group field identifier.* *Two property group fields can't have the same name.* *Please, use only letters, numbers and underscore: "0-9a-zA-Z_".*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Field width	*This parameter is used for rendering the width of the field in the user interface view.*
Field height	*This parameter is used for rendering the height of the field in the user interface view.*
Element	*This parameter sets the element type. It's used for rendering several values in the field in the user interface view.* *Values:* • *CheckBox* • *Input* • *Iframe* • *Select* • *Separator* • *SuggestBox* • *Text* • *TextArea*
Others	*Shows field extra information.*

Edit metadata

To add, modify or delete metadata there's only one operation for all, edit the metadata xml definition.

- *Click on*  ***Edit metadata***.
- *In Edit view,* ***modify the XML***.
- *At bottom right, click* ***Edit button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata XML definition

The application metada is based on a formal XML definition. The DTD (Document Type Definition) defines the structure and the legal elements and attributes of an XML document.

*You can see latest DTD in our website http://www.openkm.com/dtd/property-groups-**{major}**.**{minor}**.dtd where **{major}** and **{minor}** values are two numbers (major release and minor release). For example the location for version 3.10 of the DTD is http://www.openkm.com/dtd/property-groups-3.10.dtd*

Each application version supports only a couple of DTD definitions, read Application version compatibility table to see which are supported by your application.

Basic XML skeleton

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>

</property-groups>
```

- *The **DOCTYPE** is a formal definition of some DTD. In the previous example the **property-groups-3.10.dtd**.*
- *The XML begins with <**property-groups**> tag and ends with </**property-groups**>. Into this hicheracy are defined the metada groups.*

Metadata group definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">

  </property-group>
</property-groups>
```

- *Each <**property-group**>...</**property-group** block identify the **begining and end** of new **metadata group definition**.*
- *Each metadata group has a **label** - used by user interface - **Consulting**.*
- *Each metadata group has a **unique name**, **okg:consulting**.*
- *The metadata group name always must starts with "**okg:**" (**O**pen**KM** **G**roup).*
- *Must not use special characters for name.*

Attribute	*Description*	*Required*
		true

label	*The text shown as a label in the user interface.*	
name	*Unique field identifier.* *Two metadata groups can't have the same name.* ***Name must be unique****.* *The name must start with "**okg:**".* *Please, use only letters, numbers and underscore: "0-9a-zA-Z_".* *We recommend "snake_case" better than "camelCase" for better columns names.*	*true*
visible	*Show or hide the metadata.* *Allowed values are:* • *true.* • *false.* *By default the metadata are visible.*	*false*
readonly	*Metadata values can not be modified from user interface only from API.* *Allowed values are:* • *true.* • *false.* *By default the metadata are modificable.*	*false*
defaultValueClassName	*Set custom JAVA class for setting initial default values.*	*false*

Basic example

Available field types:

- *Checkbox*
- *Input*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Iframe*
- *Select*
- *Separator*
- *Suggestbox*
- *Text*
- *TextArea*

More information about field type elements at [Metadata fields]

> *It's a good practice to use the metadata group name as the basis to create metadata field name. For example* ***okp:consulting.code*** *use* ***okg:consulting*** *as base (changing* ***okg*** *to* ***okp*** *). That helps to find out which is the metada group asociated to each metadata field.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

Using file system DTD

If the server has no access to the Internet, then the DTD file must be accessible in some way, for example can store a copy in your application server.

> *The XML definition must be changed to refer to the new DTD location, otherwise you will get an application error while processing DTD file.*
>
> *Can be a file system (example below) or a valid intranet URL.*

```
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "file:///home/openkm/property-groups-3.10.dtd">
```

- *In the example,* ***/home/openkm/property-groups-3.10.dtd*** *is the file system path to the file.*

How metadata is stored into OpenKM

When you register a new metadata definition in OpenKM the metadata values are stored in two separated tables. One table for current metadata values and the other for history metadata values.

Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

After registering the previous metadata definition will be created two tables with these columns:

- ***OKM_PGRP_CUR_CONSULTING** (used for current metadata values)*
  - ***RGT_UUID**: ID of the node which has this metadata*
  - ***RGT_PRO_INPUT1**: Value of property okp:consulting.input1*
  - ***RGT_PRO_INPUT2**: Value of property okp:consulting.input2*
  - ***RGT_PRO_SEPARATOR**: Value of property okp:consulting.separator*
- ***OKM_PGRP_HIS_CONSULTING** (used for history metadata values)*
  - ***RGT_UUID**: ID of the node which has this metadata*
  - ***RGT_VERSION**: ID of the related version*
  - ***RGT_PRO_INPUT1**: Value of property okp:consulting.input1*
  - ***RGT_PRO_INPUT2**: Value of property okp:consulting.input2*
  - ***RGT_PRO_SEPARATOR**: Value of property okp:consulting.separator*



*The prefix used for all the columns will be **RGT_**.*

*The column with name **RGT_UUID** contains the value of the node UUID.*



Be careful with the Oracle table name and column name 32 characters limitation.

We recommend "snake_case" better than "camelCase" for better columns names.

*For example this definition would create the table field **RGT_PRO_RETURNPLACE**:*

```
<input label="Return place" name="okp:sample.returnPlace" />
```

*But this one would create the table field **RGT_PRO_RETURN_PLACE**, which is more human readable:*

```
<input label="Return place" name="okp:sample.return_place" />
```

Default column size by type

Type	Size

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Input	*128 characters*
Select	*128 characters*
SuggestBox	*128 characters*
TextArea	*256 characters*

In case you have metadata values larger should increase the value of the column with the attribute " ***dbColumnSize****". See the sample below:*



*The value of the attribute "**dbColumnSize**" will only be applied the first time you register the metadata group or each time you add a new file. After the column has been created in the database, although you change the value "**dbColumnSize**" of the attribute, the column size will not be updated. In case you want to update the column size after creation, should apply changes from database client directly.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" dbColumnSize="256"/>
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" dbColumnSize="512"/>
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata fields

These are the available field types:

Field type	*Description*	*Visible in the metadata stack panel*
Checkbox	*Metadata Checkbox field*	*Yes.*
Input	*Metadata Input field*	*No.*
Iframe	*Metadata Iframe field*	*No.*
Select	*Metadata Select field*	*Yes.*
Separator	*Metadata Separator field*	*No.*
Suggestbox	*Metadata Suggestbox field*	*Yes.*
Text	*Metadata Text field*	*No.*
TextArea	*Metadata TextArea field*	*No.*
VerticalPanel	*Metadata VerticalPanel field*	*No.*
HorizontalPanel	*Metadata HorizontalPanel field*	*No.*

Some field types can have children elements, validator or option elements. This hierarchy is permitted:

- *metadata group*
  - *Checkbox*
    - *Validator*
  - *Input*
    - *Validator*
  - *Iframe*
  - *Select*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Option*
- *Validator*

- *Separator*
- *Suggestbox*
  - *Validator*
- *Text*
- *TextArea*
  - *Validator*
- *VerticalPanel*
  - *CheckBox*
  - *Input*
  - *Iframe*
  - *Select*
  - *Separator*
  - *Suggestbox*
  - *Text*
  - *TextArea*
  - *VerticalPanel*
  - *HorizontalPanel*
- *HorizontalPanel*
  - *CheckBox*
  - *Input*
  - *Iframe*
  - *Select*
  - *Separator*
  - *Suggestbox*
  - *Text*
  - *TextArea*
  - *VerticalPanel*
  - *HorizontalPanel*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata Checkbox field

Checkbox

Best used to represent boolean data:

Attribute	Description	Required
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* *Two metadata can't have the same name and **must be unique**.* *The name must start with "**okp:**".* *Using a metadata group name as a basis for a metadata field name is a good practice. For example use "**okg:consulting**" has been used for the field name "**okp:consulting**.checkbox", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
value	*Specifies a default value for this component.* *The allowed values are **true** and **false**.*	*false*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
description	*General description.*	*false*
readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*
dbColumnSize	*Set the size of the database field where the value is stored.*	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	By default, fields are stored in a database column with 128 characters in size. Because your data length might be bigger than 128 characters, the administrator should update the default column size. *Take a look at the Metadata XML definition documentation section for a deep explanation of metadata storage in the OpenKM database,*	
defaultAccess	*Default security access policy. By default is set to "grant" so all users will see the property.*	

Child elements:

- ***Validator*** *(more information at Metadata Validator element)*
- ***Security*** *(more information at Metadata Security element)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <checkbox label="Checkbox label" name="okp:consulting.checkbox" />
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata Iframe field

The application will render into an iframe object the selected web url. In the url will be passed as query parameters the uuid and propertyGroup.

Attribute	*Description*	*Required*
label	*The text shown as label in user interface.*	*true*
name	*Unique field identifier* *Two metadata can't have the same name.* ***Name must be unique****.* *The name must start with "**okp:**".* *It's a good practice to use a metadata group name as basis for the metadata field name. For example use "**okg:consulting**" has been used for field name "**okp:consulting**.iframe", changing "**okg**" to "**okp**". That helps to find out the metada group based on metadata field name.* *Please, use only letters, numbers and underscore: "0-9a-zA-Z_".*	*true*
url	*Set the url of the web page to be load in the iframe.* *There're set two parameters to the request:* • *uuid (the node of the actual object selected in the user interface).* • *propertyGroup (the name of the metadata group).* *If your url value is http://www.openkm.com the url called into iframe will be http://www.openkm.com?uuid=03ced012-63cc-4feb-ac26-bb12ded86f73&propertyGroup=okg:consulting* *It's a good practice the URL starts with http:// or https://*	*true*
width	*The width of the HTML element. It is advisable to set the attribute width to 100% so that it is adjusted to the screen*	*false.*
height	*The height of the HTML element.*	*false.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <iframe label="Iframe label"  name="okp:consulting.iframe" url="http://localhost:8080/openkm" width="100%" heig
  </property-group>
</property-groups>
```

Metadata Select field

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two metadata can't have the same name. The name **must be unique**.* *The name must start with "**okp:**."* *Using the metadata group name as a basis for the metadata field name is a good practice. For example, use "**okg:consulting**" has been used for the field name "**okp:consulting**.select", changing "**okg**" to "**okp.**" That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
type	*Type value. Allowed types:* • *Simple (single choice allowed).* • *Multiple (multiple selections allowed).*	*true*
width	*The width of the HTML element.* *In the case of the KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	
optionsQuery	*A metadata query is used to get the selected options.*	*
className	*A class is used to get the selected options.*	*
	The application can automatically suggest values based on document content analysis when adding metadata to documents. By default, the application has some implementations that can easily be extended.	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

suggestion	• ***com.openkm.form.suggestion.DocumentContentContainsSuggestion****: Gets recommendations based on the document's content (use String.contains)* • ***com.openkm.form.suggestion.DocumentContentTokenizerSuggestion****: Gets suggestions based on the document's content (Use StringTokenizer).* • ***com.openkm.form.suggestion.DocumentTermsSuggestion****: Gets recommendations based on the document's term vectors.* *For more information, read Creating your own Suggestion Analyzer.*	*false*
parentElement	*Parent field name value. In this case, the value is hierarchical with other field values (the parent).*	*false*
description	*General description.*	*false*
read-only	*The set field cannot be modified in the user interface. By default, the value is "**false**."*	*false*
dbColumnSize	*Set the size of the database field where the value is stored.* *By default, fields are stored in a database column with 128 characters in size. Because your data length might be more extensive than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a profound explanation of metadata stored in the OpenKM database,*	
defaultAccess	*Default security access policy. By default is set to "grant" so all users will see the property.*	

Child elements:

- *Option. Couples of XML* ***<option label="label text" value="unique identifier value"/>*** *values or* ***<option label="label text" value="unique identifier value" parentValue="parent unique identifier value"/>*** *in the case is set a hierarchical relationship.*
- ***Validator*** *(more information at Metadata Validator element)*
- ***Security*** *(more details at Metadata Security element)*

***** *There're three options to get option values:*

- *Directly defined in XML.*
- *Metadata table (optionsQuery).*
- *className.*



One of these options must be defined; otherwise, you will get an error. It can not be done as a combination of several; only one can be chosen.

Basic select example

- *Type is equal to* ***simple****.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="select label" name="okp:consulting.select" type="simple">
      <option label="one" value="001" />
      <option label="two" value="002" />
      <option label="three" value="003" />
    </select>
  </property-group>
</property-groups>
```

Allow multiple choices

- *Type is equal to* ***multiple****.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="select label" name="okp:consulting.select" type="multiple">
      <option label="one" value="001" />
      <option label="two" value="002" />
      <option label="three" value="003" />
    </select>
  </property-group>
</property-groups>
```

Based on SQL query

We can also get these option values using a SQL query. This SQL sentence should be put on the ***optionsQuery*** *attribute, but the* ***table*** *attribute should be empty (otherwise, this query is a metadata query). Note that the first element returned in every row is the option value, and the second is the option label.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
                    "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Location" name="okg:location">
    <select label="Country" name="okp:location.country" type="simple"
          optionsQuery="select CT_ID, CT_NAME from COUNTRY order by CT_NAME" />
  </property-group>
</property-groups>
```

The table used in this sample is defined as follows:

```
create table COUNTRY (
  CT_ID varchar(2),
  CT_NAME varchar(32),
  primary key(CT_ID)
);

insert into COUNTRY (CT_ID, CT_NAME) values ('es', 'Spain');
insert into COUNTRY (CT_ID, CT_NAME) values ('pt', 'Portugal');
insert into COUNTRY (CT_ID, CT_NAME) values ('it', 'Italy');
```

Based on className

To help get a list from external applications or other purposes, your class can be built to retrieve the option values from those systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                    "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="select label" name="okp:consulting.select" type="simple"
          className="com.openkm.plugin.form.values.OptionSelectUserList" />
  </property-group>
</property-groups>
```

For more information, read Creating your own Option Select Values plugin.

Suggested options

Based on the content analyzer, the application can suggest some options values. At the user interface, the recommended options will be shown in green.

The application has some implementations that can easily be extended.

- ***com.openkm.form.suggestion.DocumentContentContainsSuggestion**: Get recommendations based on document content (use String.contains)*
- ***com.openkm.form.suggestion.DocumentContentTokenizerSuggestion**: Get suggestions based on document content (Use StringTokenizer).*
- ***com.openkm.form.suggestion.DocumentTermsSuggestion**: Get directions based on document term vectors.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

For more information, read Creating your own Suggestion Analyzer.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="select label"  name="okp:consulting.select" type="multiple"
          suggestion="com.openkm.form.suggestion.DocumentContentContainsSuggestion">
      <option label="one" value="001" />
      <option label="two" value="002" />
      <option label="three" value="003" />
    </select>
  </property-group>
</property-groups>
```

Hierarchical relationship

- *The field options values of "**okp:consulting.children**" have parent relation with "**okp:consulting.parent**". It's defined by **parentElement="okp:consulting.parent"**.*
- *The option value "**benefits**" of the field "**okp:consulting.children**" has a parent value "**directives**" from the field "**okp:consulting.parent**".*

Tree relationships:

- *directives*
  - *benefits*
  - *eligibility*
  - *services*
- *handbook*
  - *basic*
  - *normal*
  - *advanced*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="Parent" name="okp:consulting.parent" type="simple">
      <option value="directives" label="Directives" />
      <option value="handbook" label="Handbook Paragraphs" />
    </select>
    <select label="Children" name="okp:consulting.children" type="simple" parentElement="okp:consulting.parent">
      <option value="benefits" label="Benefits" parentValue="directives" />
      <option value="elegibility" label="Eligibility" parentValue="directives" />
      <option value="services" label="Services" parentValue="directives" />
      <option value="basic" label="Basic" parentValue="handbook" />
      <option value="normal" label="Advanced" parentValue="handbook" />
      <option value="advanced" label="Normal" parentValue="handbook" />
    </select>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
  </property-group>
</property-groups>
```

Hierarchical relationship with SQL

We use a couple of tables to get the information in this case.

```
create table LOCALIDAD (
  LOC_ID varchar(8),
  LOC_NAME varchar(32),
  primary key (LOC_ID)
);

create table UBICACION (
  UBI_ID varchar(8),
  UBI_LOC varchar(8),
  UBI_NAME varchar(32),
  primary key (UBI_ID)
);

insert into LOCALIDAD (LOC_ID, LOC_NAME) values ('l0', 'Localidad 0');
insert into LOCALIDAD (LOC_ID, LOC_NAME) values ('l1', 'Localidad 1');
insert into LOCALIDAD (LOC_ID, LOC_NAME) values ('l2', 'Localidad 2');


insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u0', 'l0', 'Ubicación 0');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u1', 'l0', 'Ubicación 1');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u2', 'l0', 'Ubicación 2');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u3', 'l1', 'Ubicación 4');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u4', 'l1', 'Ubicación 5');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u5', 'l1', 'Ubicación 6');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u6', 'l2', 'Ubicación 7');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u7', 'l2', 'Ubicación 8');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u8', 'l2', 'Ubicación 9');
```

This is the form definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.7//EN"
                          "http://www.openkm.com/dtd/property-groups-3.7.dtd">
<property-groups>
  <property-group label="Sample" name="okg:sample">
    <!-- Parent child -->
    <select label="Localidad" name="okp:sample.localidad"
          optionsQuery="select LOC_ID, LOC_NAME from LOCALIDAD" />
    <select label="Ubicacion" name="okp:sample.ubicacion" parentElement="okp:sample.localidad"
          optionsQuery="select UBI_ID, UBI_NAME, UBI_LOC from UBICACION" />
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata Separator field

Draw a separator between fields.

Attribute	Description	Required
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* *Two metadata can't have the same name. The name **must be unique**.* *The name must start with "**okp:**".* *Using a metadata group name as a basis for a metadata field name is a good practice. For example use "**okg:consulting**" has been used for the field name "**okp:consulting**.separator", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata Suggestbox field

This is an implementation of the typical AJAX suggest box component:

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* ⚠ *Two metadata can't have the same name. The name **must be unique**.* *The name must start with "**okp:**".* *Using a metadata group name as a basis for a metadata field name is a good practice. For example use "**okg:consulting**" has been used for the field name "**okp:consulting**.suggestbox", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
description	*General description.*	*false*
filterQuery	*A SQL query to filter the data. Use {0} to place the user input.*	*
valueQuery	*A SQL query to obtain the value of an identifier. Use {0} to place the identifier.*	*
className	*A class is used to get the selected options.*	*
filterMinLen	*The minimum chars in input to begin to filter results.*	*true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

dialogTitle	*The title of the HTML dialog.*	*true*
readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*
dbColumnSize	*Set the size of the database field where the value is stored.* *By default, fields are stored in a database column with 128 characters in size. Because your data length might be more significant than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a deep explanation of metadata storage in the OpenKM database,*	
defaultAccess	*Default security access policy. By default is set to "grant" so all users will see the property.*	

Child elements:

- ***Validator*** *(more information at Metadata Validator element)*
- ***Security*** *(more information at Metadata Security element)*

***** There're three options to get suggested box values (using **one of them is mandatory**):*

- *filterQuery + valueQuery*
- *className*

No other combination of them is allowed.

Based on filterQuery and valueQuery

*Now **filterQuery** and **valueQuery** could be regular SQL queries.*

You should only use {0} as the parameter the user enters to filter the values from the table.

Create COUNTRY table:

```
create table COUNTRY (
  CT_ID varchar(2),
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
  CT_NAME varchar(32),
  primary key(CT_ID)
);

insert into COUNTRY (CT_ID, CT_NAME) values ('es', 'Spain');
insert into COUNTRY (CT_ID, CT_NAME) values ('pt', 'Portugal');
insert into COUNTRY (CT_ID, CT_NAME) values ('it', 'Italy');
```

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
  <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
    filterQuery="select ct_id, ct_name from COUNTRY where ct_name like '%{0}%' order by ct_name"
    valueQuery="select ct_id, ct_name from COUNTRY where ct_id='{0}'" />
  </property-group>
</property-groups>
```

Based on className

To help to get a list from external applications or other purposes, your class can be built to retrieve the option values from those systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
  <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
    className="com.openkm.plugin.form.values.SuggestBoxUserList" />
  </property-group>
</property-groups>
```

For more information, read Creating your own Suggestbox plugin

Metadata Text field

Prints text that can contain HTML tags.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.* *From version 7.1.41 the label can render HTML. In the case of HTML value the string must be encoded, for example:* `<font color="red">red text</font>` *must be converted to* `<font color="red">red text</font>` *You can use an online XML entity encoder like* *https://coderstoolbox.net/string/#!encoding=xml&action=encode&charset=us_ascii*	*true*
name	*Unique field identifier* ⚠ *Two metadata can't have the same name. The name **must be unique**.* *The name must start with "**okp:**".* *Using the metadata group name as the basis for the metadata field name is a good practice. For example use "**okg:consulting**" has been used for the field name "**okp:consulting**.text", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <text label="text label" name="okp:consulting.text" width="200px" />
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata TextArea field

Prints text that can contain HTML tags.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* ⚠ *Two metadata can't have the same name. The name **must be unique**.* *The name must start with "**okp:**".* *Using the metadata group name as a basis for the metadata field name is a good practice. For example use "**okg:consulting**" has been used for the field name "**okp:consulting**.textarea", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
type	*Type value. Allowed types:* • *text.* • *html.*	
value	*The default value for the component.*	*false*
placeholder	*The placeholder value.*	*false*
description	*General description.*	*false*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

height	*The height of the HTML element.*	*false*
readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*
dbColumnSize	*Set the size of the database field where the value is stored.* *By default, fields are stored in a database column with 128 characters in size. Because your data length might be bigger than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a deep explanation of metadata storage in the OpenKM database,*	
defaultAccess	*Default security access policy. By default is set to "grant" so all users will see the property.*	

Child elements:

- ***Validator*** *(more information at Metadata Validator element)*
- ***Security*** *(more information at Metadata Security element)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <textarea label="textarea label" name="okp:consulting.textarea" width="200px" />
  </property-group>
</property-groups>
```

Metadata Validator element

The goal of the Validator element is to ensure that the user provides the necessary and properly formatted information needed to complete an operation successfully.

The validator can only be used in combination with other fields:

- *Metadata Checkbox field*
- *Metadata Input field*
- *Metadata Select field*
- *Metadata Suggestbox field*
- *Metadata TextArea field*

Available validators:

name	*Description*
req	*Required value.*
alpha	*Alphabetic value.*
alphanum	*Alphanumeric value.*
dec	*Decimal value.*
num	*Numeric value.*
email	*E-mail value.*
url	*URL value.*
maxlen	*Set the maximum length allowed.*
minlen	*Set the minimum length required.*
lt	*Less than.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

gt	*Greater than.*
min	*Minimum value.*
max	*Maximum value.*
regexp	*Regular expression.*
plugin	*Creates your form validator.* *More information at Creating your own Form Validator plugin.*

Parent:

- *Metadata Checkbox field*
- *Metadata Input field*
- *Metadata Select field*
- *Metadata Suggestbox field*
- *Metadata TextArea field*

Basic example

- *Value required "**req**".*
- *The value must be decimal "**dec**".*
- *The maximum length of the decimal value is six digits "**maxlen**".*

```
<input label="Decimal value" name="okp:consulting.decimal">
   <validator type="dec" />
   <validator type="req" />
   <validator type="maxlen" parameter="6" />
</input>
```

Regular expression

You can use a regular expression if you want a validation that is not included in these predefined. For example, imagine you want to restrict an input to lowercase ASCII values:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<input label="Lowercase value" name="regExp">
    <validator type="regexp" parameter="[a-z]+" />
</input>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata Option element

The option element is only used in combination with Metadata Select field. The option value represents a list of distinct values to select.

Attribute	Description	Required
label	*The text shown as label in the user interface.*	*true*
value	*Unique value identifier.* *Two options can't have the same value. **Value must be unique**.* *It's recommended to only use characters which matches this regular expression: [a-z_] (a to z in lowercase and underline, no spaces)*	*true*
parentValue	*When present, it specifies the option parent.* *The parent must be present in other select.*	*false*
selected	*When present, it specifies that an option should be pre-selected when the form is redered.*	*false*
cssClass	*CSS class.* *More information about creating your own styles at CSS.*	*false*

Parent:

- *Select. For more information, Please see: Metadata Select field.*

Basic select with selected option

- *attribute selected=true, when present, it specifies that an option should be pre-selected when the page loads.*

```
<?xml version="1.0" encoding="UTF-8"?>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="select label"  name="okp:consulting.select" type="simple">
      <option label="one" value="001" />
      <option label="two" value="002" selected="true"/>
      <option label="three" value="003" />
     </select>
  </property-group>
</property-groups>
```

Hierarchical relationship

- *The field option value of "**okp:consulting.children**" have a parent relation with "**okp:consulting.parent**". It's defined by **parentElement="okp:consulting.parent"**.*
- *The option value "**benefits**" of the field "**okp:consulting.children**" has a parent value "**directives**" from field "**okp:consulting.parent**".*

Tree relationships:

- *directives*
  - *benefits*
  - *elegibility*
  - *services*
- *handbook*
  - *basic*
  - *normal*
  - *advanced*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="Provincia" name="okp:consulting.parent" type="simple">
      <option value="directives" label="Directives" />
      <option value="handbook" label="Handbook Paragraphs" />
    </select>
    <select label="Provincia" name="okp:consulting.children" type="simple" parentElement="okp:consulting.parent">
      <option value="benefits" label="Benefits" parentValue="directives" />
      <option value="elegibility" label="Eligibility" parentValue="directives" />
      <option value="services" label="Services" parentValue="directives" />
      <option value="basic" label="Basic" parentValue="handbook" />
      <option value="normal" label="Advanced" parentValue="handbook" />
      <option value="advanced" label="Normal" parentValue="handbook" />
    </select>
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata Input field

In most cases, this field is used to enter free-form text, although it can save HTTP links, dates, and data or refer to the application folder (as an internal link to this folder).

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two metadata can't have the same name. The name **must be unique**.* *The name must start with "**okp:**".* *Using the metadata group name as the basis for the metadata field name is a good practice. For example use "**okg:consulting**" has been used for the field name "**okp:consulting**.input", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
type	*Type value. Allowed types:* • *text.* • *date.* • *link.* • *folder.*	*true*
value	*Specifies a default value for this component.* *There's a special case for date types. Data is stored in the format "**yyyyMMddHHmmss**", be aware of this format when using OpenKM API.*	*false*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*

height	*The height of the HTML element.*	*false*
placeholder	*The placeholder value.*	
description	*General description.*	*false*
readonly	*If the field cannot be modified from user interface. By default, the value is "**false**".*	*false*
timeFormat	*When the input type is "date", we can specify if we want to be able to see and modify hours and minutes. Allowed values:* • *hm* • *h* • *none*	*false*
searchType	*Choose how Lucene analyzes the string value. These are the possible values:* • *fuzzy* • *exact* *The default value is "**fuzzy**". When the value is "**fuzzy,**" Lucene analyzes and or tokenizes the string. If the value is set to "**exact,**" Lucene will not tokenize the string to make the search more exact. For example, if you have two nodes with these property values:* • ***Node 1**: "1000"* • ***Node 2**: "100"* *In the case of "**fuzzy,**" if you look for "100," both results will appear. In the case of "**exact,**" only **Node 2** will appear in the results.*	
dbColumnSize	*Set the size of the database field where the value is stored.* *By default, fields are stored in a database column with 128 characters in size. Because your data length might be more significant than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a deep explanation of metadata stored in the OpenKM database,*	

defaultAccess	*Default security access policy. By default is set to "grant" so all users will see the property.*	

Child elements:

- ***Validator*** *(more information at Metadata Validator element)*
- ***Security*** *(more information at Metadata Security element)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input" />
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata VerticalPanel field

A vertical layout that allows you to sort components inside.

Attribute	*Description*	*Required*
name	*Unique field identifier* *Two metadata can't have the same name. The name **must be unique**.* *The name must start with "**okp:**."* *Using a metadata group name as a basis for a metadata field name is a good practice. For example use "**okg:consulting**" has been used for the field name "**okp:consulting**.separator", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please, use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <vpanel name="okp:consulting.vertical1">
      <input label="Input label" name="okp:consulting.input1" />
    </vpanel>
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata HorizontalPanel field

Horizontal layout that allow you to sort components inside.

Attribute	*Description*	*Required*
name	*Unique field identifier* *Two metadatas can't have the same name.* ***Name must be unique****.* *The name must start with "**okp:**".* *It's a good practice to use a metadata group name as basis for metadata field name. For example use "**okg:consulting**" has been used for field name "**okp:consulting**.separator", changin "**okg**" to "**okp**". That helps to find out the metada group based on metadata field name.* *Please, use only letters, numbers and underscore: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* *In the case of KCenter UI must use width values based in %, for example:* • *width="50%" will works* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <hpanel name="okp:consulting.horizontal1">
      <input label="Input label" name="okp:consulting.input1" />
      <input label="Input label" name="okp:consulting.input2" />
    </hpanel>
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata Security element

The goal of the Security element is to set security for each form elements.

*There's a a configuration parameter named "**property.group.element.default.access**" when is set to true all the metadata fields are shown to the users, when false none of the metadata fields are shown.*

Sample scenarios:

- *When **property.group.element.default.access** is **enabled** all the **metadata fields are shown by default**. In this scenario have sense using **revokeUser** and **revokeRole**.*
- *When **property.group.element.default.access** is **disabled** all the **metadata fields are not shown by default**. In this scenario have sense using **grantUser** and **grantRole**.*

For more information about this parameter go to Administration > Configuration parameters .

Remember the members or administrator role (usuarlly ROLE_ADMIN) are not affected by this kind of restrictions.

The Security element can only be used in combination with other fields:

- *Metadata Checkbox field*
- *Metadata Input field*
- *Metadata Select field*
- *Metadata Suggestbox field*
- *Metadata TextArea field*

Available child security elements:

name	*Description*
grantUser	*Set access to a user.*
grantRole	*Set access to the members of the role.*
revokeUser	*Revoke access to a user.*
revokeRole	*Revoke access to the members of the role.*

Basic grant example

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Grant access to a field to a specific user: only* ***testUser*** *user is able to accessing the metadata field named* ***okp:consulting.input1***. *The* ***defaultAccess*** *property is set to* ***revoke***, *so no regular user will be able to see this property.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" defaultAccess="revoke">
        <grantUser name="testUser"/>
    </input>
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

Administrator (users with ROLE_ADMIN role) will see always all fields.

Basic revoke example

Revoke action has priority over the grant action.

- *Revoke access to a field to a specific user: the user* ***testUser*** *do not have access to the metadata field named* ***okp:consulting.input1***, *the other users what have access to the metadata group still have access to it. The* ***defaultAccess*** *property is not present, so it's using the default policy which is "****grant****" that means that all users will be able to see the property, unless there is a child* ***revokeUser*** *tag to exclude users.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1">
        <revokeUser name="testUser"/>
    </input>
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

Administrator (users with ROLE_ADMIN role) will see always all fields.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata internationalization

Feature available since OpenKM v7.1.35

Label internationalization

*The keys of the labels must always start with the "**kcenter.**" string.*

Translations

Register translations at Administration > Tools > Database query

```
-- English translations
-- Group name
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('technology.
-- Fields
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.ti
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.c
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.li
-- Options in select multiple
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
-- Options in select
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p

-- Spanish translations
-- Group name
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('technology.
-- Fields
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.ti
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.c
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.li
-- Options in select multiple
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.a
-- Options in select
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p
INSERT INTO OKM_TRANSLATION (TR_KEY, TR_LANGUAGE, TR_MODULE, TR_TEXT) VALUES ('tecnology.p
```

Metadata definition

Register metadata definition at Administration > Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
    "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
    <property-group label="kcenter.technology.group" name="okg:technology">
        <input label="kcenter.tecnology.title" type="text" name="okp:technology.title">
            <validator type="req"/>
        </input>
        <input label="kcenter.tecnology.date" type="date" name="okp:technology.date" />
        <textarea label="kcenter.tecnology.description" name="okp:technology.description"/>
        <select label="Type" name="okp:technology.type" type="multiple">
            <option label="kcenter.tecnology.description.newProduct" value="t1" selected="true"/>
            <option label="kcenter.tecnology.description.newFeature" value="t2" selected="true"/>
            <option label="kcenter.tecnology.description.customization" value="t3" />
            <option label="kcenter.tecnology.description.integration" value="t4" />
            <option label="kcenter.tecnology.description.quality" value="t5" />
            <option label="kcenter.tecnology.description.other" value="t6" />
        </select>
        <select label="kcenter.tecnology.priority" name="okp:technology.priority" type="simple">
            <option label="kcenter.tecnology.priority.low" value="low" selected="true"/>
            <option label="kcenter.tecnology.priority.medium" value="Medium"/>
            <option label="kcenter.tecnology.priority.high" value="High" />
        </select>
        <textarea label="kcenter.tecnology.comment" name="okp:technology.comment"/>
        <input label="kcenter.tecnology.link" type="link" name="okp:technology.link"/>
    </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Manage users and roles

In the Manage users and roles view you can do:

- *Manage users*.
- *Manage roles*.
- *Manage user mail accounts*.
- *Logged users*
- *Message queue*

By default the application manages users and roles in its own database.

It is also possible to connect the application to an external directory (LDAP directory for instance).

Also the login module can be extended and create your own login adapter. More information about it Creating your own login module.

When the application is connected to LDAP the manage users and roles view is automatically set to readonly mode.

In this scenario the LDAP takes full control of users and roles.

The restricted actions are:

- *Create, delete or modify a user.*
- *Create or delete a role.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Manage users

To manage users *click at the top menu*  ***Authentication button*** *and then click on submenu* ***Users link***.

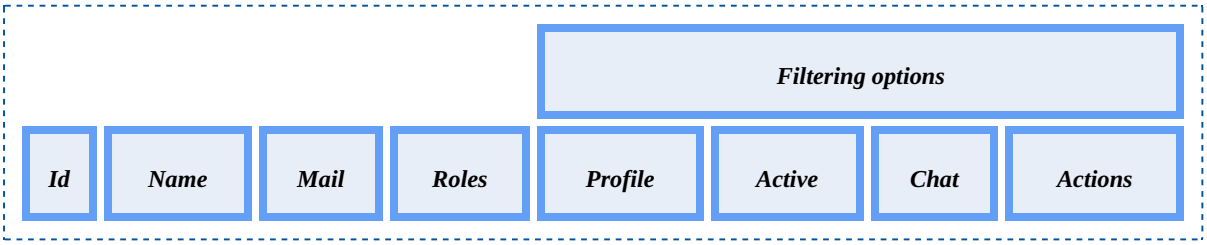
By default the application manages users and roles in its own database.

It is also possible to connect the application to an external directory (LDAP directory for instance).

The login module can be extended and create your own login adapter. More information about it *Creating your own login module*.

When the application is connected to LDAP the manage users view is automatically set to readonly mode. Almost all actions are restricted.

Users are shown in a table structure.



User parameters:

Field	*Description*	*Required*
Id	*The id the user will use to log in.* *Two users can't have the same id.*	*true*
Name	*User name.*	*true*
Mail	*User email address.*	*true*
Roles	*Roles - groups - from which the user is a member.* *User must be at least in 'ROLE_ADMIN' or 'ROLE_USER' role.*	*true*

Profile	*Each profile is a set of options available on the user interface.*	*true*
Active	*Sets user enabled or not.*	*true*
Chat	*Indicates if user is connected to the chat.* *Althought a user can have the application opened on several browser tabs. Only one of them will be logged on chat.*	*false*
Action	*Available actions:* • *Edits an user.* • *Deletes an user.* • *Sets the profile.* • *Sets the user tenant.* • *Configures mail accounts. More information here Manage user mail accounts.* • *Shows the activity log of the user.* • *Delete 2FA user secret*	*
** Mail account and twitter account are optional.*		

Create a user

- *On top right of the table, click on the* ***New user icon***.
- ***Fill all the fields***. *Remember all users must be at least in 'ROLE_ADMIN' or 'ROLE_USER' roles.*
- *Click on the* ***Create button***.

Edit a user

- *Click on the* ***Edit user icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Delete a user

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Click on the*  ***Delete user icon***.
- *Click on the* ***Delete button***.

Change the user profile

- *Click on the*  ***User config icon***.
- *Choose the profile.*
- *Click on the* ***Edit button***.



By default the profile named "default" is automatically assigned to any new user.

Export the users

- *On the top right click on the*  ***export users icon***.
- *The application will download a .csv file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Manage user mail accounts

In the user acounts view, are shown the mail user accounts. These mail accounts are used by application to periodically import mail to application repository.

***To manage user mail account**, **from Users view** click on*  ***Email account icon**.*

Mail account hicheracy

- *Mail account*
  - *Filter*
    - *Rules*

- *The mail account sets the data connection to the mail server to retrieve mails.*
- *Filters are optional and sets the appliation location - folder - where to store mails.*
- *Rules are optional and sets a couple of rules that controls what are the valid mails to import.*

When a filter is not set, by default the application will store mail into /okm:mail/userid folder.

When rules are not set, by default any incoming mail is accepted.

Mail accounts are shown in a table structure.

Mail protocol	***Mail host***	***Mail user***	***Mail folder***	***Active***

Mail account parameters:

Field	*Description*	*Required*
Mail protocol	*The protocol used to connect to the mail server.* *Available options:* • *IMAP* • *IMAPS* • *POP3* • *POP3S*	*true*
Mail host	*The mail server host.*	*true*

Mail user	*The user name to access the mail account.*	*true*
Mail password	*The password to access the mail account.* *In case of OAuth authentication, the format is:* • *oauth:tenant:clientId:clientSecret* *Sample values:* • *Tenant ID: b5a5d2e3-94f6-4b1e-b5c5-8c6e1c4e7b5e* • *Client ID: 3c2a5b1c-d7f0-4c23-b123-92f27b5e27c3* • *Client Secret: q9W1rE3tY8zP5lS3dR7mX2fC0sH6gT1hZ5aQ9vB2nF1kL0mZ8s* *In this case the mail password parameter would be:* *oauth:b5a5d2e3-94f6-4b1e-b5c5-8c6e1c4e7b5e:3c2a5b1c-d7f0-4c23-b123-92f27b5e27c3:q9W1rE3tY8zP5lS3dR7mX2fC0sH6gT1hZ5aQ9vB2nF1kL0mZ8s*	*true*
Mail folder	*The IMAP folder.* *Only applicable when mail protocol value is IMAP or IMAPS. In case of POP3 you can leave empty otherwise need to be set to INBOX.*	*false*
Mail mark seen	*Sets to mail server the seen status.* *Available options:* • *Keep (keep as is).* • *Yes (mark as seen)* • *No (mark as not seen).*	*true*
Mail mark deleted	*Set mail to be deleted.* *When checked the mail will be marked to be deleted from the mail server.*	*true*
Active	*Sets mail account enabled or not.*	*true*

Actions	*Available actions:* • *Edits a mail account.* • *Deletes a mail account.* • *Sets mail account configuration filters.* • *Browsers emails in the mail server.*	

Create a new mail account

- *At the top right click on the* ***Create new account icon***.
- ***Fill in all the fields***.
- *Click on the* ***Create button***.

> *It is a good practice before clicking on Create button, to* ***check the mail configuration***. *There's a button named* ***Check*** *that will check the mail configuration. If all is right application will get* ***OK*** *status.*

Edit a mail account

- *Click on the* ***Edit mail account icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Delete a mail account

- *Click on the* ***Delete mail account icon***.
- *Click on the* ***Delete button***.

Browser emails in the mail server

- *Click on the* ***Server list icon***.

> *In this view mail can be directly imported.*

Manage mail account filters

In the mail account configuration, rules can be set to automatically store mails on different application folders.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

For example, filters can help to automatically catalog mails, based on a "subject" value or "from" value among others.

Mail accounts filter are shown in a table structure.



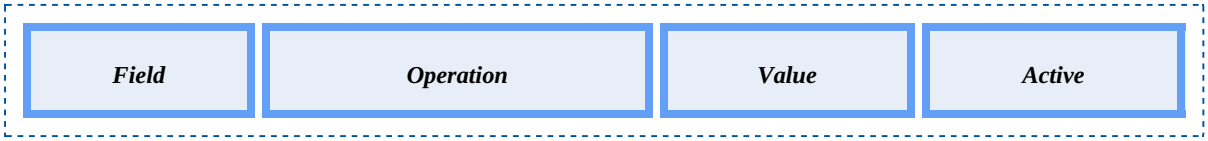
Mail account filter parameters:

Field	*Description*	*Required*
Folder	*Sets the destination folder.*	*true.*
Grouping	*Sets automatic catalog by **year/month/day** enabled.* *To prevent having too many mails in the same folder, we recommend enabling mail grouping.* *The folders year/month/years are automatically created by application.*	*false.*
Active	*Sets filter enabled or not.*	*true.*
Actions	*Available actions:* • *Edit a filter.* • *Delete a filter.* • *Show rules.*	

At least a filter sets a destination folder it also can have additional rules.

A rule can only be set into a filter.

Mail accounts rules are shown in a table structure.



Mail account rule parameters:

Field	*Description*	*Required*
Field	*Sets the mail field that will be evaluated by the rule.* *Available options are:* • *From* • *To* • *Subject* • *Content*	*true.*
Operation	*Set the operation that will be done for the evalution process.* *Available options are:* • *Contains* • *Equal*	*true.*
Value	*Sets the value that will be used for evaluation process.*	*true.*
Active	*Sets rule enabled or not.*	*true.*

Create a mail account filter

- *Click on the*  ***Show filters icon***.
- *On top right click on*  ***New filter icon***.
- ***Fill in all the fields***.
- *Click on the* ***Create button***.

Edit a mail account filter

- *Click on the*  ***Show filters icon***.
- *On top right click on*  ***Edit filter icon***.
- ***Modify the fields fields***.
- *Click on the* ***Edit button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Delete a mail account filter

- *Click on the*  ***Show filters icon***.
- *On top right click on*  ***Delete filter icon***.
- *Click on the* ***Delete button***.

Create a mail account rule

- *Click on the*  ***Show filters icon***.
- *Click on the*  ***Show rules icon***.
- *On top right click on*  ***New rule icon***.
- ***Fill in the fields.***
- *Click on the* ***Create button***.

Edit a mail account rule

- *Click on the*  ***Show filters icon***.
- *Click on the*  ***Show rules icon***.
- *On top right click on*  ***Edit rule icon***.
- ***Modify the fields.***
- *Click on the* ***Edit button***.

Delete a mail account rule

- *Click on the*  ***Show filters icon***.
- *Click on the*  ***Show rules icon***.
- *On top right click on*  ***Delete rule icon***.
- *Click on the* ***Delete button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Manage roles

To manage roles *click at the top menu*  ***Authentication button*** *and then click on submenu* ***Roles link***.

By default the application manages users and roles in its own database.

It is also possible to connect the application to an external directory (LDAP directory for instance).

Also the login module can be extended and create your own login adapter. More information about it *Creating your own login module*.

When the application is connected to LDAP the manage roles view is automatically set to readonly mode. Almost all actions are disabled.

Roles are shown in a table structure.

Id	***Active***	***Actions***

Role parameters:

Field	*Description*	*Required*
Id	*The id of the role.* *Two roles can't have the same id.*	*true*
Active	*Sets if role is enabled or not.*	*true*
Action	*Available actions:* • *Edits a role.* • *Deletes a role.* • *Sets a role tenant.*	

Create a role

- *On the top right in the table, click on the* ***New role icon***.

- ***Fill all the fields***.
- *Click on the* ***Create button***.

Edit a role

- *Click on the*  ***Edit role icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.



ROLE id can't be modified to prevent repository security inconsistences. This operation can be done only at database level.

Delete a role

- *Click on the*  ***Delete role icon***.
- *Click on the* ***Delete button***.

Export the roles

- *On the top right click on the*  ***export roles icon***.
- *The application will download a .csv file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Logged users

To manage logged users *click at the top menu*  ***Authentication button*** *and then click on submenu* ***Logged Users link***.

Logged users are shown in a table structure :

User	***Session Id***	***Remote IP***	***Remote Host***	***Creation***	***Last access***

Real time user activity log

- *Click on on*  ***Activity log icon***.
- *Will be shown as a table with all actions performed by the user today.*
- *Click on the*  ***Refresh button*** *to refresh the list.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Message queue

Message queue helps administrators to send messages across the user interface.

Helps administrators to send a message to all logged users to indicate the system will be halt and automatically logout the users 5 minutes after they received the message.

The application sends messages to a user when a background process has been completed.

The application sends warnings to a user about an action that is not allowed.

Among others messages.

Allowed parameters in the body of the messages:

Parameter	*Sample*	*Description*
uuid	*<uuid>48b29080-57d9-558e-a306-a7b48314983d</uuid>*	*The uuid of a node.*
taskInstance	*<taskInstanceId>19</taskInstanceId>*	*The task instance id of a workflow.*
taskId	*<taskId></taskId>*	*The id of a task.*

Queue of messages shown in a table structure:

Date	***Action***	***Message***	***To All***	***To Users***	***ToRoles***

Message parameters:

Field	*Description*	*Required*
Date	*Date of the message.* *Automatically set by the application.*	*true*
Action	*Available values:* • *Logout (force users who receives it to logout)*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Message	*The text of the message.*	*true*
To All	*When present, the message is sent to all the users.*	*
To Users	*When present, indicate a set of users to sent the message.*	*
To Roles	*When present, indicate a set of users which are members of some of these roles.*	*
* *At least one of the options.*		

Send a message

- *Click on the*  ***New message icon***.
- ***Fill the fields***.
- *Click on the* ***Create button***.

Edit a message

- *Click on the*  ***Edit message icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Delete a message

- *Click on the*  ***Delete message icon***.
- *Click on the* ***Delete button***.

Delete all messages queue

- *On top right click on*  ***Clear button***.



This operation can not be reverted. All the message queue will be deleted.

Refresh message queue

- *On the top right click on*  ***Refresh button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Profiles

Profiles are a set of user interface configuration parameters. They help the administrators to set the available interface features for each user.

For example:

- *Restricts the user interface hidding search and dashboard tabs and almost toolbar buttons.*
- *Restricts the user interface showing only search view.*

The application and specially the user interface comes by default with a several features and extensions.

Not all users need access to everything, specially at the begining when a lot of features can cause confusion to end user.

Our suggestion is going from less to more features as a natural process of the application knowledge.

Profiles are shown in a table structure.

Name	Active	Actions

Profiles are divided in five tabs:

- *Profile General tab*

Create a new profile

- *Click on the*  ***New profile icon***.
- ***Fill in the fields***.
- *Click on the* ***Create button***.

Edit a profile

- *Click on the*  ***Edit profile icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Delete a profile

- *Click on the*  ***Delete profile icon***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Click on the* ***Delete button***.

Clone a profile

- *Click on the*  ***Clone profile icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Profile General tab

General section

Field	Description
Name	*The name of the profile.* *Two profiles cannot share the same name.*
Active	*Sets profile enabled or not.*

Misc section

Field	Description
Advanced filters	*Enables search filtering on the users and roles pop-up view.* *Helps to find roles or users in a large list.*
Web skin	*Sets the default skin (CSS styles applied).* *Info message*
Print preview	*Enables or disables the printing function.*
Keywords enabled	*Enables or disables keyword shortcuts.*
Upload notify users	*Shows or hides user notification on the upload popup.*
Notify external users	*Enables or disables sending notifications to external users.* *By default, the application is restricted to mail users' addresses.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	For example, sending documents to an external mail address.
Increase version	*When enabled, users can choose between document upload as a minor version or a major version.*
Add documents title	*When enabled, documents have an extra field named title.* *A title is shown:* • *A column in a file explorer view.* • *Search view* • *When uploading a new document.* • *On the document properties tab.*
Show events notification	*When enabled, the user is shown a pop-up message sent by the application.* *Common notifications:* • *New documents uploaded.* • *New task assigned.* • *New emails received.*
Thumbnail visible	*When enabled, a thumbnail is shown on the desktop in search view.* *This feature can decrease the application's performance.* *For more information, see Performance configuration parameters.*
Sent Mail Storage	*Set the default mail storage location.* *Available options:* • *Mail folder ("**/okm:root/${user}/sent**")* • *Current folder (user-selected frontend UI folder).*

Quick search visible	*Show or hide top top-right quick search in UI.*
Recursive security change	*Enable or disable by default the checkbox for applying security changes recursively.*
Copy related information	*Enable or disable copy attributes while creating documents from templates.*
Extensions	*Sets the enabled extensions for the profile.*
Reports	*Sets the enabled reports for the profile.*
Workflows	*Sets the available workflows for the profile.*
Property Groups	*Sets the available property groups for the profile.*

Wizard section

The Wizard only takes effect when uploading a new document, not when updating an existing.



In case an automation definition is present, these parameters have no effect.

The automation feature covers all these cases and more.

Field	*Description*
Metadata	*Sets a metadata that will be required to be filled as part of uploading a new document.*
Workflow	*Sets workflows that will be executed as part of uploading a new document.*  *Some workflows require initial fields filled at the beginning, but not all.*
Keywords	*When enabled, the application will ask the user to add keywords as part of uploading a new document.* *The user can skip this step without adding any keywords.*

	
Categories	*When enabled, the application will ask the user to add categories as part of uploading a new document.*  *The user can skip this step without adding any category.*
OCR Data Capture	*When enabled, the application will try to capture data with the OCR feature as part of uploading a new document.*  *The application executes OCR data capture only from images and PDF MIME types.*
Cancel wizard	*When enabled, a button that allows the user to cancel the operation is shown in the Wizard pop-up.*

Pagination section

Pagination control

Field	*Description*
Enabled	*Enable or disable pagination on file explorer view.*
Page list	*Sets the distinct values in what will be shown in a number of elements per page.*  *For example, the value "10;25;50;100" sets the maximum number of elements per page that can be 10, 25, 50, or 100.* *From the user's view, the user can choose one of these options.*  *When pagination is enabled, at least one value must be set* *To separate the distinct values must be used the caracter ";"*
Type filter	*When enabled, a filter by type is shown, which helps the user to show only documents, folders, records, or mails at the same time. **
Show folders by	*By default shows or hides folders. **

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

default	
Show documents by default	*By default shows or hides the document. **
Show mails by default	*By default shows or hides mails. **
Show records by default	*By default shows or hides records. **

** When the type filter is enabled, the options show folders by default, etc... they must be active, otherwise they are not taken in consideration and all types are shown.*

Section General

Shortcuts are keyword combinations for a faster way of accomplishing a task.

Field	*Description*
Rename	*Executes the rename action on the selected node.* `F2`
Copy	*Executes copy action on the selected node.* `CTRL+SHIFT+C`
Cut	*Executes cut action on the selected node.* `CTRL+SHIFT+X`
Paste	*Executes the paste action on the selected node.* `CTRL+SHIFT+V`
Download	*Executes the download action on the selected node.* `CTRL+D`

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Testing	*Shows a pop-up to do some user interface testing.* *CTRL+G*
Log	*Shows the user interface log.* *CTRL+Z*
Create Folder	*Executes create a folder action on the selected node.* *CTRL+N*
Refresh	*Executes refresh action.* *F5*
Add Document	*Executes the add document action.* *INSERT*
Bookmarks	*Executes a bookmark action on the selected node.* *ALT+B*
Help	*Shows a help popup.* *F1*
Logout	*Executes the application logout.* *CTRL+ALT+Q*
Delete	*Executes a delete action on the selected node.* *ALT+DELETE*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Profile Components tab

Stacks section

Field	*Description*
Taxonomy	*Shows or hides taxonomy in the stack panel.*
Categories	*Shows or hides categories in the stack panel.*
Thesaurus	*Shows or hides the thesarus in the stack panel.*
Templates	*Shows or hidesthe templates in the stack panel.*
Personal	*Shows or hides personal in the stack panel.*
Mail	*Shows or hides mail in the stack panel.*
Metadata	*Shows or hides metadata in the stack panel.*
Trash	*Shows or hides trash bin in the stack panel.*

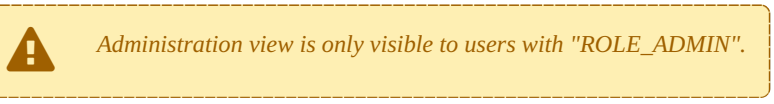
Dashboard section

Field	*Description*
User	*Shows or hides the user panel on dashboard view.*
Mail	*Shows or hides the new mail panel in the dashboard view.*
News	*Shows or hides the news panel in the dashboard view.*
General	*Shows or hides the general panel in the dashboard view.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow	*Shows or hides the workflow panel in the dashboard view.*
Keywords	*Shows or hides the tag cloud panel in the dashboard view.*
Mail Account	*Shows or hides the user mail account panel in the dashboard view.*
Task Manager	*Shows or hides the task manager panel in the dashboard view.*
Calendar	*Shows or hides the calendar panel in the dashboard view.*

Tab section

Field	*Description*
Default tab	*Sets the default tab visible when user login the application.*  
Desktop	*Shows or hides the desktop view.*
Search	*Shows or hides the search view.*
Dashboard	*Shows or hides the dashboard view.*
Administration	*Shows or hides the administration view.* 

Tab document section

The tab document is visible when a document is selected in the desktop view.

Field	Description
Properties	Shows or hides the document properties tab.
Security	Shows or hides the security tab.
Notes	Shows or hides the notes tab.
Versions	Shows or hides the versions tab.
Version download	Enables or disables the download document version.
Preview	Shows or hides the preview tab.
Metadata	Shows or hides the metadata tabs.
Relation	Shows or hides the relations tab.

Tab folder section

The tab folder is visible when a folder is selected in desktop view.

Field	Description
Properties	Shows or hides the folder properties tab.
Security	Shows or hides the folder tab.
Notes	Shows or hides the notes tab.
Preview	Shows or hides the preview tab.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata	*Shows or hides the metadata tabs.*
Relation	*Shows or hides the relations tab.*

Tab mail section

Tab mail is visible when a mail is selected in desktop view.

Field	*Description*
Properties	*Shows or hides the mail properties tab.*
Security	*Shows or hides the mail tab.*
Notes	*Shows or hides the notes tab.*
Preview	*Shows or hides the preview tab.*
Metadata	*Shows or hides the metadata tabs.*
Relation	*Shows or hides the relations tab.*

Tab record section

The tab record is visible when a record is selected in desktop view.

Field	*Description*
Properties	*Sets the default search tab visible when a user login the application..*
Security	*Shows or hides the record tab.*
Notes	*Shows or hides the notes tab.*
Preview	*Shows or hides the preview tab.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Metadata	*Shows or hides themetadata tabs.*
Relation	*Shows or hides the relations tab.*

Tab search section

Field	*Description*
Default search	*Sets the default tab visible in search view.* *Available options:* • *Simple* • *Advanced*
Simple	*Shows or hides the simple search view.*
Advanced	*Shows or hides the advanced search view.* *Advanced search view is composed of:* • *Tab basic* • *Tab advanced* • *Tab metadata*
Tab basic	*Shows or hides the basic tab in advanced search view.*
Tab advanced	*Shows or hides the advanced tabs in the advanced search view.*
Tab metadata	*Shows or hides the metadata tab in the advanced search view.*
Export CSV	*Shows or hides the export to csv feature in advanced search view.*
Export ZIP	*Enabled or disables the export to zip feature in the advanced search view.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Tab basic section

The tab basic contains the most common fields for searching.

Field	Description
Context	*Shows or hides the context field.*
Content	*Shows or hides the content field.*
Name	*Shows or hides the name field.*
Title	*Shows or hides the title field.*
Keywords	*Shows or hides the keywords field.*
Language	*Shows or hides the language list.*
User	*Shows or hides the user list.*
Date	*Shows or hides the date range fields.*

Tab advanced section

Tab advanced contains the advanced fields for doing searching.

Field	Description
Folder	*Shows or hides the folder selector.*
Category	*Shows or hides the category selector.*
Type	*Shows or hides the node type.* *In search view four types are shown:* • *Document*

	• *Folder* • *Mail* • *Record*
Document type	*Shows or hides the document type list.* *Contains a list of MIME type.* *To set the element to be shown in the list see MIME types*
Notes	*Shows or hides the the notes field.*
Mail fields	*Shows or hides the mail fields.* *Available mail fields:* • *Email from.* • *Email to.* • *Email subject.*

Profile Menu tab

These menu options have preference among toolbar options or contextual menu options.

Hierarchy:

- *Menu*
  - *Toolbar*



Preference hierarchy. Upper hierarchy takes control over the lower. If the upper option is disabled then it will be also disabled at the lower one.

For example, when download menu option is disabled then it will not be shown at the toolbar althought there will be enabled.

Menu section

These are the main menu options.

Field	*Description*
File	*Shows or hides the file menu.*
Edit	*Shows or hides the edit menu.*
Tools	*Shows or hides the tools menu.*
Bookmarks	*Shows or hides the bookmark menu.*
Templates	*Shows or hides the templates menu.*
Help	*Shows or hides the help menu.*

Menu bookmark section

Field	*Description*
Manage bookmarks	*Shows or hides manage the bookmarks menu option.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Add bookmark	*Show or hide add the bookmark menu option.*
Set home	*Shows or hides the set home menu option.*
Go home	*Shows or hides the go to home menu option.* *On loading, the application always try to open the default home.* *The default home is /okm:root that can be changed by end user.*

Menu tool section

Field	*Description*
Languages	*Shows or hides the change language menu option.*
Skin	*Shows or hides the change skin menu option.*
Debug	*Shows or hides the debug menu option.* *When enabled is shown, it helps the administrator to find out possible UI bugs.*
Administration	*Shows or hides the administration menu option.* *Open administration is on a new window.*
Preferences	*Shows or hides the user preferences menu option.*
OCR Capture data	*Shows or hides the ocr data capture menu option.*
OMR	*Shows or hides the user omr - Object Mark Recognition - menu option.*
	Shows or hides the convert menu option.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Convert	*This option is used to convert documents to different formats.* *For example .doc to PDF.*
Web page import	*Shows or hides the web page menu option.*

Menu help section

Field	*Description*
Documentation	*Shows or hides the go to documentation website menu option.*
Bug tracking	*Shows or hides the go to bug tracking website menu option.*
Support	*Shows or hides the support menu option.*
Forum	*Shows or hides the go to public forum website menu option.*
Changelog	*Shows or hides the go to changelog menu option.*
Web site	*Shows or hides the go to openkm website menu option.*
About	*Shows or hides the application information menu option.*

Menu file section

Field	*Description*
Create folder	*Shows or hides the create folder menu option.*
Find folders	*Shows or hides the find folders menu option.*
Find documents	*Shows or hides the find documents menu option.*
Find mails	*Shows or hides the find mails menu option.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Find records	*Shows or hides the find records menu option.*
Similar documents	*Shows or hides the find similar documents menu option.*
Go folder	*Shows or hides the go to folder menu option.* *This option for example is shown in the contextual menu at the search results.*
Download	*Shows or hides the download menu option.*
Download PDF	*Shows or hides the download as pdf menu option.*
Add document	*Shows or hides the add document menu option.*
Add record	*Shows or hides the add record menu option.*
Purge	*Shows or hides the purge menu option.*
Purge trash	*Shows or hides the purge all user trash menu option.*
Restore	*Shows or hides the restore from trash menu option.*
Start workflow	*Shows or hides the start workflow menu option.*
Refresh	*Shows or hides the refresh user interface menu option.*
Export	*Shows or hides the export to ZIP menu option.*
Create from template	*Shows or hides the create from template menu option.*
Send document link	*Shows or hides the send document link menu option.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Send document attachment	*Shows or hides the send document attachment menu option.*
Generate download link	*Shows or hides the generate download link menu option.*
Forward e-mail	*Shows or hides the forward email menu option.*
Write e-mail	*Shows or hides the write email menu option.*

Menu edit section

Field	*Description*
Lock	*Shows or hides the lock menu option.*
Unlock	*Shows or hides the unlock menu option.*
Update	*Shows or hides the update a document menu option.*
Edit	*Shows or hides the edit a document menu option.*
Cancel Edit	*Shows or hides the cancel edit a document menu option.*
Delete	*Shows or hides the delete menu option.*
Rename	*Shows or hides the rename menu option.*
Copy	*Shows or hides the copy menu option.*
Move	*Shows or hides the move menu option.*
Add subscription	*Shows or hides the add subscription menu option.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Remove subscription	*Shows or hides the remvoe subscription menu option.*
Add metadata group	*Shows or hides the add metadata group menu option.*
Update metadata group	*Shows or hides the update metadata group menu option.*
Remove metadata group	*Shows or hides the remove metadata group menu option.*
Add note	*Shows or hides the add note menu option.*
Remove note	*Shows or hides the remove note menu option.*
Add category	*Shows or hides the add category menu option.*
Remove category	*Shows or hides the remove category menu option.*
Add keyword	*Shows or hides the add keyword menu option.*
Remove keyword	*Shows or hides the remove keyword menu option.*
Merge PDF	*Shows or hides the merge pdf documents menu option.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Profile Notification tab

The Application sends notifications based on events ocurred on user's subscriptions. These notifications can be sent as an immediate notification mail or all the messages grouped in a single mail delivered once a day.

Grouped notifications are sent by a crontab task name "Send Grouped Notifications".

Notifications are shown on a table structure:

Immediate	***Grouped***

*By default **these two groups are empty and no notifications will be sent**. If you want to work with notification you have to choose which events you want to be subscribed to.*

Events

Common

Notification	*Description*
COMMON	*Common event.*
COPY_ATTRIBUTES	*Copy attributes*

Document

Notification	*Description*
CANCEL_DOCUMENT_CHECKOUT	*Cancel the edition of a document.*
CHECKIN_DOCUMENT	*Update a document.*
CHECKOUT_DOCUMENT	*Mark the document for edition.*
COPY_DOCUMENT	*Copy a document.*
CREATE_DOCUMENT	*Create a new document.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

DELETE_DOCUMENT	*Delete a document.*
GET_CHILDREN_DOCUMENTS	*Get documents.*
GET_DOCUMENT_CONTENT	*Get the content of a document.*
GET_DOCUMENT_CONTENT_BY_VERSION	*Get the content of a document by version.*
GET_DOCUMENT_PROPERTIES	*Get the document properties.*
GET_DOCUMENT_VERSION_HISTORY	*Get the document version history.*
LOCK_DOCUMENT	*Lock a document.*
FORCE_LOCK_DOCUMENT	*Force lock a document.* *Available from OpenKM version 7.1.18*
MOVE_DOCUMENT	*Move a document.*
PURGE_DOCUMENT	*The document is definitely removed from repository.*
RENAME_DOCUMENT	*Change the name of a document.*
SET_DOCUMENT_PROPERTIES	*Change some document properties.*
UNLOCK_DOCUMENT	*Unlock a document.*
FORCE_ UNLOCK_DOCUMENT	*Force unlock a document.*
SET_DOCUMENT_NODE_CLASS	*Set the document NodeClass*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

SET_TITLE	*Set the title*
SET_DESCRIPTION	*Set the description*

Folder

Notification	*Description*
COPY_FOLDER	*Copy a folder.*
CREATE_FOLDER	*Create a new folder.*
DELETE_FOLDER	*Delete a folder.*
GET_CHILDREN_FOLDERS	*Get folders.*
GET_FOLDER_CONTENT_INFO	*Get the content info of a folder.*
GET_FOLDER_PROPERTIES	*Get the folder properties.*
MOVE_FOLDER	*Move a folder.*
PURGE_FOLDER	*The folder is definitely removed from repository.*
RENAME_FOLDER	*Change the name of a folder.*
SET_FOLDER_STYLE	*Set the folder style.*
SET_DESCRIPTION	*Set the description*

Mail

Notification	*Description*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

CREATE_MAIL	*Create a new mail.*
GET_MAIL_PROPERTIES	*Get the mail properties.*
DELETE_MAIL	*Delete a mail.*
PURGE_MAIL	*The mail is definitely removed from repository.*
RENAME_MAIL	*Changes the name of a mail.*
MOVE_MAIL	*Move a mail.*
COPY_MAIL	*Copy a mail.*
GET_CHILDREN_MAILS	*Get mails.*
SET_MAIL_NODE_CLASS	*Set the mail NodeClass*
SET_DESCRIPTION	*Set the description*
SET_TITLE	*Set the title*

Record

Notification	*Description*
CREATE_RECORD	*Create a new record.*
GET_RECORD_PROPERTIES	*Get the properties of a record.*
DELETE_RECORD	*Delete a record.*
FORCE_LOCK_RECORD	*Force lock a record.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	Available from OpenKM version 7.1.18
FORCE_UNLOCK_RECORD	*Force unlock a record.* *Available from OpenKM version 7.1.18*
LOCK_RECORD	*Lock a record.*
PURGE_RECORD	*The record is definitely removed from the repository.*
RENAME_RECORD	*Changes the name of a record.*
UNLOCK_RECORD	*Unlock a record.*
MOVE_RECORD	*Move a record.*
COPY_RECORD	*Copy a record.*
GET_CHILDREN_RECORDS	*Get records.*
SET_RECORD_TITLE	*Set the record title.*
SET_RECORD_NODE_CLASS	*Set the record NodeClass*
SET_DESCRIPTION	*Set the description*
SET_TITLE	*Set the title*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Profile Other tab

Toolbar section

Set the toolbar buttons that are shown.



To enable the toolbar option it must also be enabled in the menu tab, otherwise it will not be visible in the toolbar.

Field	Description
Create folder	*Shows or hides the create folder menu option.*
Find folders	*Shows or hides the find folders menu option.*
Find documents	*Shows or hides the find documents menu option.*
Find mails	*Shows or hides the find mails menu option.*
Find records	*Shows or hides the find records menu option.*
Similar documents	*Shows or hides the find similar documents menu option.*
Download	*Shows or hides the download document menu option.*
Download PDF	*Shows or hides the download document as pdf menu option.*
Print	*Shows or hides the print document menu option.*
Lock	*Shows or hides the lock menu option.*
Unlock	*Shows or hides the unlock menu option.*
Add document	*Shows or hides the add document menu option.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Add record	*Shows or hides the add record menu option.*
Write mail	*Shows or hides the write mail menu option.*
Edit	*Shows or hides the edit a document menu option.*
Update	*Shows or hides the update a document menu option.*
Cancel Edit	*Shows or hides the cancel edit a document menu option.*
Delete	*Shows or hides the delete menu option.*
Add metadata group	*Shows or hides the add metadata group menu option.*
Remove metadata group	*Shows or hides the remove metadata group menu option.*
Start workflow	*Shows or hides the start workflow menu option.*
Add subscription	*Shows or hides the add subscription menu option.*
Remove subscription	*Shows or hides the remove subscription menu option.*
Refresh	*Shows or hides the refresh user interface menu option.*
Home	*Shows or hides the go to home menu option.*
Scanner	*Shows or hides the scanner tool menu option.*
Uploader	*Shows or hides the bulk upload menu option.*
Splitter resize	*Shows or hides the splitter resize menu option.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

History	*Shows or hides the history menu option.*
OMR	*Shows or hides the OMR - Object Mark Recognition - menu option.*

File browser section

Sets the file browser columns that are shown.

Field	*Description*
Status	*Shows or hides the status column.*
Massive	*Shows or hides the checkbox column.*
Icon	*Shows or hides the icon column.*
Name	*Shows or hides the name column.*
Title	*Shows or hides the title column.*
Language	*Shows or hides the language column.*
Size	*Shows or hides the size column.*
Last modified	*Shows or hides the last mofied date column.*
Author	*Shows or hides the author column.*
Version	*Shows or hides the version column.*
Column 1 to Colum 10	*Shows or hides the column based on metadata value.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Relation types

The application can sets distinct relations between nodes (documents, folders, mails, records). In relation types are set the relation types definitions.

There're tree types or relations:

- *Bidirectional*
- *Parent-child*
- *Many-to-many*

Relation types are shown in a table structure.

Type	Title	Title parent to child	Title child to parent

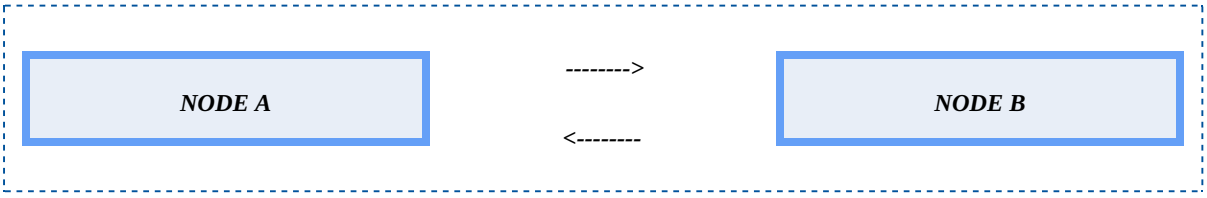
Types

Bidirectional

This releation is set always between two nodes.

A is linked to B like B is linked to A. The relation that links boths elements is the same.

An example of this relation type is "Language translation" relation. A is tranlations of B like B is translation of A.

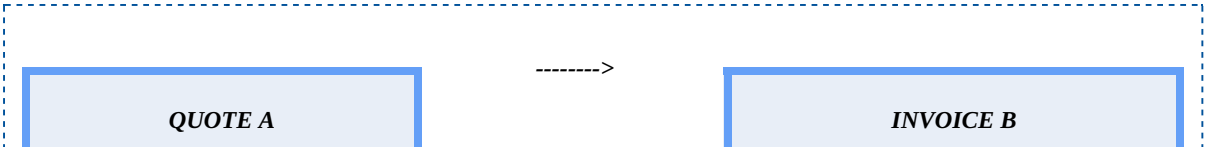


Parent - child

This relelation is always set between two nodes.

A is linked to B and B is linked to A, but the relation that links both elements is diferent.

An example of this relation type is "Invoice - Quote". B is an invoice of A. A is a quote of B.



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

<--------

Many-to-many

This relation is set between two or more nodes.

All nodes are linked to one another.

And example of this relation type is "Stapling". A,B,C, etc. are stapled.

Create a relation

- *On the top right click on* ***New relation type icon***.
- ***Choose one type*** *(bidirectional, parent-child or many-to_many).*
- ***Fill in the fields***.
- *Click on* ***Create button***.

On the bidirectional type is needed to fill the field "Title".

On the parent-child type are needed to fill the fields "Title parent to child" and "Title child to parent".

On many-to-many is needed to fill the field "Title".

Edit a relation

- *Click on* ***Edit a relation icon***.
- ***Modify the fields***.
- *Click on* ***Edit button***.

Delete a relation

Only relations that are not used can be deleted. If some nodes that are using that relation type exist, then the relation can not be deleted.

- *Click on* ***Delete a relation icon***.
- *Click on* ***Delete button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

CSS

The application allows setting your own CSS - Cascade Style Sheets - in the user interface.

The CSS applied to the user interface is set in Profile General tab section.

That helps the administrator to change the look and feel of the application.

For example, change the font size for Cyrillic or logographic languages.

CSS is shown in a table structure.

Name	Context	Active	Actions

Create a new CSS

It cannot be two css with the same name.

- *At the top right, click on the **New CSS icon**.*
- ***Fill in the fields**.*
  - *Name.*
  - *Context available is frontend or mobile.*
  - *Active.*
- *Click on **Create button**.*

Edit a CSS

- *Click on **Edit CSS icon**.*
- ***Modify the fields**.*
- *Click on the **Edit button**.*

Delete a CSS

- *Click on **Delete CSS icon**.*
- *Click on **Delete button**.*

Clone a CSS

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Click on*  ***Clone CSS icon***.
- ***Modify the fields***.
- *Click on* ***Clone button***.

Download a CSS

- *Click on*  ***Download CSS icon***.
- ***Elapsed some seconds, will be retrieved a CSS file***.

Samples



*OpenKM 7.1.x **comes with two UI**, the classical UI (frontend) and a new one named KCenter.*

Because the technologies used in both UI are pretty different, the changes in the CSS style should consider which UI will be used by the users.

New UI samples (KCenter)

Large font sample

```
body, td, a, p, div, input, textarea, select, b-tab {
  font-size: 1.2rem !important; // also can be used 14px;
  font-weight: 400; // Standard weight
}
```

Medium font sample

```
body, td, a, p, div, input, textarea, select, b-tab {
  font-size: 1.1rem !important; // also can be used 13px;
  font-weight: 400; // Standard weight
}
```

Arabic sample



The following example will work from OpenKM version 7.1.31 and upper.



In the case of Arabic can be used several fonts:

- *"Noto Sans Arabic"*
- *"Noto Naskh Arabic"*
- *"Noto Kufi Arabic"*
- *"IBM Plex Sans Arabic"*
- *"Tajawal"*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
/*
 * Notes about some attribute values:
 * font-size: 1.2rem !important;      // Also can be used 14px;
 * font-weight: 400;                  // Standard weight value
*/
body, td, a, p, div, input, textarea, select, b-tab, button, ion-app, ion-menu, ion-header, ion-item, ion-content, ion-foo
  font-size: 1.2rem !important;
  font-weight: 400;
  font-family: "Noto Kufi Arabic" !important;
}

/* Properties */
li, td, th {
  font-size: 1.2rem !important;
}

/* breadcrumb */
li.breadcrumb-item.mt-1 a {
  font-size: 1.2rem !important;
}
#rootNodes div {
 min-height: 35px !important;
}

/* navbar */
#okm-mainMenu .navbar-nav > li.active > a {
 padding-top: 12px !important;
 padding-bottom: 11px !important;
    line-height: unset !important;
}

#okm-mainMenu .navbar-nav > li > a {
 padding-top: 12px !important;
    line-height: unset !important;
}

/* search */
.okm-forceEmptyLabel-topPadding {
 padding-top: 18px !important;
}

/* button */
.btn-group.okm-small-button > button {
 height: 30px !important;
}
```

Reports

The application comes with Jasper reports engine. That helps administrator to create own reports.

The JasperReports Library is one of the world's most popular reporting engine. It is able to use data coming from any type of data source and produce pixel-perfect documents that can be viewed, printed or exported in a variety of document formats including HTML, PDF, Excel, LibreOffice and Word.

More information about how to build your own reports at Creating your own reports.

More information about jasper report engine version that comes into application at Application version compatibility table.

Keep on mind that any font used in the reports should be installed in the OpenKM server. For example, if you are using Arial (sample provided reports use this font) you should ensure Arial font is installed. For example, if you are using Ubuntu the package to install is ***ttf-mscorefonts-installer****.*

```
$ apt-get install ttf-mscorefonts-installer
```

In RedHat or CentOS the package is ***msttcore-fonts****. You need to restart the server after this package is installed.*

If not sure about which fonts are available, execute this scripting code from Administration:

```
import java.awt.*;

for (String font : GraphicsEnvironment.getLocalGraphicsEnvironment().getAvailableFontFamilyNames()) {
   print("Font: " +  font);
}
```

Reports are shown in a table structure.

Name	File name	Active

Create a new report

- *On the top right click on* ***New report icon****.*
- ***Fill in the fields****.*
- ***Choose a report file*** *(*.rep or *.jrxml)*
- *Click on* ***Create button****.*

Edit a report

- *Click on*  ***Edit a report icon***.
- ***Modify the fields***.
- ***Optionally choose a report file*** *(*.rep or *.jrxml)*
- *Click on* ***Edit button***.

Delete a report

- *Click on*  ***Delete a report icon.***
- *Click on* ***Delete button***.

Show report report parameters

> *Optionally, reports can require parameters for execution.*
>
> *For example a range of dates. These parameters will be shown as a form in the user interface when report will be executed.*

- *Click on*  ***Report params icon.***

Execute a report

- *Click on*  ***Execute report icon.***
- ***Fill in the fields*** *(when report comes with parameters).*
- *Choose report output format
  -  *PDF format.*
  -  *RTF format.*
  -  *CSV format.*

Reports pack

Most of these reports are based on Activity log events audit trail.

For more information Activity log.

Template reports

Name	Use Activity log	Require parameters	Description
JavaReportPattern.jrxml	No.	No.	Template pattern that use Java to retrieve data.
SQLReportPattern.jrxml	No.	No.	Template pattern that use SQL to retrieve data.
subreport-sample.rep	No.	No.	Subreport sample. *Support for subreports is available starting from version 8.1.15.*

Authentication

Name	Use Activity log	Require parameters	Description
-	Yes.	Yes.	Users login, filtered by date range.
UsersLogout.rep	Yes.	Yes.	Users logout, filtered by date range.
UsersSessionExpiration.rep	Yes.	Yes.	Users that session expired, filtered by date range.

Documents

Name	Use Activity log	Require parameters	Description

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

DocumentCanceledCheckout.rep	*Yes.*	*Yes.*	*Lists documents that cancelled the edition, filtered by date range.*
DocumentCheckin.rep	*Yes.*	*Yes.*	*List documents updated, filtered by date range.*
DocumentCheckout.rep	*Yes.*	*Yes.*	*List documents on edition, filtered by date range.*
DocumentCreate.rep	*Yes.*	*Yes.*	*List documents created, filtered by date range.*
DocumentDelete.rep	*Yes.*	*Yes.*	*List documents deleted, filtered by date range.*
DocumentGetChildren.rep	*Yes.*	*Yes.*	*Show when the users listed the documents into some folder or record. Results are filtered by date range.* *To list documents into a folder is called internally method named "getChildren".*
DocumentContent.rep	*Yes.*	*Yes.*	*Show when the users retrieved the content of a document. Results are filtered by date range.* *For example, when a user download or preview a document (among other actions) is retrieving the content of the document.*
DocumentContentByVersion.rep	*Yes.*	*Yes.*	*Show when the users retrieved a specific version of the content of a document. Results are filtered by date range.*
DocumentGetProperties.rep	*Yes.*	*Yes.*	*Show when user retrieved the properties of a document. Results are filtered by date range.*

			For example, when document is selected, is called internally a method named "getProperties" to fill the contents of properties tab in user interface.
DocumentGetVersionHistory.rep	*Yes.*	*Yes.*	*Shown when the users retrieved the list of all versions of a document. Results are filtered by date range.*
DocumentGetPropertyGroupProperties.rep	*Yes.*	*Yes.*	*List when has been retrieved the fields of a metadata group.* *For example, when a metadata group is shown in users interface, is called internally a method named "getPropertyGroupProperties" to fill the metadata group values.*
DocumentLock.rep	*Yes.*	*Yes.*	*List documents updated, filtered by date range.*
DocumentMove.rep	*Yes.*	*Yes.*	*List documents moved, filtered by date range.*
DocumentPurge.rep	*Yes.*	*Yes.*	*List documents purged, filtered by date range.* *When a document is purged it will definitively disappear from application repository.*
DocumentRenamed.rep	*Yes.*	*Yes.*	*List documents renamed, filtered by date range.*
DocumentUnlock.rep	*Yes.*	*Yes.*	*List documents unlocked, filtered by date range.*
DocumentsAccessibleByUserOrdered.jrxml	*No.*	*No.*	*Retrieve a list of available documents - security filtered - ordered by folder name.*

			Optimized for csv format. *Not recommended for large repositories, can take a lot of time.*
DuplicatedFiles-v-7.1.23.jrxml	*No.*	*No.*	*This report only can be used for version 7.1.23 or higher.* *List of duplicated files into application repository, based on document checksum.*
DuplicatedFiles.jrxml	*No.*	*No.*	*This report only can be used for version 7.1.22 or lower.* *List of duplicated files into application repository, based on document checksum.*
ListDocumentsWithChecksum.rep	*No.*	*Yes.*	*List of all documents into some folders and subfolders with checksum value.*

Folders

Name	*Use Activity log*	*Require parameters*	*Description*
FolderCopy.rep	*Yes.*	*Yes.*	*List folders copyed, filtered by date range.*
FolderCreate.rep	*Yes.*	*Yes.*	*List folders created, filtered by date range.*
FolderDelete.rep	*Yes.*	*Yes.*	*List folders deleted, filtered by date range.*
FolderGetChilden.rep	*Yes.*	*Yes.*	*Show when the users listed the folders into some folder or record. Results are filtered by date range.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

			To list folders into a folder is called internally method named "getChildren".
FolderGetContentInfo.rep	*Yes.*	*Yes.*	*Shown when the internal method named "getContentInfo" has been executed.* *To do some operations is needed to calculate folder deep, for example before starting folder ZIP download.* *There's an internal method named "getCotentInfo" that does it.*
FolderGetProperties.rep	*Yes.*	*Yes.*	*Show when user retrieved the properties of a folder. Results are filtered by date range.* *For example, when folder is selected, is called internally a method named "getProperties" to fill the contents of properties tab in user interface.*
FolderMove.rep	*Yes.*	*Yes.*	*List folders moved, filtered by date range.*
FolderPurge.rep	*Yes.*	*Yes.*	*List folders purged, filtered by date range.* *When a folder is purged it will definitively disappear from application repository.*
FolderRename.rep	*Yes.*	*Yes.*	*List folders renamed, filtered by date range.*
FolderSecurity.rep	*No.*	*Yes.*	*List folders and subfolders showing security regarding roles and/or users.*
FolderGetSize.rep	*No.*	*Yes.*	*List folders and subfolders showing the size of each one.*

Mails

Use	*Require*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Name	Activity log	parameters	Description
MailCreate.rep	Yes.	Yes.	List mails created, filtered by date range.
MailGetProperties.rep	Yes.	Yes.	Shown when a user retrieved the properties of a mail. Results are filtered by date range. For example, when mail is selected, is called internally a method named "getProperties" to fill the contents of properties tab in user interface.
MailPurge.rep	Yes.	Yes.	List mails purged, filtered by date range. When a mail is purged it will definitively disappear from application repository.
MailRename.rep	Yes.	Yes.	List mails renamed, filtered by date range.
MailMove.rep	Yes.	Yes.	List mails moved, filtered by date range.
MailCopy.rep	Yes.	Yes.	List mails copyed, filtered by date range.
MailGetChilden.rep	Yes.	Yes.	Show when the users listed the mails into some folder or record. Results are filtered by date range. To list mails into a folder is called internally method named "getChildren".

Others

Name	Use Activity log	Require parameters	Description
TenantUserQuota.jrxml	No.	No.	Document size (in MB) by user and tenant.
WorkflowWorkload.jrxml	No.	No.	Each workflow task with the time spent on it.

RegisteredUsers.jrxml	*No.*	*No.*	*Registered users.*

Subreports



Quick guide how to create a subreport

Restrictions:

- *The name of the subreport must always start with "**subreport-**", for example, **subreport-roles.jrxml**.*
- *In the main report, you must create a parameter with the same name as the subreport (without the extension). For example, if the subreport is named **subreport-roles.jrxml**, the parameter should be named **subreport-roles**, and the associated class should be **net.sf.jasperreports.engine.JasperReport**.*
- *The expression in the subreport must correspond to the value of the created parameter. Following the example from the previous point, it should be **$P{subreport-roles}**.*
- *Both reports must share the same database connection.*

XML sections sample

In the main JRXML, define the subreport parameter:

```
<parameter name="subreport-roles" class="net.sf.jasperreports.engine.JasperReport"/>
```

Then in the subreport element, use the parameter reference:

```
<subreport>
   <reportElement x="0" y="0" width="200" height="100"/>
   <subreportExpression><![CDATA[$P{subreport-roles}]]></subreportExpression>
</subreport>
```

The key points are:

- *The parameter name must match the filename without the extension used in the REP (ZIP file)*
- *Parameter type must be JasperReport.*
- *Use $P{parameter-name} in subreportExpression.*
- *Remove any file path references from subreportExpression.*

How it works in the background:

- *The OpenKM core loads the subreport from the REP (ZIP file)*
- *Adds it to the params map with the correct name*

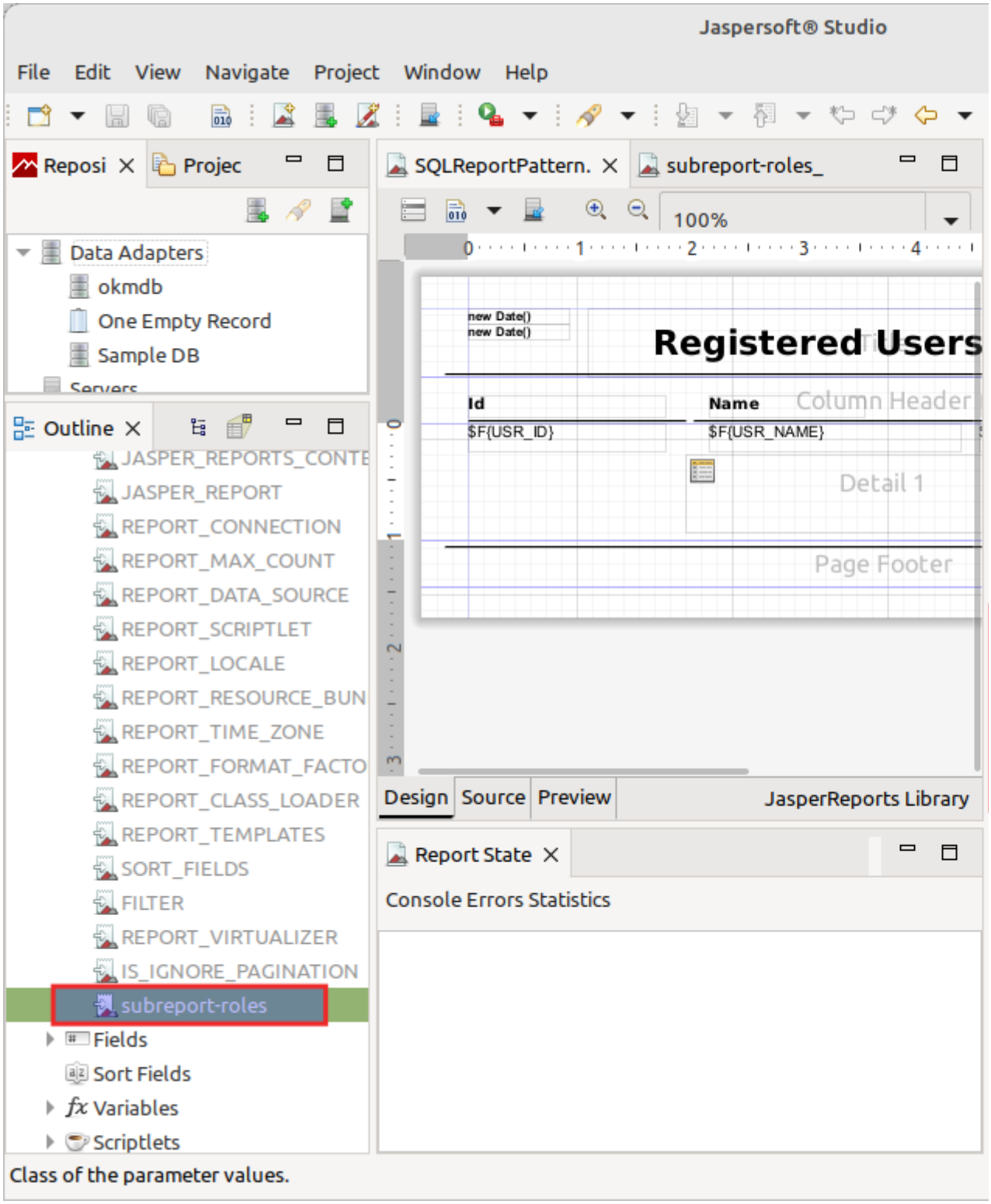
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *JasperReports finds it via parameter reference instead of file path*

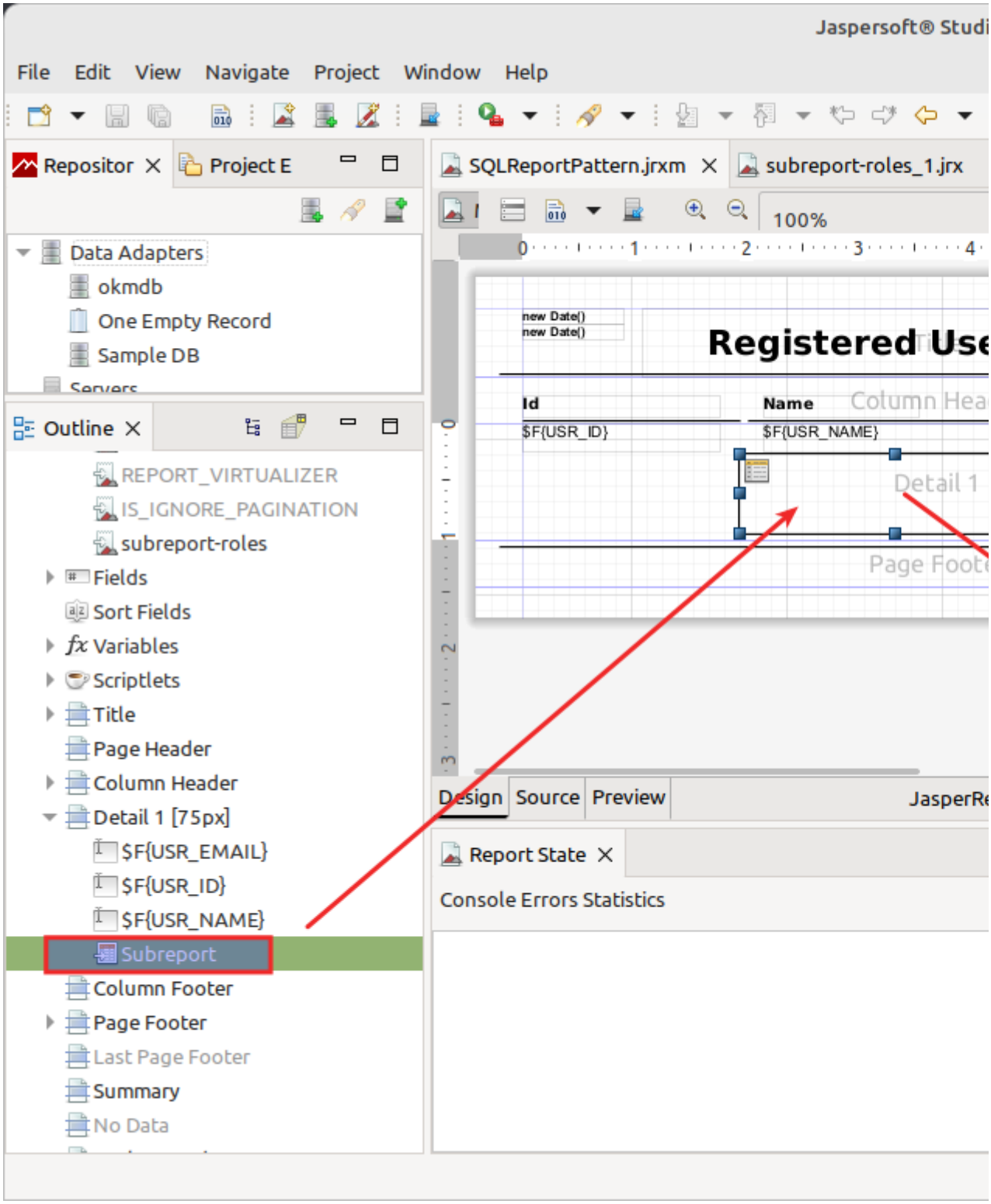
Screenshots

Subreport parameter

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Subreport expression



Suggestions during development

The dynamic configuration of subreports through parameters will work once the report is deployed in OpenKM, but it will not work in the Jasper Studio application. On the other hand, configuring subreports with file system paths will work in Jasper

Studio but will likely not work in OpenKM (we do not recommend using paths).

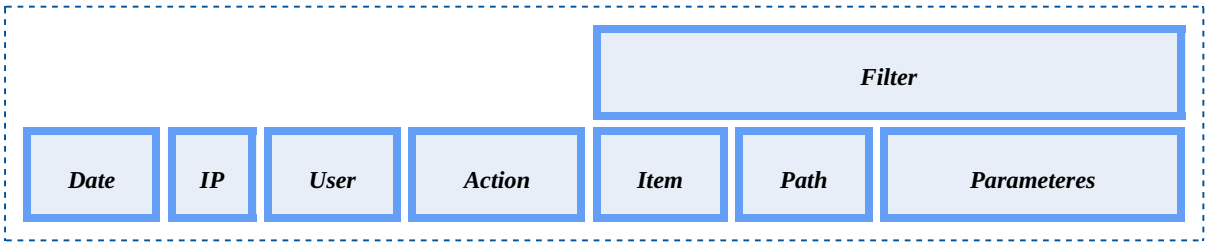
During development, you will likely want to test the report from the Jasper Studio application before deploying it to production. For this reason, we recommend setting the report path directly (file system path) during this phase of development, and once completed, modify it to use dynamic access via the parameter mentioned earlier.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Activity log

This application performs a complete audit trail. Audit trail is mainly a datastore service. It defines a data record structure that will be used for storing audit information.

The activity log is shown in a table structure.



Filter options:

Field	*Description*	*Required*
From date	*Start of date range.*	*true.*
To date	*End of date range.*	*true.*
User	*Specific user.*	*false.*
Action	*Any action.*	*false.*

Auth actions

Field / Property	*Description*
LOGIN	*User log in.*
LOGOUT	*User log out.*
SESSION_EXPIRATION	*User session expiration date*
GRANT_USER	*Adds user grant on a node.*
REVOKE_USER	*Removes user grant on a node.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

GRANT_ROLE	*Add role grant on a node.*
REVOKE_ROLE	*Removes user grant on a node.*
CHANGE_SECURITY	*Several security changes are applied in the same action.*

Document actions

Field / Property	*Description*
CANCEL_DOCUMENT_CHECKOUT	*Cancels document edition.*
CHECKIN_DOCUMENT	*Updates document.*
CHECKOUT_DOCUMENT	*Edits document.*
CREATE_DOCUMENT	*Creates a new document.*
DELETE_DOCUMENT	*Deletes a document.*
GET_CHILDREN_DOCUMENTS	*Retrieves a document list in a node.* *List of documents in a folder or record.*
GET_DOCUMENT_CONTENT	*Retrieves binary data.*
GET_DOCUMENT_CONTENT_BY_VERSION	*Retrieves binary data by document version.*
GET_DOCUMENT_PROPERTIES	*Retrieves all document properties.* *It does not include metadata. It has a direct relationship with the panel properties when a document is selected.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

GET_DOCUMENT_VERSION_HISTORY	*Retrieves a list of all versions of a document.*
LOCK_DOCUMENT	*Locks a document.*
MOVE_DOCUMENT	*Moves a document.*
PURGE_DOCUMENT	*Purges a document.* *A purged document is definitively removed from the application.*
RENAME_DOCUMENT	*Renames a document.*
SET_DOCUMENT_PROPERTIES	*Sets document properties.* *For example when a user is subscribed to a document.*
SET_DOCUMENT_NODE_CLASS	*Set the node class.*
UNLOCK_DOCUMENT	*Unlocks a document*
DOWNLOAD_DOCUMENT	*Downloads a document.*
DOWNLOAD_PREVIEW	*Previews a document.*
SET_DOCUMENT_ANNOTATIONS	*Set annotations in a document.*
GET_DOCUMENT_ANNOTATIONS	*Get annotations of a document.*
SET_DOCUMENT_DISPOSITION_STEP	*Set the disposition step.*

Folder actions

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Field / Property	*Description*
COPY_FOLDER	*Copies a folder.*
CREATE_FOLDER	*Creates a folder.*
DELETE_FOLDER	*Deletes a folder.*
GET_CHILDREN_FOLDERS	*Retrieves a folder list in a node.* *List of folders in a folder or record.*
GET_FOLDER_CONTENT_INFO	*Counts nodes into a folder (documents, folders, records and mails that are into).*
GET_FOLDER_PROPERTIES	*Retrieves all folder properties.* *Does not include metadata. it has a direct relationship with panel properties when a folder is selected.*
MOVE_FOLDER	*Moves a folder.*
PURGE_FOLDER	*Purges a folder.* *A purged folder is definitively removed from application.*
RENAME_FOLDER	*Renames a folder.*
SET_FOLDER_STYLE	*Sets folder style.* *Default folder icons can be changed.*
EXPORT_FOLDER_AS_JAR	*Folder contents exported as a JAR file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Mail actions

Field / Property	*Description*
CREATE_MAIL	*Creates a mail.*
GET_MAIL_PROPERTIES	*Retrieves all mail properties.* ⚠ *Does not include metadata. It has a direct relationship with panel properties when a mail is selected.*
DELETE_MAIL	*Deletes a mail.*
PURGE_MAIL	*Purges a mail.* ⚠ *A purged mail is definitively removed from application.*
RENAME_MAIL	*Renames a mail.*
MOVE_MAIL	*Moves a mail.*
COPY_MAIL	*Copies a mail.*
GET_CHILDREN_MAILS	*Retrieves mail list in some node.* ℹ *List of mail in a folder or record.*
DOWNLOAD_MAIL	*Downloads a mail*
SEND_MAIL_LINK	*Send a mail with a node link.*
SEND_MAIL_ATTACHMENT	*Send a mail with document attachments.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

CREATE_MAIL_ATTACHMENT	*Create a mail with document attachments.*
DELETE_MAIL_ATTACHMENT	*Delete a mail with document attachments.*
GET_MAIL_ATTACHMENTS	*Retrieves a mail with attachments.*
SEND_MAIL_LINK	*Send a mail link.*
DOWNLOAD_MAIL	*Download a mail.*
SET_MAIL_NODE_CLASS	*Set node class.*
SET_MAIL_DISPOSITION_STEP	*Set the disposition step.*

Record actions

Field / Property	*Description*
CREATE_RECORD	*Creates a record.*
GET_RECORD_PROPERTIES	*Retrieves all record properties.* *Does not include metadata. It has a direct relationship with panel properties when a record is selected.*
DELETE_RECORD	*Deletes a record.*
PURGE_RECORD	*Purges a record.* *A purged record is definitively removed from application.*
RENAME_RECORD	*Renames a record.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

MOVE_RECORD	*Moves a record.*
COPY_RECORD	*Copies a record.*
GET_CHILDREN_RECORDS	*Retrieves record list in some node.* *List of record in a folder or record.*
SET_RECORD_TITLE	*Sets title.*
SET_RECORD_NODE_CLASS	*Set record node class.*
ADD_RECORD_NODE_CLASS_CHILDREN	*Add node class children.*
SET_RECORD_DISPOSITION_STEP	*Set the disposition step.*

Property group actions

Field / Property	*Description*
ADD_PROPERTY_GROUP	*When add metadata to a node (document, folder, mail or record).*
REMOVE_PROPERTY_GROUP	*When remove metadata to a node (document, folder, mail or record).*
SET_PROPERTY_GROUP_PROPERTIES	*When update metadata to a node (document, folder, mail or record).*
GET_PROPERTY_GROUP_PROPERTIES	*Retrieves all metadata properties of a node (document, folder, mail or record).*
GET_PROPERTY_GROUP_PROPERTY	*Retrieves a single metadata group properties of a node (document, folder, mail or record).*

Node actions

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Field / Property	Description
SUBSCRIBE_NODE	*When an user subscribe a node (document, folder or record).*
UNSUBSCRIBE_NODE	*When an user desubscribe from a node (document, folder or record).*
NODES_TO_ZIP	*Exports nodes to ZIP.*
RESTORE_NODE_VERSION	*Restore version of a node (document, folder or record).*

Repository actions

Field / Property	Description
PURGE_TRASH	*Trash is purged.* *All related nodes are definitively removed from application.*

Admin actions



These actions are only done by users with Administration grants.

Field / Property	Description
ADMIN_ACTIVITY_LOG	*The administrator uses "activity log".*
ADMIN_ACTIVE_SESSIONS	*The administrator uses Logged users.*
ADMIN_AI_PROMPT_CREATE	*Creates an AI prompt*
ADMIN_AI_PROMPT_DELETE	*Deletes an AI prompt.*
ADMIN_AI_PROMPT_EDIT	*Edits an AI prompt.*

ADMIN_AUTOMATION_ADD_ACTION	*Adds an action to an automation*
ADMIN_AUTOMATION_ADD_VALIDATION	*Adds a validation to an automation*
ADMIN_AUTOMATION_CREATE	*Creates an automation*
ADMIN_AUTOMATION_DELETE	*Deletes an automation*
ADMIN_AUTOMATION_DELETE_ACTION	*Deletes an action from an automation*
ADMIN_AUTOMATION_DELETE_VALIDATION	*Deletes a validation from an automation*
ADMIN_AUTOMATION_EDIT	*Edits an automation*
ADMIN_AUTOMATION_EDIT_ACTION	*Edits an action within an automation*
ADMIN_AUTOMATION_EDIT_VALIDATION	*Edits a validation within an automation*
ADMIN_CACHE_STATS	*Displays cache statistics*
ADMIN_CHECK_EMAIL	*Checks email poll.* *The application reads email accounts to import emails.*
ADMIN_CONFIG_CREATE	*Creates a configuration parameters.*
ADMIN_CONFIG_DELETE	*Deletes a configuration parameter.*
ADMIN_CONFIG_EDIT	*Edits a configuration parameter.*
ADMIN_CRONTAB_ACTIVE	*Activates or manages scheduled tasks.*
ADMIN_CRONTAB_EDIT	*Edits a crontab task.*
ADMIN_CRONTAB_EXECUTE	*Executes a crontab task.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_CSS_CLONE	*Clones an existing CSS style.*
ADMIN_CSS_CREATE	*Creates a new CSS style*
ADMIN_CSS_DELETE	*Deletes a CSS style*
ADMIN_CSS_UPDATE	*Updates an existing CSS style*
ADMIN_CUSTOM_ACTION_CREATE	*Creates a custom action.*
ADMIN_CUSTOM_ACTION_DELETE	*Deletes a custom action*
ADMIN_CUSTOM_ACTION_EDIT	*Edits a custom action*
ADMIN_DATABASE_QUERY	*Executes a database query.*
ADMIN_DATABASE_QUERY_JDBC	*Executes a database query using JDBC*
ADMIN_DATABASE_UPDATE	*Executes an update database query.*
ADMIN_ENTITY_TENANT_EDIT	*Edits tenant-related settings for an entity*
ADMIN_FORCE_OPTIMIZE_INDEXES	*Forces optimization of database indexes*
ADMIN_FORCE_REBUILD_INDEXES	*Forces a rebuild of database indexes*
ADMIN_FORCE_TEXT_EXTRACTOR	*Forces text extraction for indexing or processing*
ADMIN_HIBERNATE_STATS	*Displays Hibernate statistics*
ADMIN_LANGUAGE_CREATE	*Creates a new language.*
ADMIN_LANGUAGE_DELETE	*Deletes a language.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_LANGUAGE_EDIT	*Edits a language.*
ADMIN_LANGUAGE_IMPORT	*Imports a language.*
ADMIN_LOGCAT_DOWNLOAD	*Downloads log files*
ADMIN_LOGCAT_LIST	*The administrator user logcat tool.*
ADMIN_LOGCAT_PURGE	*Deletes or purges log files*
ADMIN_LOGCAT_VIEW	*The administrator user logcat tool.*
ADMIN_LOGGED_USERS	*The administrator uses Logged users.*
ADMIN_MAIL_ACCOUNT_CHECK	*Checks a mail account.*
ADMIN_MAIL_ACCOUNT_CREATE	*Creates a mail account.*
ADMIN_MAIL_ACCOUNT_DELETE	*Deletes a mail account.*
ADMIN_MAIL_ACCOUNT_EDIT	*Edits a mail account.*
ADMIN_MAIL_FILTER_CREATE	*Creates a mail account filter.*
ADMIN_MAIL_FILTER_DELETE	*Deletes a mail account filter.*
ADMIN_MAIL_FILTER_EDIT	*Edits a mail account filter.*
ADMIN_MAIL_FILTER_RULE_CREATE	*Creates a mail account rule.*
ADMIN_MAIL_FILTER_RULE_DELETE	*Delete a mail account rule.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_MAIL_FILTER_RULE_EDIT	*Edits a mail account rule.*
ADMIN_MAIL_SERVER_IMPORT	*Imports mail server settings*
ADMIN_MESSAGE_QUEUE_CREATE	*Creates a new message.* *For more information see Message queue.*
ADMIN_MESSAGE_QUEUE_DELETE	*Deletes a message.*
ADMIN_MESSAGE_QUEUE_UPDATE	*Updates message.*
ADMIN_MIME_TYPE_CREATE	*Creates a new MIME type*
ADMIN_MIME_TYPE_DELETE	*Deletes MIME type.*
ADMIN_MIME_TYPE_EDIT	*Edits MIME type.*
ADMIN_NODE_CLASS_ACTIVE	*Activates or deactivates a node class*
ADMIN_NODE_CLASS_BAC_LIST	*Displays the list of node class BAC (Business Access Control) settings.*
ADMIN_NODE_CLASS_CREATE	*Creates a new node class.*
ADMIN_NODE_CLASS_DELETE	*Deletes a node class.*
ADMIN_NODE_CLASS_EDIT	*Edits an existing node class*
ADMIN_NODE_CLASS_LOCK	*Locks a node class to prevent modifications.*
ADMIN_NODE_CLASS_SECTION_CREATE	*Creates a new section within a node class.*

ADMIN_NODE_CLASS_SECTION_DELETE	*Deletes a section from a node class.*
ADMIN_NODE_CLASS_SECTION_EDIT	*Edits an existing section of a node class.*
ADMIN_NODE_CLASS_SECTION_LIST	*Displays the list of sections within a node class.*
ADMIN_NODE_CLASS_UNLOCK	*Unlocks a previously locked node class.*
ADMIN_NODE_RELATION_TYPE_CREATE	*Creates a new node relation type.*
ADMIN_NODE_RELATION_TYPE_EDIT	*Edits an existing node relation type.*
ADMIN_OCR_TEMPLATE_CLONE	*Clones an existing OCR template.*
ADMIN_OCR_TEMPLATE_CONTROL_FIELD_CREATE	*Creates ocr template control field.*
ADMIN_OCR_TEMPLATE_CONTROL_FIELD_DELETE	*Deletes ocr template control field.*
ADMIN_OCR_TEMPLATE_CONTROL_FIELD_EDIT	*Edits ocr template control field.*
ADMIN_OCR_TEMPLATE_CREATE	*Creates ocr template.*
ADMIN_OCR_TEMPLATE_DELETE	*Deletes ocr template.*
ADMIN_OCR_TEMPLATE_FIELD_CREATE	*Creates ocr template field.*
ADMIN_OCR_TEMPLATE_FIELD_DELETE	*Deletes ocr template field.*
ADMIN_OCR_TEMPLATE_FIELD_EDIT	*Edits ocr template field.*
ADMIN_OCR_TEMPLATE_UPDATE	*Updates an existing OCR template.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_PROCESS_DEFINITION_DELETE	*Deletes a workflow process definition.*
ADMIN_PROCESS_INSTANCE_ADD_COMMENT	*Adds comment on a workflow process instance.*
ADMIN_PROCESS_INSTANCE_DELETE	*Deletes a workflow process instance.*
ADMIN_PROCESS_INSTANCE_END	*Workflow process instance finished.*
ADMIN_PROCESS_INSTANCE_RESUME	*Workflow process instance resumed.*
ADMIN_PROCESS_INSTANCE_SUSPEND	*Workflow process instance suspended.*
ADMIN_PROCESS_INSTANCE_VARIABLE_ADD	*Adds variable on a workflow process instance.*
ADMIN_PROCESS_INSTANCE_VARIABLE_DELETE	*Deletes variable on a workflow process instance.*
ADMIN_PROFILING_STATS_LIST	*Displays a list of profiling statistics.*
ADMIN_PROFILING_STATS_VIEW	*Views detailed profiling statistics.*
ADMIN_PROPERTY_GROUP_CREATE	*Creates a new property group.*
ADMIN_PROPERTY_GROUP_DELETE	*Deletes a property group.*
ADMIN_PROPERTY_GROUP_EDIT	*Edits an existing property group.*
ADMIN_PROPERTY_GROUP_EXISTS_EXCEPTION	*Throws an exception if a property group already exists.*
ADMIN_PROPERTY_GROUP_LIST	*Lists metadata groups.*
ADMIN_RELATION_TYPE_DELETE	*Deletes a relation type.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_REPORT_DELETE	*Deletes a report.*
ADMIN_REPORT_EDIT	*Edits a report.*
ADMIN_REPORT_CREATE	*Creates a report.*
ADMIN_REPORT_EXECUTE	*Executes a report.*
ADMIN_REPOSITORY_CHECKIN	*Updates a node from a repository view.*
ADMIN_REPOSITORY_EDIT	*Edits a node from a repository view.*
ADMIN_REPOSITORY_LIST	*Refreshes folder contents from repository view.*
ADMIN_REPOSITORY_REINDEX	*Reindexes Lucene repository.*
ADMIN_REPOSITORY_REMOVE_CONTENT	*Purges folder lo dentro*
ADMIN_REPOSITORY_REMOVE_CURRENT	*Deletes all folder todo*
ADMIN_REPOSITORY_REMOVE_MIXIN	*Removes a mixin from a repository.*
ADMIN_REPOSITORY_SAVE	*Saves repository configurations or changes.*
ADMIN_REPOSITORY_SEARCH	*Performes a Lucene search.*
ADMIN_REPOSITORY_UNLOCK	*Unlocks a node from a repository view.*
ADMIN_ROLE_ACTIVE	*Enables or disables a role.*
ADMIN_ROLE_CREATE	*Creates a role.*
ADMIN_ROLE_DELETE	*Deletes a role.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_ROLE_EDIT	*Edits a role.*
ADMIN_SCRIPTING	*Executes scripting.*
ADMIN_SHARD_CREATE	*Creates a new database shard*
ADMIN_SHARD_EDIT	*Edits an existing database shard.*
ADMIN_SQL_REPORT_CREATE	*Creates a new SQL report.*
ADMIN_SQL_REPORT_DELETE	*Deletes an SQL report.*
ADMIN_SQL_REPORT_EDIT	*Edits an existing SQL report.*
ADMIN_SQL_REPORT_EXECUTE	*Executes an SQL report.*
ADMIN_STAMP_BARCODE_CREATE	*Creates a stamped barcode.*
ADMIN_STAMP_BARCODE_DELETE	*Deletes a stamped barcode.*
ADMIN_STAMP_BARCODE_EDIT	*Creates a new tenant configuration.*
ADMIN_STAMP_IMAGE_CREATE	*Creates a stamp image.*
ADMIN_STAMP_IMAGE_DELETE	*Deletes a stamp image.*
ADMIN_STAMP_IMAGE_EDIT	*Edits a stamp image.*
ADMIN_STAMP_TEXT_CREATE	*Creates a stamp text.*
ADMIN_STAMP_TEXT_DELETE	*Deletes a stamp text.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_STAMP_TEXT_EDIT	*Edits a stamp text.*
ADMIN_TASK_INSTANCE_ADD_COMMENT	*Adds comment on a workflow task instance.*
ADMIN_TASK_INSTANCE_END	*Workflow task instance ended.*
ADMIN_TASK_INSTANCE_RESUME	*Resumes a workflow task instance.*
ADMIN_TASK_INSTANCE_SET_ACTOR	*Sets actor on a a workflow task instance.*
ADMIN_TASK_INSTANCE_START	*Workflow task instance started.*
ADMIN_TASK_INSTANCE_SUSPEND	*Workflow task instance suspeded.*
ADMIN_TASK_INSTANCE_VARIABLE_ADD	*Adds variable on a workflow task instance.*
ADMIN_TASK_INSTANCE_VARIABLE_DELETE	*Deletes variable on a workflow task instance.*
ADMIN_TENANT_CONFIG_CREATE	*Creates a new tenant configuration.*
ADMIN_TENANT_CONFIG_DELETE	*Deletes a tenant configuration.*
ADMIN_TENANT_CONFIG_EDIT	*Edits an existing tenant configuration.*
ADMIN_TENANT_CREATE	*Creates a new tenant.*
ADMIN_TENANT_DELETE	*Deletes a tenant.*
ADMIN_TENANT_EDIT	*Edits an existing tenant.*
ADMIN_TOKEN_SIGNAL	*Workflow task signal sent from administration view.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ADMIN_TWITTER_ACCOUNT_CREATE	*Creates a twitter account.*
ADMIN_TWITTER_ACCOUNT_DELETE	*Deletes a twitter account.*
ADMIN_TWITTER_ACCOUNT_EDIT	*Edits a twitter account.*
ADMIN_USER_ACTIVE	*Enables or disables a user.*
ADMIN_USER_CONFIG_EDIT	*Changes a user profile.*
ADMIN_USER_CREATE	*Creates a user.*
ADMIN_USER_DELETE	*Deletes a user.*
ADMIN_USER_EDIT	*Edits a user.*
ADMIN_USER_PROFILE_ACTIVE	*Activates or deactivates a user profile.*
ADMIN_USER_PROFILE_CLONE	*Clones a user profile.*
ADMIN_USER_PROFILE_CREATE	*Creates a profile.*
ADMIN_USER_PROFILE_DELETE	*Deletes a profile.*
ADMIN_USER_PROFILE_EDIT	*Edits a profile.*
ADMIN_WORKFLOW_REGISTER	*Registers a workflow.*

Adds a validation to an automationâ€¦

Misc actions

Field / Property	*Description*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

MISC_OPENKM_START	*Starts application.*
MISC_OPENKM_STOP	*Stops application.*
MISC_STATUS	*Application check.* *Several internal checks to ensure application is alive:* • *Database connection.* • *Repository connection.*
MISC_TEXT_EXTRACTION_FAILURE	*Text extraction failure.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Automation

Automation is a set of controls that perform actions based on a set of rules.

Preliminar concepts.

Each rule is composed of:

Field	Description
Order	*Indicates the order of the evaluation.*  *The rules with less*
Name	*Rule description.*
Events	*Available events:* • *Document create.* • *Document update.* • *Document delete.* • *Document rename.* • *Document purge.* • *Document move.* • *Document version restore.* • *Document download from UI.* • *Document download from UI for preview.* • *Document print.* • *Document stamp.* • *Document virus detected* • *Document set node class* • *Folder create.* • *Folder delete.* • *Folder move.* • *Task update* • *Task delete.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Exports folder as JAR.*
- *Mail creation.*
- *Mail delete.*
- *Mail move.*
- *Mail download from UI.*
- *Mail set node class.*
- *Mail import attachments.*
- *Record create.*
- *Record delete.*
- *Record move.*
- *Record set node class.*
- *Add metadata group.*
- *Set metadata group.*
- *Remove metadata group.*
- *Text extraction.*
- *Exports several nodes as ZIP.*
- *Task create.*
- *User create*
- *User login.*
- *User logout.*
- *Note create.*
- *Note update.*
- *Note delete.*

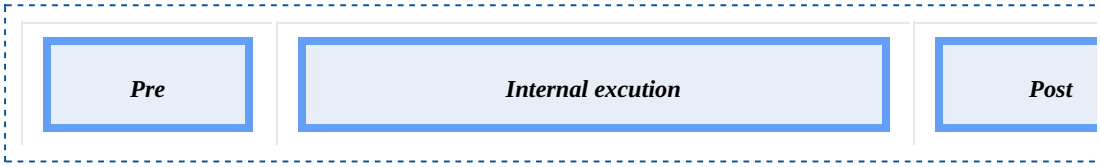
At

The evaluation time.

Available values:

- *Pre.*
- *Post.*

Understanding stages on automation execution:



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	Pre indicate rule execution before an internal event is executed. *Post indicate rule execution after the internal event is executed.* *For example a rules based on "Document creation" event.* • *On "pre" stage the rule is triggered when the document has not yet been created.* • *On "post" stage the rules is triggered after the document has been created*
Evaluation	*If all automation validations should evaluate to true or just one of them.* *For example, if you have two validations like "NameMatches" and "PathContains", in case of evaluation is set t* *both conditions must be met. If is set to "OR", only one need to be true.*
Exclusive	*When enabled, indicates no order rules should be executed.*
Active	*Enables or disables the rule.*
Validators	*List of validators. Each validator is a condition that must be complied.* *Validations are also ordered. It's a good practice to first set positions that will have more possibilit to comply.* *When empty, always comply.*
Actions	*List of actions. An action is something that will be done with the node that caused the execution of the rule.* *Nodes types:* • *Document* • *Folder* • *Mail* • *Record* *Example of an action, can be renaming a document immediately after creation.*

Understanding Validators and Actions

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

The application provides a complete list of available validators and actions from Plugins view. Application plugins' architecture helps administrators to add new features to an existing application.

By default application comes with built in plugins:

- *Built-in Automation Validator plugins.*
- *Built-in Automation Action plugins.*

Also you can create your own plugins:

- *Creating your own Automation Validation.*
- *Creating your own Automation Action.*

> *More information about how to Register a new plugin.*

Creating a rule

> *Rules are composed of:*
>
> - *General fields.*
> - *Actions and validators definions.*
>
> *A rule creation has two steps. First define the general fields, then the actions and validations definitions.*

- *On top right click on*  ***Create new rule***.
- ***Fill in the fields***.
- *Click on* ***Create button***. *(the application will return to the initial screen).*
- *Click on*  ***Definition icon***.
- *Add actions and validations according to convenience.*
  - *Click on*  ***Add validation icon***.
  - *Click on*  ***Add action icon***.
- *When finished, at the top left click on* ***Automation rules button***. *(the application will return to the initial screen).*

Editing a rule

Edit general parameters

- *Click on*  ***Edit rule***.
- ***Modify the fields***.
- *Click on* ***Edit button***.

Edit action and validation definitions

- *Click on*  ***Definition icon***.
- *Modifiy actions and validations according to convenience.*
  - *Validations*
    - *Click on*  ***Add validation icon***.
    - *Click on*  ***Edit validation icon***.
    - *Click on*  ***Delete validation icon***.
  - *Actions*
    - *Click on*  ***Add action icon***.
    - *Click on*  ***Edit action icon***.
    - *Click on*  ***Delete action icon***.

Deleting a rule

- *Click on*  ***Delete rule***.
- *Click on* ***Delete button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Built-in Automation Validator plugins

Name	Apply at	Parameters	Description
ContainsFolder	Post	Folder name	Valid when the node path contains a folder name. The value of the parameter folder name is **test**. Node with path **/okm:root/folder1** triggered the event.. the folder **/okm:root/folder1/test** exists ,then the validation return is **valid**. The value of the parameter folder name is **test**. Node with path **/okm:root/folder1** triggered the event.. A folder **/okm:root/folder1/test** doesn't exist, then the validation return is not valid.
HasCategory	Post		
HasKeyword	Post	Keyword name	Valid when the node path has a keyword with the same name assigned.
HasPropertyGroup	Post	Metadata group name	Valid when the node path has a metadata group with the same name assigned.
HasPropertyGroupValue	Post	Metadata field name Metadata field value	Valid when the node path has a metadata group field with the same value assigned.
IsCreatedFromTemplate	Pre, Post		

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

			Valid when the node has been created from a template.
IsNotCreatedFromTemplate	*Pre*, *Post*		*Valid when the node has NOT been created from a template.*
IsOCRDataCaptureFile	*Post*		*Valid when the document is supported by OCR.* *Supported MIME type are:* • *PNG* • *GIF* • *JPEG* • *BMP* • *TIFF* • *PDF*
NameContains	*Post*	*Name*	*Valid when the node name contains the name value into.* *The value of the parameter name is **test**.* *Node with path **/okm:root/folder/sometest** triggered the event.* *Folder **sometest** contains the value **test**, then the validation return is **valid**.* *The value of the parameter name is **test**.* *Node with path **/okm:root/folder/somefolder** triggered the event.*

			*Folder **somefolder** does not contains the value **test** then the validation return is **not valid**.*
PathContains	*Post*	*Path*	*Valid when the node path starts with a path.* *The value of the parameter path is **/okm:root/folder1**.* *Node with path **/okm:root/folder1/test/temporary** triggered the event.* ***/okm:root/folder1/test/temporary** starts with path **/okm:root/folder1**, then the validation return is **valid**.* *The value of the parameter path is **/okm:root/folder1**.* *Node with path **/okm:root/test/temporary** triggered the event.* ***/okm:root/test/temporary** does not start with path **/okm:root/folder1**, then the validation return is **not valid**.*
PathContainsFolderName	*Pre, Post*	*Folder name*	*Valid when the node path contains a folder name.* *The value of the parameter folder name is **folder1**.* *Node with path **/okm:root/folder1/test/temporary** triggered the event.* ***/okm:root/folder1/test/temporary** contains a folder named **folder1**, then the validation return is **valid**.* *The value of the parameter folder name is **folder1**.* *Node with path **/okm:root/test/temporary/folder1** triggered the event.* ***/okm:root/test/temporary/folder1** contains a folder named **folder1**, then the validation return is **valid**.* *The value of the parameter folder name is **folder1**.* *Node with path **/okm:root/test/temporary** triggered the event.*

			/okm:root/test/temporary does not contains a folde named ***folder1***, then the validation return is ***not valid***.
ParentPathContainsFolderName	*Pre, Post*	*Folder name*	*Valid when the node parent path contains a folder name.* *The value of the parameter folder name is* ***folder1***. *Node with path* ***/okm:root/folder1/test/temporary*** *triggered th event.* *The parent path* ***/okm:root/folder1*** *contains a folder named* ***folder1***, *then the validation return is* ***valid***. *The value of the parameter folder name is* ***folder1***. *Node with path* ***/okm:root/test/folder1/test1/test2*** *triggered the event.* *The parent path* ***/okm:root/test/folder1*** *contains a folder named* ***folder1***, *then the validation return is* ***valid***. *The value of the parameter folder name is* ***folder1***. *Node with path* ***/okm:root/folder1/temporary*** *triggered the event.* *The parent path* ***/okm:root*** *does not contains a folder named* ***folder1***, *then the validation return is* ***not valid***. *The value of the parameter folder name is* ***folder1***. *Node with path* ***/okm:root/test/test1/temporary*** *triggered the event.* *The parent path* ***/okm:root/test*** *does not contains a folder named* ***folder1***, *then the validation return is* ***not valid***.
isUser	*Pre, Post*	*User id*	*Validates whether the authenticated user matches the user provided as a parameter.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Built-in Automation Action plugins

Name	Apply at	Parameter	Description
AddCategory	Post	Category	Adds a category to the node that triggered the event.
AddCategoryToWizard	Post		Shows a category pop-up as part of the process of uploading a new document.  Only takes effect when a user uploads a new document from the user interface.
AddDigitalSignatureToWizard	Post		Shows a digital signature pop-up as part of the proces uploading a new document.
AddKeyword	Post	Keyword name	Adds a keyword to the node that triggered an event.
AddKeywordToWizard	Post		Shows a keyword popup as part of the process of uploading a new document.  Only takes effect when a user uploads a new document from the user interface.
AddOCRDataCaptureToWizard	Post		Shows an OCR data capture pop-up as part of the pro of uploading a new document.  Only takes effect when a user uploads a new document from the user interface.
AddPropertyGroup	Post	Metadata	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

		group name	*Add the Metadata group.*
AddPropertyGroupToWizard	*Post*	*Metadata group name*	*Shows a Metadata group pop-up as part of the proces uploading a new document.* *Only takes effect when a user uploads a new document from the user interface.*
AddWorkflowToWizard	*Post*	*Workflow name*	*Starts a workflow as part of the process of uploading new document.* *A pop-up will be shown if a workflow requires a user to set parameters for starting the workflow. Otherwise, it will b transparent to the end user.* *Only takes effect when a user uploads a new document from the user interface.*
CatalogByMetadata	*Post*		*Used by the file plan to automatically catalog docume in the repository.*
CheckUniqueDocumentName	*Pre*		*In case exist a document with the same name exists in whole repository, it raises an exception.*
CompactDocumentHistory	*Post*		*Compact the document history, removing all older versions.*
ConvertToPDF	*Post*		*Convert a document to PDF.*
ConvertToPDFA	*Post*		*Convert a document in PDF format to PDFA.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

CreateDestinationPathByCatalogManager	*Post*		*Used by the fileplan to automatically create the destination path of a node.*
DelCadReferences	*Pre*		*Delete the OpenKM document references of an AutoC document.*
DeleteNode	*Post*		*Deletes a node that is moved into the user's trash node*
DocumentCreationNameToLowerCase	*Pre*		*Force the name and the extension of the document to lowercase.*
DownloadAsDocumentVersion	*Post*		*When downloading a document, rename the document including an end the version number.*
DownloadControl	*Post*		*Registers the activity log that triggered a download.* *Available download causes:* • *Folder or document download as a ZIP.* • *Document download from the user interface.* • *Document download from the user interface to preview.* • *Document download from the user interface to print.* • *Folder exported as JAR.* • *Email download from the user interface.*
ExecuteScripting	*Pre* *Post*	*Code*	*Executes a script.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ForceGrantActualUser	*Post*	*Permission*	*Grants permission to the user who caused the event in node that triggered the event.* *The super user makes the security change*
ForceRevokeActualUser	*Post*	*Permission*	*Revoke permission to the user who caused the event in node that triggered the event.* *The super user makes the security change*
GrantRole	*Post*	*Role name* *Permission*	*Grants a role permission to the node that triggered the event.*
GrantUser	*Post*	*User name* *Permission*	*Grants a user permission to the node that triggered the event.*
MoveToAZFolder	*Post*	*Folder destination* *Folder deep;*	*Create a folder hierarchy based on the name of the no and deep, and move the node into it.* *Document named "Testing.pdf"* *Folder destination = /okm:root/move/into* *With deep = 3* *The document will be moved at /okm:root/move/into/T/E/S/Testing.pdf*
MoveToFolder	*Post*	*Folder destination*	*Move the node to a new location.*
MoveToYearMonthDayFolder	*Post*	*Folder destination*	*Move the node to a new location with a subfolder structure year / month / day*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

			Document named "Testing.pdf" *Folder destination = /okm:root/move/into* *Current date = 2020-05-08* *The document will be moved at /okm:root/move/into/2020/05/08/Testing.p*
NewCadReferences	*Post*		*Create the OpenKM document references of an AutoC document.*
OCRDataCapture	*Post*		*Captures data from the document.*
OMRDataCapture	*Post*		*Captures data from the document.*
PurgeNode	*Post*		*Purge a node. The node is removed from the repositor*
RevokeAllRoles	*Post*		*Revokes all roles granted to the node that triggered th event.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

RevokeAllUsers	*Post*		*Revoke all users granted to the node that triggered the event.*
RemoveCategory	*Post*	*Category*	*Removes the category from the node that triggered the event.*
RemoveKeyword	*Post*	*Keyword name*	*Remove the keyword from the node that triggered the event.*
RenameUniqueDocument	*Pre*		*Ensures new documents will have a unique name.*
RevokeRole	*Post*	*Role name*	*Revokes the role granted to the node that triggered the event.*
RevokeUser	*Post*	*User name*	*Revokes the user granted to the node that triggered the event.*
Send Mail	*Post*	*Role names* *User names* *Message*	*Send mail to the selected users and roles.* *The message format is the same used in notification messages. For more information, take a look at Notification configuration parameters.*
SetAutoStartToWizard	*Pre, Post*		*Set the autostart to the wizard*
SetInQuarantine	*Post*		*When the antivirus engine detects a virus in a docume*

			moves the document into the quarantine folder /okm:root/quarantine
SetLanguage	*Post*	*Language*	*Sets the language to the document node that triggered event.* *Language value is based on ISO 639-1.* *For example, for Spanish the value is "es" for English "en", etc.*
SetRelationsFromXRef	*Post*		*Set the relations from xref.*
SetSummarization	*Post*		*Set the document summarization.* *Must have enabled the KEA Summarizati service.*
SetTextExtracted	*Post*	*String*	*Replaces the original content text extracted from the document node that triggered the event.*
StampDocument	*Post*	*Stamp id* *Stamp type*	*Stamp a document.* *The stamp ID must be an existing stamp I definition.* *Stamp type values:* • *"text" for text stamping.* • *"images" for image stamping.*
VidSignerDocumentSigned	*Post*		*Mark a document as signed by VidSigner.* *Must have a VidSigner account.*
VidSignerDocumentToSign	*Post*		*Sent a document to sign by VidSigner.*

			Must have a VidSigner account.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Stamp

Stamp feature helps you to insert an image into a document.

*This option is hidden when **Stamp** extension is not enabled. Read about it at Enable extensions .*

More information at Stamp extension.

You can set three types of stamp:

- *Based on images.*
- *Based on text.*
- *Based on barcodes.*

- ***To manage text stamps** click at the top menu **Stamp button** and then click on submenu **Text Stamp**.*
- ***To manage image stamps** click at the top menu **Stamp button** and then click on submenu **Image Stamp**.*
- ***To manage barcode stamps** click at the top menu **Stamp button** and then click on submenu **Barcode Stamp**.*

Stamp are shown in a table structure.

Id	Name	Descripcion	Text	Script	Color	Users	Active

Create new stamp based on text

When use stamp based on text, the text is converted to an image and then stamped into a document.

Fields description:

Field	Description
Name	*Stamp name.*

Description	*Stamp description.*
Text	*The text to be shown in stamp.*

Script	*When is set the result of the script will be stamped text.*
	
	List of variables what can be used directly:
	
	Variable / **Description**
	document — *Document object of the document node.*
	For more information see com.openkm.bean.Document class at API description.
	
	Sample:
	
	`import java.util;`
	
	`String value = "";`
	`value = document.getActualVersion().getName();`
	`value += "-" + String.valueOf(new Date());`
	
	`return value;`
	
	Sample:
	
	`import org.springframework.web.context.*;`
	`import com.openkm.util.*;`
	`import com.openkm.api.*;`
	
	`WebApplicationContext cc = (WebApplicationContext) ContextWrapper.getContext();`
	`OKMPropertyGroup okmPrpGrp = cc.getBean(OKMPropertyGroup.class);`
	`String prpGrpName = "okg:sample";`
	
	`if (okmPrpGrp.hasGroup(null, document.getUuid(), prpGrpName)) {`
	`  Map props = okmPrpGrp.getProperties(null, document.getUuid(), prpGrpName);`
	`  return "Number: " + props.get("okp:sample.number");`
	`} else {`
	`  return "No property group assigned: " + prpGrpName;`
	`}`

Font	*unicode.ttf*

	The file of the font must be copied into the $TOMCAT_HOME/lib folder.
Layer	*Stamped image position.* *Available values:* • *Under content.* • *Over content.*
Opacity	*The text opacity. The range of values allowed are from 0 to 1.* *An **Opacity** equals to 0 means a transparent text or image and you won't see it. Set to 0.5 to see an semi-transparent text or set to 1 to be opaque.* *Also play with layer to stamp over or below the default text layer.*
Size	*Text size.*
Align	*Text align.* *Available values:* • *Left.* • *Center.* • *Right.*
Rotation	*Text rotation. The range of values allowed are from 0 to 360.*
Colour	*The text colour.*
Expr. X	*Expression to set the position in X coordinates.*
Expr. Y	*Expression to set the position in Y coordinates.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Active	*When enabled the stamp will be available to the users.*
Users	*List of users who can use the stamp.*

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently the following macros are defined:

- *PAGE_WIDTH*
- *PAGE_HEIGHT*
- *PAGE_CENTER*
- *PAGE_MIDDLE*

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

- *Click on*  ***Create new Stamp icon***.
- ***Fill the form***.
- *Click on* ***Create button***.

Create new stamp based on image

Fields description:

Field	*Description*
Name	*Stamp name.*
Description	*Stamp description.*
Image	*The image to be added into the document.*
Script	*When is set the result of the script will be stamped text.* *List of variables what can be used directly:*

Variable	*Description*

	document *Document object of the document node.*
	
	For more information see com.openkm.bean.Document class at API description.
	
	Sample:
	
	`import java.util;`
	
	`String value = "";`
	`value = document.getActualVersion().getName();`
	`value += "-" + String.valueOf(new Date());`
	
	`return value;`

Layer	*Stamped image position.* *Available values:* • *Under content.* • *Over content.*
Opacity	*The text opacity. The range of values allowed are from 0 to 1.*
Expr. X	*Expression to set the position in X coordinates.*
Expr. Y	*Expression to set the position in Y coordinates.*
Active	*When enabled the stamp will be available to the users.*
Users	*List of users who can use the stamp.*

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently the following macros are defined:

- *IMAGE_WIDTH*
- *IMAGE_HEIGHT*

- *PAGE_WIDTH*
- *PAGE_HEIGHT*
- *PAGE_CENTER*
- *PAGE_MIDDLE*

So to center a stamp in the page you can use:

Expr. X	*PAGE_CENTER - IMAGE_WIDTH / 2*
Expr. Y	*PAGE_MIDDLE - IMAGE_HEIGHT / 2*

- *Click on*  ***Create new Stamp icon***.
- ***Fill the form***.
- *Click on* ***Create button***.

Edit stamp

- *Click on*  ***Edit icon***.
- ***Modify the form***.
- *Click on* ***Edit button***.

Delete stamp

- *Click on*  ***Delete icon***.
- *Click on* ***Delete button***.

Test stamp

- *Click on*  *icon.*



Staps what use scripting with variable document can not be tested from administration.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AI Prompt

AI prompt feature helps you to create your own AI prompts.

This feature remains hidden unless an AI model has been activated beforehand. Read about it at AI configuration parameters .

You can set two types of AI prompts:

- *Based on text.*
- *Based on plugins.*

Standard AI prompts can be executed on any document, as long as the content of the document's body is available.

In the case of AI prompts based on plugins, it is possible to set whether they apply to nodes or specific text. For nodes, it is possible to restrict the MIME types on which execution is allowed.

AI prompt are shown in a table structure.



Create AI prompt

Fields description:

Field	Description
Name	*AI prompt name.*
Description	*AI prompt description.*
Prompt	*The AI prompt to be executed over documents or text* *This field will only be displayed as long as no plugin is selected; otherwise, it will remain hidden.*
Plugin	*This list will display the AI prompt plugins that are installed in the application.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Areas	*Indicate in which section of the user interface the AI Prompt will be displayed.* *Areas available:* • *Anywhere* • *Email compose*
Actions	*Actions are additional operations that can be performed with the result of executing an AI prompt.* *Actions available:* • *Add note.* • *Set description.* *Actions will only appear available if the AI prompt has been executed on a node.*

- *Click on*  ***Create new AI prompt icon***.
- ***Fill the form***.
- *Click on* ***Create button***.

Edit AI prompt

- *Click on*  ***Edit icon***.
- ***Modify the form***.
- *Click on* ***Edit button***.

Delete AI prompt

- *Click on*  ***Delete icon***.
- *Click on* ***Delete button***.

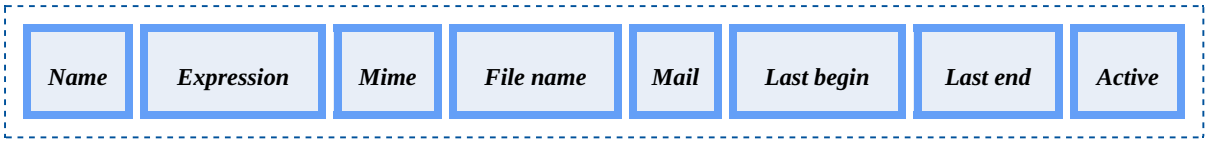
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Crontab

The application implements a time-based job scheduler named Crontab. In crontab view, there is a set of jobs that are executed periodically at fixed times or intervals.

Typically automated application maintenance or administration is crontab jobs.

Crontab is shown in a table structure.



Crontab parameters

Field	*Description*	*Required*
Name	*Job name.*	*true*
Expression	*Sets when the job is executed.* *Takes into consideration the status of the application when the task will be executed.* *For example, if you stop the application at 00:00 to perform a backup, it delays the task until you consider the backup process will be completed.*	*true*
Class name	*More information at Creating your own Crontab plugin.*	*true*
Mail	*When present, a mail is sent as part of the task once the execution has been completed. You can use several mail addresses using a comma "," as a separator.*	*false*
Last begin	*Shows the last time a process started.*	-
Last end	*Shows the last time a process was completed.*	-
Active	*Enables or disables the process execution.*	*true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Understanding expression syntax

> *More information https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/scheduling/support/CronSequenceGenerator.htm*

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at least one of the two day fields (day of month, or day of week) match the current time. The scheduler examines crontab entries once every minute. The time and date fields are:

```
* * * * * command to execute
â”, â”, â”, â”, â”,
â”, â”, â”, â”, â”, â”,
â”, â”, â”, â”, â”, â”""â”€â”€â”€â”€â”€ day of week (MON, TUE, WED, THU, FRI, SAT, SUN)
â”, â”, â”, â”, â""â”€â”€â”€â”€â”€â”€â”€â”€â”€ month (1 - 12)
â”, â”, â”, â""â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€ day of month (1 - 31)
â”, â”, â""â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€ hour (0 - 23)
â”, â""â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€ min (0 - 59)
â""â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€ seconds (0 - 59)
```

Field	*Values*
seconds	*0-59*
minute	*0-59*
hour	*0-23*
day of month	*1-31*
month	*1-12*
day of week	*MON, TUE, WED, THU, FRI, SAT, SUN*

Samples

*"0 0 * * * *" = every hour of every day.*

"/10 * * * * *" = every ten seconds.*

*"0 0 8-10 * * *" = 8, 9 and 10 o'clock of every day.*

*"0 0 6,19 * * *" = 6:00 AM and 7:00 PM every day.*

*"0 0/30 8-10 * * *" = 8:00, 8:30, 9:00, 9:30, 10:00 and 10:30 every day.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*"0 0 9-17 * * MON-FRI" = on the hour nine-to-five weekdays*

"0 0 0 25 12 ?" = every Christmas Day at midnight

Create a new crontab job

Crontab job is based in architecture.

More information at Creating your own Crontab plugin.

Edit a crontab job

- *Click on*  ***Edit crontab job icon***.
- ***Modify the fields***.
- *Click on* ***Edit button***.

Execute a crontab job

- *Click on*  ***Execute crontab job icon***

Delete a crontab job

Crontab job is based in plugin architecture deployed into $TOMCAT_HOME/plugins. Must delete the jar file in the plugin folder to remove the associated crontab job.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Built in crontab tasks

Task	Default time	Execution	Description
Antivirus Checker Worker	0 */5 * * * *	Execution every 5 minutes.	Process the antivirus check queue. For more information Statistics.
Check disk space	0 0 0 * * *	Daily execution at 00:00	When disk space is low, send an email alert to the administrator.
Check Repository Integrity	0 0 0 * * SAT	Execution each Saturday at 00:00	Check repository integrity and make a report.
Lucene Delayed Indexer	0 */5 * * * *	Execution every 5 minutes.	Indexing records in the background without switching the repository to read-only status. The column NBS_FORCE_INDEX_REBUILD in the table OKM_NODE_BASE is used to mark the records that should be indexed.
Mail Extractor Worker	0 */5 * * * *	Execution every 5 minutes.	Process the mail extractor queue.
Mail Extractor Issues	0 0 0 * * *	Daily execution at 00:00	Process the mail extractor issues
Metadata Refresh	0 0 0 * * *	Daily execution at 00:00	Metadata refresh
Process Disposition	0 0 0 * * *	Daily execution at 00:00	When the file plan is enabled, this cron processes the disposition state value.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Process Pending Tasks	*0 */5 * * * ****	*Execution every 5 minutes.*	*Execute the pending task.* *For more information Statistics.*
Purge Activity Log	*0 0 0 * * ****	*Daily execution at 00:00*	*Delete activity log*
Purge Dashboard Activity	*0 0 0 * * ****	*Daily execution at 00:00*	*Clean older than one-month dashboard activity log.*
Purge Download Preview Tokens	*0 */5 * * * ****	*Execution every 5 minutes.*	*Delete expired preview tokens that have not been consumed.*
Purge Empty Repository Directories	*0 0 0 * * ****	*Daily execution at 00:00*	*Purge empty folders into the repository*
Purge Node Versions	*0 0 0 * * ****	*Daily execution at 00:00*	*Purge all the history of the documents and keep only the current version in the application.*
Purge Old Trash Nodes	*0 0 0 * * ****	*Daily execution at 00:00*	*Purge old nodes in the trash.*
Purge Orphan Repository Files	*0 0 0 * * SUN*	*Execution at 00:00 every Sunday*	*Purge the files in the datastore that are not in the application database.*
Purge Pending Downloads	*0 0 0 * * ****	*Daily execution at 00:00*	*Delete pending downloads*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Purge Temporal Printing Files	*0 0 0 * * ***	*Daily execution at 00:00*	*Delete temporary printing files.*
Purge Tomcat Logs	*0 0 0 * * ***	*Daily execution at 00:00*	*Delete the old Tomcat logs.*
Purge Trash Batch	*0 */5 * * * ***	*Execution every 5 minutes.*	*It does an incremental purge of the trash in the background.*
Repository Info	*0 0 0 * * ***	*Daily execution at 00:00*	*Update repository stats.*
Send Grouped Notifications	*0 0 0 * * ***	*Daily execution at 00:00*	*Send daily mail notifications grouped into a single mail.* *For more information Profile Notification tab.*
Sync LDAP users	*0 0 0 1 1 **	*Executes at 00:00 on January 1st of every year*	*Synchronize AD user data to the OpenKM database.*
Text Extractor Issues	*0 0 0 * * ***	*Daily execution at 00:00*	*Collect text extraction issues and send an email warning to the administrator.*
Text Extractor Worker	*0 */5 * * * ***	*Execution every 5 minutes.*	*Process text extraction queue.* *For more information Statistics.*
User Mail Importer	*0 */30 * * * **	*Execution every 30 minutes.*	*Import the emails from the registered user's mail accounts.*
SyncLdapUsers	*0 0 0 1 1 **	*Executed once a week*	*Synchronize users and roles from the AD to the OpenKM database.*

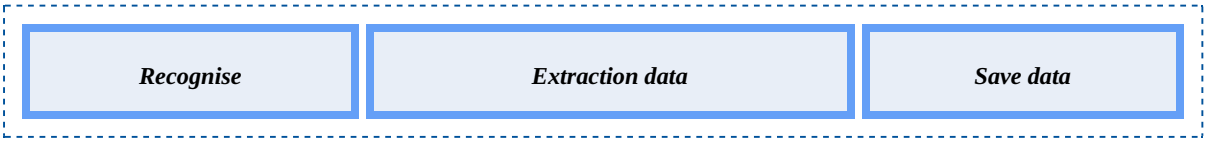
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OCR templates

OCR Templates allow to create zonal OCR templates which allows to recognise and extract estructured text from scanned images.

For best results, images should be scanned at least at 200 dpi resolution to get good text recognition from the OCR engine.

Understanding stages on the OCR templates execution:



On recognition, the application based on ocr template's definition tries to identify the document. A document will be considered recognised only if an OCR template checks 100%

On extraction data, the values are extracted from the document.

On save data, the dynamic values got in the extraction data stage are saved into metadata groups. Optionaly are saved as static values defined on the OCR template (normally used to identify the document type).

There're two field types on OCR templates:

- *Validation Control fields.*
- *Data Capture fields.*

***Validation Controls fields** are used only on the **recognition** stage.*

***Data Capture fields** are used on the **extraction data** stage and optionaly can be used also on **recognition** stage.*

Allowed MIME types on the recognise stage are:

- *PNG*
- *JPG*
- *GIF*
- *BMP*
- *PDF*

Validation Control field description

Validation Control fields are used only in the Recognise stage in helping on the document identification by checking a document's section pattern.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Each Validation Control field is composed of:

Field	*Description*	*Required*
Name	*Name of the field.*	*true.*
Order	*Order of the evaluation*	*true.*
Type	*List of **OCR Validation Controls**. Each validator is a condition that must be complied.*	*true.*
Value	*A value.*	*
Pattern	*A Java regex pattern.*	*
Rotation	*Indicates the image rotation that must be applied.* *Available image rotations:* • *-270* • *-180* • *-90* • *0* • *90* • *180* • *270*	*true.*
Custom	*Sets commands to process images by the OCR. Several commands can be set , one per line.* *For example to highlight the red color* ***/usr/bin/convert ${fileIn} -matte (+clone -fuzz 50% -transparent red) -compose DstOut -composite -negate ${fileOut}***	*false.*
OCR	*OCR command line.* *When present it will be executed in place of the system.ocr configuration parameter.*	

	More information Configuration parameters. *For example to force digits capture.* ***/usr/bin/tesseract ${fileIn} ${fileOut} digits***	*false.*
Empty value allowed	*When checked indicates ocr empty text value is also valid in identifying document process.*	*false*
***** These parameters are used into to the Validation Control. Can be required or not depending on which Validation Control is used.*		

Data Capture field description

Data Capture fields are used to retrieve data in the extraction stage but also can be used in the Recognise stage in helping a document identification by checking a document section pattern.

Each Data Capture field is composed of:

Field	*Description*	*Required*
Name	*Name of the field.*	*true.*
Order	*Order of evaluation*	*true.*
Type	*List of the* ***OCR Data Capture***.	*true.*
Property	*Metadata field where will be stored the captured data.*	*true.*
Pattern	*A Java regex pattern.*	*
Rotation	*Indicates the image rotation that must be applied.* *Available image rotations:* • *-270* • *-180* • *-90*	*true.*

	• *0* • *90* • *180* • *270*	
Custom	*Sets commands to process the images by the OCR. Several commands can be set, one per line.* *For example to highlight the red color* ***/usr/bin/convert ${fileIn} -matte (+clone -fuzz 50% -transparent red) -compose DstOut -composite -negate ${fileOut}***	*false.*
OCR	*OCR command line.* *When present it will be executed in place of the system.ocr configuration parameter.* *More information Configuration parameters.* *For example to force digits capture.* ***/usr/bin/tesseract ${fileIn} ${fileOut} digits***	*false.*
Use to recognize	*When it's set to true, indicates that the field will also be used on the recognise stage.*	*false.*
***** This parameter is used in the Data Capture. It can be required or not depending on which Data Capture is used.*		

Understanding OCR Validation Control and OCR Data Capture

The application provides a complete list of available OCR Validation Controls and OCR Data Capture from Plugins view. Application plugins architecture help the administrators to add new features to the existing application.

By default application comes with built in plugins:

- *Built in OCR Validation Control plugins.*
- *Built in OCR Data Capture plugins.*

Also your own plugins can be created:

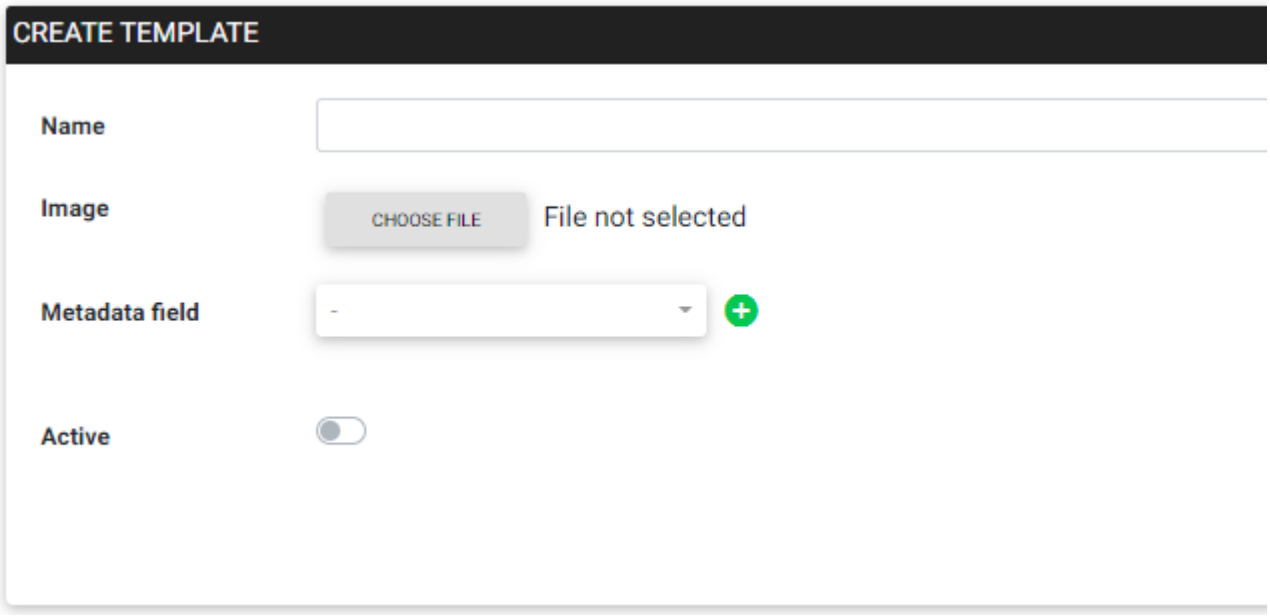
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Creating your own OCR Validation Control plugin*.
- *Creating your own OCR Data Capture plugin*.

> *For more information about how to Register a new plugin.*

Create a new OCR template

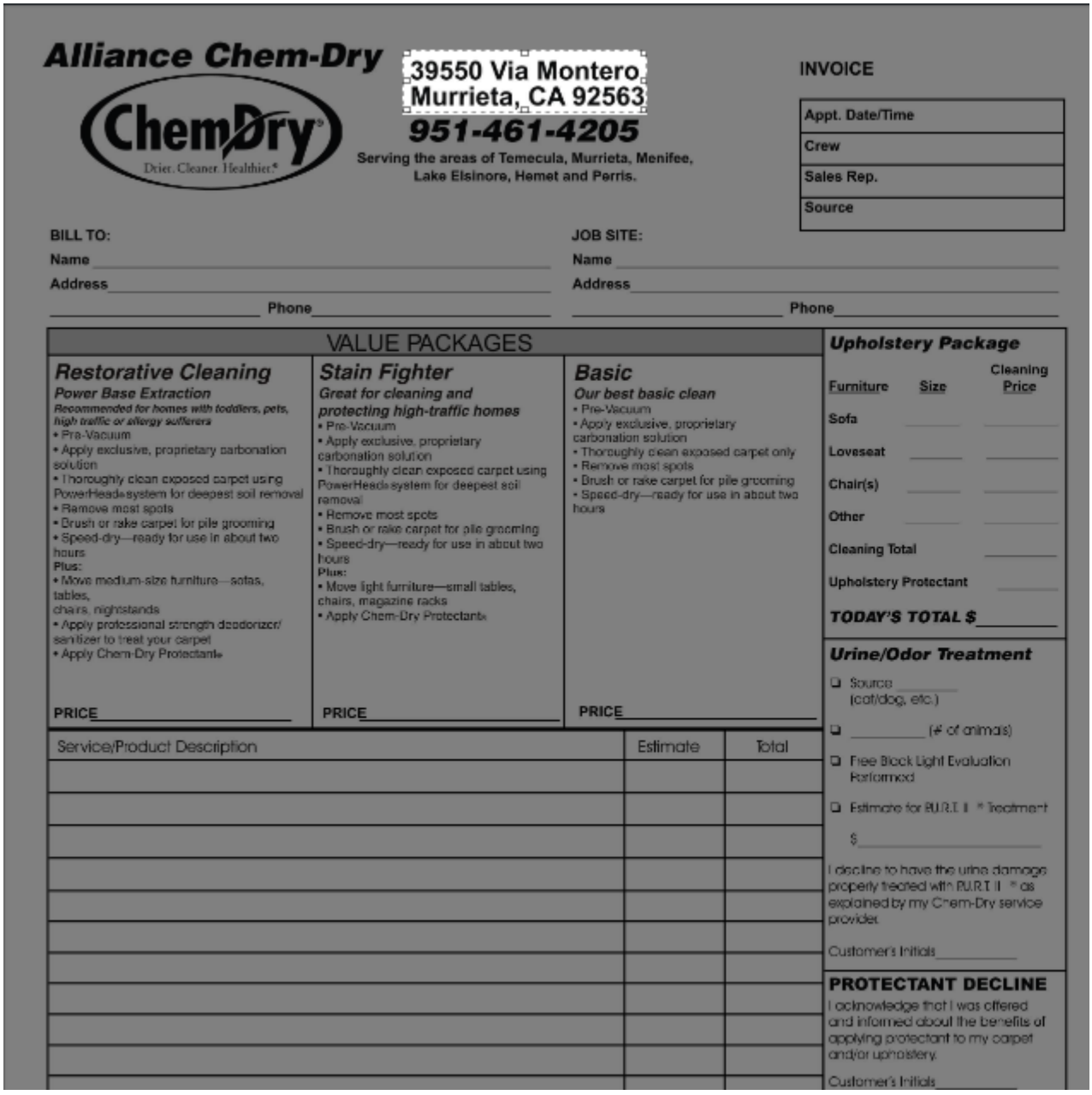
- *On the top right click* ***New OCR templace icon***.
- ***Fill in the fields***.
  - *Metadata field (static metadata values, normally used to identify document type).*
- *Chose the file (allowed MIME type).*
- *Click on* ***Create button***.



- *Add Validation Controls fields according to your convenience.*
  - *Validation Controls*
    - *Click on* ***Fields***.
      - *To create a new Validation Control:*
        - *Click on* ***New Validation Control Field icon***.
        - ***Fill the fields***.
        - ***Choose the area with mouse on the left image***.
        - *Click on* ***Create button***.
      - *To edit a Validation Control:*

- *Click on* ***Edit a Validation Control icon***.
- ***Modify the fields***.
- *Click on* ***Edit button***.

- *To delete a Validation Control:*
  - *Click on* ***Delete a Validation Control icon***.
  - *Click on* ***Delete button***.



Alliance Chem-Dry

ChemDry
Dries. Cleans. Healthier.®

39550 Via Montero
Murrieta, CA 92563
951-461-4205
Serving the areas of Temecula, Murrieta, Menifee, Lake Elsinore, Hemet and Perris.

INVOICE

Appt. Date/Time
Crew
Sales Rep.
Source

BILL TO:
Name
Address
Phone

JOB SITE:
Name
Address
Phone

VALUE PACKAGES

Restorative Cleaning
Power Base Extraction
Recommended for homes with toddlers, pets, high traffic or allergy sufferers
• Pre-Vacuum
• Apply exclusive, proprietary carbonation solution
• Thoroughly clean exposed carpet using PowerHeads system for deepest soil removal
• Remove most spots
• Brush or rake carpet for pile grooming
• Speed-dry—ready for use in about two hours
Plus:
• Move medium-size furniture—sofas, tables, chairs, nightstands
• Apply professional strength deodorizer/ sanitizer to treat your carpet
• Apply Chem-Dry Protectant®
PRICE

Stain Fighter
Great for cleaning and protecting high-traffic homes
• Pre-Vacuum
• Apply exclusive, proprietary carbonation solution
• Thoroughly clean exposed carpet using PowerHeads system for deepest soil removal
• Remove most spots
• Brush or rake carpet for pile grooming
• Speed-dry—ready for use in about two hours
Plus:
• Move light furniture—small tables, chairs, magazine racks
• Apply Chem-Dry Protectant®
PRICE

Basic
Our best basic clean
• Pre-Vacuum
• Apply exclusive, proprietary carbonation solution
• Thoroughly clean exposed carpet only
• Remove most spots
• Brush or rake carpet for pile grooming
• Speed-dry—ready for use in about two hours
PRICE

Service/Product Description	Estimate	Total

Upholstery Package

Furniture	Size	Cleaning Price
Sofa		
Loveseat		
Chair(s)		
Other		
Cleaning Total		
Upholstery Protectant		

TODAY'S TOTAL $

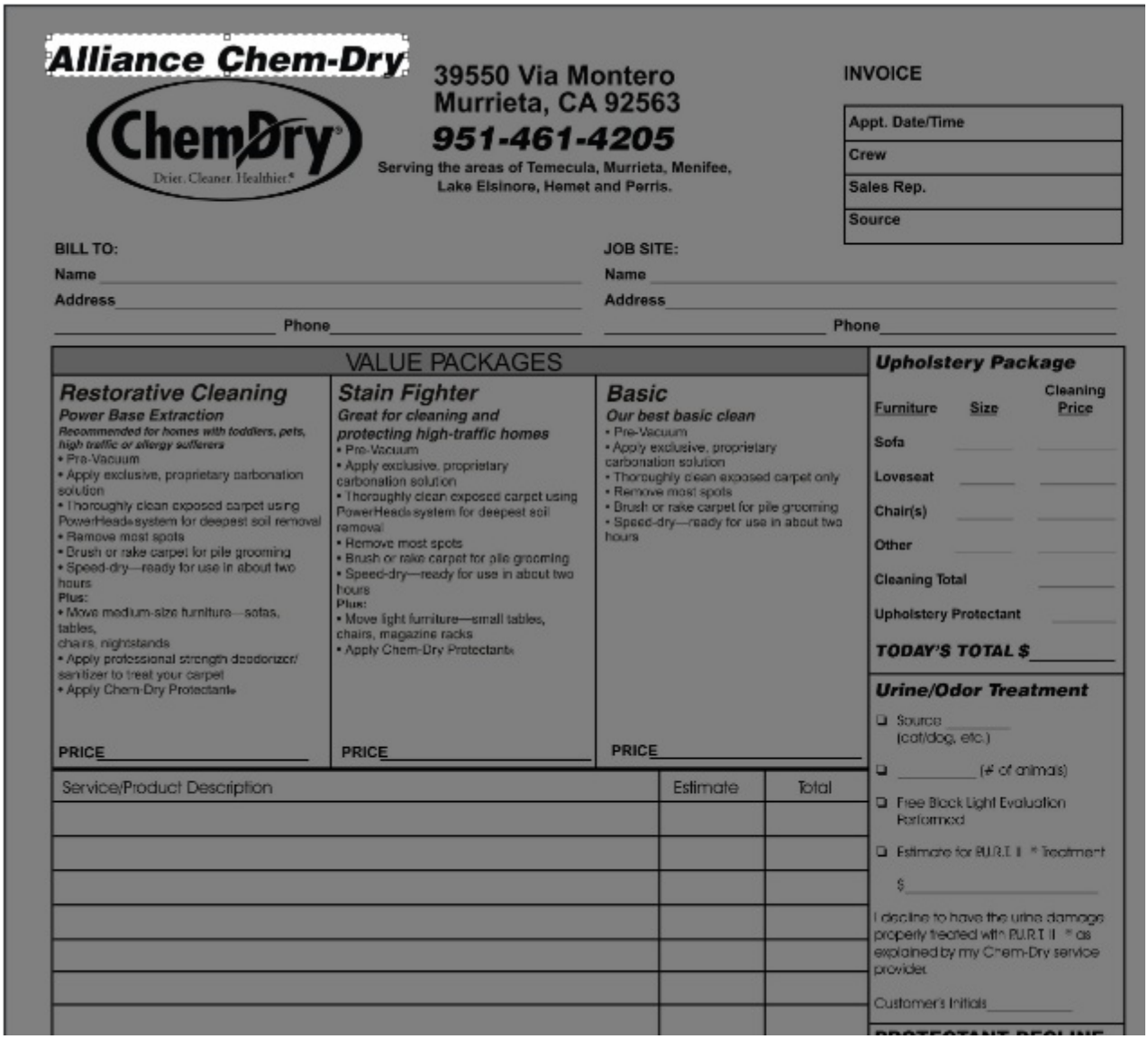
Urine/Odor Treatment
❑ Source (cat/dog, etc.)
❑ (# of animals)
❑ Free Black Light Evaluation Performed
❑ Estimate for P.U.R.I.® Treatment
$
I decline to have the urine damage properly treated with P.U.R.I.® as explained by my Chem-Dry service provider.
Customer's Initials

PROTECTANT DECLINE
I acknowledge that I was offered and informed about the benefits of applying protectant to my carpet and/or upholstery.
Customer's Initials

- *Add Data Capture fields according to your convenience.*
  - *Data Capture*

- *Click on*  ***Fields***.
- *To create a new Data Capture*
  - *Click on*  ***New Data Capture Field icon***.
  - ***Fill in the fields***.
  - ***Choose the area with mouse on the left image***.
  - *Click on* ***Create Button***.
- *To edit a Data Capture:*
  - *Click on*  ***Edit a Data Capture icon***.
  - ***Modify the fields***.
  - *Click on* ***Edit button***.
- *To delete a Data Capture:*
  - *Click on*  ***Delete a Data Capture icon***.
  - *Click on* ***Delete button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


Alliance Chem-Dry

ChemDry
Drier. Cleaner. Healthier.®

39550 Via Montero
Murrieta, CA 92563
951-461-4205
Serving the areas of Temecula, Murrieta, Menifee, Lake Elsinore, Hemet and Perris.

INVOICE

Appt. Date/Time
Crew
Sales Rep.
Source

BILL TO:
Name
Address
Phone

JOB SITE:
Name
Address
Phone

VALUE PACKAGES

Restorative Cleaning
Power Base Extraction
Recommended for homes with toddlers, pets, high traffic or allergy sufferers
• Pre-Vacuum
• Apply exclusive, proprietary carbonation solution
• Thoroughly clean exposed carpet using PowerHeads system for deepest soil removal
• Remove most spots
• Brush or rake carpet for pile grooming
• Speed-dry—ready for use in about two hours
Plus:
• Move medium-size furniture—sofas, tables, chairs, nightstands
• Apply professional strength deodorizer/ sanitizer to treat your carpet
• Apply Chem-Dry Protectant®
PRICE

Stain Fighter
Great for cleaning and protecting high-traffic homes
• Pre-Vacuum
• Apply exclusive, proprietary carbonation solution
• Thoroughly clean exposed carpet using PowerHeads system for deepest soil removal
• Remove most spots
• Brush or rake carpet for pile grooming
• Speed-dry—ready for use in about two hours
Plus:
• Move light furniture—small tables, chairs, magazine racks
• Apply Chem-Dry Protectant®
PRICE

Basic
Our best basic clean
• Pre-Vacuum
• Apply exclusive, proprietary carbonation solution
• Thoroughly clean exposed carpet only
• Remove most spots
• Brush or rake carpet for pile grooming
• Speed-dry—ready for use in about two hours
PRICE

Service/Product Description	Estimate	Total

Upholstery Package

Furniture	Size	Cleaning Price
Sofa		
Loveseat		
Chair(s)		
Other		
Cleaning Total		
Upholstery Protectant		

TODAY'S TOTAL $

Urine/Odor Treatment
❑ Source (cat/dog, etc.)
❑ (# of animals)
❑ Free Black Light Evaluation Performed
❑ Estimate for P.U.R.T.® Treatment
$
I decline to have the urine damage properly treated with P.U.R.T.® as explained by my Chem-Dry service provider.
Customer's Initials

Edit a OCR template

- *Click on*  ***Edit OCR template icon***.
- ***Modify de fields***.
- *Click on* ***Edit button***.

Edit a OCR template fields

Edit a OCR Validation Control

Click on  ***Fields***.

- *Click on*  ***Edit a Validation Control icon***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- ***Modify the fields***.
- *Click on* ***Edit button***.

Edit a OCR Data Capture

Click on ***Fields***.

- *Click on* ***Edit a Data Capture icon***.
- ***Modify the fields***.
- *Click on* ***Edit button***.

Delete a OCR template fields

Delete a OCR Validation Control

Click on ***Fields***.

- *Click on* ***Delete a Validation Control icon***.
- *Click on* ***Delete button***.

Delete a OCR Data Capture

Click on ***Fields***.

- *Click on* ***Delete a Data Capture icon***.
- *Click on* ***Delete button***.

Check OCR template

- *Click on* *Check icon and the OCR template will be checked.*

In the Capture Data field is shown the result of the text extraction by the OCR.

In the Validation Control field is shown the result of the compliance validation (true or false).

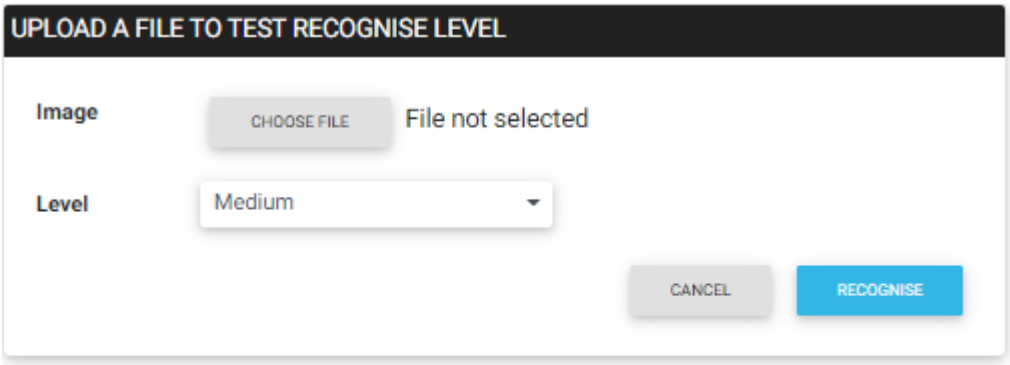
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OCR TEMPLATE DEFINITION CHECK: Test

Field	Class	Type	Zone
Title	String	Data Capture field	
Street name	String contains	Validation Control field	

Check recognise a document

- *On top right click on* ***Recognise icon***.
- ***Choose the file to be recognised*** *(allowed MIME type).*
- ***Choose the recognise level***.
- *Click on* ***Recognise button***.



Name	Confidence level	Control	Recognised
Test	50 %	(1 of 2)	✕

Recognise level, use more or less speed to identify a document. By default application uses slow.

- *In the fast level, it takes only in consideration the Validation Control fields.*
- *In the medium level, it also takes in consideration that the Data Capture fields must not be empty.*
- *In the slow level, it also takes in consideration that the Data Capture fields are right (complish with patterns etc...).*

Delete a OCR template

- *Click on*  ***Delete OCR template icon***.
- *Click on* ***Delete button***.

Enable OCR Data Capture

Feature	*Description*
Profile General tab	*When the OCR data capture is enabled from the Profile General tab, the OCR Data Capture process will be executed automatically on each new uploaded document.* *In case that is present an automation definition, these parameters have no effect.* *With the automation feature all these cases and more are covered.*
Automation	*Validations:* • ***Is OCRDataCaptureFile*** *More information about Built-in Automation Validator plugins.* *Actions:* • ***OCR DataCapture*** • ***Add OCRDataCaptureToWizard*** *More information about Built-in Automation Action plugins.*

Built in OCR Validation Control plugins

Name	*Value*	*Pattern*	*Description*
String contains	*Optional*	*Optional.*	*Returns the evaluation of a "contains operation" with a value or a pattern within the text extracted by OCR.* *In case the pattern field is empty and the value field is not empty, it returns true if the text extracted by OCR contains the value of the value field.* *In case the pattern field is not empty and the value field is empty, it returns true if the text extracted by OCR matches the pattern only one time.* *In case the pattern field is not empty and the value the field is not empty, it returns true if the text extracted by OCR matches the pattern only one time and the matched pattern contains the value field.* *In case the pattern and value fields are empty, it returns false.*
String equals	*Optional.*	*Optional.*	*It returns the evaluation of equal operations with a value or a pattern with the text extracted by OCR.* *In case the pattern field is empty and the value field is not empty, it returns true if the text extracted by OCR equals the value of the value field.* *In case the pattern field is not empty and the value field is empty, it returns true if the text extracted by OCR matches the pattern only one time.* *In case the pattern field is not empty and the value field is not empty, it returns true if the text extracted by OCR matches the pattern only one time and the matched pattern is equal to the value field.* *In case the pattern field is empty and the value field is empty, it returns false.*
Similar	*Optional.*	*Optional*	*Returns the evaluation of similar operations with some values or a pattern with the*

String			*text extracted by OCR.* *The "similar operation" counts the number of changes that must be done to transform the expected value with the text extracted by ocr. This is the **distance between both texts**.* *This plugin works in combination with a configuration parameter named ocr.control.similar.distance. By default the value of this parameter is 5.* *The field value is mandatory, can not be empty.* *In case a pattern field is empty and the value field is not empty, it returns true if the distance between the text extracted by OCR and the value of the value field is less than the one set in ocr.control.similar.distance parameter.* *In case a pattern field is not empty and the value field is not empty, it returns true if the text extracted by OCR matches the pattern only one time and the distance between the matched text and the value of the value field is less than set in ocr.control.similar.distance parameter.* *The field value is mandatory and the field pattern is optional.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Built in OCR Data Capture plugins

Name	*Pattern*	*Description*
String	*Optional.*	*Return a text value.* *In case the pattern is empty returns text extracted by OCR.* *In case the pattern is not empty, returns the text extracted by OCR that matches the pattern only one time.* *In case the text extracted by OCR is empty, it returns and Exception.* *In case the pattern is not empty and the text extracted by OCR doesn't match the pattern only one time,it returns an Exception.*
Number	*Optional.*	*Return a number.* *In case the pattern is empty, it returns a number based on the text extracted by OCR.* *In case the pattern is not empty,it returns a number based on the text extracted by OCR that matches the pattern.* *In case the text extracted by OCR is empty, it returns an exception.* *In case the pattern is not empty but the text extracted by OCR doesn't match the pattern, it returns an exception.* *In case the text extracted can not be converted into a number, it returns an exception.*
Date	*Mandatory.*	*Returns a Calendar value.* *In case the pattern is not empty, it returns a Calendar based on the text extracted by OCR and the pattern.* *In case the pattern is empty returns an exception.* *In case the text extracted by OCR is empty it returns an exception.* *In case the text extracted can not be converted to a Calendar, it returns an exception.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Barcode	*Optional.*	*Return barcode text.* *In case the pattern is empty, it returns a text extracted by a Barcode engine.* *In case the pattern is not empty, it returns the text extracted by a Barcode engine that matches the pattern.* *In case the text extracted by a Barcode engine is empty, it returns an exception.* *In case the pattern is not empty but text extracted by a Barcode engine doesn't match the pattern, it returns an Exception.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Languages

The application can work with several languages, here are set the languages terms definitions.

Languages are shown in a table structure.

Id	Flag	Name	Translation status

The "en-GB" translation is used by application as default translation.

Sometimes translations are not fully completed and some terms are missing. The "en-GB" is used to identify which are the missing terms and how many are.

When a term is missing in other language the "en-GB" term is used in its place.

en-GB translation can not be removed.

Create a new language

- *Click on* ***new language icon***.
- *Fill the fields*
  - *The id must be compliant with ISO 639 (2 levels).*
  - *Language name*
  - *The icon should be 16x11 px*
- *Click on* ***Create button***.

More information about ISO 639.

Edit a language

- *Click on* ***Edit language icon***.
- ***Change the fields***.
- *Click on* ***Edit button***.

Delete a language

- *Click on* ***Delete language icon***.
- ***Change the fields***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Click on* ***Delete button***.

Edit language translations

> *On the right top the translation can be filtered by key.*

- *Click on* ***Edit language translation icon***.

Languages translations are shown in a table structure.

Module	*Key property*	*English translation*	*Translation*

> *By default are* ***tree modules*** *that indicate where translations are used:*
>
> - *frontend (used on frontend)*
> - *mobile (used on mobile view)*
> - *extension (used on frontend by extension modules)*
>
> ***Two key property can not be equals***.

- ***Change the fields***.
- *Click on* ***Edit button***.

Export a language translations

- *Click on* ***Export language translations icon***.
- *Application* ***retrieve a SQL file*** *with language translation into.*

Import a language translations

> *Download your language package from Language translations.*

- *On right bottom* ***choose a Language package file***.
- *Click on* ***Add new translation button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Language translations

Language	File	Number of terms
Arabic	OpenKM_8.1_ar-AR.sql	559
Azeri	OpenKM_8.1_az-AZ.sql	264
Belgian/Flemish	OpenKM_8.1_nl-BE.sql	97
Bosnian	OpenKM_8.1_bs-BS.sql	88
Bulgarian	OpenKM_8.1_bg-BG.sql	97
Català	OpenKM_8.1_ca-ES.sql	115
Chinese simple	OpenKM_8.1_zh-CN.sql	116
Chinese traditional	OpenKM_8.1_zh-TW.sql	116
Czech	OpenKM_8.1_cs-CZ.sql	645
Danish	OpenKM_8.1_dk-DK.sql	96
Deutsch	OpenKM_8.1_de-DE.sql	616
Deutsch - Austria	OpenKM_8.1_de-AT.sql	97
Dutch	OpenKM_8.1_nl-NL.sql	118
English	OpenKM_8.1_en-GB.sql	661
English - USA	OpenKM_8.1_en-US.sql	88

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Español	***OpenKM_8.1_es-ES.sql***	*661*
Español Colombia	***OpenKM_8.1_co-ES.sql***	*88*
Euskera	***OpenKM_8.1_eu-ES.sql***	*181*
Farsi	***OpenKM_8.1_fa-FA.sql***	*88*
French	***OpenKM_8.1_fr-FR.sql***	*690*
Gallego	***OpenKM_8.1_gl-ES.sql***	*88*
Greece	***OpenKM_8.1_el-GR.sql***	*113*
Hebrew	***OpenKM_8.1_he-IL.sql***	*251*
Hungarian	***OpenKM_8.1_hu-HU.sql***	*631*
Indonesian	***OpenKM_8.1_id-ID.sql***	*88*
Italian	***OpenKM_8.1_it-IT.sql***	*170*
Japanese	***OpenKM_8.1_ja-JP.sql***	*96*
Korean	***OpenKM_8.1_ko-KR.sql***	*214*
Latvian	***OpenKM_8.1_lv-LV.sql***	*88*
Lithuanian	***OpenKM_8.1_lt-LT.sql***	*88*
Macedonian	***OpenKM_8.1_mk-MK.sql***	*88*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Norwegian	***OpenKM_8.1_nb-NO.sql***	*118*
Polish	***OpenKM_8.1_pl-PL.sql***	*191*
Português do Brasil	***OpenKM_8.1_pt-BR.sql***	*710*
Romanian	***OpenKM_8.1_ro-RO.sql***	*88*
Russian	***OpenKM_8.1_ru-RU.sql***	*115*
Serbian	***OpenKM_8.1_sr-SR.sql***	*98*
Slovak	***OpenKM_8.1_sk-SK.sql***	*116*
Suomi	***OpenKM_8.1_fi-FI.sql***	*111*
Swedish	***OpenKM_8.1_sv-SE.sql***	*147*
Thai	***OpenKM_8.1_th-TH.sql***	*88*
Turkish	***OpenKM_8.1_tr-TR.sql***	*119*
Vietnamese	***OpenKM_8.1_zh-VN.sql***	*116*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

How to create a new translation



Before starting with a new translation, please check if is available in Language translations section.

A sample of new language base on Oromo from Ethiopia

*The first step is identifying what is your language code based on ISO 639 (2 level). Looking at the table of language identifiers, the code might be "**om-ET**".*

*The second step will get the country flag, in this case, Ethiopia. Download the flag icons. The flags icon name are based ISO 369 (one level), that means we are looking for a file named "**et.gif**".*

The third step will be to create the new language. Go to Administration > Languages and create a new language with these values:

- *Click on*  ***new language icon***.
- *Fill the fields*
  - *The "id" might be "**om-ET**".*
  - *The "name" might be "Oromo".*
  - *The "icon" might be the file named "**et.gif**".*
- *Click on **Create button**.*

Once the new language has been created must set the translations:

- *Click on*  ***Edit language icon***.
- ***Change the fields***.
- *Clic,k on **Edit button**.*

Finally, think on sharing with the community your translation.

- *Click on*  ***Export language translations icon***.
- *The application **retrieves a SQL file** with language translation into.*
- *Contact us at the contact website form and we will tell you the way for sharing with us the translation.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Utilities

The utilities section at Administration is a collection of useful features designed to be used by IT staff.

To take advantage of most of these feature some technical skills are needed .

These features can be categorized as:

Category	*Description*
Security	• *Admin permissions* • *Download tokens* • *Purge data* • *Purge trashes* • *Password encryption*
Performace	• *Cache stats* • *Clean conversion cache* • *Hibernate stats* • *Profiling stats*
Low level access	• *Database query* • *Execute JAR* • *Scripting*
Check specific configuration	• *Check email* • *Check text extraction* • *LDAP Search* • *System properties* • *Create support zip*
Import & export	• *Database import* • *Repository import* • *Repository export*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Search engine	• *List indexes* • *Rebuild indexes*
Others	• *Folder style* • *Orphan Files* • *Plugins* • *Repository checker* • *Shards* • *Thesaurus*
Experimental features	• *LogCat*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Admin permissions

The Admin permission feature helps the administrators to set the Administration access levels to users who have access to the application.

For example, it provides access to some users only for managing users and roles.

On this view it is shown a table with all the users who have access to administration based on admin role.

More information about default admin role at Security configuration parameters.

Change user level access

- *Click on*  ***Edit user level access icon***.
- ***Change level access***.
- *Click on* ***Edit button***.

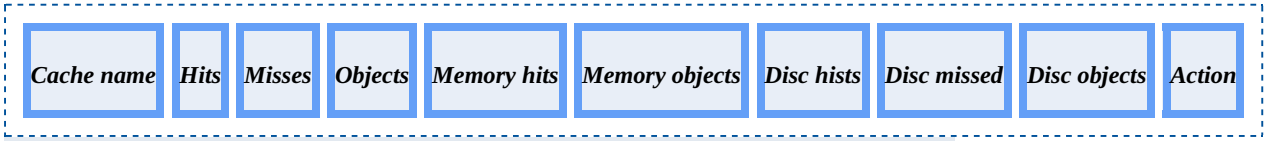
Cache stats

OpenKM takes advantage of cached objects for fast retrieving data.

> *For example the users and roles list, when application is configured to working with LDAP, are cached to reduce the traffic between application and LDAP server.*
>
> *The associated cached name for LDAP are:*
>
> - *com.openkm.cache.ldapPrincipalAdapter.general*
> - *com.openkm.cache.ldapPrincipalAdapter.name*

Also cache can be used to collect data useful to find out configuration parameters changes needed for better application perfomance. For this purpose the application comes with a cache audit trail.

Cache stats are shown in a table structure:



Field	Description	Audit trail required
Cache name	Unique class name of the cached class.	Not applicable.
Hits	Number of successful queries.	true
Misses	Number of failed queries.	true
Objects	Number of cached objects.	false
Memory hits	Number of successful queries in cache memory.	true
Memory missed	Number of failed queries in cache memory.	true
Memory objects	Number of objects in cache memory.	false
Disc hits	Number of successful queries in cache disk.	true

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Disc missed	*Number of failed queries in cache disk.*	*true*
Disc objects	*Number of objects in cache disk.*	*false*
Action	*Clean cached objects.*	*Not applicable.*

Clean all cache

- *On the right top click on*  ***Reset all button***.

Refresh cache stats

- *On the top right click on*  ***Refresh icon***.

Enable cache stats audit trail

- *On the top right click on*  ***Enable cache audit trial button***.

Enables the fields stats:

- *Hits.*
- *Misses.*
- *Memory Hits.*
- *Memory missed.*
- *Memory objects.*
- *Disc hits.*
- *Disc missed.*

Disable cache stats audit trail

- *On top right click on*  ***Disable cache audit trial button***.

Disable the fields stats:

- *Hits.*
- *Misses.*
- *Memory Hits.*
- *Memory missed.*
- *Memory objects.*
- *Disc hits.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Disc missed.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Check email

The check email feature helps the administrator to check the application mail configuration.

Check email

- *Fill the fields*
  - *From*
  - *To*
  - *Subject*
  - *Content*
- *Click on **Check button***



If the application mail configuration application is correctly set it will return an "OK" otherwise will be shown the cause of the error.



*If the **send.mail.from.user** configuration property is set to **false**, the "From" field will be ignored and will make use of the **mail.from** JavaMail configuration property. Note that depending on the SMTP server configuration only the address configured in **mail.smtp.user** may be able to send messages, this mean, the user who appears in the "From" field.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Check text extraction

This feature helps the administrator to check the text extraction process.

It can check the extraction process based on:

- *Document to be processed*
  - *Document UUID (see tab properties when document is selected on general application user interface).*
  - *Document path.*
  - *Document outside application (choose a document from file system to be uploaded).*
- *List of available text extractors.*

Check text extraction

- ***Choose a document** to be processed*
  - *OpenKM document:*
    - *Fill field document UUID or document path.*
  - *External document:*
    - *Choose a document from file system to be uploaded.*
- ***Choose one text extractor** from available text extractors list.*
- *Click on **Check button**.*



After the process is completed it will show the text extracted by the selected text extractor.



The text extraction process can take some seconds or minutes depending on the size of the document to be processed and the file MIME type.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Clean conversion cache

The application comes with integrated file conversion cache.

File conversion cache is stored in the file system, by default into $TOMCAT/repository/cache folder.

A classical example of document conversion is generating the thumbnail of each document shown in the default application user interface. For example to get a thumbnail of a Microsoft Word document is needed convert the first page as an image. For doing this process application consumes some CPU.

To prevent it consumes more CPU than necessary, the documents are converted only when it's necessary, first time, or after document update. In other cases the file is retrieved from cached files.

The administrator can take control of these cached files and clean by type or all.

Cached files:

Type	*Description*
DXF	*Caused by autocad pre visualization conversion.*
PDF	*Several causes:* • *Document pre visualization.* • *User conversion by demand.*
SWF	*Document pre visualization.*
Thumbnail properties	*Document thumbnail in application desktop view.*
Thumbnail Search	*Document thumbnail in application search view.*
Thumbnail Lightbox	*Document thumbnail in lightbox.*

Clean conversion cache

- ***Choose cached file to be cleaned***
- *Click on* ***Clean button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



The operation can take few seconds or minutes depending the number of files to be removed from the file system.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Create support zip

This feature allow the user to download a zip file with important info about current installation. It contains the following files:

- ***env.properties**: environment properties*
- ***openkm.log**: application log*
- ***openkm.properties**: OpenKM configuration file*
- ***openkm.xml**: OpenKM configuration file*
- ***server.xml**: tomcat configuration file*
- ***system.properties**: system properties*



This feature will download a zip file to your computer.



It will only download application log for the current day. Old logs will not be donwloaded.

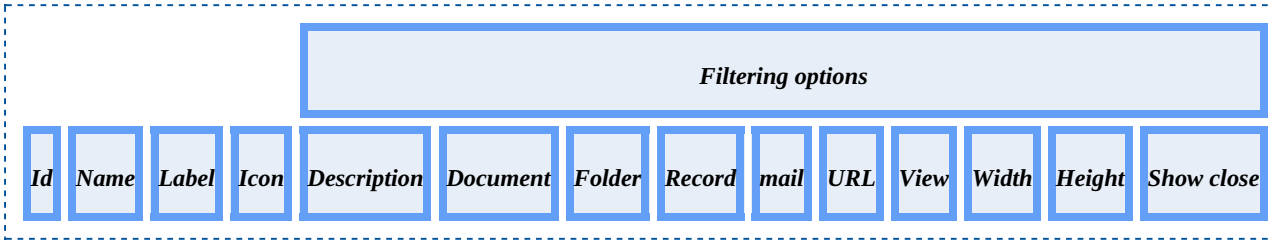
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Custom actions

The Custom actions feature helps to add custom actions options in the Kcenter user interface

> *Sample at Custom actions in KCenter*

Users are shown in a table structure.



User parameters:

Field	*Description*	*Required*
Id	*The internal Id of the custom action.*	*Set by the application*
Name	*The name of the custom action.*	*true*
Label	*The label of the custom action.* > *When you wish to internationalize the label can use a language key value*	*true*
Icon	*The MDI icon value*	*true*
URL	*URL to be requested by the custom action*	***
Description	*Description of the custom action.*	*false*
View	*Allowed options:* • *Popup-wizard (the custom action is shown in a popup)*	*true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	• *Full-screen (the custom action is shown in the whole user interface screen)* • *Run-javascript (the script will be executed in the UI without showing any kind of panel)*	
Width	*A number or % is allowed.* *Samples:* • *300* • *80%*	*
Height	*A number or % is allowed.* *Samples:* • *300* • *80%*	*
Apply to document	*When enabled the custom action will be available for documents.*	*true*
Apply to folder	*When enabled the custom action will be available for documents.*	*true*
Apply to record	*When enabled the custom action will be available for records.*	*true*
Apply to mail	*When enabled the custom action will be available for mail.*	*true*
Show close button	*When enabled it is shown a close button*	**
Script	*The javascript code that will be executed by the UI*	****

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Action	*Available actions:* •  *Edits a custom action.* •  *Deletes a custom action*	

** Mandatory when Popup-wizard is selected.*

*** Option is shown when Popup-wizard is selected.*

**** Mandatory when Popup-wizard or Full-screen are selected*

***** Mandatory when Run-javascript is selected.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Database import

The database importer helps administrator to import data from a database source to a destination folder within current OpenKM.

Field	*Description*
DB URL	*Connection url of a source OpenKM database.*
DB user	*Name of the user of the source OpenKM database.*
DB password	*Password of the user of the source OpenKM database.*
Source folder	*UUID of the source folder from the source OpenKM.*
Destination folder	*UUID of the destination folder within current OpenKM.*
Thread pool size	*Number of threads for executing the process.*
Source tenant id	*Id of the tenant from a source OpenKM.*
Concurrent documents	*Check if you want to import this data.*
Concurrent folders	*Check if you want to import this data.*
Concurrent records	*Check if you want to import this data.*
Concurrent mails	*Check if you want to import this data.*
Nodes	*Check if you want to import this data.*
Users and roles	*Check if you want to import this data.*
Node relations	*Check if you want to import this data.*
Bookmarks	*Check if you want to import this data.*

Import

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Set the DB URL.*
- *Set the DB user.*
- *Set the DB user password*
- *Set the source folder.*
- *Set the destination folder*
- *Set the thread pool size*
- *Set the source tenant id.*
- *Check the database import options.*
- *Click on **Import button**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Database query

This feature helps administrator accessing database data, easily and quickly without thirdparty database software client.

The results of the queries can be exported to SQL or CSV file formats.

Allowed types:

Type	Description	List	Export
JDBC	*SQL queries supported by the configured database.*	*When choosen is shown a list of available tables.*	*Yes.*
Hibernate	*Neutral database queries based on hibernate queries format.*		*No.*

- *Is allowed writing more than one query.*
- *Each query must be written on a single line, break lines are not allowed.*
- *Execute will process all written queries in same order has been written, top to bottom. Queries are executed as a FIFO (first in first out).*
- *In case some query fails, that will not stop the execution of the others.*

Execute JDBC query

- *On bottom right **choose type JDBC**.*
- *Write some valid SQL.*
- *Click on **Execute button**.*
- *Optionaly the results can be exported to SQL or CSV*
  - *Click on*  *Export to sql icon.*
  - *Click on*  *Export to csv icon.*

Example:

```
// nodeBase = documents + folders + mails + records
SELECT count(*) from OKM_NODE_BASE;
SELECT count(*) from OKM_NODE_DOCUMENT;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
SELECT count(*) from OKM_NODE_FOLDER;
SELECT count(*) from OKM_NODE_MAIL;
SELECT count(*) from OKM_NODE_RECORD;

// select first 10 records
SELECT * from OKM_NODE_BASE LIMIT 10;
SELECT * from OKM_NODE_DOCUMENT LIMIT 10;
SELECT * from OKM_NODE_DOCUMENT_VERSION LIMIT 10;
SELECT * from OKM_NODE_FOLDER LIMIT 10;
SELECT * from OKM_NODE_MAIL LIMIT 10;
SELECT * from OKM_NODE_RECORD LIMIT 10;
```

Execute Hibernate query

- *On bottom right* ***choose type Hibernate***.
- *Write some valid HQL.*
- *Optionally you can check the "Show SQL" option.*
- *Click on* ***Execute button***.

When "Show SQL" option is checked is shown in results the real sql executed.

Example:

```
select count(*) from NodeBase;
select count(*) from NodeFolder;
```

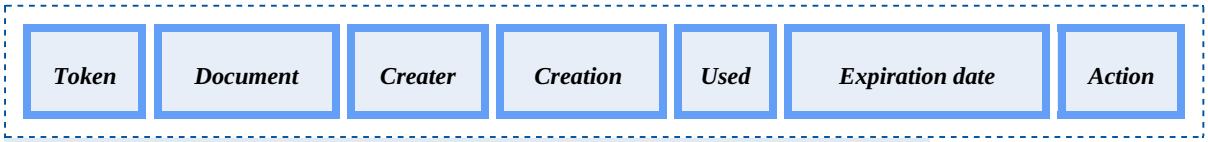
Allowed sentences

- *On bottom right* ***choose a valid sql file***.
- *Click on* ***Import SQL script button***.

Download tokens

The download token feature helps application users to share documents with external users without need to be logged into the application. With Download tokens log feature, the administrator can take control over the documents that has been shared, the users who shared it and when had been downloaded.

Download tokens log are shown is a table structure:



Field	Description
Token	*The unique token UUID.*
Document	*The shared document.*
Creater	*Id of the user who shared the document.*
Creation	*Date when the document had been shared.*
Used	*Indicates if the document has been downloaded or not.*
Expiration time	*Indicates the date when the document will not be longer downloaded.*
Action	*Deletes a token*

Delete token

- *Click on*  ***Delete token button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Execute JAR

A JAR file is a type of compressed file, which is actually using the same compression algorithms and format as ZIP files.

JAR files normally contain a number of Java Classes and some metadata (a manifest file). Java classes are part of applications which are written in Java.

This feature helps administration to create its own classes, packaged into JAR file and execute from OpenKM.

Examples of usage:

- *When administrator wants to do some operation across all repository that take effect on a lot of nodes, based on some logic or special conditions. Can take advantage building minimal code with OpenKM API for it. Some examples could be, change security, remove or move files, rename files based on their metadata values, among others.*
- *Check crontab task.*

Requisits:

- *Must implement the class "**public static void main(String[] args)**".*

```
package com.openkm;

public class CronTabTest {
    public static void main(String[] args) {
        // Some stuff here
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Hibernate stats

With the Hibernate stats feature the administrator can take control of value information about database queries, finding out performance problems and the possible causes.

By default Hibernate stats feature is disabled because the log analysis consumes hardware resources.

This feature should only be enabled by administrator in order to investigate possible performance problems with application.

Each time the application is restarted all the previous analysis is removed. It's important that wh he or she n the administrator enables this feature, takes in consideration that the application should not be stopped.

Information provided by Hibernate stats

Information provided by Hibernate stats are shown in several tables:

- *General data.*
- *Query statistics.*
- *Entity statistics.*
- *Collection statistics.*
- *2nd level cache statistics.*

Time columns are finished with dur. name:

- *Max dur (Maximum duration).*
- *Min dur (MInimum duration).*
- *Avr dur (Average duration).*
- *Total dur (Accumulated total time).*

Columns with time values are in miliseconds.

When the application presents some performance problem, in most cases it will be caused by some query that need a huge time to complete its execution.

*On Query statistics are shown, all the database queries executed by application, the most important column value is **Max dur.** which indicates the Maximum duration of the query.*

Enable Hibernate stats

This option is only available when Hibernate stats is stopped.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *On right top click on*  *Enable button.*

Clear Hibernate stats

- *On right top click on*  *Clear button.*

Disable Hibernate stats

> *This option is only available when Hibernate stats is running.*

- *On right top click on*  *Disable button.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

LDAP Search

The LDAP Search feature helps administrator to checking LDAP configuration.

> *Application LDAP integration is based on a subset of LDAP queries, for example retrieving LDAP users or LDAP groups among others.*
>
> *More information at LDAP configuration.*

These are the available parameters:

Field	*Description*
Host	*The LDAP host.*
User	*The LDAP user distinguised name.*
Password	*The LDAP user password.*
Search base	*The LDAP search base.*
Search filter	*The filter used in LDAP search.*
Attribute	*The atrribute returned by LDAP search.*
Referral	*In almost all cases this value is empty. Use "**follow**" to indicate several domain servers working together (balanced).*

Example of a query:

Field	*Description*
Host	*ldap://192.168.1.34:389*
User	*CN=Administrator,OU=openkm,DC=company,DC=com*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Password	**secret**
Search base	*DC=company,DC=com*
Search filter	*objectclass=user*
Attribute	*sAMAccountName*
Referral	*In almost all cases this value is empty. Use "**follow**" to indicate several domain servers working together (balanced).*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

List indexes

The List index feature helps the administrator to take control about how the objects (documents, folders, mails and records) are indexed into Lucene search engine.

At the main screen is shown a table with the Lucene indexed fields of each object.

Common indexed fields:

Field	*Description*
#	*Number*
_hibernate_class	*The hibernate class name:* • *com.openkm.dao.bean.NodeFolder* • *com.openkm.dao.bean.NodeDocument* • *com.openkm.dao.bean.NodeMail* • *com.openkm.dao.bean.NodeRecord*
uuid	*Object unique identifier.*
parent	*Parent node.*
context	*Context values:* • *okm_root* • *okm_metadata* • *okm_categories* • *okm_thesaurus* • *okm_templates* • *okm_personal* • *okm_mail* • *okm_personal* • *okm_trash*
tenant	*Tenant identifier.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

author	*Id of the user author.*
created	*Indicate when object was created.*
name	*Object name.*
userPermission	*User permissions.*
rolePermission	*Role permissions.*



*Depending the type of object (**_hibernate_class**) can be more indexed fields.*

Search indexes

- *On top right click on*  ***Search indexes button***.
- *In input box copy some object UUID (from desktop view, you can get at Properties tab any object UUID, select the UUID and press CTRL+C to copy).*
- *Click on* ***Search button***.
- *From the table list click on*  ***List icon***.



*The filtering input box use the sintaxis **FIELD : value** , here some examples:*

- *uuid:ce3338a5-b1ec-481b-9d59-f5aeb3fcf42b*
- *name:invoice.**
- *text:telephone*

Let's see an example. Imagine you have two documents called "Parana.doc" and "Parana.pdf". You can locate them by this search:

```
name:paran*
```

But be careful because using quotation marks you won't get any result:

```
name:"paran*"
```

If you want to locate all PDF documents which name starts with "Paran", the right query is:

```
name:paran*pdf
```

This query would also work (note the dot before que extension):

```
name:paran*.pdf
```

*But this query will failt because you cannot use a * or ? symbol as the first character of a search. This is a Lucene limitation.*

```
name:*pdf
```

If you want to lookup documents with a given extension, I would recommend using the MIME Type property this way:

```
mimeType:application/pdf
```

In this query we are looking for all documents which name starts with "parana" and has a keyword called "dog":

```
name:parana* AND keyword:song
```

You can learn more about query syntaxt at Lucene Query Syntax.

Enable show terms

When show terms enabled, is shown and extra fields named terms on documents objects with the terms used by Lucene search engine for indexing that document.

- *On top right click on* ***Show terms button***.

Disable show terms

- *On top right click on* ***Show terms button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

LogCat

The LogCat feature helps administrator to accessing Tomcat log files directly from application.

Logs are shown is a table structure:

File	Action

*The main Tomcat log file is named **catalina.log**.*

This file rotate each day at night creating a new catalina.log file and saving older with name catalina.log.year-month-day

Purge all logs

The purge option will clean all Tomcat logs.

- *On top right click on*  ***Purge button***.

Download file as ZIP

- *Choose a log file.*
- *Click on*  ***download as ZIP icon***.

Browser log file

- *Choose a log file.*
- *Click on*  ***browse icon***.
- *Use filter options.*

Filter options:

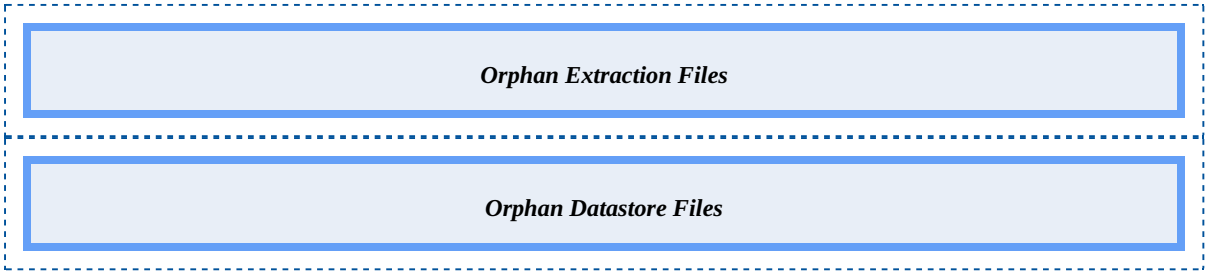
Option	Description
Begin	*Indicate the row position.* *By default 0 indicate the starting of the document.*
End	*Indicate the row position.*

	By default -1 indicate the end of the document.
String	*Filter by some text.*

Orphan Files

The Orphan files feature to display all orphan extraction files and orphan datastore files.

Both are shown in a table structure:



Filter action

- *To search files click on*  ***Filter button***.

Password encryption

The password encryption feature helps the administration to encrypt a password that will be used by the OpenKM. For example, with the LDAP integration.

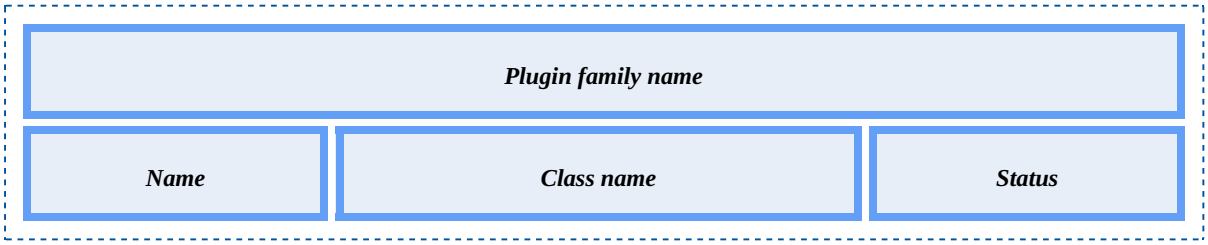
Field	*Description*
Password	*The password to be encrypted*
Algorithm	*Allowed algorithms:* • *DES* • *AES*
Method	*Allowed methods:* • *Server* • *Cluster*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Plugins

The application comes with a plugins feature system that offers an easy way to customize and extend OpenKM functionalities. By default application comes with general purpose plugins implementation, these can be disabled, changed by others or increased in number depending the type of the plugin. For example in antivirus plugin has sense disable or replace by other, on case text extractor has sense disable or increase the number of available text extractors.

Plugins are shown in a table structure:



Plugin families:

Plugin family	*Module*	*Description*
Automation validations	*Automation*	*Validations are used in automation feature.* *More information at:* • *Automation* • *Creating your own Automation Validation*
Automation actions	*Automation*	*Validations are used in automation feature.* *More information at:* • *Automation* • *Creating your own Automation Action*
Template parsers	*OCR Zone*	*Data extraction parser used in OCR Zone.* *More information at:* • *OCR templates* • *Creating your own OCR Data Capture plugin*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Template control parsers	*OCR Zone*	*Document identification check point used in OCR Zone.* *More information at:* • *OCR templates* • *Creating your own OCR Validation Control plugin*
Option select values	*Metadata*	*Retrieve select values from server side logic. For example helps on retrieving data from third-party system.* *More information at:* • *Metadata* • *Metadata Select field* • *Creating your own Option Select Values plugin*
Field validators	*Metadata*	*Validation based on server side logic. Helps on implementing complex validation logics that are not covered from default UI validators.* *More information at:* • *Metadata* • *Metadata Validator element* • *Creating your own Form Validator plugin*
Principal adapters	*Authentication*	*Authentication adapters are used to retrieve users and roles information. For example connect to some LDAP, or external database that stores users and roles.* *More information at:* • *Creating your own Principal Adapter*
Version number adapters	*Document*	*Implements version number logic.* *More information at:* • *Creating your own Version Number Adapter*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Document converters	*Conversion*	*Conversion service is used to convert document to other formats.* *More information at:* • *Creating your own Document Converter*
Metadata adapters	*UI*	*Visualization logic for metadata UI.* *More information at:* • *Creating your own Stack Metadata Adapter (deprecated)*
Text extractor	*Search engine*	*Text extractor based on document MIME TYPE.* *More information at:* • *Creating your own Text extractor*
Antivirus	*Document*	*Antivirus checker. Helps on integrating with any command line antivirus.* *More information at:* • *Creating your own Antivirus Analyzer plugin*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Profiling stats

The Profiling stats feature gives low level value information to the administrator to find out application performance problems.

By default the Profiling stats feature is disabled because the log analysis consumes hardware resources.

This feature should only be enabled by the administrator in order to investigate possible performance problems with application.

Each time the application is restarted all the previous analysis are removed. It's important when administrator enable this feature take into consideration that the application should not be stopped.

Profiling stats are shown is a table structure:



Field	Description
Class	*The Java class name.*
Method	*The Java method name into the class.*
Calls	*Number of calls.*
Max dur.	*Maximum duration of the calls.*
Min dur.	*Minimum duration of the calls.*
Avr dur.	*Average duration of the calls.*
Total dur.	*Accumulated duration time.*
Action	*List of all calls.*

The duration time is in miliseconds.

Enable profiling stats

- *On top right click on*  ***Enable button***.

Disable profiling stats

- *On top right click on*  ***Disable button***.

Clear profiling stats

- *On top right click on*  ***Clear button***.

Refresh profiling stats

- *On top right click on*  ***Refresh button***.

List of detailed calls

- *Click on*  ***View icon***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Purge data

The purge data feature helps administrators to delete users and roles grants assigned to objects (documents, folders, mails or records).

Submenu operations:

- *Purge Mail Account Users.*
- *Purge Permission Roles*
- *Purge Permission Users*
- *Purge Stamp Users*
- *Purge Subscription Users*
- *Purge Orphan Cache Files*
- *Purge Orphan Datastore Files*
- *Purge Orphan Extraction Files*

Purge

This operation can not be reverted, use with care.

- *In any list click on*  ***Delete icon***.

Filter

- *To search in any list click on*  ***Filter button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Purge trashes

The purge trashes feature helps the administrator to clean all users trashes.

When a users deletes objects (folders, documents, mails or records) they go into users trash to prevent accidentally deletes and be able to recover lost data.

Although the users have a feature to clean their own trash, usually they do not do it. That could cause to having a lot of garbage into the trashes.

The administrator periodically should check the repository growing an specially the size of the trashesthey .

Purge

This operation can not be reverted. Use with care.

- ***Write Yes** into the input box.*
- *Click on **Purge button**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Rebuild indexes

The Rebuild Indexes feature helps the administrator to rebuild Lucene indexes.

Indexes	*Description*
Text extractor	*Extracts all document contents and rebuild the indexes.*
Lucene indexes	*Rebuild indexes with actual contents.* *When you choose this option you can rebuild all node types or only the selected. Available options are:* • *NodeDocument for rebuilding nodes of type document.* • *NodeFolder for rebuilding nodes of type folder.* • *NodeMail for rebuilding nodes of type mail.* • *NodeRecord for rebuilding nodes of type record.*
Optimize indexes	*Remove older contents that no longer exists in repository.*

Rebuild

- *Choose the index to rebuild.*
- *Click on **Execute button**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Repository checker

The repository checker feature helps the administrator to verify the repository integrity.

When the administrator thinks the system could be compromised, for example a hard disk failure or similar,he can use Repository checker feature to verify the application integrity.

Option	*Description*
Path	*The path from where will be started the verification process.* *Common values:* • */okm:root* • */okm:trash* • */okm:personal* • */okm:mail* • */okm:templates*
Fast	*The fast verification will only check the existence of the last document version.*
Versions	*Verify the existence of all document versions. This option has priority over fast option.*
Checksum	*Calculate the checksum of repository document and compares with database value.*

Check repository

The process can take several minutes or hours depending on the repository size and the repository options checked.

The repository checker process will show a process status in %, before it is shown the application calculates the total amount of nodes that should be checked in a huge repository this calculation process can take several minutes, please be patient.

- *Set the path.*
- *Choose repository checker options (fast, versions, checksums).*
- *Click on **Check button**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Repository export

The repository export feature helps the administrator to export the whole or part of the OpenKM repository to the file system.

The output result will be the same folders and files structure that are set into the application.

Field	Description
Repository path	*The repository path to export.* *Common values:* • */okm:root* • */okm:templates* • */okm:personal* • */okm:mail*
Filesystem path	*Server destination path.* *Linux example:* */mnt/export* *Windows example:* *c:/export*
Metadata	*When enabled, metadata values are exported.*
History	*When enabled, all documents versions are exported.*

Export

- *Set the repository path.*
- *Check the repository export options.*
- *Click on Export button.*

Understanding output format

Case metadata and history are not checked

The result will be a folder structure with documents and mails into. Although a document has several versions only will be

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

exported the actual version.

Case only metadata is checked

The result will be a folder structure with documents and mails into. Although a document has several versions only will be exported the actual version.

For each folder, document or mail will be created an ***.okm*** *file. For example for a file named "Administration.html" will be created in the same path a file named "Administration.html.okm". The* ***.okm*** *files contains into all the metadata information associated to the node in a JSON format.*

Example of .okm file:


```
{"uuid":"8299d059-7894-468b-ba48-3f3b59ebc37c",
 "author":"okmAdmin",
 "name":"redacted_481.pdf",
 "path":"/okm:root/redacted_481.pdf",
 "created":{"year":2015,
  "month":6,
  "dayOfMonth":1,
  "hourOfDay":17,
  "minute":18,
  "second":0},
  "lastModified":{"year":2015,
     "month":6,
     "dayOfMonth":2,
     "hourOfDay":10,
     "minute":47,
     "second":39},
  "language":"en",
  "keywords":[],
  "categories":[],
  "subscriptors":[],
  "notes":[{"user":"system",
     "date":{"year":2015,
       "month":6,
       "dayOfMonth":2,
       "hourOfDay":10,
       "minute":47,
       "second":40},
     "text":"New version 1.1 by okmAdmin: Signed",
     "path":"7d11d393-df46-49a1-8f29-ae6d0ea4a093"}],
  "propertyGroups":[],
  "version":{"author":"okmAdmin",
      "name":"1.1",
      "created":{"year":2015,
        "month":6,
        "dayOfMonth":2,
        "hourOfDay":10,
        "minute":47,
        "second":39},
      "size":139215,
      "mimeType":"application/pdf",
      "comment":"Signed"},
 "grantedUsers":{"okmAdmin":15},
 "grantedRoles":{"ROLE_USER":15}
}
```



Take in consideration that the JSON output format can change if the main objects (folder, document, mail or record) are added, removed or modified fields.

Case only history is checked

The result will be a folder structure with documents and mails. All document versions will be saved.

Example of output for a document with version 1.0 and 1.1:

- *Administration.html (size = 0 bytes)*
- *Administration.html#v1.0# (contents of version 1.0)*
- *Administration.html#v1.1# (contents of version 1.1)*

Case metadata and history are checked

The result will be a folder structure with documents and mails. All document versions will be saved.

For each folder, document or mail will be created a ***.okm*** *file. For example for a file named "Administration.html" will be created in the same path a file named "Administration.html.okm". The* ***.okm*** *file contains all the metadata information associated to the node in a JSON format.*

Example of output for a document with version 1.0 and 1.1:

- *Administration.html (size = 0 bytes)*
- *Administration.html.okm (metadata associated to the document)*
- *Administration.html#v1.0# (contents of version 1.0)*
- *Administration.html#v1.0#.okm (metadata associated to version 1.0)*
- *Administration.html#v1.1# (contents of version 1.1)*
- *Administration.html#v1.1#.okm (metadata associated to version 1.1)*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Repository import

The repository importer helps administrator to import data from the file system.

Initial import (import standard folders and documents from your openkm server).

Import from previous openkm export.

Field	Description
Filesystem path	*Server import path.*
Repository path	*The repository destination path.* *Common values:* • */okm:root* • */okm:templates* • */okm:personal* • */okm:mail*
Metadata	*When enabled, the importer assumes the metadata information (.okm files) are present.*
History	*When enabled, the importer assumes all document versions of the document are present.* *More information about document versions format at Repository export.*
Restore UUIDs	*When enabled, the importer will keep the same node UUID.* *When applicationis integrated with thirdparty software, usually the thirdparty software stores in their own database UUID of the node as main reference. This references should not be lost, that's why on these scenarios are needed to keep out the UUID's.*

Specially when importing from previous openkm, export must take in consideration that you must use the same import options that had been chosen while exporting, otherwise the import will fails.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Import

- *Set the file system path.*
- *Set the repository path.*
- *Check the repository import options.*
- *Click on **Import button**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Scripting

Scripting was an advanced feature that enables administrators to execute some BeanShell scripts in folders, triggered on every notified event (for example uploading documents).

This feature has been enhanced by Automation (see ExecuteScripting at Built-in Automation Action plugins), which helps you to do the same things with a few mouse clicks. For compatibility reasons there is also an Automation task where you can put a BeanShell code.

OpenKM uses BeanShell. For more information check http://www.beanshell.org/intro.html.

BeanShell is a small, free, embeddable Java source interpreter with object scripting language features, written in Java. BeanShell dynamically executes standard Java syntax and extends it with common scripting conveniences such as loose types, commands, and method closures like those in Perl and JavaScript.

See some examples at Scripting samples.

Execute script as part of application starting process

Can be executed as an script as a part of application starting process. To do so, just create a ***start.bsh*** *file in* ***$TOMCAT_HOME****.*

For example, we use this feature to create a complete environment (create custom users, register property groups, register workflow) each time the OpenKM demo is restarted.

Execute script as part of application stopping process

Can be executed a an script as part of the application stopping process. To do so, just create a ***stop.bsh*** *file in* ***$TOMCAT_HOME****.*

Scripting in Automation

In the Automation Scripting execution, there're several variables that are directly injected in scripting environment that can be used.

List of variables what can be used directly:

Variable	Description
systemToken	The system token.

node	*Involved node (NodeDocument, NodeFolder, etc.)*
uuid	*Node unique identifier.*
file	*Only available when uploading or updating a new document content.*
userId	*The user id.*

Example:

```
import com.openkm.util.*;
import com.openkm.api.*;

OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);

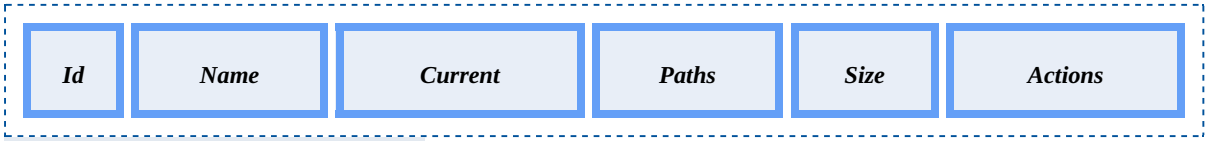
// Variable uuid is injected by automation to the scripting environment
okmProperty.addKeyword(null, uuid, "test");
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Shards

The shards feature to manage disk sharding.

Shards are shown is a table structure:



Field	Description
Id	*The unique id of the shar.*
Name	*The name of the shard.*
Current	*Boolean if it is active.*
Paths	*Paths of the datastores.*
Size	*Total size.*
Actions	*New Shard.* *Refresh.* *Refresh size.* *Edit a shard.*

Shard actions

- *To create new shard click on* ***New shard button***.
- *To refresh click on* ***Refresh button***.
- *To refresh size click on* ***Refresh size button***.
- *To edit shard click on* ***Edit shard button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

System properties

The system properties feature show the JVM variables.

Most common JVM variables:

Key	Description of Associated Value
java.version	Java Runtime Environment version
java.vendor	Java Runtime Environment vendor
java.vendor.url	Java vendor URL
java.home	Java installation directory
java.vm.specification.version	Java Virtual Machine specification version
java.vm.specification.vendor	Java Virtual Machine specification vendor
java.vm.specification.name	Java Virtual Machine specification name
java.vm.version	Java Virtual Machine implementation version
java.vm.vendor	Java Virtual Machine implementation vendor
java.vm.name	Java Virtual Machine implementation name
java.specification.version	Java Runtime Environment specification version
java.specification.vendor	Java Runtime Environment specification vendor
java.specification.name	Java Runtime Environment specification name
java.class.version	Java class format version number
java.class.path	Java class path
java.library.path	List of paths to search when loading libraries
java.io.tmpdir	Default temp file path
java.compiler	Name of JIT compiler to use

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

java.ext.dirs	*Path of extension directory or directories*
os.name	*Operating system name*
os.arch	*Operating system architecture*
os.version	*Operating system version*
file.separator	*File separator ("/" on UNIX)*
path.separator	*Path separator (":" on UNIX)*
line.separator	*Line separator ("\n" on UNIX)*
user.name	*User's account name*
user.home	*User's home directory*
user.dir	*User's current working directory*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow

In the Workflow tenant view are shown the registered workflows in the current tenant. In this view, can enable workflows for the current tenant.

Workflow are shown in a table structure.

Workflow name	***Action***

Description:

Property	*Description*
Workflow name	*Must register the workflow name with the exact name used in the workflow definition*
Actions	*Available actions:* •  *New.* •  *Edit.* •  *Delete*

Register workflow

- *Click on the right top add Workflow tenant.*
- ***Set the workflow name***.
- *Click on* ***Create button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow forms definition

Workflow forms are based on formal XML definition. The DTD (Document Type Definition) defines the structure and the legal elements and attributes of an XML document.

*You can see the latest DTD in our website http://www.openkm.com/dtd/workflow-forms-**{major}.{minor}**.dtd where **{major}** and **{minor}** values are two numbers (major release and minor release). For example the location for version 2.6 of the dtd is http://www.openkm.com/dtd/workflow-forms-2.6.dtd*

Each application version supports only a couple of DTD definitions, read: Application version compatibility table to see which are supported by your application.

Basic XML skeleton

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>

</workflow-forms>
```

- *The **DOCTYPE** is a formal definition of some DTD. In the previous example the **workflow-forms-2.6.dtd**.*
- *The XML begins with **<workflow-forms> tag** and ends with **</workflow-forms>**. Into this hicheracy are defined the metada groups.*

Workflow task form definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.0//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.0.dtd">
<workflow-forms>
  <workflow-form task="evaluate price">

  </workflow-form>
</workflow-forms>
```

- *Each **<workflow-form> ... </workflow-form>** block identify the **beginning and end** of new **workflow form task definition**.*
- *Each task has a **unique name**, **"evaluate price"**.*
- *Must not use special characters for name.*

Attribute	*Description*	*Required*
task	*Unique field identifier.*	*true*

	Two metadata groups can't have the same name. ***Name must be unique****.* *The name must start with "**okg:**".* *Must not use special character for task name. We suggest (0-9a-zA-Z_).*	
defaultValueClassName	*Set custom JAVA class for setting initial default values.*	*false*

Two form task can't have the same name.

Must not use special character for task name. We suggest (0-9a-zA-Z_).

Basic example

Available field types:

- *Checkbox*
- *Button*
- *Download*
- *Input*
- *Iframe*
- *Select*
- *Separator*
- *Suggestbox*
- *Text*
- *TextArea*
- *Upload*
- *VerticalPanel*
- *HorizontalPanel*

More information about field type elements at Workflow fields

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.0//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.0.dtd">
<workflow-forms>
  <workflow-form task="evaluate price">
    <input label="Purchase price" name="price" data="price" readonly="true" />
    <textarea label="Purchase description" name="description" data="description" readonly="true" />
    <button name="approve" label="Approve" transition="approve"/>
    <button name="deny" label="Deny" transition="deny"/>
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Using file system DTD

If the server has no access to the Internet, then the DTD file must be accessible in some way, for example can store a copy in your application server.



The XML definition must be changed to refer to the new DTD location, otherwise you will get and application error while trying to process a DTD file.

It can be a file system (example below) or a valid intranet URL.

```
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.6//EN"
                                 "file:///home/openkm/workflow-forms-2.6.dtd">
```

- *In the example, **/home/openkm/workflow-forms-2.6.dtd** is the file system path to the file.*

Workflow fields

These are the available field types:

Field type	Description
Checkbox	Workflow Checkbox field
Button	Workflow Button field
Download	Workflow Download field
Input	Workflow Input field
Select	Workflow Select field
Separator	Workflow Separator field
Suggestbox	Workflow Suggestbox field
Text	Workflow Text field
TextArea	Workflow TextArea field
Upload	Workflow Upload field
VerticalPanel	Workflow VerticalPanel field
HorizontalPanel	Workflow HorizontalPanel field

Some field types can have children elements, validator or option elements. This hierarchy is permitted:

- *workflow form*
  - *Checkbox*
    - *Validator*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Button*
- *Download*
    - *Node*
    - *Validator*
- *Input*
    - *Validator*
- *Print*
    - *Node*
    - *Validator*
- *Select*
    - *Option*
    - *Validator*
- *Separator*
- *Suggestbox*
    - *Validator*
- *Text*
- *TextArea*
    - *Validator*
- *Upload*
    - *Validator*
- *VerticalPanel*
    - *Checkbox*
    - *Button*
    - *Donwload*
    - *Input*
    - *Print*
    - *Select*
    - *Separator*
    - *Suggestbox*
    - *Text*
    - *TextArea*
    - *Upload*
    - *VerticalPanel*
    - *HorizontalPanel*
- *HorizontalPanel*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Checkbox*
- *Button*
- *Donwload*
- *Input*
- *Print*
- *Select*
- *Separator*
- *Suggestbox*
- *Text*
- *TextArea*
- *Upload*
- *VerticalPanel*
- *HorizontalPanel*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow Button field

Buttons are typically used by users - in the user interface - for decision-making.

For example, two buttons, accept and deny. One to accept something and the other one to deny.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*A unique field identifier* *Two fields can't have the same name in the exact task definition.* ***The Name must be unique****.*	*true*
transition	*If present, a process instance token will follow the indicated transition.*	*false*
confirmation	*The button will show as a popup with an operation confirmation message. The message text used is the confirmation value.*	*false*
validate	*The default value is "true".*	*false*
style	*Icon style applied to the button. Possible values:* • *yes* • *no* • *add* • *delete* • *download* • *downloadZip* • *home* • *view* • *change* • *compact* • *clean* • *search* • *save*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	• *comment*	
color	*Color applied to the button. Possible values:* • *success* • *primary* • *secondary* • *danger* • *warning* • *info* • *light* • *dark*	*false*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.7//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.7.dtd">
<workflow-forms>
  <workflow-form task="evaluate price">
    <button name="deny" label="Deny" transition="deny"/>
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow Download field

This workflow is used to download a document.

Attribute	Description	Required
label	*The text shown as a label in user interface.*	*true*
name	*Unique name identifier* *Two fields can't have the same name in the same task definion.* ***Name must be unique***.	*true*
width	*The width of the HTML element.* *In the case of KCenter UI must use width values based in %, for example:* • *width="50%" will works* • *width="100px" will be ignored in the KCenter UI.*	*false*
data	*When present,it is an identifier used to load data.* *For example to dynamically load the uuid or path of the node to download.*	*false*

Child elements:

- *Node. For more information see Workflow Node element.*
- *Validator. For more information see Workflow Validator element.*

Basic download example

- *Download button with node element.*
- ***uuid** is a valid document uuid.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                               "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<workflow-forms>
  <workflow-form task="download test">
    <download name="download" label="Download document">
      <node label="download" uuid="afb8ac33-912d-4515-e29b-1c01b693809a" />
    </download>
  </workflow-form>
</workflow-forms>
```

Dynamic download example

- *Download button with node element.*
- ***uuid** is a valid document uuid.*
- ***data** is a valid mapping variable. This **variable** is **used to dynamically set uuid attribute** into node element.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                               "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="download test">
    <download name="download" label="Download document" data="dynamic">
      <node label="download"/>
    </download>
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow Validator element

The goal of the Validatior element is to ensure that the user provides necessary and properly formatted information needed to successfully complete an operation.

Validator can only be used in combination with other fields:

- *Workflow Input field*
- *Workflow Suggestbox field*
- *Workflow Upload field*
- *Workflow Download field*
- *Workflow Print field*
- *Workflow Checkbox field*
- *Workflow Text field*
- *Workflow TextArea field*
- *Workflow Select field*

Available validators:

name	*Description*
req	*Required value.*
alpha	*Alphabetic value.*
alphanum	*Alphanumeric value.*
dec	*Decimal value.*
num	*Numeric value.*
email	*E-mail value.*
url	*URL value.*
maxlen	*Sets the maximum length allowed.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

minlen	*Sets the minimum length required.*
lt	*Less than.*
gt	*Greater than.*
min	*Minimum value.*
max	*Maximum value.*
regexp	*Regular expression.*
plugin	*Create your own form validator.* *More information at: Creating your own Form Validator plugin.*

Parent:

- *Workflow Input field*
- *Workflow Suggestbox field*
- *Workflow Upload field*
- *Workflow Download field*
- *Workflow Print field*
- *Workflow Checkbox field*
- *Workflow TextArea field*
- *Workflow Select field*

Basic example

- *Value required "**req**".*
- *Value must be decimal "**dec**".*
- *The maximum length of the decimal value is 6 digits "**maxlen**".*

```
<input label="Decimal value" name="decimal">
   <validator type="dec"/>
   <validator type="req"/>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    <validator type="maxlen" parameter="6"/>
</input>
```

Regular expression

If you want a validation which is not included in these prefefined, you can use a regular expression. For example, imagine you want to restrict an input to lowecase ASCII values:

```
<input label="Lowercase value" name="regExp">
    <validator type="regexp" parameter="[a-z]+" />
</input>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow Node element

This option element is only used in combination with:

- *Workflow Download field*

The option value represents a list of distinct values in a Selected field.

Attribute	*Description*	*Required*
label	*The text shown as a label in the user interface.*	*true*
path	*Node path. Usually a document path.*	*false*
uuid	*Node uuid. Usually a document valid uuid.*	*false*

Parent :

- *Workflow Download field*

Basic download example

- *Download button with node element.*
- ***uuid** is a valid document uuid.*

```
<workflow-form task="download test">
  <download name="download" label="Download document">
    <node label="download" uuid="afb8ac33-912d-4515-e29b-1c01b693809a" />
  </download>
</workflow-form>
```

Take it as an example. Normally it wants to assign dynamically a uuid, for it is used data attribute.

```
<download name="download" label="Download document" data="dynamic" />
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow Upload field

Upload is used to upload a document.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique name identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
data	*When present, it is an identifier used to load data.* ℹ *For example, dynamically set the folderPath or folderUuid.*	*false*
type	*Values:* • *create (the operation will be to create a new document)* • *update (the process will be to update the existing document)* ℹ *When it is not present, the default value is set to create.*	*false*
folderPath	*The folder path where the document will be stored.*	*false*
folderUuid	*The folder uuid is where the document will be stored.*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

documentName	*Forced name of the uploaded document.*	*false*
documentUuid	*The uuid of the document is to be updated.*	*false*
allowedExtensions	*Set the document extensions that are allowed to be uploaded. For example:* `pdf;png`	*false*
multiple	*When true, it allows the upload of several files. By default, the value is "**false**".*	*false*



To upload a new document, you should set the folderPath or folderUuid, optionally documentName, if we want to force the document name.

To update an existing document, you should set document Uuid and put type to edit.

Child elements:

- *Validator. For more information, see the Workflow Validator element.*

Basic upload example

- *Upload a new document to a folder.*
- ***folderUuid** is a valid folder uuid. You can also use **folderPath** and set a path, which is more human-readable.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="upload test">
    <upload name="upload" label="Upload document" folderUuid="fca2d85e-0e01-418d-b699-c0dd0f8190da"
  </workflow-form>
</workflow-forms>
```

Document updated

- *Create a new version of a given document.*
- ***documentUuid** is the UUID of the document to be replaced.*
- *Note the **type** parameter which is set to update.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="upload test">
    <upload name="upload" label="Upload document" type="update" documentUuid="e6a06309-c1cf-42be-98fd-7d7
```

```
  </workflow-form>
</workflow-forms>
```

Upload an example with dynamic parameters.

- *Upload a document into a non-predefined folder.*
- ***data** is a valid mapping variable used to dynamically set the folderPath attribute.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                               "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="upload test">
    <upload name="upload" label="Upload document" data="updData" />
  </workflow-form>
</workflow-forms>
```

This is the Java code that sets the data and should be executed before the task which displays the form element:

```
import org.jbpm.graph.def.ActionHandler;
import org.jbpm.graph.exe.ExecutionContext;

import com.openkm.bean.form.Upload;

public class UploadAction implements ActionHandler {
    private static final long serialVersionUID = 1L;

    @Override
    public void execute(ExecutionContext ctx) throws Exception {
        Upload upd = new Upload();
        upd.setFolderPath("/okm:root/dstFld");
        ctx.setVariable("updData", upd);

        // Go to next node
        ctx.getToken().signal();
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow Checkbox field

Checkbox

Best used to represent boolean data:

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
value	*Specifies a default value for this component.* ⚠ *The allowed values are **true** and **false**.*	*false*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
description	*General description.*	*false*
readonly	*Set field will not be able to be modified in the user interface. By default, the value is "**false**".*	*false*
data	*When present, it is an identifier used to load data.* ℹ *For example, to dynamically load the true or false value of the checkbox.*	

Child elements:

- *Validator. For more information, see Workflow Validator element.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                               "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="checkbox test">
    <checkbox label="Checkbox label" name="checkbox" />
  </workflow-form>
</workflow-forms>
```

Workflow Input field

In most cases is used to enter free-form text, although it can be used to save HTTP links, date data, or refer to the application folder (as an internal link to this folder).

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
type	*Type value. Allowed types:* • *text.* • *date.* • *link.* • *folder.* • *password.*	*true*
value	*Can specify a default value for this component.* *There's a particular case for date types. Data is stored in the format **"yyyyMMddHHmmss"; take** care of it when you're using OpenKM API.*	*false*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
description	*General description.*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

data	*When present, it is an identifier used to load data.* *For example, to dynamically load the Input value.*	
readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*
timeFormat	*When the input type is the date, we can specify if we want to be able to see and modify hours and minutes. Allowed values:* • *hm* • *h* • *none*	*false*

Child elements:

- *Validator. For more information, see the Workflow Validator element.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="input test">
    <input label="Input label" name="input" />
  </workflow-form>
</workflow-forms>
```

Workflow Select field

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* ⚠ *Two areas can't have the same name in the exact task definition. The name **must be unique**.*	*true*
type	*Type value. Allowed types:* • *simple (single choice allowed).* • *multiple (multiple selections allowed).*	*true*
value	*Can specify a default value for this component.*	*false*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
description	*General description.*	*false*
optionsQuery	*A metadata query or SQL query is used to get the select options.*	*****
className	*A class is used to get the select options.*	*****
data	*When present, it is an identifier used to load data.*	*false*
optionsData	*When present, it is an identifier used to load options.*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*

Child elements:

- *Option. Couples of XML <**option label="label text" value="unique identifier value**"/> values or <**option label="label text" value="unique identifier value" parentValue="parent unique identifier value**"/> in the case is set some hierarchical relationship.*
- *Validator. For more information, see the Workflow Validator element.*

***** There're three options to get option values:*

- *Directly defined in XML.*
- *Metadata table (optionsQuery).*
- *className.*



One of these options must be defined. Otherwise, you will get an error. A combination of several cannot be done; only one can be chosen.

Basic select example

- *Type is equal to **simple**.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.13//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.13.dtd">
<workflow-forms>
  <workflow-form task="select test">
    <select label="select label" name="select" type="simple">
      <option label="one" value="001" />
      <option label="two" value="002" />
      <option label="three" value="003" />
    </select>
  </workflow-form>
</workflow-forms>
```

Allow multiple choice

- *Type is equal to **multiple**.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.13//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.13.dtd">
<workflow-forms>
  <workflow-form task="select test">
    <select label="select label"  name="select" type="multiple" >
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
      <option label="one" value="001" />
      <option label="two" value="002" />
      <option label="three" value="003" />
    </select>
  </workflow-form>
</workflow-forms>
```

Based on SQL query

We can also get these option values using a SQL query. This SQL sentence should be put on the ***optionsQuery*** *attribute, but the* ***table*** *attribute should be empty (otherwise, this query is a metadata query). Note that the first element returned in every row is the option value, and the second is the option label.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="select test">
    <select label="country" name="country" type="simple"
          optionsQuery="select CT_ID, CT_NAME from COUNTRY order by CT_NAME"/>
  </workflow-form>
</workflow-forms>
```

The table used in this sample is defined as follows:

```
create table COUNTRY (
  CT_ID varchar(2),
  CT_NAME varchar(32),
  primary key(CT_ID)
);

insert into COUNTRY (CT_ID, CT_NAME) values ('es', 'Spain');
insert into COUNTRY (CT_ID, CT_NAME) values ('pt', 'Portugal');
insert into COUNTRY (CT_ID, CT_NAME) values ('it', 'Italy');
```

Based on className

To help get a list from external applications or other purposes, you can build your class to retrieve from these systems the option values.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="select test">
    <select label="select label" name="select" type="simple"
          className="com.openkm.plugin.form.values.OptionSelectUserList" />
  </workflow-form>
</workflow-forms>
```

For more information, read: Creating your own Option Select Values plugin.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Hierarchical relationship

- *The field option value of "**childrenSelect**" has a parent relation with "**parentSelect.**" It's defined by **parentElement="parentSelect."***
- *The option value "**benefits**" of the field "**childrenSelect**" has a parent value "**directives**" from the area "**parentSelect**".*

Tree relationships:

- *directives*
  - *benefits*
  - *eligibility*
  - *services*
- *handbook*
  - *basic*
  - *normal*
  - *advanced*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.13//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.13.dtd">
<workflow-forms>
  <workflow-form task="select test">
    <select label="Parent" name="parentSelect" type="simple">
      <option value="directives" label="Directives" />
      <option value="handbook" label="Handbook Paragraphs" />
    </select>
    <select label="Children" name="childrenSelect" type="simple" parentElement="parentSelect">
      <option value="benefits" label="Benefits" parentValue="directives" />
      <option value="elegibility" label="Eligibility" parentValue="directives" />
      <option value="services" label="Services" parentValue="directives" />
      <option value="basic" label="Basic" parentValue="handbook" />
      <option value="normal" label="Advanced" parentValue="handbook" />
      <option value="advanced" label="Normal" parentValue="handbook" />
    </select>
  </workflow-form>
</workflow-forms>
```

Hierarchical relationship with SQL

We use a couple of tables to get the information in this case.

```
create table LOCALIDAD (
  LOC_ID varchar(8),
  LOC_NAME varchar(32),
  primary key (LOC_ID)
);

create table UBICACION (
  UBI_ID varchar(8),
  UBI_LOC varchar(8),
  UBI_NAME varchar(32),
```

```
  primary key (UBI_ID)
);

insert into LOCALIDAD (LOC_ID, LOC_NAME) values ('l0', 'Localidad 0');
insert into LOCALIDAD (LOC_ID, LOC_NAME) values ('l1', 'Localidad 1');
insert into LOCALIDAD (LOC_ID, LOC_NAME) values ('l2', 'Localidad 2');


insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u0', 'l0', 'Ubicación 0');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u1', 'l0', 'Ubicación 1');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u2', 'l0', 'Ubicación 2');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u3', 'l1', 'Ubicación 4');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u4', 'l1', 'Ubicación 5');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u5', 'l1', 'Ubicación 6');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u6', 'l2', 'Ubicación 7');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u7', 'l2', 'Ubicación 8');
insert into UBICACION (UBI_ID, UBI_LOC, UBI_NAME) values ('u8', 'l2', 'Ubicación 9');
```

This is the form definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.13//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.13.dtd">
<workflow-forms>
  <workflow-form task="task">
    <!-- Parent child -->
    <select name="localidad" label="Localidad"
          optionsQuery="select LOC_ID, LOC_NAME from LOCALIDAD" />
    <select name="ubicacion" label="Ubicacion" parentElement="localidad"
          optionsQuery="select UBI_ID, UBI_NAME, UBI_LOC from UBICACION" />
  </workflow-form>
</workflow-forms>
```

Workflow Separator field

Draw a separator between fields.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
description	*General description.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="separator test">
    <input label="Input label" name="input1" />
    <separator label="Separator label" name="separator" />
    <input label="Input label" name="input2" />
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow Suggestbox field

This is an implementation of the typical AJAX suggest box component:

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
value	*Can specify a default value for this component.*	*false*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
description	*General description.*	*false*
filterQuery	*A Hibernate query to filter the data. Use {0} to place the user input.*	*
valueQuery	*A Hibernate query to obtain the value of an identifier. Use {0} to place the identifier.*	*
className	*A class is used to get the selected options.*	*
filterMinLen	*The minimum chars in input to begin to filter results.*	*true*
dialogTitle	*The title of the HTML dialog.*	*true*
data	*When present, it is an identifier used to load data.*	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*

Child elements:

- *Validator. For more information, see the Workflow Validator element.*

** There're three options to get suggested box values (using **one of them is mandatory**):*

- *filterQuery + valueQuery*
- *className*



No other combination of them is allowed.

Based on filterQuery and valueQuery (without table attribute)

*Now **filterQuery** and **valueQuery** could be regular SQL queries.*



You should only use {0} as the parameter the user enters to filter the values from the table.

Create COUNTRY table:

```
create table COUNTRY (
  CT_ID varchar(2),
  CT_NAME varchar(32),
  primary key(CT_ID)
);

insert into COUNTRY (CT_ID, CT_NAME) values ('es', 'Spain');
insert into COUNTRY (CT_ID, CT_NAME) values ('pt', 'Portugal');
insert into COUNTRY (CT_ID, CT_NAME) values ('it', 'Italy');
```

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.5//EN"
                          "http://www.openkm.com/dtd/property-groups-3.5.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
  <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
    filterQuery="select ct_id, ct_name from country where ct_name like '%{0}%' order by ct_name"
    valueQuery="select ct_id, ct_name from country where ct_id='{0}'" />
  </property-group>
</property-groups>
```

Based on className

To help get a list from external applications or other purposes, your class can be built to retrieve the option values from those systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.5//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.5.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
  <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
    className="com.openkm.plugin.form.values.SuggestBoxUserList" />
  </property-group>
</property-groups>
```

For more information, read Creating your own Suggestbox plugin

Workflow Text field

Print a text that can contain HTML tags.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.* *From version 7.1.41 the label can render HTML. In the case of HTML value the string must be encoded, for example:* `<font color="red">red text</font>` *must be converted to* `<font color="red">red text</font>` *You can use an online XML entity encoder like* *https://coderstoolbox.net/string/#!encoding=xml&action=encode&charset=us_ascii* *If set the data value from the workflow transition, then must not convert the value.*	*true*
name	*Unique field identifier* *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
data	*When present, it is an identifier used to load data.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                                "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="text test">
    <text label="text label" name="consulting_text" width="100px"/>
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow TextArea field

Print a text that can contain HTML tags.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true.*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition, and the name **must be unique**.*	*true.*
type	*Type value. Allowed types:* • *text.* • *html.*	
value	*The default value for the component.*	*false.*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will works* • *width="100px" will be ignored in the KCenter UI.*	*false.*
height	*The height of the HTML element.*	*false.*
description	*General description.*	*false.*
data	*When present, it is an identifier used to load data.*	*false.*
readonly	*Set field will not be able to be modified in the user interface. By default, the value is "**false**".*	*false.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Child elements:

- *Validator. For more information, see the Workflow Validator element.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                               "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="textarea test">
    <textarea label="textarea label" name="textarea" width="200px"/>
  </workflow-form>
</workflow-forms>
```

Workflow Option element

The option element is only used in combination with Workflow Select field. The option value represents a list of distinct values to select.

Attribute	*Description*	*Required*
label	*The text shown as label in the user interface.*	*true*
value	*Unique value identifier.* *Two options can't have the same value. **Value must be unique**.*	*true.*
parentValue	*When present, it specifies the option parent.* *The parent must be present in other select.*	*false*
selected	*When present, it specifies that an option should be pre-selected when the form is rendered.*	*false*
cssClass	*CSS class.* *More information about creating your own styles at: CSS.*	*false*

Parent:

- *Select. For more information, Please see: Workflow Select field.*

Basic select with selected option

- *attribute selected=true, when present, it specifies that an option should be pre-selected when the page loads.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="textarea test">
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    <select label="select label"  name="test" type="simple">
      <option label="one" value="001" />
      <option label="two" value="002" selected="true"/>
      <option label="three" value="003" />
    </select>
  </workflow-form>
</workflow-forms>
```

Hierarchical relationship

- *The field option value of "**children**" have a parent relation with "**parent**". It's defined by **parentElement="parent"**.*
- *The option value "**benefits**" of the field "**children**" has a parent value "**directives**" from field "**parent**".*

Tree relationships:

- *directives*
    - *benefits*
    - *eligibility*
    - *services*
- *handbook*
    - *basic*
    - *normal*
    - *advanced*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="textarea test">
    <select label="Provincia" name="parent" type="simple">
      <option value="directives" label="Directives" />
      <option value="handbook" label="Handbook Paragraphs" />
    </select>
    <select label="Provincia" name="children" type="simple" parentElement="parent">
      <option value="benefits" label="Benefits" parentValue="directives" />
      <option value="elegibility" label="Eligibility" parentValue="directives"/>
      <option value="services" label="Services" parentValue="directives"/>
      <option value="basic" label="Basic" parentValue="handbook" />
      <option value="normal" label="Advanced" parentValue="handbook"/>
      <option value="advanced" label="Normal" parentValue="handbook"/>
    </select>
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow VerticalPanel field

A vertical layout that allows you to sort components inside.

Attribute	*Description*	*Required*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="verticalpanel test">
    <vpanel name="vertical1" />
      <input label="Input label" name="input1" />
    </vpanel>
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Workflow HorizontalPanel field

A horizontal layout that allows you to sort components inside.

Attribute	*Description*	*Required*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.6//EN"
                        "http://www.openkm.com/dtd/workflow-forms-2.6.dtd">
<workflow-forms>
  <workflow-form task="horizontalpanel test">
    <hpanel name="horizontal1" />
      <input label="Input label" name="input1" />
    </hpanel>
  </workflow-form>
</workflow-forms>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

User guide



You can use the Quick start guide to learn the basics quickly and to find detailed info on the user sections.

OpenKM front is the Back office view of the content repository. The features are organized into several sections:

- *The **Menu bar** at the top is available for all the features organized by type.*
- *The **Quick search box** is at the top right.*
- *The **Toolbar** where are the most commonly used features.*
- *The **Desktop** tab enables you to navigate the repository.*
- *The **Search** tab enables a query of the repository.*
- *The **Dashboard** tab provides a set of dashboards.*
- *The **Administration** tab provides a set of administration tools.*
- *The **Status** panel at the bottom provides general information.*



*Only members with the "**ROLE_ADMIN**" role have access to the **Administration** tab.*

The OpenKM front allows you to manage any content: scanned images, photographs, office documents, and even large video files or CAD files, among others. The built-in workflow feature helps companies automate processes and get complete control of the document cycle of the documents, saving time and money. It also allows the social activities around the content to connect people to other people, information to information, and people to information, helping to manage, more efficiently, the collective intelligence of the company's human resources.

The OpenKM Profiles feature helps administrators to set what is displayed for each user of the application.

OpenKM provides the solution to:

- *Controls the growing volume of information to be managed.*
- *Ends the diversification of information storage sources.*
- *Prevents the duplication of the data.*
- *Obtains valid information through the control of document versions.*
- *Reduces the time and resources used in locating information.*
- *It avoids knowledge leaks or wrongful access by users, therefore complying with the obligations set in the LOPD (Spanish Data Protection Law).*
- *Restricts unauthorized users' access to the information.*

Through the OpenKM system, all relevant and valuable information can be accessed and found in less time to resolve problems

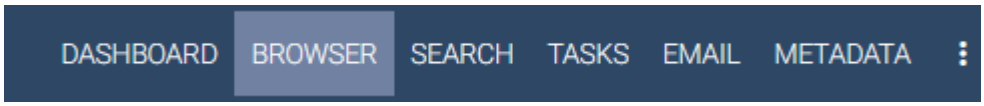
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

and make decisions.

Menu bar

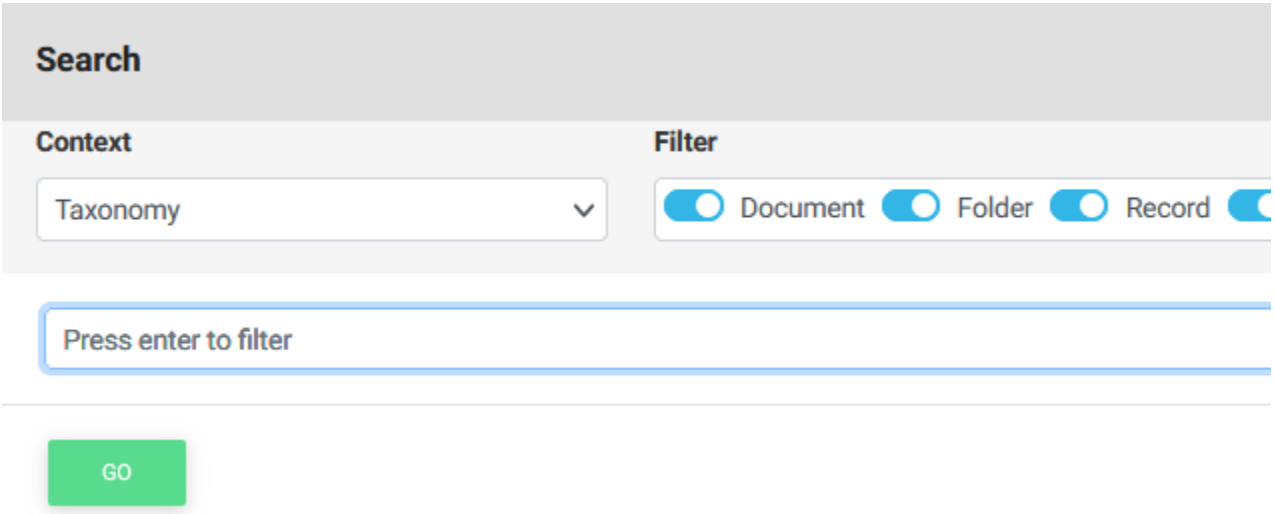
The menu bar at the top has all features organized by type:

- *File.*
- *Edit.*
- *Tools.*
- *Bookmarks.*
- *Templates.*
- *Helps.*



Quick search box

The Quick search box is at the top right corner, providing fast access to the Default search in the Search panel.



The minimum text length for doing a query is three characters.

ToolBar

The toolbar section has the most commonly used features, like downloading documents, creating new folders, and editing or uploading files.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Desktop

Most user activity is done there. Users can work directly on folders, documents, emails, or records.

The desktop is distributed in three sections:

- ***Navigation** panel on the left-hand side.*
- ***Browser** panel at top center.*
- ***Properties** panel at the bottom center.*

Navigation

The navigation on the left-hand side is a stack panel. Each stack contains a tree (folder and records hierarchy) structure from a root that enables quick access to documents and emails.

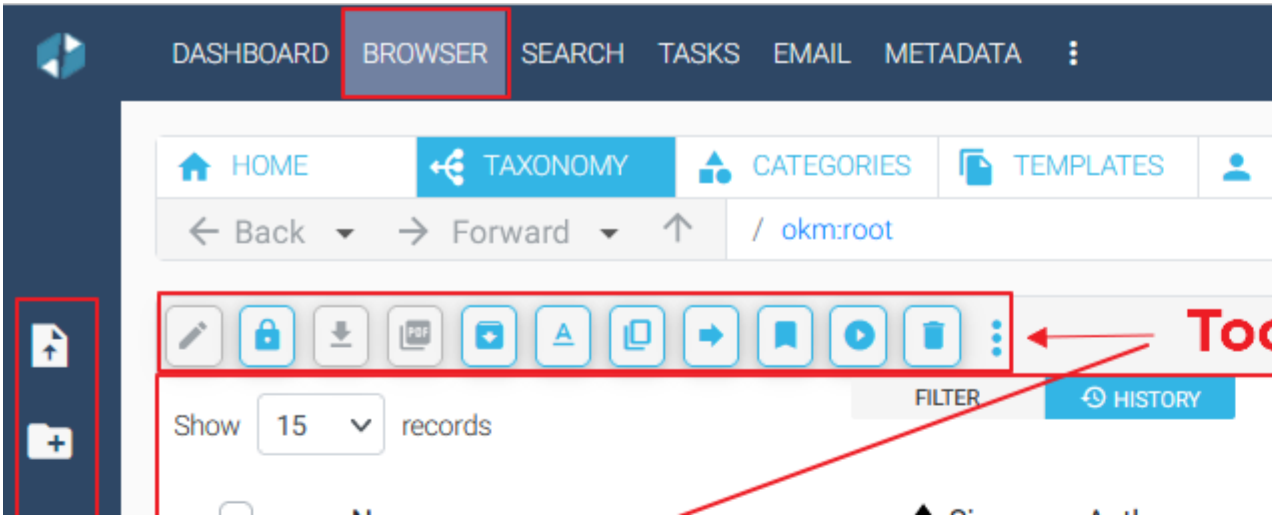
Browser

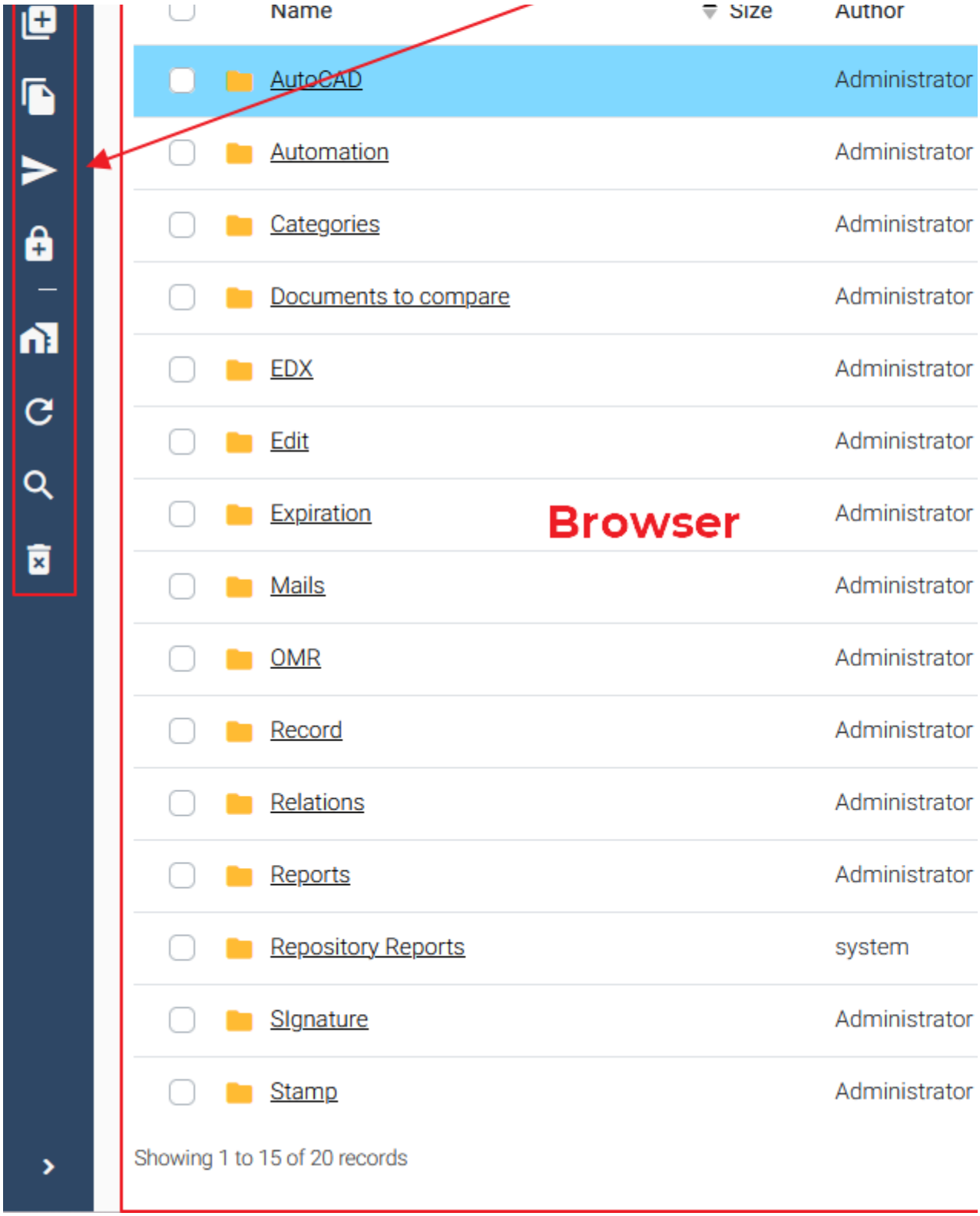
The browser panel at the top center shows a table of nodes (documents, folders, emails, and records) on the selected navigation node.

Properties

The properties panel at the bottom center shows a collection of tabs that contain the grouped properties or features, like a preview associated with the selected node. The selected node can be a navigator node or a browser node.

> *The selected node at the navigator panel determines the content shown in the browser panel. If the last node chosen by the user is at the navigator or browser panel, the properties panel will show the properties of the active node at the navigator or browser.*





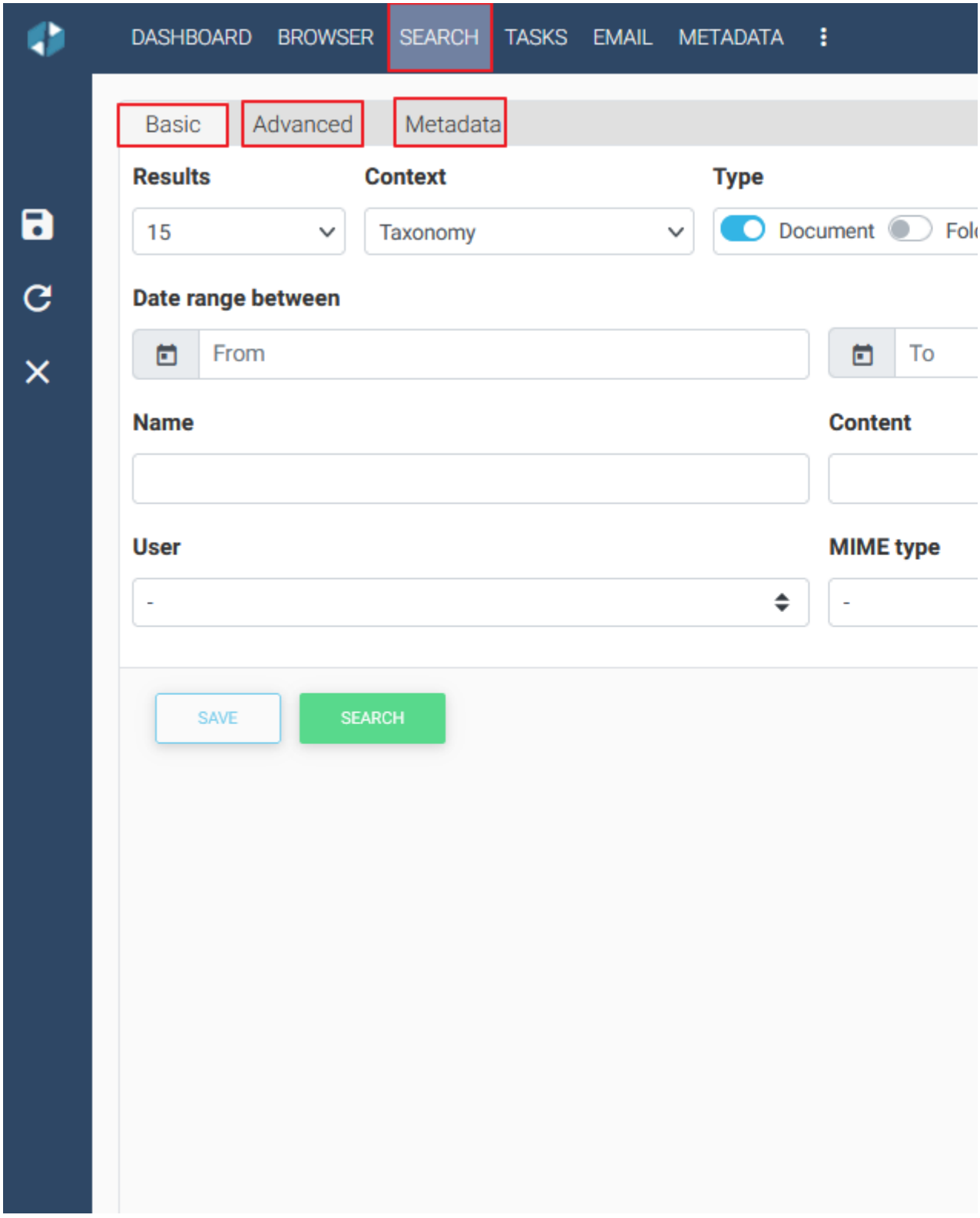
Search

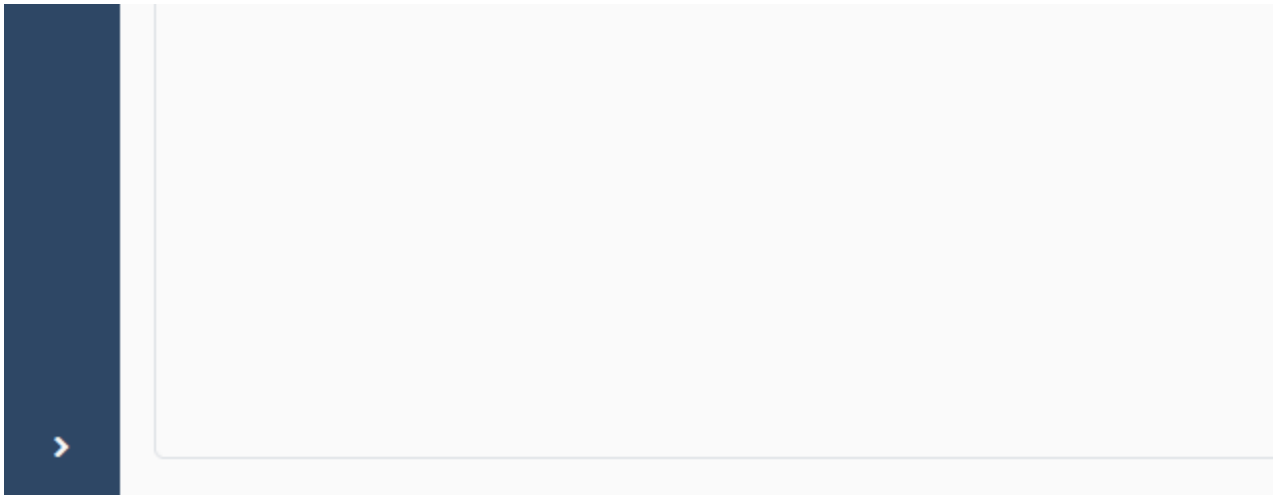
The search tab enables a query of the repository. This tab gives access to different searches to query the content of the

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

application:

- ***The default search*** *is a full-text search, also available from the top right corner of the screen.*
- ***Advanced search*** *enables you to search keywords, metadata, and additional criteria such as location in the application.*



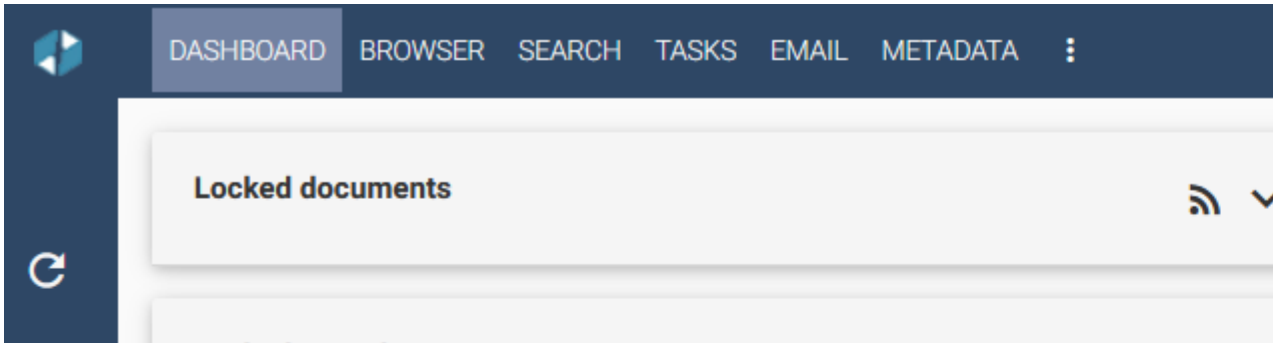


Dashboard

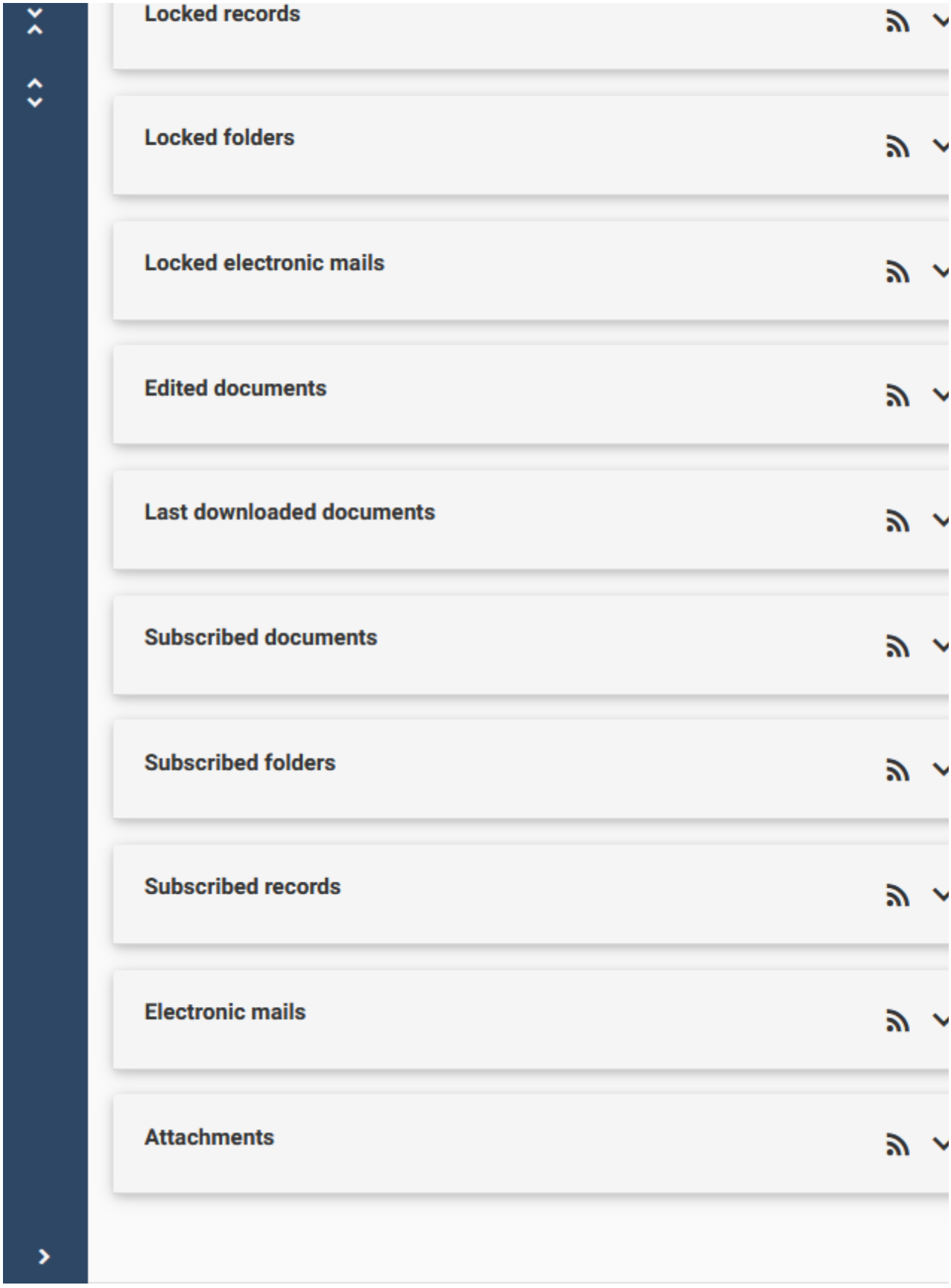
The dashboard panel provides a set of dashboards; each user has a set of personal dashboards that display the information relevant to them and help to have a global view of the application activity.

The dashboard panel :

- ***User view**. Displays the actions that have been carried out in the system*
- ***General view**. Displays a global view of the application activity.*
- ***News**. Displays information based on scheduled user search queries.*
- ***Mail**. It's virtual navigation across all mail repositories.*
- ***Mail account**. It helps users to manage their mail accounts.*
- ***Workflow**. Users work on company business process tasks.*
- ***Keyword map**. Displays a tag cloud based on all repository keywords.*
- ***Tasks**. Helps with creating, administering, assigning, and following basic company tasks.*
- ***Calendar**. Assists in following the status of the primary company task.*
- ***Messaging**. Displays internal OpenKM messaging.*
- ***Forum**. Displays a forum.*
- ***Wiki**. Displays a wiki.*



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Administration

The administration panel is a set of tools that help the administrator to manage the application.

> *More information at Administration guide.*

Status

The status panel provides general information:

- *OpenKM news alert.*
- *The connected user.*
- *User repository size.*
- *Chat.*
- *Alerts.*
- *HTML documents on the edition.*
- *Message panel.*



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Quick start

Before starting a production environment, you should also read these sections:

- *Essential fundamentals.*
- *Plan your data organization.*

With OpenKM, storing and sharing information and collaborating effectively is simple.

This quick start guide gives you an introduction to some of the features of OpenKM:

- *Logging in.*
- *Create a taxonomy.*
- *Add documents.*
- *Edit documents.*
- *Search documents.*

Logging in

You need to log in to OpenKM.

- *Enter the following on the login page:*
  - ***Username**: your username.*
  - ***Password**: your password.*
- *Choose your **language**.*
- *Click on the **Login** button.*

Ask your administrator for your login URL. By default, the OpenKM server URL is http://127.0.0.1:8080/openkm

The default administrator credentials are:

- ***Username**: okmAdmin*
- ***Password**: admin*

*Do not forget to **change the administrator password**.*

Create a taxonomy

The dictionary says:

> *"Taxonomy (from Greek taxis meaning arrangement or division and nomos meaning law) is the*

> *science of classification according to a pre-determined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval. In theory, the development of a good taxonomy takes into account the importance of separating elements of a group (taxon) into subgroups (taxa) that are mutually exclusive, unambiguous, and taken together, include all possibilities."*
>
> *A **taxonomy** is a way to group things. Like a controlled vocabulary focuses on concepts, i n practice, a good taxonomy should be simple, easy to remember, and easy to use.*

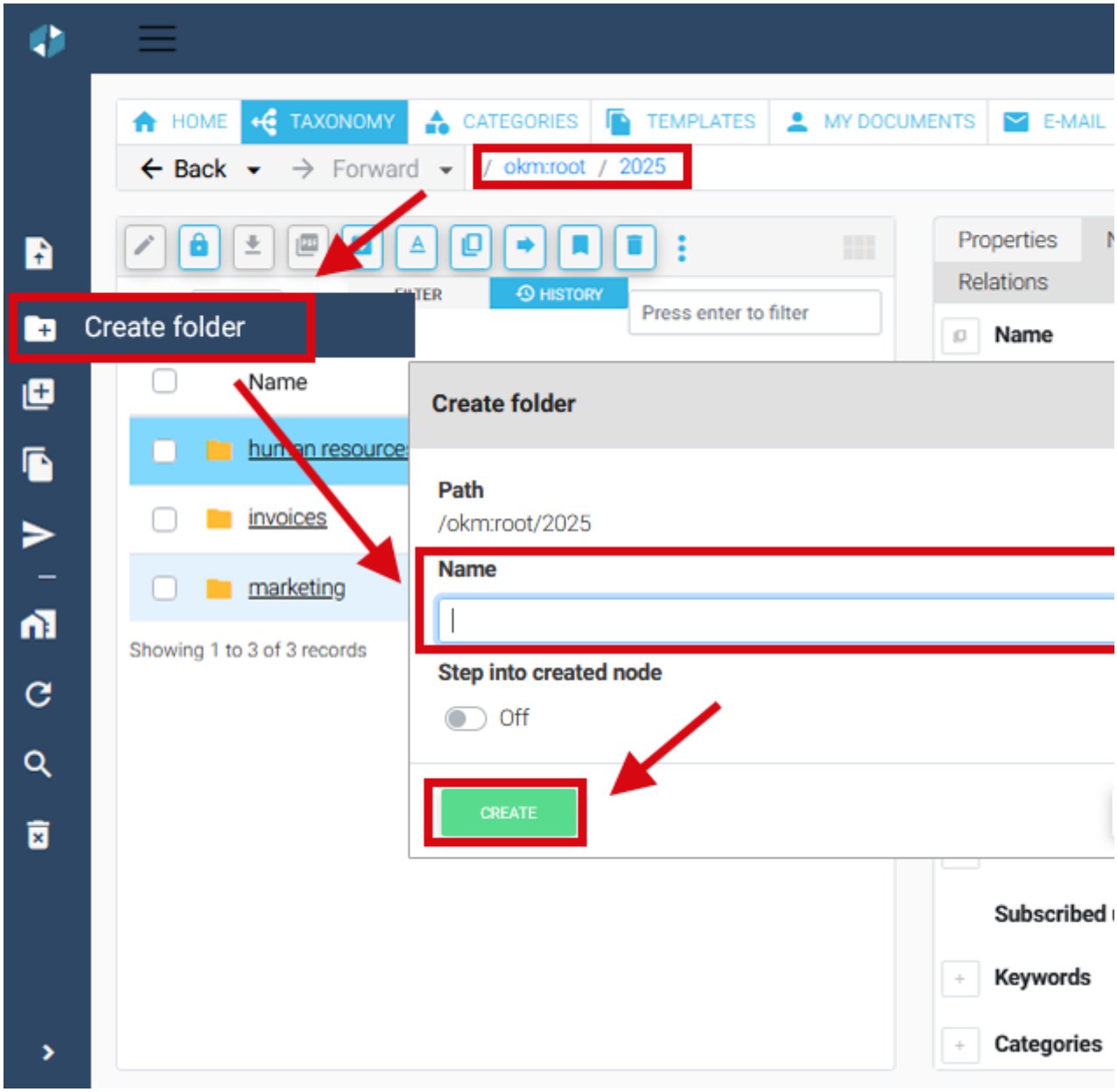
The first step should be analyzing the company organization and defining a folder hierarchy to store the organized documents.

Take this one as a sample:

/okm:root/	***2015/***	***invoices/***
		human resources/
		marketing/
	2016/	***invoices/***
		human resources/
		marketing/

To create a folder:

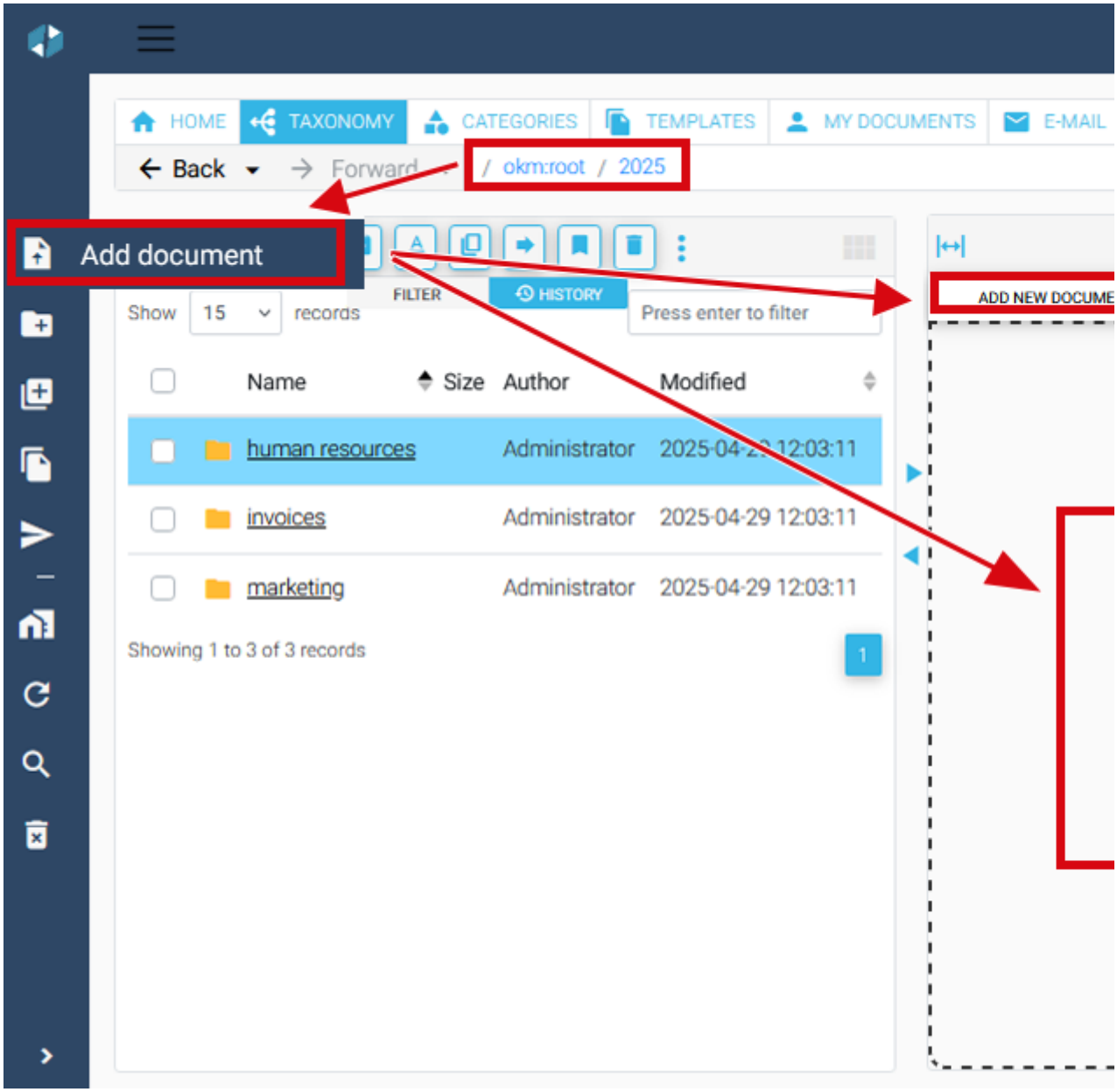
- ***Select a node** from the taxonomy where to create the folder.*
- *Click on the **folder icon** in the toolbar.*
- *Set the **name**.*
- *Click on **Create** button.*



We suggest spending some time thinking about and designing your taxonomy. We suggest less than 200 elements per folder for the best UI performance.

Add documents

- ***Select a node*** *from the taxonomy where to add the document.*
- *Click the* ***add document*** *icon from the toolbar.*
- ***Select the document*** *from your computer or* ***drag & drop****.*



Edit documents

*The document edition is based on the "**check-out / check-in**" model.*

*"**Check-out**" and "**check-in**" are standard terms in many document management systems, including OpenKM. Their purpose is to prevent conflicts in an environment where multiple people might want to simultaneously edit the same piece of content in OpenKM, a document.*

*The term "**check-out**" describes the process of getting a version of a document. By checking out a document, a user can prevent others from editing that content.*

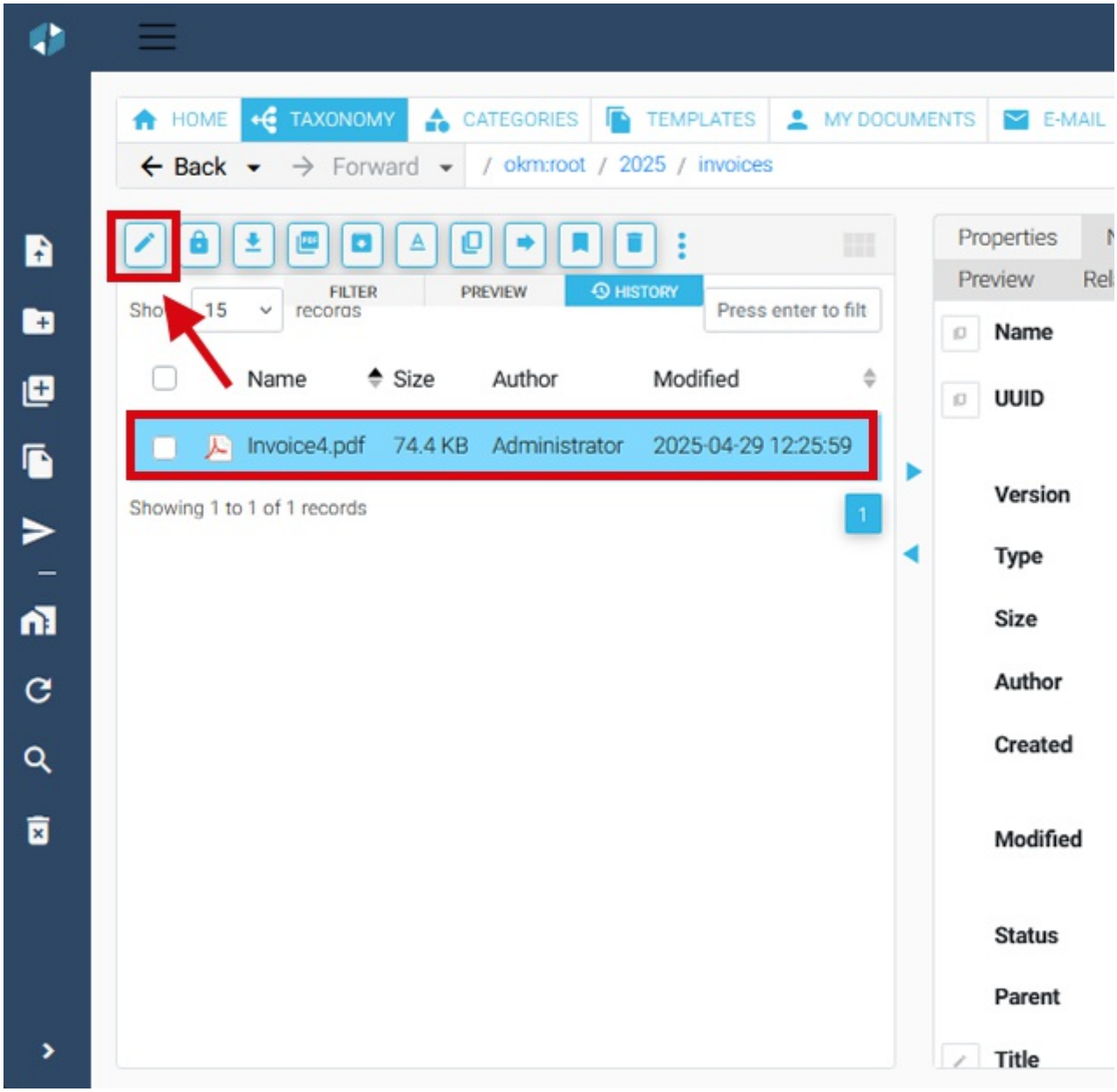
*The term "**check-in**" describes updating a document to replace the previous version. By checking in the item, the user can allow others to edit the document without worrying about overriding changes that others have made.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

When a document is checked out to a user, that user can work on that document, and other users cannot. Other users must wait for the user to check the document before they can edit. While the document is checked out, other users can see only the last version that was checked in; they can't see the changes the current user has made to the document while it's been checked out.

First must check out the document:

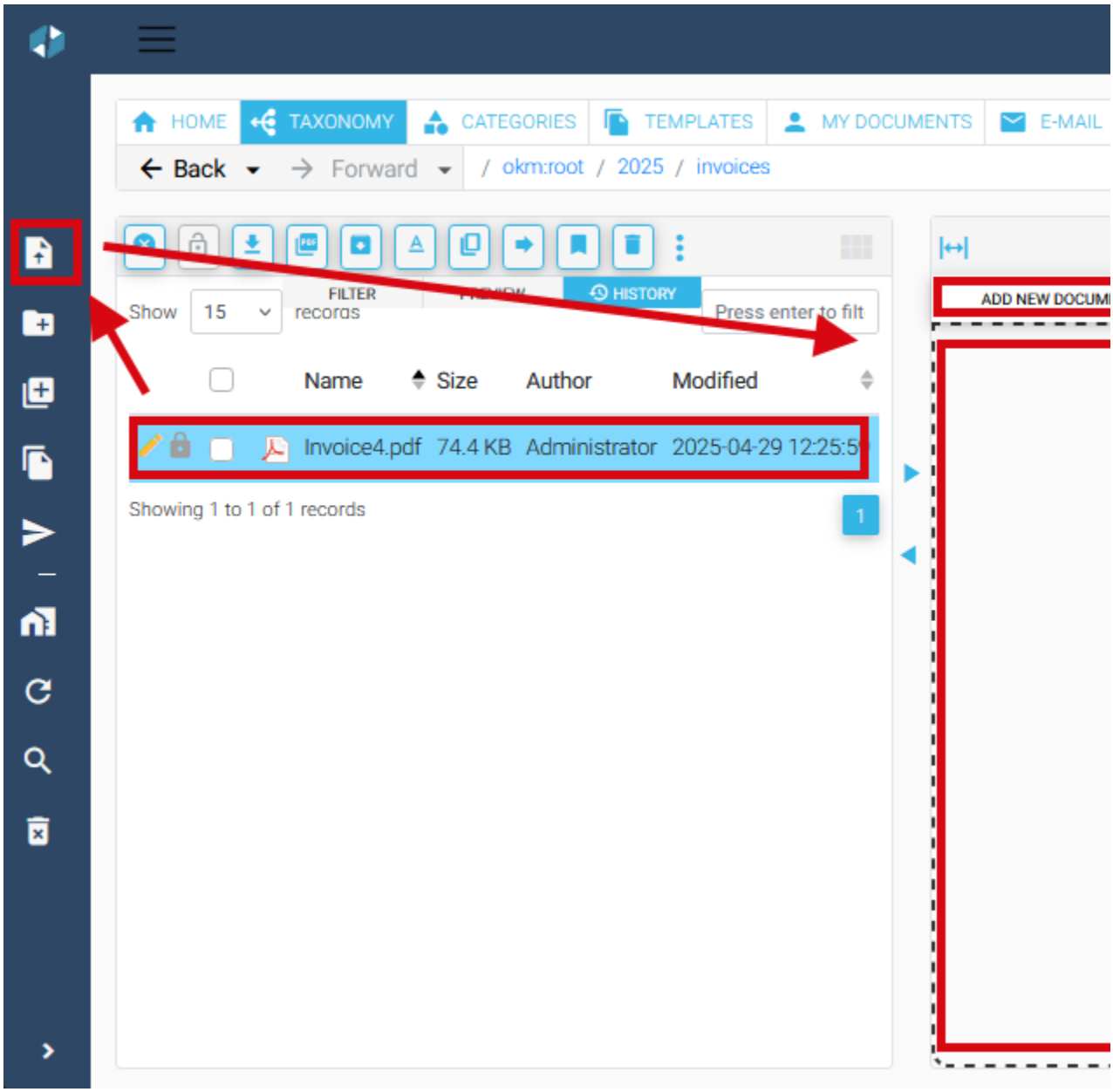
- ***Select a document***.
- *Click on the* ***Edit icon***.



Finally, check the document to upload the changes:

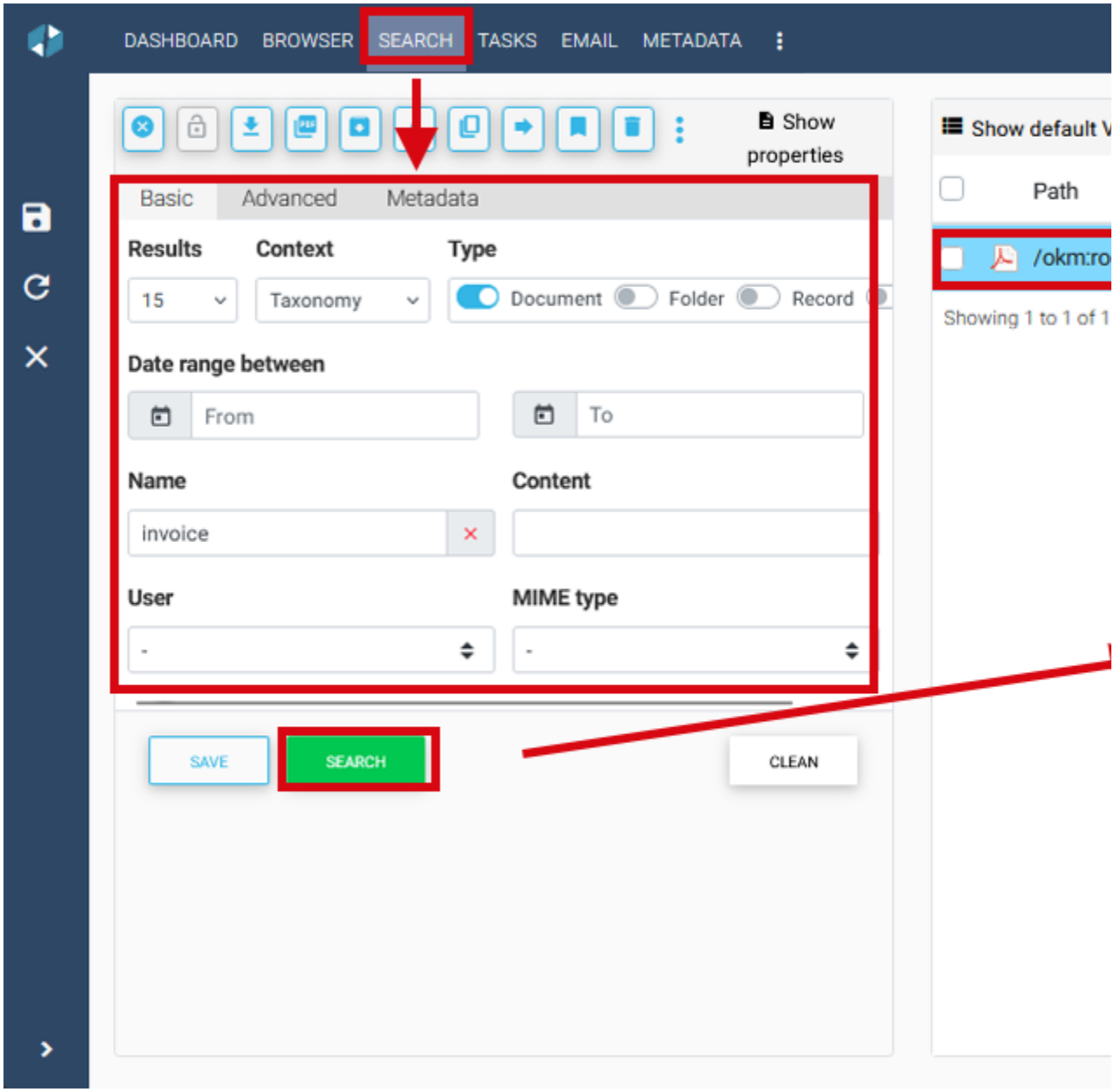
- *Select the* ***edited document***.

- *Click on the **Add document icon**.*
- ***Select the document** from your computer or drag & drop.*



Search documents

- *Go to **search view**.*
- ***Write the term** for the query.*
- *Click on the **Search** button.*


DASHBOARD
BROWSER
SEARCH
TASKS
EMAIL
METADATA
Show properties
Basic
Advanced
Metadata
Results
Context
Type
15
Taxonomy
Document
Folder
Record
Date range between
From
To
Name
Content
invoice
User
MIME type
SAVE
SEARCH
CLEAN
Show default V
Path
/okm:ro
Showing 1 to 1 of 1

Essential fundamentals

Basic concepts

OpenKM has built-in with several types of nodes:

- *Folder.*
- *Document.*
- *Mail.*
- *Record.*

*A node type **folder** is a **storage unit or a container**. As a document, it also has a name but is not a fil, but it can contain documents, records and other folders into.*

*A node type **document** is a **file**. A document serves to establish one or several facts, and can be relied upon as a proof thereof.*

*A node type **mail**, most commonly referred to as **email** or **e-mail**, is a method of exchanging digital messages from an author to one or more recipients. A mail can contain documents (in mail context are named attachments).*

*In essence there are basically two kinds of nodes of type **record**:*

- *Non-digital records.*
- *Digital records.*

*A **non-digital record** is only a set of metadata groups.*

*A **digital record** can contains documents, mails, folders and other record into.*

> ***Non-digital record sample:***
>
> *When OpenKM is used to store the location of papers in the real world. Each record only is a set of metadata fields that helps user to locate papers and there's not binary information associated (like associated scanned document, etc.).*
>
> ***Digital record sample:***
>
> *OpenKM is used to store business records, where each record represents the result of some business process transaction. The digital record then, is used to store documents, mails associated with that process.*

Understanding taxonomy, keywords, categories and metadata groups.

Spending time on nodes categorization is a good practice. A good categorization will help users quickly finding fast of the desired information, using less time, saving money and being more efficient in your business process.

OpenKM helps you on it providing basically four ways for recategorizing your repository:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Taxonomy.*
- *Keywords.*
- *Categories.*
- *Metadata groups.*



You can use one of these features to categorize your repository or a combination of all.

Taxonomy

The dictionary says:

"Taxonomy (from Greek taxis meaning arrangement or division and nomos meaning law) is the science of classification according to a pre-determined system, with the resulting catalogue used to provide a conceptual framework for discussion, analysis, or information retrieval. In theory, the development of a good taxonomy takes into account the importance of separating elements of a group (taxon) into subgroups (taxa) that are mutually exclusive, unambiguous, and taken together, include all possibilities."

*Basically, a **taxonomy** is a way to group things together. Like a controlled vocabulary focuses on concepts, i n practice, a good taxonomy should be simple, easy to remember, and easy to use.*



For a best performance at UI we suggest - when possible - a taxonomy where each parent has 200 children or less.

Each folder in a taxonomy represents a "concept" and lists a couple of nodes (folders, documents, mails or records) associated within.

Keywords

A keyword, in the context of document management is a particular word or phrase that describes the contents. Setting keywords is the most simple and basic system to categorizing the nodes in your repository but at same time is the less powerfull. For large repositories we encourage using metadata groups.

Limitations of keywords usage:

- *When you have thousands of keywords searching by keywords can be inefficient (not able to refine search and you get a lot of results with same keywords, the users not followed a restricted way to catalog). A tag cloud diagram in this cases is usually an unusable graphical representation of keywords.*
- *It's difficult for the users setting keywords because they do not know all the keyword dictionary of the company.*
- *Users write keywords in similar way but not always the same (for example users use the words "building", "construction" and "edifice" indistinctly).*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

You can get best results when:

- *You limit the number of users who are able to set keywords.*
- *You use a controlled keywords dictionary (more information at http://www.w3.org/2004/02/skos/).*
- *You share a complete company keywords dictionary to your users. That helps understanding how is mapped the knowledge in your organization and helps users finding it.*

Categories

In OpenKM, the categories are a virtual taxonomy that helps on navigation of taxonomy nodes across other concepts. Each category node represents a "concept" and list a couple of nodes (folders, documents, mails or records) associated within.

We suggest using categories rather than keywords.

Metadata

Metadata groups are data that describes other data providing information about a certain nodes content. Metadata groups helps on creating controlled classification and a powerful way to locate nodes and refine query searchs.

With metadata groups you can simulate the keywords and categories features. We encourage expend some time investigating metadata feature and we suggest use it rather keywords and categories.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Plan your data organization

Before and intensive use of your application you should plan your data organization in the repository.

> *Take the information that follows only as a general idea about the basic steps you should consider before an intensive use of your application.*

A good analysis of your organization and the understanding of the application can help you finding the best way to solve your problems. OpenKM is not a closed application, something more than a couple of features focused on document management. OpenKM is a flexible application and can be used on several ways. It's good idea to expend some time understanding your organization, your business logic and process involved and how the OpenKM features can work together in your day-by-day helping your organization making it more efficient and saving money.

Some questions you should do yourself about your organization:

- *Where is your information and documents ?*
- *What kind of documents do you have ?*
- *Are your documents on electronic format or not ?*
- *What format of document do you have ? PDF, MS Office, LibreOffice, etc.*
- *How are you actually finding the document ? How would you like to find it, which terms you consider relevant ?*
- *Who can access the information and on which way ?*
- *Do you need a work flow process ?*

Which are the objective of the previous questions ?

- *Identify on a table all the distinct types of information you have (invoices, budgets, legal papers, etc.).*
- *Identify the format of the documents (PDF, DOC, etc.)*
- *Identify non digitalized information that should be processed (scanned). The digitalization process can be a basic process.*
- *Identify for each type of information (document type) what information is relevant (metadata). For example invoices usually have invoice number, range of date, customer among others. The metadata will help users to retrieve fast and efficiently the information.*
- *Identify the security that must be applied for each document type. Who can read, write or delete information. Also your security needs will help you on setting the level of the application log you need. OpenKM provide you a full audit on any user action that can be configured in a more or less restrictive way.*
- *Identify workflows and people involved in workflows.*

> *The goal is to create a plan with calendar and milestones.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Which are the goals you are looking for the implantation of the application ? Usually could be:

- *Set some order in my organization. Application "force" the users to work in some way, that benefits your company because helps you "setting rules" that all users must follow.*
- *Consolidate the intellectual capital. Consolidating all your company knowledge on a single point you have all the company knowledge available to be share among all your users and setting rules on it.*
- *Help on accomplish with legal regulation.*

Try to set and specification on a single document answering this questions and others. It will help you to plan how to organize the information into.

Good practice

- *Consider OpenKM as a project that affects all your departments and usually your customers.*
- *Choose the leaders.*
- *Do gradual implantation of the project. Can start with some department or group of users or for some business logic process. Think the implantation as a virus in your company. A good leader should demonstrate the benefits of the new and show to the other users. Users usually receipt any news on the company as something negative if they perceive the application as something is helping on their work (efficient search, saving time and solving real problems, etc.) will be the first interested on using it. One of the major goals of OpenKM is the intuitive UI what helps on this process (after two weeks the user feels comfortable using the application).*
- *Think easy, the best solution is usually finding the easiest way to solve problems.*
- *Listen your users and try to get them involved in the project.*
- *Periodically review your plan, try to refine and identify possible problems.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Working with the menu bar

At the menu bar on the top all available features are organized by type. The menu options is formed by all the pull down menus with their different options. If an option within a menu appears greyed-out, this option is not available

The menu bar options are:

- *File.*
- *Edit.*
- *Tools.*
- *Bookmarks.*
- *Templates.*
- *Help.*

The features described below are the default features. In your menu bar options can be shown others not described here if you have OpenKM extensions enabled.

Massive column in the tables, indicates you can check several nodes at the same time in browser panel to apply the same action.

File

Feature	*Description*	*Massive*
Find folder	*Finds folder in the repository.* *The search can be executed only on the context of the selected stack.*	*No.*
Find document	*Finds a document in the repository.* *The search can be executed only on the context of the selected stack.*	*No.*
Find mail	*Finds a mail in the repository.* *The search can be executed only on the context of the selected stack.*	*No.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Find record	*Finds a record in the repository.* *The search can be executed only on the context of the selected stack.*	*No.*
Find similar document	*Finds similar documents in the repository. The feature is based on the search engine heuristic that is based on contents of the selected document.* *The search can be executed only on the context of the selected stack.* *This feature is only enabled when a document in Browser view is selected .*	*No.*
Create folder	*Creates a new folder into the selected node of the navigator panel.*	*No.*
Add document	*Uploads a new document into the selected node of the navigator panel.* *When the option "Import documents from ZIP" is checked you import the contents of a ZIP file respecting the hierarchy of the folder tree. If you have problems with filename encodings, please take a look at Client configuration. You can also upload RAR or JAR MIME-TYPE files.* *When the option "Notify to users" is checked you can send a mail notification to other application users.* *Optionally, based on **Administration** > Profiles configuration or **Administration** > Automation a wizard is shown. Wizard can require to you for setting metadata groups, categories or keywords.*	*No.*
Add record	*Creates a new record into the selected node of the navigator panel.*	*No.*
Download document	*Downloads the selected document in browser panel.*	*No.*
Download document	*Download the selected document in browser panel in PDF format.* *The feature is enabled when is possible convert the document to PDF format.*	*No.*

as PDF		
Send link	*Sends a URL to the selected document in browser panel by mail .*	*Yes.*
Send document attachment	*Sends the selected document in browser panel as an attachment by mail.*	*Yes.*
Generate download link	*Generates a download token of the selected document in browser panel and send the URL.* *For more information take a look at Download tokens.*	*No.*
Forward e-mail	*Forwards the selected mail in a browser panel.*	*No.*
Write mail	*Directly writes a new mail from user interface.*	*No.*
Create from template	*Creates a new document based on an existing one at templates space.*	*No.*
Export to ZIP	*Downloads the selected nodes compress into ZIP file. If you have problems with filename encodings, please take a look at Client configuration.*	*Yes.*
Start workflow	*Start a workflow on the selected node in browser panel.* *You can start a workflow on any node type (document, folder, mail or record).* *Optionally some workflows require you to fill a form as part of starting.* *The available workflows are set in **Administration** > Profiles.*	*No.*

Refresh	*Refresh the user interface.*	*No.*
File uploader	*Bulk upload feature.* *This feature needs to have Java enabled in your browser. More information at Bulk Uploader.* *If you're using a recent version of Goggle Chrome, keep on mind that Java applets won't work. Please take a look at Browser configuration for more info.*	*No.*
Restore	*Restores the selected node from the trash bin.*	*No.*
Purge	*Purges the selected node from the trash bin.*	*No.*
Purge trash	*Cleans the user trash bin.*	*No.*
Exit	*Exits the application.* *Users should always exit the application with this menu option and not only closing the web browser. Anyway after 30 minutes the web session of the user in the server will be automatically terminated.*	*No.*

Example of allowed queries on find folder, document, mail or record.

- *Add an asterisk "*****" as a placeholder for any unknown or wildcard terms, for example, "***2015**", "**2015***", "**20*15**" or "***2015***".*
- *Add an asterisk "**?**" as a placeholder for any unknown or wildcard character, for example the query "**?015**" will return occurrences where first character doesn't matter.*
- *The query "**2015**" is a synonym of "**2015***". On this search view application always add "*****" for you at the end.*

Edit

Feature	Description	Massive
Lock	*Locks the selected document in a browser panel.*	*Yes.*
Unlock	*Unlocks the selected document in a browser panel.* *Only the user who locked the document or an user with administration credentials (force unlock) can unlock the node.*	*Yes.*
Edit	*Edits a document (check-out).*	*Yes.*
Update	*Uploads a new document version (check-in).* *When uploading a new version of a document, you can notify users about the document changes. The application sends the notification by mail.*	*No.*
Cancel edit	*Cancels a document edition.* *Only the user who edited the document or user with administration credentials (force cancel edit) can cancel edition of the node.*	*Yes.*
Delete	*Deletes selected node.* *When a node has been deleted, you can restore it from the trash bin. Each user has its own trash.*	*Yes.*
Copy	*Copies the selected node.*	*Yes.*
Move	*Moves the selected node.*	*Yes.*
Rename	*Renames the selected node.* *You can cancel renaming using the ESC key.*	*No.*

Add note	*Adds a note to the selected node.*	*Yes.*
Add category	*Adds a category to the selected node.* *Categories are a virtual taxonomy space that helps on the navigation of taxonomy nodes across other concepts*	*Yes.*
Add keyword	*Adds a keyword to the selected node.* *When thesaurus is enabled you can choose terms from there (SKOS).*	*Yes.*
Merge PDF	*Merges two or more PDF files. The* ***result of the process is a new PDF documen*** *t.* *Musts chooses two or more PDF files to be merged.* *You can change the process order of the PDF documents.* *You can set the name of the new PDF document result of merging the selected PDF files.*	*No.*
Add metadata group	*Adds a metadata group to the selected node.*	*Yes.*
Update metadata group	*Updates a metadata group fields values of the selected node.*	*Yes.*
Remove metadata group	*Deletes the metadata group of the selected node.*	*Yes.*
Add subscription	*Subscribes to the selected node.* *You can subscribe on any node type (document, folder, mail or record). In*	*No.*

	case you are subscribed to node of type folder or record, you'll be notified by mail with any change in that node and children hierarchy. The user will receive notification based on the events definition of its own profile (only will receive notifications on events enabled from its own profile). More information about configuring notification events on Profiles.	
Remove subscription	*Un subscribe to the selected node.*	*No.*

*The document edition is based on the "**check-out / check-in**" model.*

*"**Check-out**" and "**check-in**" are common terms in many document management systems, including OpenKM. Their purpose is to prevent conflicts in an environment where multiple people might want to edit the same piece of content - in OpenKM a document - at the same time.*

*The term "**check-out**" describes the process of getting a version of a document. By checking out a document, a user can prevent others from editing that content.*

*The term "**check-in**" describes the process of updating a document to replace the previous version. By checking in the item, the user can allow others to edit the document, without needing to worry about overriding changes that others have made.*

When a document is checked out to a user, that user can work on that document, and other users cannot. Other users must wait for the user to check in the document before they can edit. While the document is checked out, other users usually can see only the last version that was checked in; they can't see the changes that the current user has made on the document while it's been checked out.

Tools

Feature	*Description*	*Massive*
Languages	*Changes the user interface language.*	*No.*
Show administration	*Shows the administration tab on a separate browser tab.* *This option is only available for Administrators users.*	*No.*
User configuration	*Shows the user configuration. You can change your password and mail address.*	*No.*
OCR data capture	*Applies Object Character Recognition on the selected node.*	*No.*

	The data captured is stored in a metadata group. *This option is only available if you have enabled a OCR template. For more information about it take a look at **Administration** > OCR templates.*	
Web page import	*Captures a web page and import as PDF file.* *This option is only available if you have configured the parameter "**system.html2pdf**". More information at Preview parameters.*	*No.*
Convert	*Transforms the selected document to other format.* *Available options:* • *When **Save** checkbox is checked a new document is created in the repository.* • *When **Download** checkbox is checked the converted document is automatically download.*	*Yes.*
DXF preview	*When enabled the autocad extension feature, preview autocad files as DXF.* *This option is only available if you have enabled the **Autocad preview** feature.*	*No.*
PDF preview	*When enabled the autocad extension feature, preview autocad files as PDF.* *This option is only available if you have enabled the **Autocad preview** feature.*	*No.*
Add new document encrypted	*Upload a new document and crypt.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
Encrypt	*Encrypts the selected document into the repository.*	*No.*

document	*When a document is encrypted all previous version history will be removed (because previous versions history are not encrypted).* *This option is only available if you have enabled the **Cryptography** feature.*	
Edit encrypted document	*Edits an encrypted document.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
Update encrypted document	*Uploads a new version of an encrypted document.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
Download decrypted document	*Downloads version decrypted of the selected encrypted document.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
Decrypt document	*Decrypt the selected document in the repository.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
Send new message	*Sends new internal message between OpenKM and internal messaging.* *This option is only available if you have enabled the **Messaging** feature.*	*No.*
Sign document	*Applies digital signature to the selected document.* *This option is only available if you have enabled the **Digital signature** feature.*	*No.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Export to Dropbox	*Exports the selected node to Dropbox.* *This option is only available if you have enabled the **Dropbox** feature.*	*Yes.*
Import from Dropbox	*Import data from Dropbox to selected folder at Navigator.* *This option is only available if you have enabled the **Dropbox** feature.*	*No.*
Export to Google Drive	*Exports the selected node to Goggle Drive.* *This option is only available if you have enabled the **Goggle Drive** feature.*	*Yes.*
Import from Google Drive	*Imports data from Goggle Drive to selected folder at Navigator.* *This option is only available if you have enabled the **Goggle Drive** feature.*	*Yes.*
Create new document	*Creates a new document and starts editing with Goggle Drive integration view.* *Goggle Drive integration allows collaborative edition with several users on edition at same time.* *This option is only available if you have enabled the **Goggle Drive** feature.*	*No.*
Edit	*Edits the selected document with Goggle Drive integration view.* *Goggle Drive integration allows collaborative edition with several users on edition at same time.* *This option is only available if you have enabled the **Goggle Drive** feature.*	*No.*
HTML Editor	*You can enable or disable integrated HTML editor.* *This option is only available if you have enabled the **HTML Editor 4***	*No.*

	feature.	
Live Editor	*You can enable or disable the Live Edit feature. When enabled you edit the selected document in their native application (for example MS Office, OppenOffice or LibreOffice).* *This option is only available if you have enabled the **Live Edit** feature.* *This feature needs to have Java enabled in your browser. More information at Live Edit.* *If you're using a recent version of Goggle Chrome, keep on mind that Java applets won't work. Please take a look at Browser configuration for more info.*	*No.*

Bookmark

Feature	*Description*	*Massive*
Home	*Go to home node.*	*No.*
Set default home	*Change default home for selected node.* *When user log in, the application opens in desktop view the home node of the user.* *By default home node is "**/okm:root**".* *You can select nodes in navigation and browser panels from desktop.* *More information at Working with desktop.*	*No.*
Add Bookmark	*Bookmark the selected node.* *You can select nodes in navigation and browser panels from desktop.* *More information at Working with desktop.*	*No.*
Edit	*Show bookmark manager*	*No.*

Bookmark		

From bookmark view you can see the list of nodes added in bookmark by user.

When you click on a bookmark node the application opens the node in desktop view.

Templates

*Template menu dynamically build a menu hierarchy based on "**/okm:templates**" nodes. When you click on any menu option at templates, automatically starts the process of creating a new node (usually a new document based on template document).*

Help

Feature	*Description*
Documentation	*Opens a new browser tab with official openkm documentation URL.*
Support request	*Opens a new browser tab with official community support URL.*
Version notes	*Opens a new browser tab with application changelog URL*
About OpenKM	*Shows a pop-up with information of application.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Working with the toolbar

The toolbar section has the most common features like downloading documents, creating new folders, editing or uploading files, among others. The toolbar is made up of buttons with the main functions that the user will most frequently use. As with the Menu bar, occasionally the buttons will appear greyed out, meaning that they are not available.



The features described below are the default ones. Your toolbar can show others not described here if you have OpenKM extensions enabled.

A massive column in the tables indicates you can check several nodes at the same time in a browser panel to apply the same action.

Feature	*Description*	*Massive*
Find folder	*Find the older in the repository.* *The search can be executed only in the context of the selected stack.*	*No.*
Find document	*Finds a document in the repository.* *The search is executed only in the context of the selected stack.*	*No.*
Find mail	*Find mail in the repository.* *The search is executed only in the context of the selected stack.*	*No.*
Find record	*Find a record in the repository.* *The search can be executed only in the context of the selected stack.*	*No.*
Find	*Finds a similar document in the repository. The feature is based on a search engine heuristic*	*No.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

similar document	*that, based on the contents of the selected document, returns similar documents.* *The search can be executed only in the context of the selected stack.* *This feature is only enabled when a document is selected in a Browser view.*	
Create folder	*Creates a new folder in the selected node of the navigator panel.*	*No.*
Download document	*Downloads the selected document in a browser panel.*	*No.*
Download document as PDF	*Downloads the selected document in a browser panel in PDF format.* *The feature is enabled when it is possible to convert the document to PDF format.*	*No.*
Lock	*Locks the selected document in a browser panel.*	*Yes.*
Unlock	*Unlocks the selected document in a browser panel.*	*Yes.*
Add document	*Uploads a new document into the selected node of the navigator panel.* *When the option "Import documents from ZIP" is checked, you import the contents of a ZIP file, respecting the hierarchy of the folder tree. If you have problems with filename encodings, please take a look at Client configuration.* *When the option "Notify users" is checked, you can send a mail notification to other application users.* *Optionally, based on **Administration** > Profiles configuration or **Administration** > Automation, a wizard is shown. Wizard can require you to set metadata groups, categories, or keywords.*	*No.*
Print	*Prints the selected document in a browser panel.*	*No.*

Edit	*Edits a document (check-out).*	*Yes.*
Update	*Uploads a new document version (check-in).*	*No.*
Delete	*Deletes the selected node.*	*Yes.*
Cancel edit	*Cancels document edition.*	*Yes.*
Add metadata group	*Adds a metadata group to the selected node.*	*Yes*
Delete metadata group	*Deletes the metadata group of the selected node.*	*Yes.*
Add subscription	*Subscribes to the selected node.* *You can subscribe to any node type (document, folder, mail, or record). In case you are subscribed to a particular node of type folder or record, you'll be notified by mail when any change in that node and its children hierarchy occurs. Keep in mind that in order to receive these mail notifications, you have to configure which events you want to be notified in the user profile.* *See Profile Notification tab for more info.*	*No.*
Remove subscription	*Unsubscribes to the selected node.*	*No.*
Start workflow	*Starts a workflow on the selected node in a browser panel.* *You can start a workflow on any node type (document, folder, mail or record).*	*No.*

Refresh	*Refreshes the user interface.*	*No.*
Home	*Goes to the home node.*	*No.*
Previous node	*Goes to the last node visited.*	*No.*
History	*Manages the history of nodes visited.*	*No.*
Next node	*Goes to the next node visited.*	*No.*
Stamp	*Applies a stamp on a document.* *This option is only available if you have enabled the **Stamp** feature.*	*No.*
Menu options	*Displays an extended menu with custom options.* *This option is only available if you have enabled the **Menu options** feature.*	*No.*

Working with desktop

Almost all user's activity is done there. Users can work directly on folders, documents, mails or records.

The desktop is distributed in tree sections:

- ***Navigation** panel at left hand side.*
- ***Browser** panel at top center.*
- ***Properties** panel at bottom center.*

Navigation

The navigation at left hand side is an stack panel. Each stack contains a tree (folder and records hierarchy) structure from root that enables easily access documents and mails.

The navigation panel is divided on several stacks:

- *Taxonomy.*
- *Categories.*
- *Metadata.*
- *Templates.*
- *My documents.*
- *E-mail.*
- *Trash.*

Each stack contains a tree (folder and records hierarchy) to help on organizing the nodes into the repository and help the user to locate fast and efficiently the data.

*The **taxonomy** stack is the space for collaborative work, that means is the space where several users modify and share the company data.*

*The **categories** stack is a virtual taxonomy space that helps on navigation of taxonomy nodes across other concepts.*

> *Categories is a hierarchical tree structure of category and subcategory folders. The idea is these folders and sub folders represents conceptual terms in a tree structure.*

*The **metadata** stack is a virtual taxonomy space that helps on the navigation of taxonomy nodes across the metadata field values.*

> *Metadata view is a hierarchical tree structure of metadata groups. Each possible metadata value is represented as a tree node structure.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Only fields of type Select, SuggestBox and Checkbox are displayed (free text fields like Inputs or Text area are not shown).

You can create your own Stack Metadata Adapter visualization, more information at: Creating your own Stack Metadata Adapter (deprecated).

*The **templates** stack is the space to set templates for all the company. That means is the space where company templates are set to be used by several users for creating new company documents.*

*The **My documents** stack is a private space for each user, that means that only a user is able to get access in this section and is the default where personal data is stored.*

*The **E-mail** stack is a private space for each user, that means only a user is able to access their data in this section and is the default place where are stored the mails of the user.*

*The **Trash** stack is a private space for each user, that means that only a user is able to get access in this section. When a user deletes a node it goes automatically to its own trash. When a user empty its own trash the document is definitively removed from the application.*

Administrators have no restriction for accessing all repository areas, including private user sections like "My Document", "E-mail" or "Trash".

The selected node from navigator panel determines the content shown at browser panel. All the nodes shown at the browser panel have the same parent (selected node on tree).

In case of virtual stacks (categories, metadata and thesaurus) the nodes shown at browser panel do not have the same parent (selected node on tree), really they are shown there because they have a relation (category, metadata or thesaurus value) with the selected node on tree.

*When you do a **right click** with the mouse button on a node at the tree is shown a **contextual menu**. The available menu options in contextual menu depending on selected stack.*

A complete description of all menu options are available at: Working with the menu bar.

*Available menu options at **Taxonomy** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Add document.*

- *Add record.*
- *Add bookmark.*
- *Set default home.*
- *Export to ZIP.*

*Available menu options at **Categories** stack are:*

- *Create folder.*
- *Delete.*
- *Move.*

*Available menu options at **Templates** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Export to ZIP.*

*Available menu options at **My document** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Export to ZIP.*

*Available menu options at **E-mail** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy*

*Available menu options at **Trash** stack are:*

- *Restore.*
- *Purge.*
- *Purge trash.*

Metadata and Thesaurus stack view has no menu options.

Those folders to which the user is subscribed will be displayed in a different color to differentiate them:

- *You are subscribed to a folder **without subfolders** into.*
- *You are subscribed to a folder **with subfolders** into.*
- *You are subscribed to a **read only** folder **without subfolders** into.*
- *You are subscribed to a **read only** folder **with subfolders** into.*

Browser

The browser panel at the top center shows a table of nodes (documents, folders, mails and records) that are in the selected node of the navigation panel.

*At the top of the browser panel there's a **path** that indicates the actual location in the navigation panel (for example "**/okm:root/**"). Below, there is a list of the nodes related with the selected node in the navigator panel.*

Usually are listed the nodes that have the selected node as parent, except on Categories, Metadata and Thesaurus view where the nodes listed are related with the selected node and have not the same parent.

By default the browser table shows the columns:

- *Status (icons display node status, for example if the node is locked is shown the **Lock**  **icon**).*
- *Checkbox (to apply actions on several nodes at same time).*
- *Icon (indicate node type).*
- *Name (name of the node).*
- *Title.*
- *Language (only for documents).*
- *Size.*
- *Update date.*
- *Author.*
- *Version (only for documents).*

*Also in browser table can be shown extra columns based on metadata field values. For more information about how enable it take a look at: **Administration** > Profiles.*

*When you do a **right click** on the mouse button at some node it is shown a **contextual menu**. You will have distinct menu options depending on the selected node type and the visible stack panel combination.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

A complete description of all menu options is available at: Working with the menu bar.

*Available menu options when **Taxonomy, My documents** stack is visible are:*

- *Download.*
- *Edit.*
- *Update.*
- *Cancel edit.*
- *Lock.*
- *Unlock.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Convert.*
- *Add note.*
- *Add category.*
- *Add keyword.*
- *Add metadata group.*
- *Update metadata group.*
- *Merge PDF.*
- *Add bookmark.*
- *Set default home.*
- *Export to ZIP.*
- *OCR data capture.*
- *OMR data capture.*
- *Send link.*
- *Send document attachment.*
- *Forward e-mail.*

*Available menu options when **Categories, Metadata** and **Thesaurus** stack is visiable are:*

- *Download.*
- *Edit.*
- *Update.*
- *Cancel edit.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Lock.*
- *Unlock.*
- *Delete.*
- *Rename.*
- *Add note.*
- *Add category.*
- *Add keyword.*
- *Add metadata group.*
- *Update metadata group.*
- *Go to (document, folder, mail, record). Open node location in taxonomy view.*
- *Send link.*
- *Send document attachment.*
- *Forward e-mail.*

*Available menu options when **Templates** stack is visible are:*

- *Download.*
- *Edit.*
- *Update.*
- *Cancel edit.*
- *Lock.*
- *Unlock.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Convert.*
- *Add note.*
- *Add category.*
- *Add keyword.*
- *Add metadata group.*
- *Update metadata group.*
- *Merge PDF.*
- *Export to ZIP.*
- *Create from template.*
- *Add bookmark.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Send link.*
- *Send document attachment.*
- *Forward e-mail.*

> ***Create from template*** *helps you to create new documents based on existing templates. More information at: Creating templates.*

Available menu options when ***E-mail*** *stack is visible are:*

- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Convert.*
- *Add note.*
- *Add category.*
- *Add keyword.*
- *Add metadata group.*
- *Update metadata group.*
- *Merge PDF.*
- *OCR data capture.*
- *OMR data capture.*
- *Send link.*
- *Send document attachment.*
- *Forward e-mail.*

Available menu options when ***Trash*** *stack is visible are:*

- *Restore.*
- *Purge.*

The ***checkbox*** *helps you to* ***select several nodes*** *at same time and* ***apply an action for all****.*

> ***You can do actions on several nodes at the same time*** *(you have a checkbox in each row for selecting the node). The massive actions allowed are:*
>
> - *Delete.*
> - *Move.*
> - *Copy.*
> - *Add notes.*

- *Add categories.*
- *Add keywords.*
- *Add metadata groups.*
- *Change metadata group field values.*

*On **making a subscription** to node an **icon** will appear at the side of node indicating that **you are subscribed**. Keep on mind that in order to receive these mail notifications you have to configure which events you want to be notified in the user profile. See Profile Notification tab for more info.*

When there is no checkbox selected the action will be applied to the selected node (blue row mark the selected node in browser view).

The Massive icon displays a contextual menu for doing quick actions:

Feature	*Description*
Select all	*Selects all the nodes.*
Select all folders	*Selects all the nodes of type folder.*
Select all document	*Selects all the nodes of type document.*
Select all mails	*Selects all the nodes of type mail.*
Select all records	*Selects all the nodes of type record.*
Remove selection	*Unselect all the nodes.*

Pagination

*When **pagination feature** is enabled is shown and extra toolbar:*



Feature	*Description*
Show / hide folders	*Shows / hides nodes of type folder in the browser panel.*
Show / hide documents	*Shows / hides nodes of type document in the browser panel.*

Show / hide mails	*Shows / hides nodes of type mail in the browser panel.*
Show / hide records	*Shows / hides nodes of type record in the browser panel.*
Filter	*Filters nodes.*
Paginate *checkbox*	*Enables or disables the pagination feature.*
Limit *select*	*Sets the limit of nodes to be shown at the same time.*
Go to the beginning	*Goes to the beginning of the list.*
Go to previous	*Goes to previous list.*
Go to next	*Goes to next list.*
Go to the end	*Goes to the end of the list.*
Order *select*	*Sets the column value for that ordering.*
Reverse *checkbox*	*Reverses the ordering.*

Mouse features

- ***Double-click*** *on a node of type folder to* ***open the folder***.
- ***Double-click*** *on a node of type document to* ***download the document***.
- ***Right click*** *on a node will show a* ***contextual menu***.
- ***Drag & drop***. *You can drag & drop nodes keeping the left mouse button held down and moving to destination node in the Navigator panel.*

Properties

The properties panel at bottom center, show a collection of tabs that contains the grouped properties or features like preview

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

associated with the selected node. The selected node can be a navigator node or a browser node.

> *Basically the selected node at the navigator panel determines the content shown at the browser panel and depending if the last selected node by the user is at navigator or browser panel the properties panel will shown the properties of the active node at navigator or browser.*

> *The features described below are the default features. In your properties tab can be shown others not described here if you have OpenKM extensions enabled.*

Tabs on nodes of type document

Tab	*Description*
Properties	*Displays properties of the selected document.* *From this view you can:* • ***Set the keywords*** *of the document.* • ***Change the title*** *of the document.* • ***Set relations between the document and categories***. *From this tab is shown a* ***list with the users subscribed to the node***. *For a quick node sharing, there are two special icons named* ***URL*** *and* ***WebDAV*** *when clicking on them, it will copy in your clipboard a browser or WebDAV URL.*
Notes	*Displays a list of notes related with the document. From this view you can add, edit or remove your own notes.* *Administrators can manage notes of other users.*
History	*Displays a list with the versions of the document.* *From this view you can:* • ***Restore to a previous version*** *of the document.* • ***Preview any document version*** *on the right panel.* • ***Compare two versions of the document*** *on the preview panel.* • ***Compact the history***. *The action of compacting the history will remove all the document version except the active.*

This action freezes repository size but must be take in consideration that this action can not be reverted.

Security

Displays the security grants of the document. From this view you can change the security grants.

- *Click on **Update button** to change the security grants.*

When "Recursive permission changes" checkbox is checked the changes will be applied recursivelly.

You should also be interested on reading:

- *Security configuration.*
- *Security configuration parameters.*
- *"**Advanced filter**" option on Profile General tab.*

Preview

Displays a pre visualization of the document.

Relations

Displays the relations of the document to other nodes (documents, folders, mails or records). From this view you can manage the relations of the document, adding or removing relation.

To set a new relation:

- *First must **select the type** from the relation type list.*
- *Then **choose one** of the defined relations.*
- ***Select a node in the repository**, four icons for searching are shown: document, folder, mail or record.*

There're three types or relations:

- *Bidirectional*
- *Parent-child*
- *Many-to-many*

More information at Relation types.

Activity log

*This view performs a complete audit trail of the document. From this view you can filter by specific actions. When the checkbox named **View** is not checked, the most common actions like*

GET_CHILDREN_DOCUMENTS *are automatically filtered.*

The reason why most common actions are filtered when checkbox name View is not checked is that these events usually create a large list of actions and often are not relevant from the audit trail point of view.

This option is only available if you have enabled the ***Activity log*** *feature.*

Forum

Displays a forum associated with the document.

From this view you can:

- ***Add a new topic***.
- ***Post a reply*** *of the topic.*
- ***Quote a reply*** *of the topic.*
- ***Edit a reply*** *of the topic.*
- ***Delete a topic***.

To delete or edit a post of the topic you must be the owner or have administrator grants.

This option is only available if you have enabled the ***Forum*** *feature.*

Workflow

Display the workflows associated with the document.

From this view you can:

- *Show the* ***detail*** *of the workflow variables.*
- ***Show a graph*** *where it is marked the actual node.*

This option is only available if you have enabled the ***Workflow*** *feature.*

Text extracted

Displays the text extracted of the document.

Only when the document has been processed by text extractor worker you will see the text. For more information: ***Administration*** *>* *Statistics*.

This option is only available if you have enabled the ***Text extracted*** *feature.*

Tabs on nodes of type folder

Tab	*Description*
Properties	*Displays properties of the selected folder.* *From this view you can:* • ***Change the folder style***. • ***Set the keywords*** *of the folder.* • ***Change the title*** *of the folder.* • ***Set relations between the folder and categories***. *This tab shows a* ***list with the users subscribed to the node***. *For a quick node sharing, there are two special icons named* ***URL*** *and* ***WebDAV*** *when clicking on them, it will copy in your clipboard a browser or WebDAV URL.* *More information: Folder style.*
Notes	*Displays a list of notes related with the folder. From this view you can add, edit or remove your own notes.* *Administrators can manage other user's notes.*
Security	*Displays the security grants of the folder. From this view you can change the security grants.* • *Click on* ***Update button*** *to change the security grants.* *When the "Recursive permission changes" checkbox is checked the changes will be applied recursivelly.* *You could be interested on reading:* • *Security configuration.* • *Security configuration parameters.* • *"**Advanced filter**" option on Profile General tab.*
Relations	*Displays the relations of the folder to other nodes (documents, folders, mails or records). From this view*

you can manage the relations of the folder, adding or removing relation.

To set a new relation:

- *First must **select the type** from the relation type list.*
- *Then **choose one** of the defined relations.*
- ***Select a node in the repository**, four icons for searching are shown: document, folder, mail or record.*

There're three types or relations:

- *Bidirectional*
- *Parent-child*
- *Many-to-many*

More information at: Relation types.

Activity log

This view performs a complete audit trail of the folder. From this view you can filter by specific actions.

*When the checkbox named **View** is not checked, the most common actions like **GET_CHILDREN_DOCUMENTS** are automatically filtered.*

The reason why most common actions are filtered when checkbox name View is not checked is that these events usually create a large list of actions and often are not much relevant from the audit trail point of view.

*This option is only available if you have enabled the **Activity log** feature.*

Forum

Display a forum associated with the folder.

From this view you can:

- ***Add new topic**.*
- ***Post a reply** to the topic.*
- ***Quote a reply** tof the topic.*
- ***Edit a reply** of the topic.*
- ***Delete a topic**.*

To delete or edit a post of the topic you must be the owner or have administrator grants.

	*This option is only available if you have enabled the **Forum** feature.*
Workflow	*Display the workflows associated with the folder.* *From this view you can:* • *Show the **detail** of the workflow variables.* • ***Show a graph** where is marked the actual node.* *This option is only available if you have enabled the **Workflow** feature.*

Tabs on nodes of type mail

Tab	*Description*
Properties	*Display properties of the selected mail.* *From this view you can:* • ***Set the keywords** of the mail.* • ***Change the title** of the mail.* • ***Set relations between the mail and categories**.* *This tab shows a **list with the users subscribed to the node**.* *For quick node sharing, there are two special icons named **URL** and **WebDAV** when clicking on them, it will copy in your clipboard a browser or WebDAV URL.*
Notes	*Displays a list of notes related with the mail. From this view you can add, edit or remove your own notes.* *Administrators can manage notes of other users.*
Security	*Displays the security grants of the mail. From this view you can change the security grants.* • *Click on **Update button** to change the security grants.* *When "Recursive permission changes" checkbox is checked the changes will be applied recursivelly.*

You might be interested on reading:

- *Security configuration.*
- *Security configuration parameters.*
- *"**Advanced filter**" option on Profile General tab.*

Preview

Displays a pre visualization of the mail.

From this view you can also download the mail attachments or copy them to a destination folder.

Relations

Displays the relations of the mail to other nodes (documents, folders, mails or records). From this view you can manage the relations of the document, adding or removing relation.

To set a new relation:

- *First must **select the type** from the relation type list.*
- *Then **choose one** of the defined relations.*
- ***Select a node in the repository**, for it are shown four icons for searching a document, folder, mail or record.*

There're three types or relations:

- *Bidirectional*
- *Parent-child*
- *Many-to-many*

More information at: Relation types.

Activity log

*This view performs a complete audit trail of the mail. From this view you can filter by specific actions. When the checkbox named **View** is not checked, the most common actions like **GET_CHILDREN_DOCUMENTS** are automatically filtered.*

The reason why most common actions are filtered when checkbox name View is not checked is that these events usually create a large list of actions and often they are not relevant from the audit trail point of view.

	*This option is only available if you have enabled the **Activity log** feature.*
Forum	*Displays a forum associated with the mail.* *From this view you can:* • ***Add a new topic**.* • ***Post a reply** to the topic.* • ***Quote a reply** of the topic.* • ***Edit a reply** of the topic.* • ***Delete a topic**.* *To delete or edit a post of the topic you must be the owner or have administrator grants.* *This option is only available if you have enabled the **Forum** feature.*
Workflow	*Display the workflows associated with the mail.* *From this view you can:* • *Show the **detail** of the workflow variables.* • ***Show a graph** where is marked the actual node.* *This option is only available if you have enabled the **Workflow** feature.*

Tabs on nodes of type record

Tab	*Description*
Properties	*Displays properties of the selected record.* *From this view you can:* • ***Set the keywords** of the record.* • ***Change the title** of the record.* • ***Set relations between the mail and categories**.* *This tab is shown as a **list with the users subscribed to the node**.* *For a quick node sharing, there are two special icons named **URL** and **WebDAV** when*

clicking on them will copy in your clipboard a browser or WebDAV URL.

Notes

Displays a list of notes related with the record. From this view you can add, edit or remove your own notes.

Administrators can manage notes of other users.

Security

Displays the security grants of the record. From this view you can change the security grants.

- *Click on **Update button** to change the security grants.*

When "Recursive permission changes" checkbox is checked the changes will be applied recursivelly.

You might be interested on reading:

- *Security configuration.*
- *Security configuration parameters.*
- *"**Advanced filter**" option on Profile General tab.*

Relations

Displays the relations of the record to other nodes (documents, folders, mails or records). From this view you can manage the relations of the record, adding or removing relation.

To set a new relation:

- *First must **select the type** from the relation type list.*
- *Then **choose one** of the defined relations.*
- ***Select a node in the repository**, four icons for searching are shown document, folder, mail or record.*

There're three types or relations:

- *Bidirectional*
- *Parent-child*
- *Many-to-many*

More information at: Relation types.

Activity log	*This view performs a complete audit trail of the record. From this view you can filter by specific actions.* *When the checkbox named **View** is not checked, the most common actions like **GET_CHILDREN_DOCUMENTS** are automatically filtered.* *The reason why most common actions are filtered when checkbox name View is not checked is that these events usually create a large list of actions and often they are not relevant from the audit trail point of view.* *This option is only available if you have enabled the **Activity log** feature.*
Forum	*Displays a forum associated with the record.* *From this view you can:* • ***Add a new topic**.* • ***Post a reply** to the topic.* • ***Quote a reply** of the topic.* • ***Edit a reply** to the topic.* • ***Delete a topic**.* *To delete or edit a post of the topic you must be the owner or have administrator grants.* *This option is only available if you have enabled the **Forum** feature.*
Workflow	*Display the workflows associated with the record.* *From this view you can:* • *Show the **detail** of the workflow variables.* • ***Show a graph** where is marked the actual node.* *This option is only available if you have enabled the **Workflow** feature.*

Shortcuts

Shortcuts can be used from desktop view.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Available shortcuts:

Shortcut	*Description*
F2	*Rename a node.*
CTRL+SHIFT+C	*Copy a node.*
CTRL+SHIFT+X	*Cut a node.*
CTRL+SHIFT+V	*Paste a node.*
CTRL+D	*Downloading.*
CTRL+G	*Show UI testing popup.*
CTRL+Z	*Show UI log.*
CTRL+N	*Create folder.*
F5	*Refresh desktop view.*
INSERT	*Add new document.*
ALT+B	*Bookmark a node.*
F1	*Show help.*
CTRL+ALT+Q	*Exit application.*
ALT+DELETE	*Delete a node*

Shortcuts are configurable from Profiles.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Working with search

The search tab enables the query the repository. The search view is distributed in four sections:

- *Searches panel.*
- *Search panel.*
- *Control panel.*
- *Results.*

Search panels

The Search panel at left hand side is an stack panel.

> *Once a search has been done, it can be saved for future sessions by entering a name that identifies it and clicking on the "Save search" button.*

This panel have two stacks:

- *Stored searches.*
- *User news.*

***Stored searches** shows a list of user's searches that has been saved to be use later. With a double-click on stored search, you can execute the search again. With right click on a stored search will be shown a contextual menu with this options:*

Feature	*Description*
Run	*Run stored search.*
Delete	*Delete an stored search.*

> *The stored searches and users news are only visible for the user who created them.*

***User news** shows a list of user searches that are used as push service (usually executed every 30 minutes). These searches are periodically executed and the results are shown at **Dashboards** > **News** panel.*

> *User news helps users to be periodically updated about repository news based on his own queries.*
>
> *Will be shown only new results that have appeared since the last time the query was executed.*

Search panel

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

This panel at the top center gives access to different searches to query the content of the application:

- ***Default search** is a full-text search. Default search is also available from the top right corner on the screen.*
- ***Advanced search** enables you to search on keywords, metadata and additional criteria such as location in the application.*

*You can **switch between default or advanced** search **checking or un checking** the "**View advanced mode**" checkbox.*

*Default search engine configuration uses **stemming** and **stopwords**.*

- ***Stemming** is the term used in linguistic morphology and information retrieval to describe the process for reducing inflected (or sometimes derived) words to their word sterm, base or root form generally a written word form. The stem needs not to be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. For instance, if you type "fishing" in a full-text search field, search results will return documents with "fishing", "fish", "fisher". In that case, the stem is "fish".*
- ***Stopwords** usually refer to the most common words in a language, there is no single universal list of stop words used by all processing of natural language tools, and indeed not all tools even use such a list. The stopword are removed from search. For example "the", "is", "at", "which" and "on" are stopwords.*

Operators and wildcards

*By default internally the **search engine use the "AND" operator**.*

You can use wildcards:

- *Add an asterisk "*****" as a placeholder for any unknown or wildcard terms, for example, "***2015**", "**2015***", "**20*15**" or "***2015***".*
- *Add an asterisk "**?**" as a placeholder for any unknown or wildcard character, for example the query "**?015**" will return occurrences where first character doesn't matter.*
- *The query "**2015**" is a synonim of "**2015***". On this search view application always add "*****" for you at the end.*

Default search

Default search is a full-text search which only search by document content.

Default search shows results nodes only in "/okm:root" context, results from other context are automatically filtered.

Always at the top right corner you have quick access with Quick search box feature.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Advanced search

Advanced search enables you to search on keywords, metadata and additional criteria such as location in the application.

The advanced search organizes the query fields on tree tabs:

- *Basic (most common fields).*
- *Advanced.*
- *Metadada.*

The available criteria from basic view :

Field	*Description*	*Wildcard allowed*
Context	*Sets the context where is executed the query.* *Available contexts:* • *Taxonomy* • *Templates* • *My documents* • *E-mail* • *Trash*	*No*
Content	*The content of the node.* *Document content or mail body.*	*Yes*
Name	*The name of the node.*	*Yes*
Title	*The title of the node.*	*Yes*
Keyword	*The associated keywords of the node.*	*Yes*
Language	*The language of the document.* *Only the nodes of typed document has language value.*	*Yes*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

User	*The user who created the node.*	*No*
Date range	*The range of dates where the node has been modified (or created).* *Each selected date is set to 00 hours, 00 minutes, 00 seconds and 00 miliseconds. That means if you want to do a query of a node modified the day 2015-08-15 must set a range from "2015-08-15" to "2015-08-16" (internally you're querying from "2015-08-15 00:00:00" to "2015-08-16 00:00:00").* *It is not mandatory to set both dates for range ("**from**" or the "**to**"), you can set only one date.*	*No*

The available criteria from advanced view :

Field	*Description*	*Wildcard allowed*
Folder	*Restrict results to folders.* *When Recursive checkbox is checked also considers the subfolders.*	*No*
Category	*Restricts results to categories. The nodes must be associated to a specific category.*	*No*
Type	*Restrict the types of node:* • *Document* • *Folder* • *Mail* • *Record* *When Mail checkbox is checked are shown extra fields (from, to , subject)*	*No*
MIME type	*Set the document MIME type.*	*No*
Notes	*The content of the notes.*	*Yes*

From	*The from mail field.* *This field is only available if you have **checked the Mail checkbox**.*	*Yes*
To	*The to mail field.* *This field is only available if you have **checked the Mail checkbox**.*	*Yes*
Subject	*The subject mail field.* *This field is only available if you have **checked the Mail checkbox**.*	*Yes*

From metadata view you can filter by any metadata field from existing metadata groups:

- *Click on* ***Add metadata group*** *button.*
- ***Choose one metadata group*** *from the list.*
- ***Choose a property*** *from the list.*
- *Click* ***Add*** *button.*

On almost input and text area fields you can use wildcards.

Control panel

Control panel at the right top corner has several features:

Featureright	*Description*
Switch to Advanced mode checkbox	*When checked is shown the advanced mode search view.*
Results compact view checkbox	*When checked the results are shown in a compact table with minimal details.*
View metadata groups	*When checked are shown metadata groups associated to each node in results view.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

checkbox	*This option is only available when "Results compact view" is not checked.*
Save as user news checkbox	*When checked if the user saves a search, will be set to user news search.*
 ***Save search button***	*Once a search has been done, it can be saved for future sessions by entering a name that identifies it and clicking on the "Save search" button.* *Depending if "Save as user news" checkbox is checked or not will be saved as "stored search" or "user news".*
Page results lists	*Set the limit of nodes to be shown in the results list.*
 ***Clean button***	*Resets the search.*
 ***Search***	*Executes the search.*
Results	*Indicates the number of results (total).*
 ***Go to previous***	*Goes to previous list.*
 ***Go to next***	*Goes to next list.*
 ***Export to CSV***	*Downloads a CSV file with results.*
 ***Export to ZIP***	*Downloads a ZIP file with results.*

Results

The Results panel at the bottom center shows a table of nodes (documents, folders, mails and records) what are the result of querying the repository. Results panel are shown in:

- *Compact view (a table with a single row for each node).*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Full view (a detailed description for each node).*

> *Stars indicates the relevance of the document found. This value is established by the Search engine according to the search field and the results found. The list of documents found is ordered from most to least relevant.*

When you do a right click on compact view will be shown a contextual with the options:

Feature	*Description*
Download	*Downloads the node.* *Only shown in document or mail nodes.*
Go to document	*Goes to the document location.* *Only shown in document nodes.*
Go to folder	*Goes to folder location.* *Only shown in folder nodes.*
Go to e-mail	*Goes to mail location.* *Only shown in mail nodes.*
Go to record	*Goes to the record location.* *Only shown in record nodes.*
Preview	*Previews the node.* *Only shown in document or mail nodes.*

From full view you see detailed information of each node and also are available some features:

Feature	*Description*
Download	*Download the node.* *Only shown in document or mail nodes.*
Notes	*Show notes. From this view can manage the notes of the node.*
Preview	*Preview the node.* *Only shown in document or mail nodes.*
Options	*Show extra mail extra fields (cc, bcc, reply, extra fields).* *Only shown in document or mail nodes.*

Working with Lucene queries

*Keep on mind that this information is only valid for **Lucene query advanced search**, not for regular search. To get this **Lucene query** tab go to **Administration** > **Profiles**, choose the right profile and enabled it at **Components** > **Seach** > **Tab Lucene query**.*

With this new feature you will be able to create complex Lucene queries, so you won't be limited by the current search form fields. But keep on mind this can be difficult unles you know how to write these queries. Let's define a couple of items:

- ***Terms**: A query is broken up into terms and operators. A term is a single word like "cat" or "dog". Multiple terms can be combined with boolean operators for a more specific query. For example:*

  ```
  cat AND dog
  ```

- ***Fields**: When you perform a Lucene seach, you can specify a field or use the default field. In OpenKM the default field contains the document text extracted. You can search by a field writing the field name, followed by a colon ":" and a term. For example, this query will search for all nodes in taxonomy which name is "animals":*

  ```
  context:okm_root AND name:animals
  ```

You can search for a phrase using quotes:

```
name:"big animals"
```

Term modifiers

Lucene support different term modifiers to create complex searches.

Wildcard searches

You can use single and multiple character wildcard searches within terms.

- ***Single**: To perform a single character wirldcard search use the "?" symbol. For example, to search for "text" or "test":*

  ```
  te?t
  ```

- ***Multiple**: To perform a multiple characted wildcard, use the "*" symbol. For example, to search for "test", "tests" or "tester":*

  ```
  test*
  ```

You can't use a "" or "?" at the begining of a query.*

Proximity searches

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Lucene also supports finding words that are in a specific distance. To perform a proximity search use the tild "~" symbol and the end of the phrase.

For example, to search for a "cat" and "dog" within 10 words of each other in a document:

```
"cat dog"~
```

Range searches

Range queries allow to match documents whose field values are between an specified lower and upper bound. Sorting is done lexicographically.

For example, to look for documents which field "name" is between "cat" and "horse" but not will include these terms:

```
name:{cat TO horse}
```

If you want these two terms to be included, use this query:

```
name:[cat TO horse]
```



Inclusive range queries are denoted by square brackets. Exclusive range queries are denoted by curly brackets.

Boolean operators

You can combine several terms using boolean operators. Lucene supports "AND", "+", "OR", "NOT" and "-".



Boolean operators must be in UPPERCASE.

Working with Lucene queries

With this new feature you will be able to create complex Lucene queries, so you won't be limited by the current search form fields. But keep on mind this can be difficult unles you know how to write these queries. Let's define a couple of items:

- ***Terms**: A query is broken up into terms and operators. A term is a single word like "cat" or "dog". Multiple terms can be combined with boolean operators for a more specific query. For example:*

  ```
  cat AND dog
  ```

- ***Fields**: When you perform a Lucene seach, you can specify a field or use the default field. In OpenKM the default field contains the document text extracted. You can search by a field writing the field name, followed by a colon ":" and a term. For example, this query will search for all nodes in taxonomy which name is "animals":*

  ```
  context:okm_root AND name:animals
  ```

You can search for a phrase using quotes:

```
name:"big animals"
```

Term modifiers

Lucene support different term modifiers to create complex searches.

Wildcard searches

You can use single and multiple character wildcard searches within terms.

- ***Single**: To perform a single character wirldcard search use the "?" symbol. For example, to search for "text" or "test":*

```
te?t
```

- ***Multiple**: To perform a multiple characted wildcard, use the "*" symbol. For example, to search for "test", "tests" or "tester":*

```
test*
```

You can't use a "" or "?" at the begining of a query.*

Proximity searches

Lucene also supports finding words that are in a specific distance. To perform a proximity search use the tild "~" symbol and the end of the phrase.

For example, to search for a "cat" and "dog" within 10 words of each other in a document:

```
"cat dog"~
```

Range searches

Range queries allow to match documents whose field values are between an specified lower and upper bound. Sorting is done lexicographically.

For example, to look for documents which field "name" is between "cat" and "horse" but not will include these terms:

```
name:{cat TO horse}
```

If you want these two terms to be included, use this query:

```
name:[cat TO horse]
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Inclusive range queries are denoted by square brackets. Exclusive range queries are denoted by curly brackets.

Boolean operators

You can combine several terms using boolean operators. Lucene supports "AND", "+", "OR", "NOT" and "-".

Boolean operators must be in UPPERCASE.

The "OR" operator is the default conjuntion operator: this means that if there is no boolean operator between two terms, the "OR" operator is used. These two queries are equivalents:

```
cat animal
```

And this one:

```
cat OR animal
```

AND

The AND operator matches documents where both terms exists. The symbol "&&" can be also used to replace the word "AND".

Let's look for documents with both "cat" and "animal" words in the content:

```
cat AND animal
```

+

This is called the required operator and force that the term placed after the "+" to be included in the results.

For example, to search for documents that must contain "animal" and may contain "cat":

```
+animal cat
```

NOT

The NOT operator excludes documents that contains the term after the "NOT". The symbol "!" can be used to replace the word "NOT".

For example, to search for documents that contains "animal" but not "cat":

```
animal NOT cat
```

The NOT operator cannot be used with just one term.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

-

This operator excludes documents which contains the term given after the "-" character:

For example, to seach documents which contains "animal" but not "cat":

```
animal -cat
```

Grouping

Lucene supports using parentheses to group clauses to form sub queries. This can be very useful if you want to control the boolean logic for a query.

For example, to search for "cat" or "dog" and "animal" use this query:

```
(car OR dog) AND animal
```

Escaping special characters

Lucene supports escaping some special characters that are part of the query syntax using the slash "\" before the character to be escaped. This is the current list of special characters:

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \
```

The "OR" operator is the default conjuntion operator: this means that if there is no boolean operator between two terms, the "OR" operator is used. These two queries are equivalents:

```
cat animal
```

And this one:

```
cat OR animal
```

AND

The AND operator matches documents where both terms exists. The symbol "&&" can be also used to replace the word "AND".

Let's look for documents with both "cat" and "animal" words in the content:

```
cat AND animal
```

+

This is called the required operator and force that the term placed after the "+" to be included in the results.

For example, to search for documents that must contain "animal" and may contain "cat":

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
+animal cat
```

NOT

The NOT operator excludes documents that contains the term after the "NOT". The symbol "!" can be used to replace the word "NOT".

For example, to search for documents that contains "animal" but not "cat":

```
animal NOT cat
```



The NOT operator cannot be used with just one term.

-

This operator excludes documents which contains the term given after the "-" character:

For example, to seach documents which contains "animal" but not "cat":

```
animal -cat
```

Grouping

Lucene supports using parentheses to group clauses to form sub queries. This can be very useful if you want to control the boolean logic for a query.

For example, to search for "cat" or "dog" and "animal" use this query:

```
(car OR dog) AND animal
```

Escaping special characters

Lucene supports escaping some special characters that are part of the query syntax using the slash "\" before the character to be escaped. This is the current list of special characters:

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \
```

Working with metadata fields

*Because OpenKM use lucene restricted characters like ":", the field name must be sanitized. For example, a field named in OpenKM as "**okp:consulting.text**", should be sanitized as "**okp_consulting_text**" replacing the character ":" and "." by "_".*

```
okp_consulting_text:value
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



In case of working with metadata fields it's needed to enable the ***sanitize.lucene.fields*** *configuration properties, otherwise you won't be able to search by metadata fields.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Working with the dashboard

The dashboard panel provides a set of dashboards, each user has a set of personal dashboards that display the information relevant to them, and helps to have a global view of the application activity.

The dashboard shows:

- *Displays the actions that have been carried out in the system*
- *Displays a global view of application activity.*
- *Displays information based on the scheduled user search queries.*
- *Users work on the company's business process tasks (workflows)*



The results listed in the dashboard boxes depend on the security level access for each user.

The information is organized in several boxes:

Box	*Description*
Locked documents	*Displays a list of the actual documents locked by the user.*
Locked records	*Displays a list of the actual records locked by the user.*
Edited documents	*Displays a list of the actual documents checked out by the user.*
Last downloaded documents	*Displays a list of the last 20 documents downloaded by the user.*
Subscribed documents	*Lists subscriptions into documents by the user.*
Subscribed folders	*Lists subscriptions into folders by the user.*
Subscribed mails	*Lists subscriptions into emails by the user.*
Electronic mails	*Displays a list of the last 20 emails uploaded by the user.*
Attachments	*Displays a list of the last 20 documents attached to emails uploaded by the user.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Last modified documents	*Displays a list of the last 20 documents modified by the user.*
Last created documents	*Displays a list of the last 20 documents created by the user.*
Last created folders	*Displays a list of the last 20 folders created by the user.*
Last created records	*Displays a list of the last 20 records created by the user.*
Last created documents	*Displays a list of the last 20 documents created.*
Last modified documents	*Displays a list of the last 20 documents modified.*
Last created folders	*Displays a list of the last 20 folders created.*
Last created records	*Displays a list of the last 20 records created.*

Each box indicates the number of new items pending to be viewed by the user.

When a user clicks on a row, the application will automatically open up the repository destination node.

News

The News view displays information based on scheduled user search queries. Each query is shown in a separate box.

More information about registering "Users news" at: Working with search.

Workflow

In the Workflow panel, the workflows are shown in execution. The Workflow panel is organized in two sections:

- *Workflow lists.*
- *Workflow detail.*

Workflow lists

In the Workflow lists panel are shown the workflows on execution organized in two boxes:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Pending tasks.*
- *Unassigned pending task.*



The workflows at the "Pending task" box have yet to be assigned to the logged-in user and are waiting for him.

The workflows at the "Unassigned pending task" box have not yet been assigned to a single user. There is a group of users who can auto assign the workflow themself (one of them is the logged user). The workflow is waiting for this automatic assignment.

Workflow detail

In the Workflow detail panel itis shown the workflow details. From this view, you can see low-level information data like "process instance", "process definition", or "data", and also ***go to the next workflow task, completing the workflow form*** *.*

To complete a workflow task:

- ***Click on workflow in the Pending task*** *box. It will be shown the workflow form.*
- ***Fill in the workflow form****.*
- ***Click on any of the available buttons*** *to submit the values.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

User email signature

There is an option to add a signature for emails created by OpenKM.

*Just place a HTML file called **signature.html** inside your personal folder "**/okm:personal/username**" and when you create a new email this content will be used as your signature.*

You can download an example here: signature.html

You can create a signature with any HTML valid code.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Activity log extension

*Digital Signature viewer helps you to **view** the activity log of a node*

*This option is only available if you have enabled the **Activity log** extension.*

More information at Enable extensions.

This extension requires additional software configuration.

Activity log

- *Choose a node.*
- ***Show the Activity log** tab.*
- *Filter by:*
  - *Action*
  - *Date range*

Features:

Feature	*Description*
Action	*The action that has been performed on the node.*
User	*The user who performed the action.*
Date	*The date on which the action was performed.*
Parameters	*Additional information related to the action performed.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AutoCAD preview extension

AutoCAD previewer allows one to perform basic operations like zoom in/out, rotate, show/hide layers and print. AutoCAD preview extension allows to view AutoCAD files using SVG and JavaScript to display and manipulate graphics.

*This option is only available if you have enabled the **AutoCAD** feature.*

This feature needs third-party software installation. Ask your OpenKM technical staff for it.

Toolbar



Feature	*Description*
Change icon Page	*Allows to change to previous or next icon page.*
Layer List	*List of the layers of the drawing.*
Print	*Print the document.*
Zoom In	*Performs a zoom in.*
Zoom Out	*Performs a zoom in.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	***Zoom Extends***	*Allows you to move the zoom in zone.*
	***Zoom Window***	*Allows you to draw a window for do zoom in.*
	***Toggle Black/White***	*Allows you to toggle black/white color in the drawing.*
	***Background Color***	*Allows you to select a specific color for the background of the drawing.*
	***File Load***	*Not implemented yet.*
	***Search Text***	*Allows you to search individual text string in the drawing.*
	***About***	*Information about CADViewer JS version.*
	***Layer Off***	*Turn layer off.*
	***Swap Layers***	*Allows to swap between layers.*
	***All Layers On***	*Turn all layers on.*
	***Measurement***	*Allows to select to points in the drawing to measure the distance between them.* *Needs to calibrate measurement first.*
	***Calibrate Measurement***	*Allows to calibrate measurement to define units of measure in drawing.*
	***Toggle Birds-Eye Window***	*Open the birds-eye window.*
	***Adjust Line Thickeness***	*Adjust the line thickness of the drawing.*
	***Save Redline***	*Save current red lines.*

	***Load Redline***	*Allows to load red lines previously saved.*
	***Delete Single Redline***	*Allows to delete one red line by selecting it in the drawing.*
	***Delete Last Redline***	*Delete last red line.*
	***Redline Color***	*Allows to change the color of the red line.*
	***Redline Thickness***	*Allows to change the thickness of the red line.*
	***Make Stickynote***	*Allows to insert an sticky note.*
	***Clear Visible Redlines***	*Clear all visible red lines.*
	***Freehand Redline***	*Allows to insert a red line with free form.*
	***Redline Cloud***	*Allows to insert a red line with a cloud shape.*
	***Redline Filled Polygon***	*Allows to insert a red line with a filled polygon shape.*
	***Redline Polyline***	*Allows to insert a red line with polyline shape.*
	***Redline Filled Rectangle***	*Allows to insert a red line with a full rectangle shape.*
	***Redline Rectangle***	*Allows to insert a rectangle red line.*
	***Redline Arrow***	*Allows to insert a red line with an arrow.*
	***Redline Text***	*Allows to write a text and select where to place it in the drawing.*

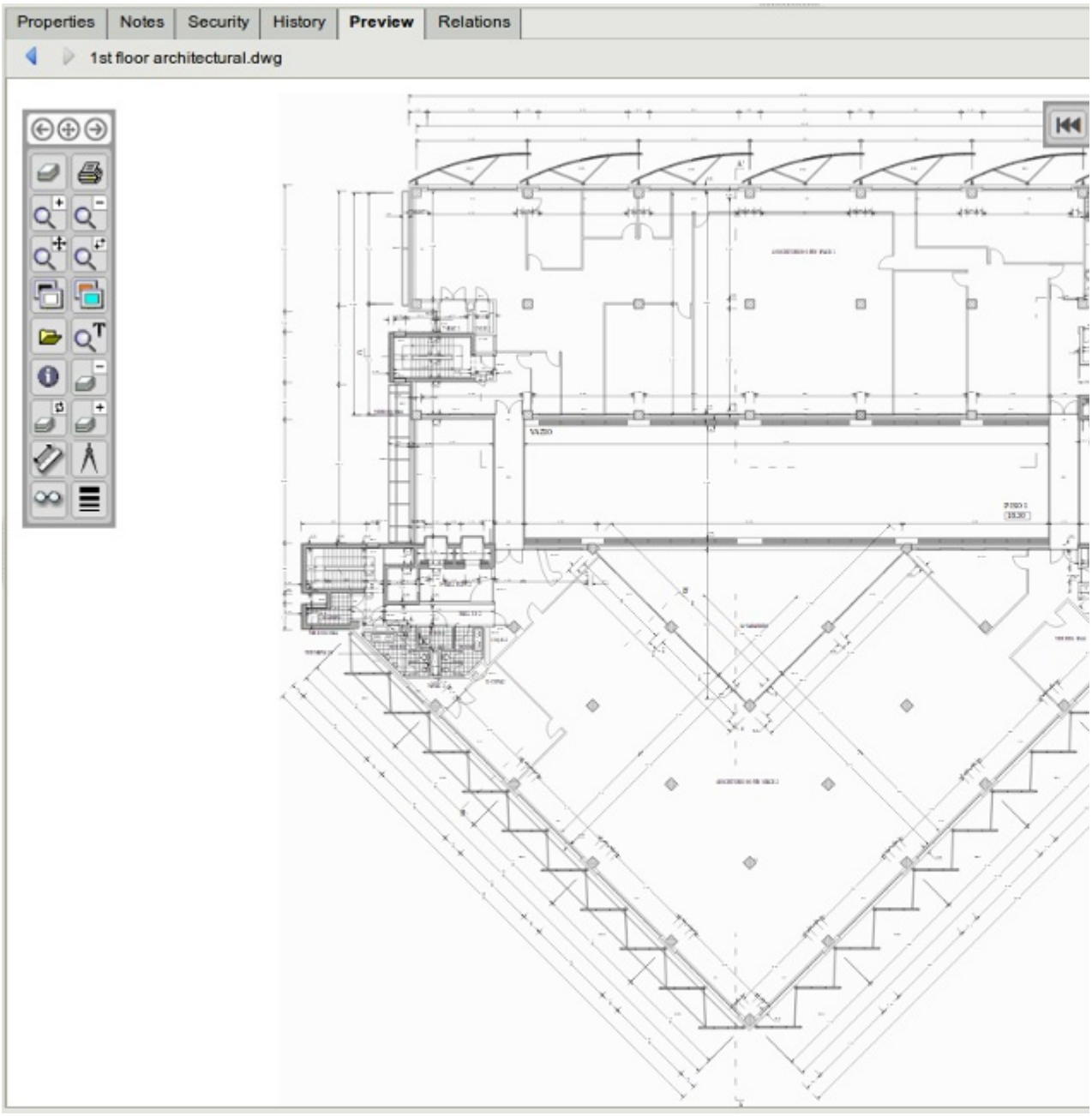
	***Zoom Extents***	*Allows to move the zoom-in zone.*
	***Zoom Window***	*Allows to draw a window for do zoom-in.*



	Feature	*Description*
	***First Page***	*Go to the first page in the drawing.*
	***Previous Page***	*Go to the previous page in the drawing.*
	***Next Page***	*Go to the next page in the drawing.*
	***Last Page***	*Go to the last page in the drawing.*

Default preview

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


Properties
Notes
Security
History
Preview
Relations
1st floor architectural.dwg

Digital Signature viewer extension

*Digital Signature viewer helps you to **view** and **verify** signed documents.*

*This option is only available if you have enabled the **Digital Signature viewer** extension.*

More information at Enable extensions.

This extension requires additional software configuration.

Verify a signed document

- *Choose a signed document.*
- ***Show the Digital Signature** viewer tab.*

Features:

Feature	*Description*
Integrity	*Display the result of checking the digital signature.*
Trust	*Display the result of checking if the signer certificate is trusted.*
Revocation	*Display the result of checking the revocation list from the certificate.*
Validity	*Display the result of checking certificate validation or signing time.*
Download the signed version	*You can download the signed document.*
Certificate information	• *You can download the certificate.* • *Display the subject.* • *Display the issuer.*
Validity information	*Display the signing time.*
Signed properties	• *Display signing reason.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

	• *Display signing location.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Document expiration extension

Document expiration is a notification system alert which can be used to advice users nearly document expiration and expired documents.



*This option is only available if you have enabled the **Document expiration** feature.*



More information at Enable extensions.

Add document expiration

- ***Choose a document***.
- ***Add** metadata group "**Expiration**"*.
- ***Fill the form***.
- *Click **Add button***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Extra tab workspace extension

The ExtraTab allows to easily add new tabs with a name and specific URL.

*This option is only available if you have enabled the **ExtraTab** feature.*

There are several parameters passed as request parameter in URL request:

- *The authentication **token** (encrypted)*
- *The **lang** (language id)*
- *The node **uuid***
- *The **userId***

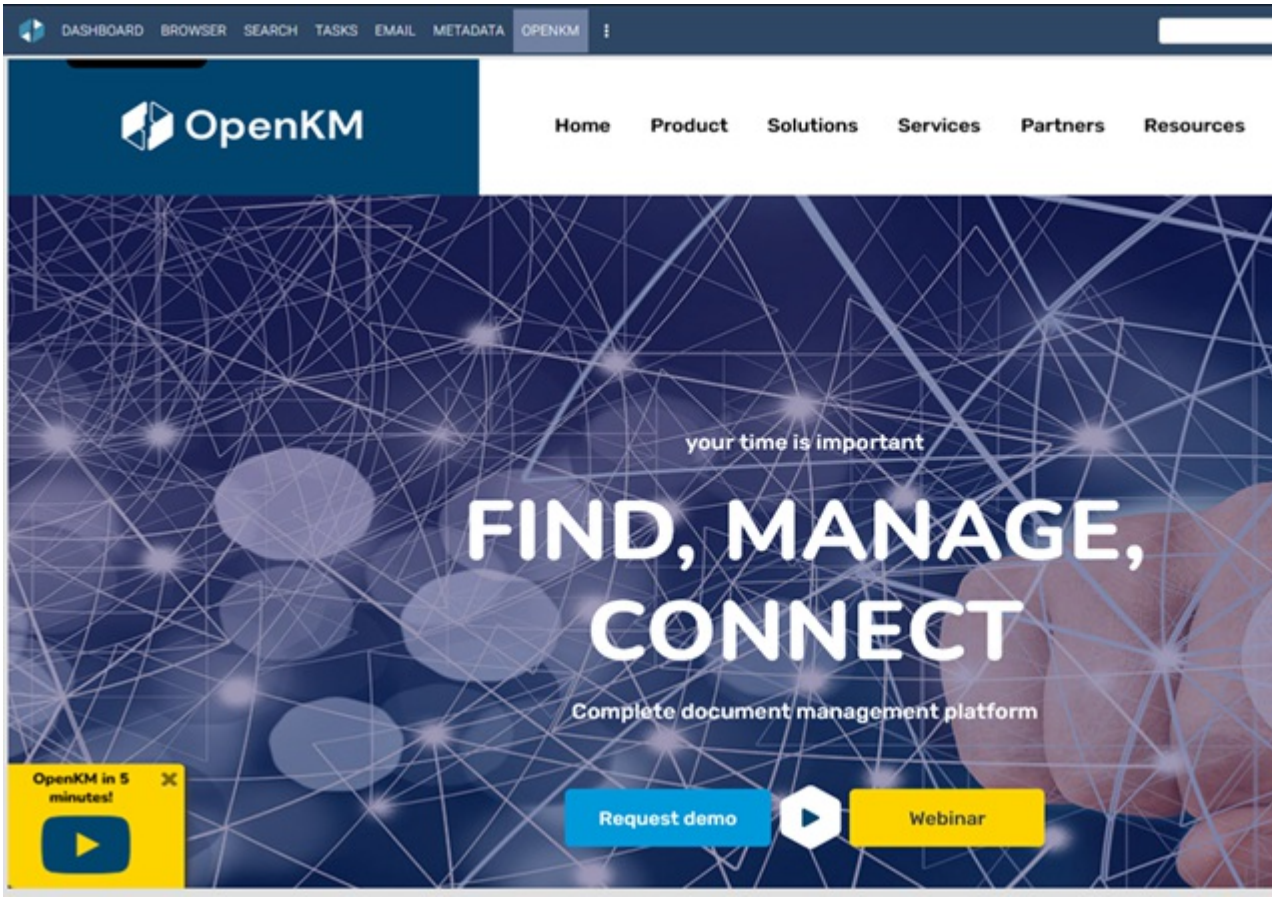
Configuration

Field / Property	Type	Description
extension.extra.tab.workspace	**List**	*Title and url pair. You can add single line (where the character ; is the separator between title and url) or several lines.* `OpenKM;https://www.openkm.com` *If **URL does not start with http** it's considered as **OpenKM relative path** and will be merged with your OpenKM server URL like **YOUR_OPENKM_URL/url***
webservices.visible.properties	**List**	*To make accessible to all the users the values of the parameter "**extension.extra.tab.workspace**", must be **added the value "extension.extra.tab.workspace"** in the current list.* *If you forget to add the property name, only administrator users will be able to get the value so some extensions may produce errors for regular users.*

More information at Configuration parameters.

Sample

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

External preview extension

The External Preview feature helps you to preview documents based on external URL request.

*This option is only available if you have enabled the **External preview** extension.*

There are two parameters passed as request parameter in URL request:

- *The document **uuid**.*
- *The **downloadUrl**.*

Configuration

Field / Property	Type	Description
extension.external.preview	List	*Each row couple of values is a preview content declaration, the format is:* ***mime-type;url*** *application/illustrator;http://localhost:8180/openkm/test.jsp* *application/epub+zip;http://localhost:8180/openkm/epub_previewer.jsp* *application/x-mspublisher;http://localhost:8180/openkm/mspreviewer_previewer.jsp*

More information at Configuration parameters.

Sample

Configuration

Create file named test.jsp and copy into $TOMCAT_HOME/webapps/openkm folder.

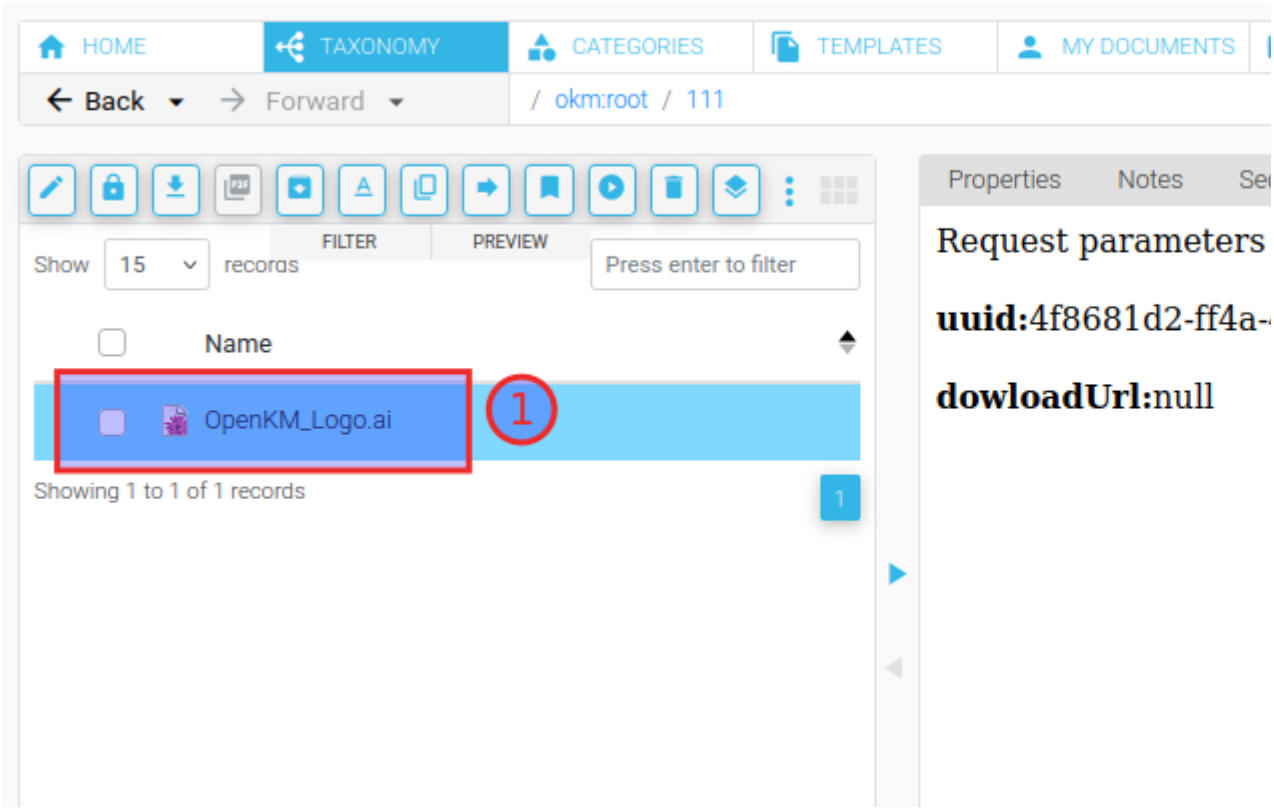
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
 <br />
 <b>dowloadUrl:</b><%=request.getParameter("downloadUrl")%><br />
</body>
</html>
```

Usage

- *Choose a document with one of the MIME type set in configuration parameters.*
- *Show preview tab. Will be shown the external URL based on document mime-type.*



External tab document extension

The External Tab Document feature helps you to show tab content in a properties tab based on external URL request when a node document is selected.

*This option is only available if you have enabled the **External tab document** feature.*

There are several parameters passed as request parameter in URL request:

- *The authentication **token** (encrypted)*
- *The document **uuid***
- *The **userId***

Configuration

Field / Property	Type	Description
extension.external.tab.document	**List**	*Each row is a couple of values with a tab content declaration, the format is:* ***name;url*** `name1;http://localhost:8180/openkm/test.jsp` `name2;http://www.openkm.com`
webservices.visible.properties	**List**	*To make accessible to all the users the values of the parameter "**extension.external.tab.document**", must be **added the value "extension.external.tab.document"** in the current list.* *If you forget to add the property name, only administrator users will be able to get the value so some extensions may produce errors for regular users.*

More information at Configuration parameters.

Sample

Configuration

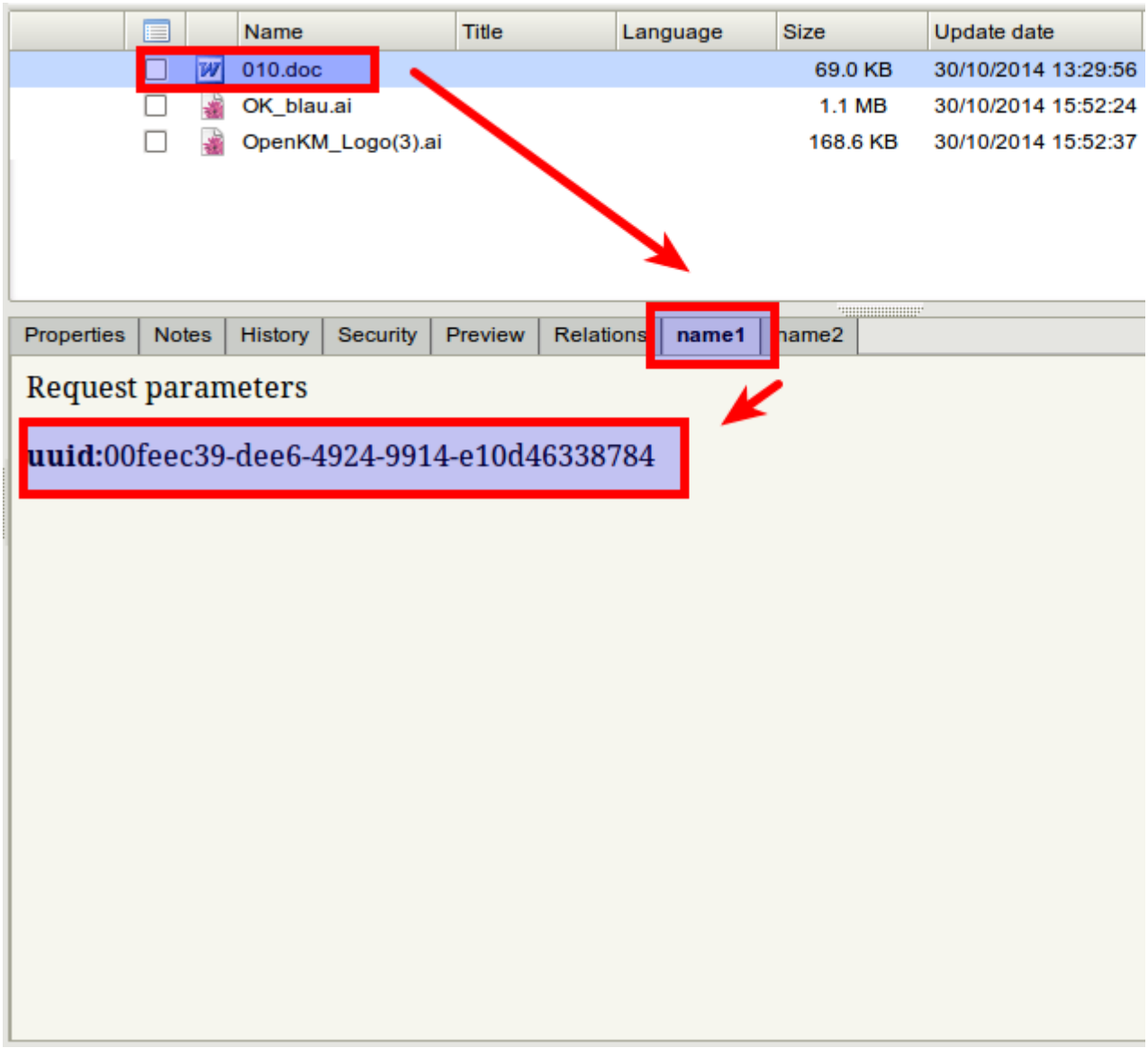
Create file named test.jsp and copy into $TOMCAT_HOME/webapps/openkm folder.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

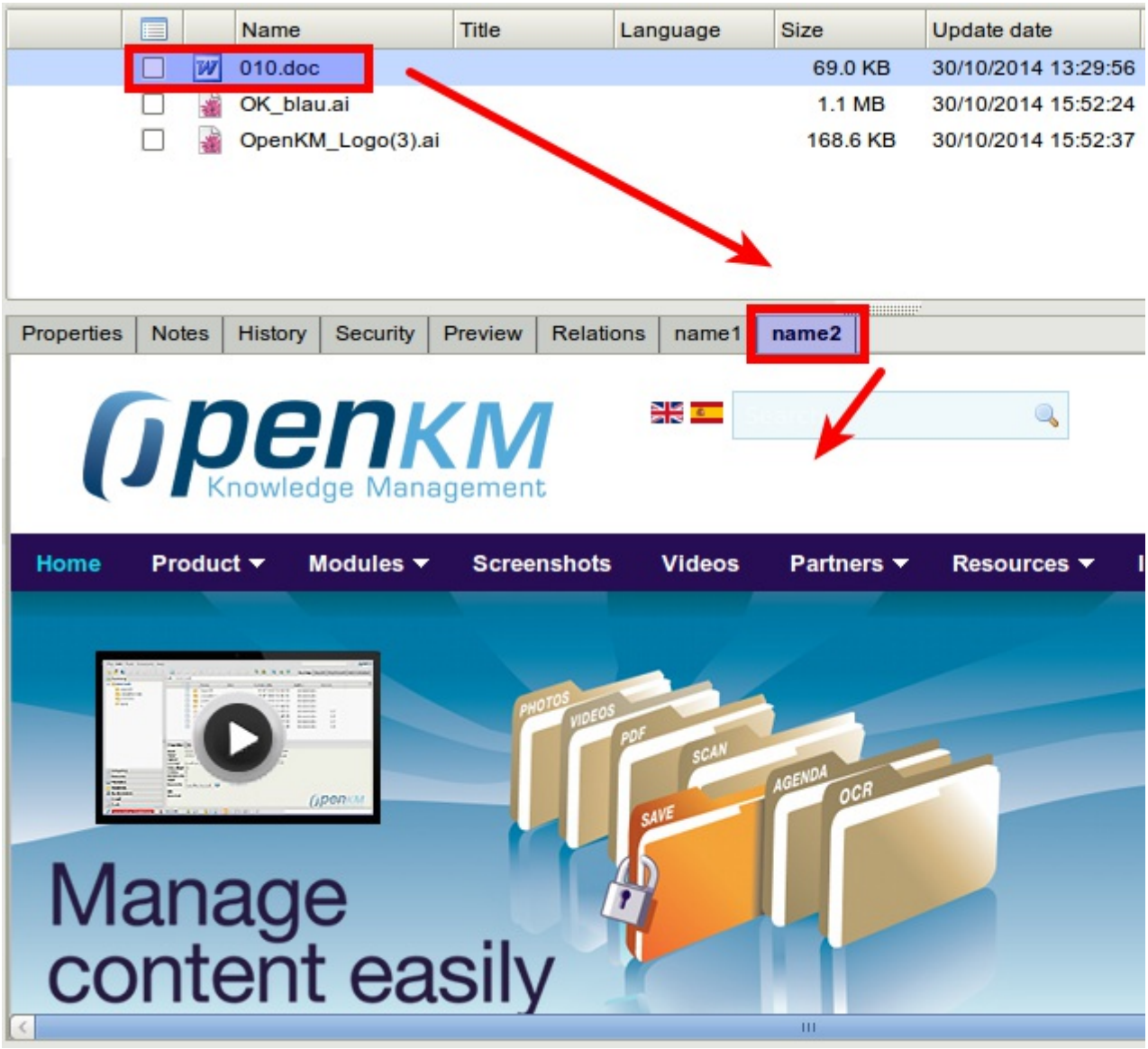
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a document***.
- ***Click on "name1" tab***.



- ***Click on "name2" tab.***


Name
Title
Language
Size
Update date
010.doc
69.0 KB
30/10/2014 13:29:56
OK_blau.ai
1.1 MB
30/10/2014 15:52:24
OpenKM_Logo(3).ai
168.6 KB
30/10/2014 15:52:37
Properties
Notes
History
Security
Preview
Relations
name1
name2
OpenKM
Knowledge Management
Home
Product
Modules
Screenshots
Videos
Partners
Resources
PHOTOS
VIDEOS
PDF
SCAN
AGENDA
OCR
SAVE
Manage
content easily

External tab folder extension

The External Tab Folder feature helps you to show tab content in properties tab based on external URL request when a node folder is selected.

*This option is only available if you have enabled the **External tab folder** feature.*

There are several parameters passed as request parameter in URL request:

- *The authentication **token** (encrypted)*
- *The folder **uuid***
- *The **userId***

Configuration

Field / Property	Type	Description
extension.external.tab.folder	**List**	*Each row is a couple of values with a tab content declaration, the format is:* ***name;url*** `name1;http://localhost:8180/openkm/test.jsp` `name2;http://www.openkm.com`
webservices.visible.properties	**List**	*To make accessible to all the users the values of the parameter "**extension.external.tab.folder**", must be **added the value "extension.external.tab.folder"** in the current list.* *If you forget to add the property name, only administrator users will be able to get the value so some extensions may produce errors for regular users.*

More information at Configuration parameters.

Sample

Configuration

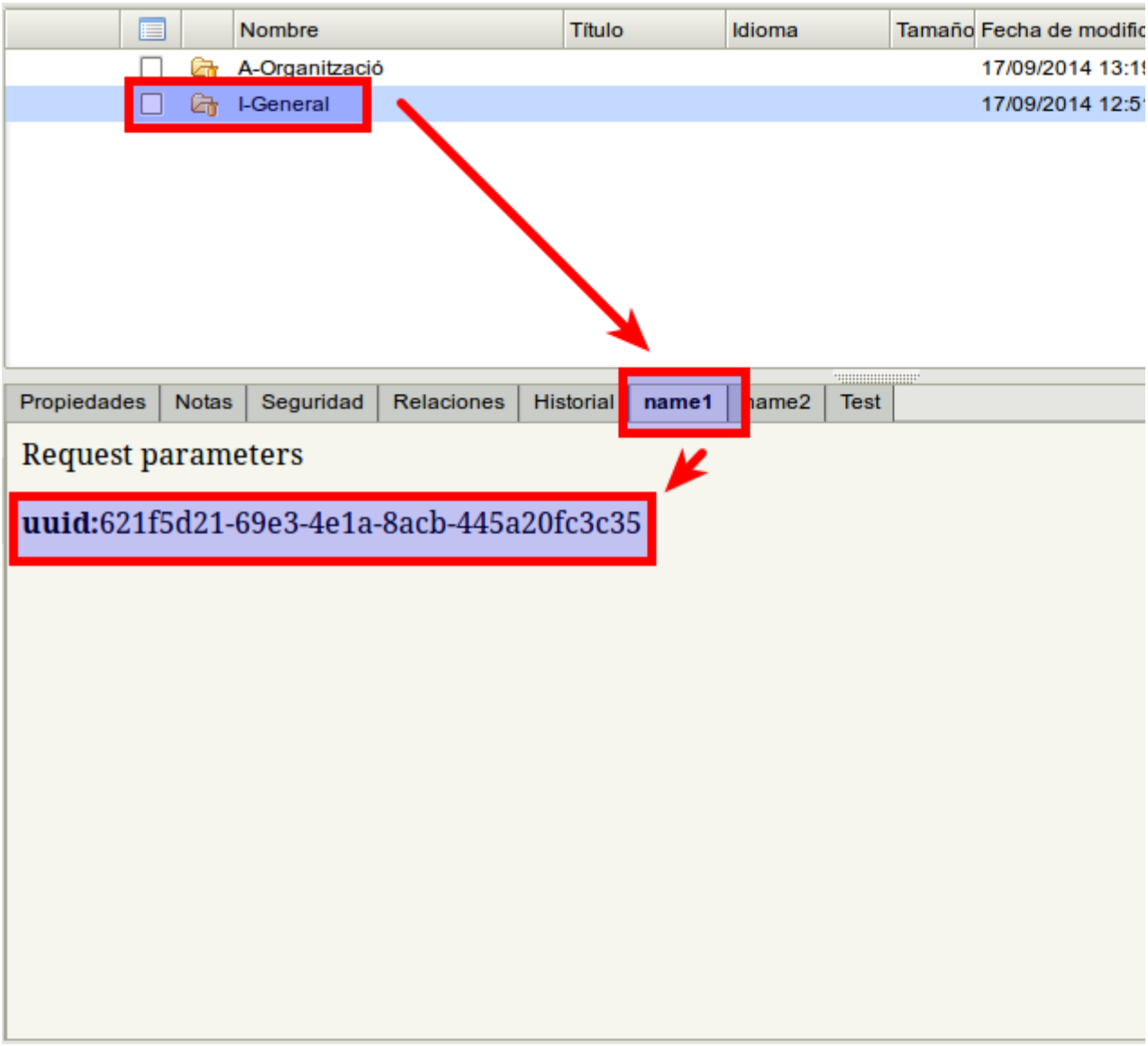
Create file named test.jsp and copy into $TOMCAT_HOME/webapps/openkm folder.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

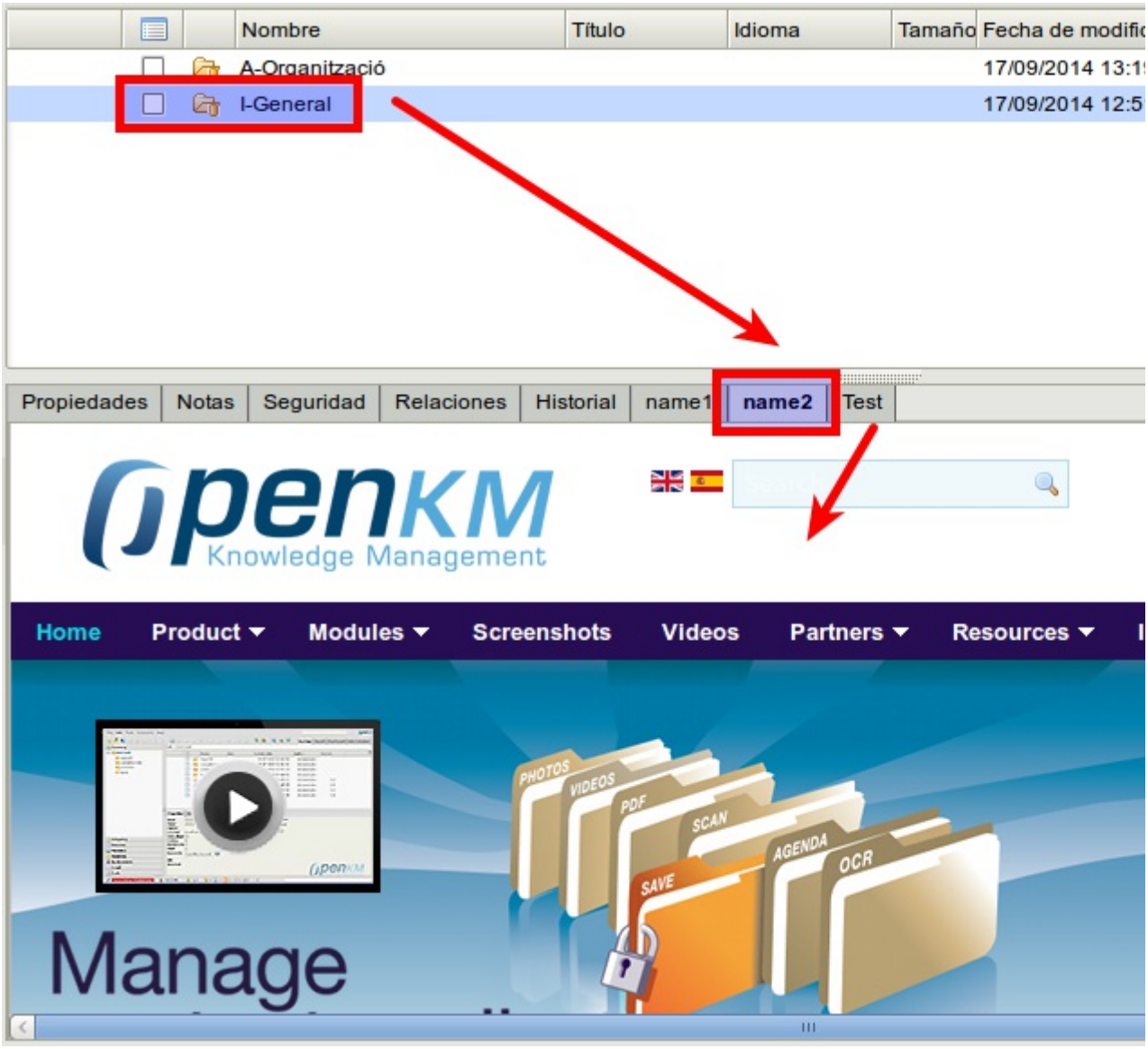
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a folder***.
- ***Click on "name1" tab***.



- ***Click on "name2" tab.***


Nombre
Título
Idioma
Tamaño
Fecha de modific
A-Organització
17/09/2014 13:1
I-General
17/09/2014 12:5
Propiedades
Notas
Seguridad
Relaciones
Historial
name1
name2
Test
OpenKM
Knowledge Management
Home
Product
Modules
Screenshots
Videos
Partners
Resources
PHOTOS
VIDEOS
PDF
SCAN
AGENDA
OCR
SAVE
Manage

External tab mail extension

The External Tab Mail feature helps you to show a tab content in the properties tab based on external URL request when a node mail is selected.

*This option is only available if you have enabled the **External tab mail** feature.*

There are several parameters passed as request parameter in URL request:

- *The authentication **token** (encrypted)*
- *The mail **uuid***
- *The **userId***

Configuration

Field / Property	Type	Description
extension.external.tab.mail	**List**	*Each row is a couple of values with a tab content declaration, the format is:* ***name;url*** `name1;http://localhost:8180/openkm/test.jsp` `name2;http://www.openkm.com`
webservices.visible.properties	**List**	*To make accessible to all the users the values of the parameter "**extension.external.tab.mail**", must be **added the value** "**extension.external.tab.mail**" in the current list.* *If you forget to add the property name, only administrator users will be able to get the value so some extensions may produce errors for regular users.*

More information at Configuration parameters.

Sample

Configuration

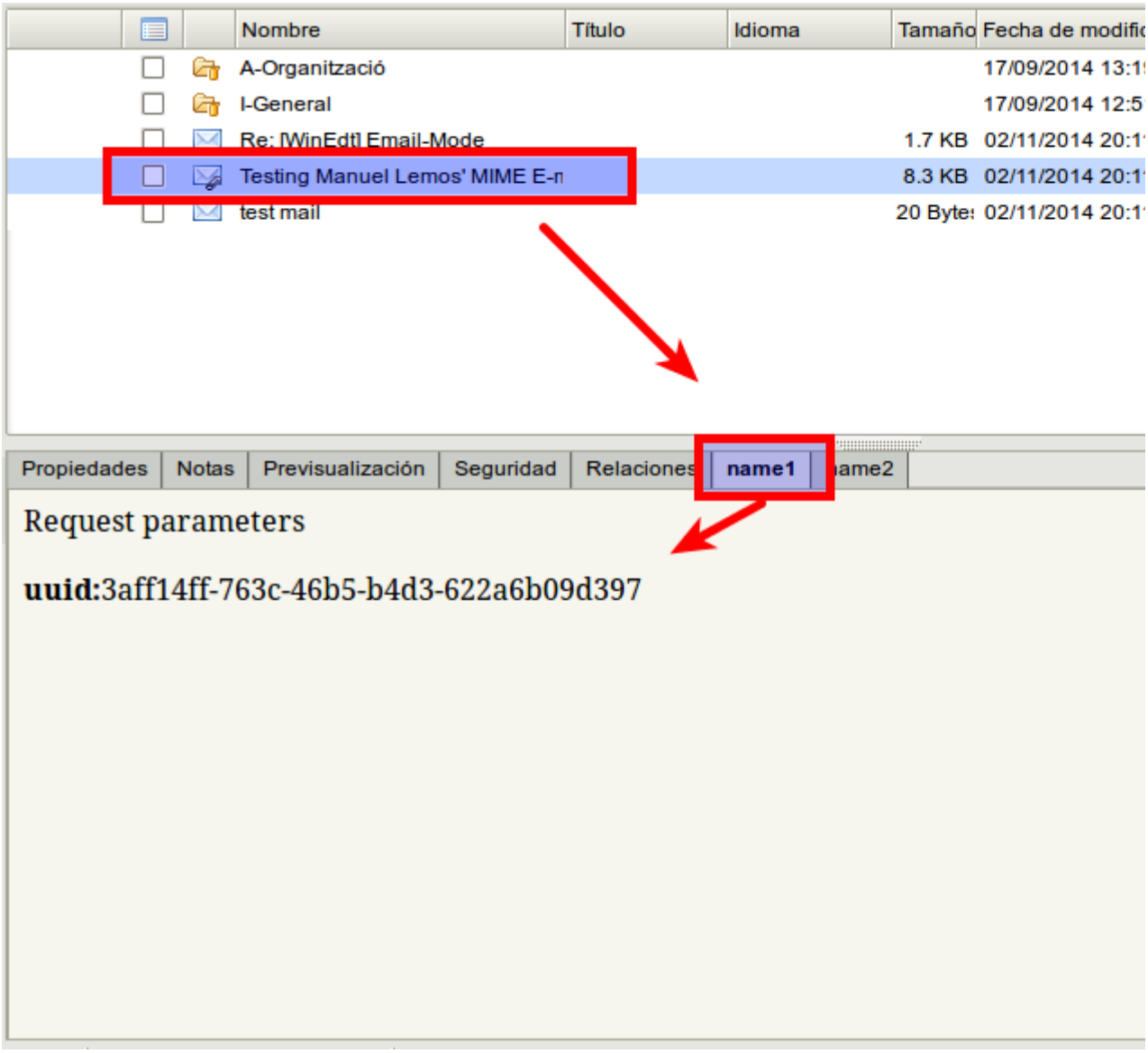
Create file named test.jsp and copy into $TOMCAT_HOME/webapps/openkm folder.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

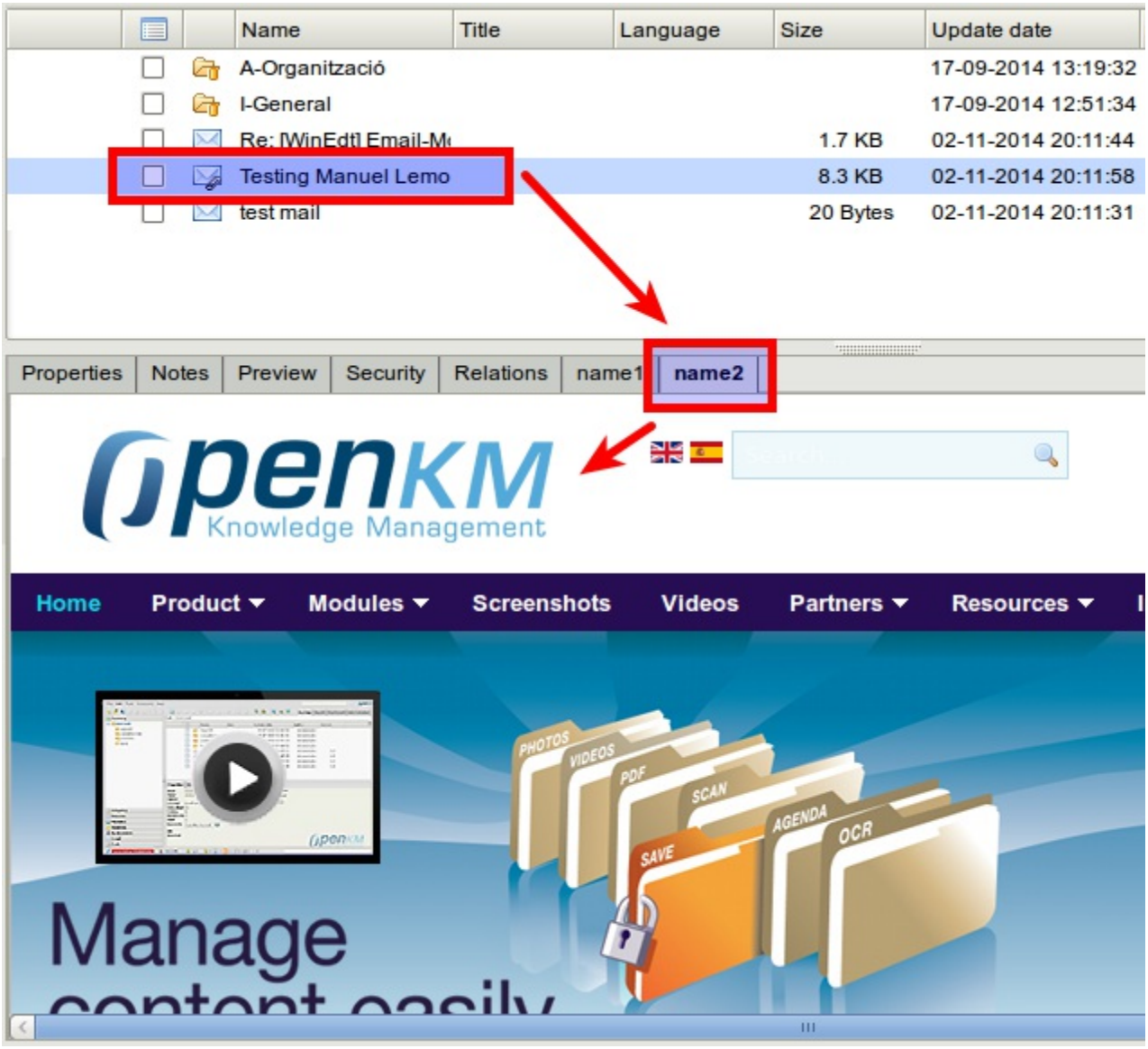
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a mail***.
- ***Click on "name1" tab***.



- ***Click on "name2" tab.***


Name
Title
Language
Size
Update date
A-Organització
17-09-2014 13:19:32
I-General
17-09-2014 12:51:34
Re: [WinEdt] Email-M
1.7 KB
02-11-2014 20:11:44
Testing Manuel Lemo
8.3 KB
02-11-2014 20:11:58
test mail
20 Bytes
02-11-2014 20:11:31
Properties
Notes
Preview
Security
Relations
name1
name2
OpenKM
Knowledge Management
Home
Product
Modules
Screenshots
Videos
Partners
Resources
PHOTOS
VIDEOS
PDF
SCAN
SAVE
AGENDA
OCR
Manage

External tab record extension

The External Tab Record feature helps you to show a tab content in properties tab based on external URL request when a node record is selected.

*This option is only available if you have enabled the **External tab record** feature.*

There are several parameters passed as request parameter in URL request:

- *The authentication **token** (encrypted)*
- *The record **uuid***
- *The **userId***

Configuration

Field / Property	Type	Description
extension.external.tab.record	**List**	*Each row is a couple of values with a tab content declaration, the format is:* ***name;url*** *name1;http://localhost:8180/openkm/test.jsp* *name2;http://www.openkm.com*
webservices.visible.properties	**List**	*To make accessible to all the users the values of the parameter "**extension.external.tab.record**", must be **added the value** "**extension.external.tab.record**" in the current list.* *If you forget to add the property name, only administrator users will be able to get the value so some extensions may produce errors for regular users.*

More information at Configuration parameters.

Sample

Configuration

Create file named test.jsp and copy into $TOMCAT_HOME/webapps/openkm folder.

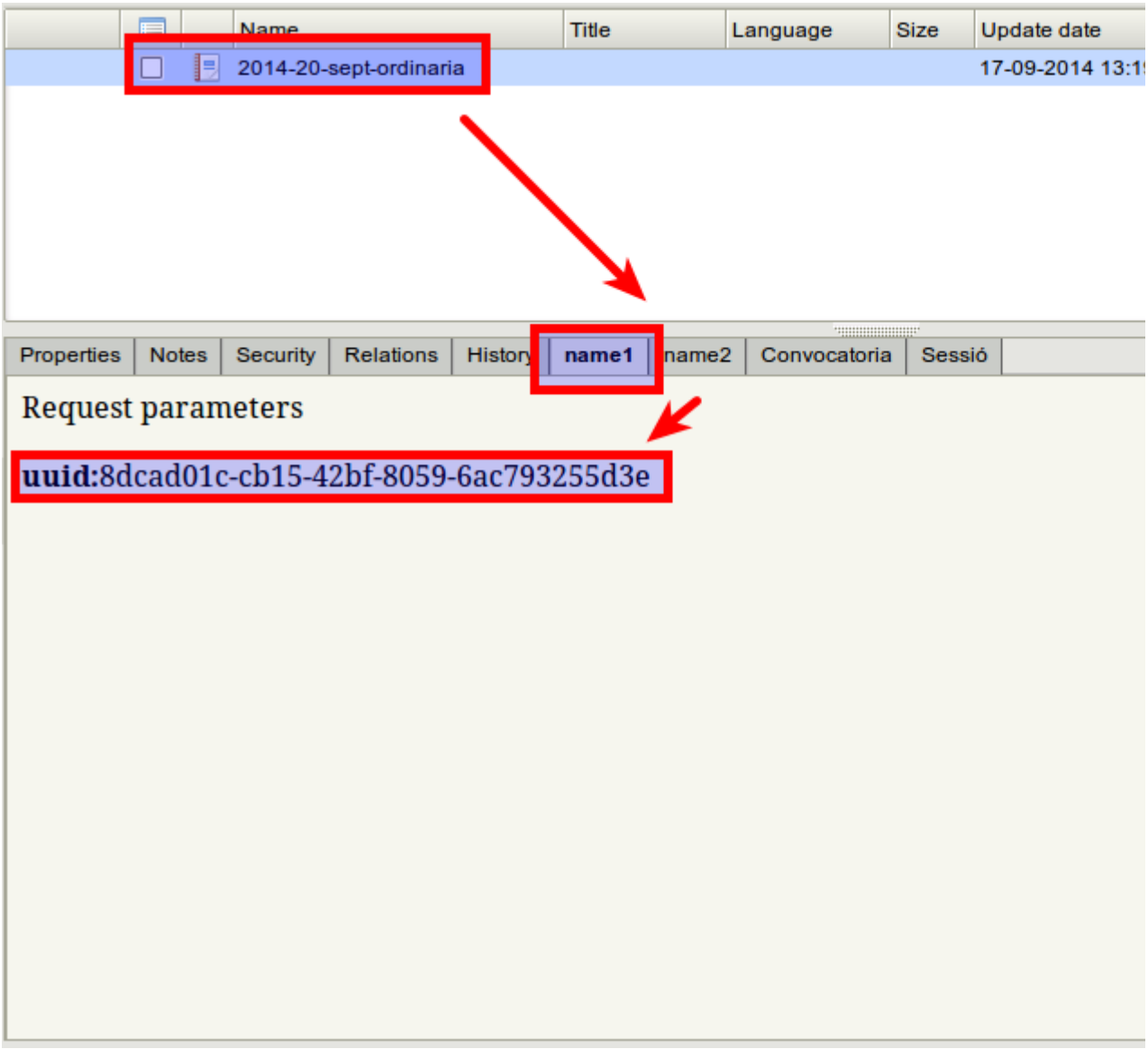
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

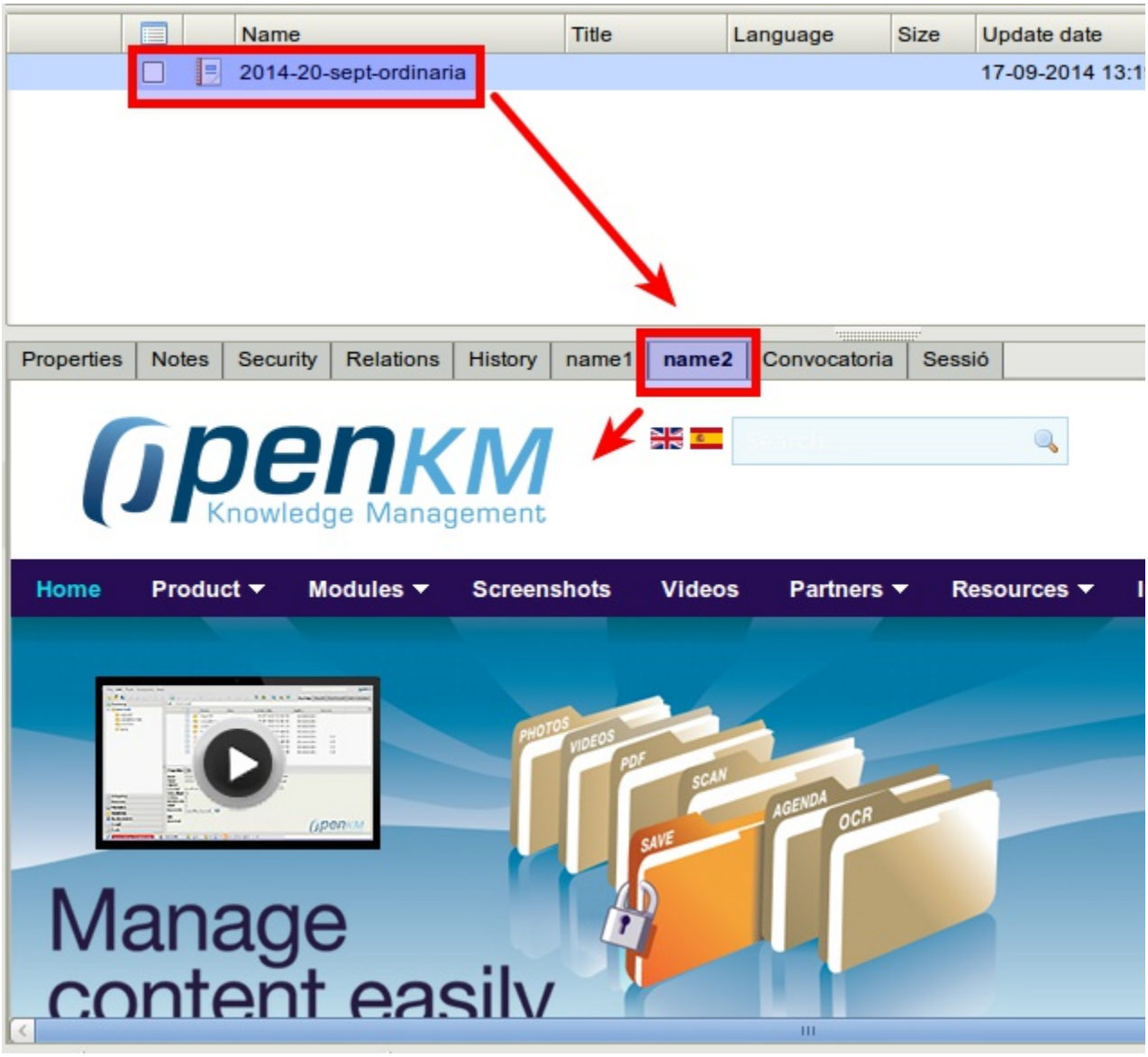
Usage

- ***Choose a record***.
- ***Click on "name1" tab***.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



- ***Click on "name2" tab.***


Name
Title
Language
Size
Update date
2014-20-sept-ordinaria
17-09-2014 13:1
Properties
Notes
Security
Relations
History
name1
name2
Convocatoria
Sessió
Home
Product
Modules
Screenshots
Videos
Partners
Resources
PHOTOS
VIDEOS
PDF
SCAN
SAVE
AGENDA
OCR
Manage
content easily

External widget dashboard extension

The External widget dashboard feature helps you to show card content in the dashboard panel based on external URL request.

This feature is available from the OpenKM version 8.1.10.

*This option is only available if you have enabled the **External dashboard widget** feature.*

There are several parameters passed as request parameters in URL request:

- *The authentication **token** (encrypted)*
- *The record **uuid***
- *The **userId***

Configuration

Field / Property	Type	Description
extension.external.widget.dashboard	**List**	*Each row is a couple of values with a tab content declaration, the format is: **label;url;height*** *Label 1;http://localhost:8180/openkm/test.jsp;370* *Label 2;http://www.openkm.com;240* *The default height of the card in the dashboard is 370 pixels.*
webservices.visible.properties	**List**	*To make accessible to all the users the values of the parameter "**extension.external.widget.dashboard**", must be **added the value "extension.external.widget.dashboard"** in the current list.* *If you forget to add the property name, only administrator users will be able to get the value so some extensions may produce errors for regular users.*

More information at Configuration parameters.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Sample

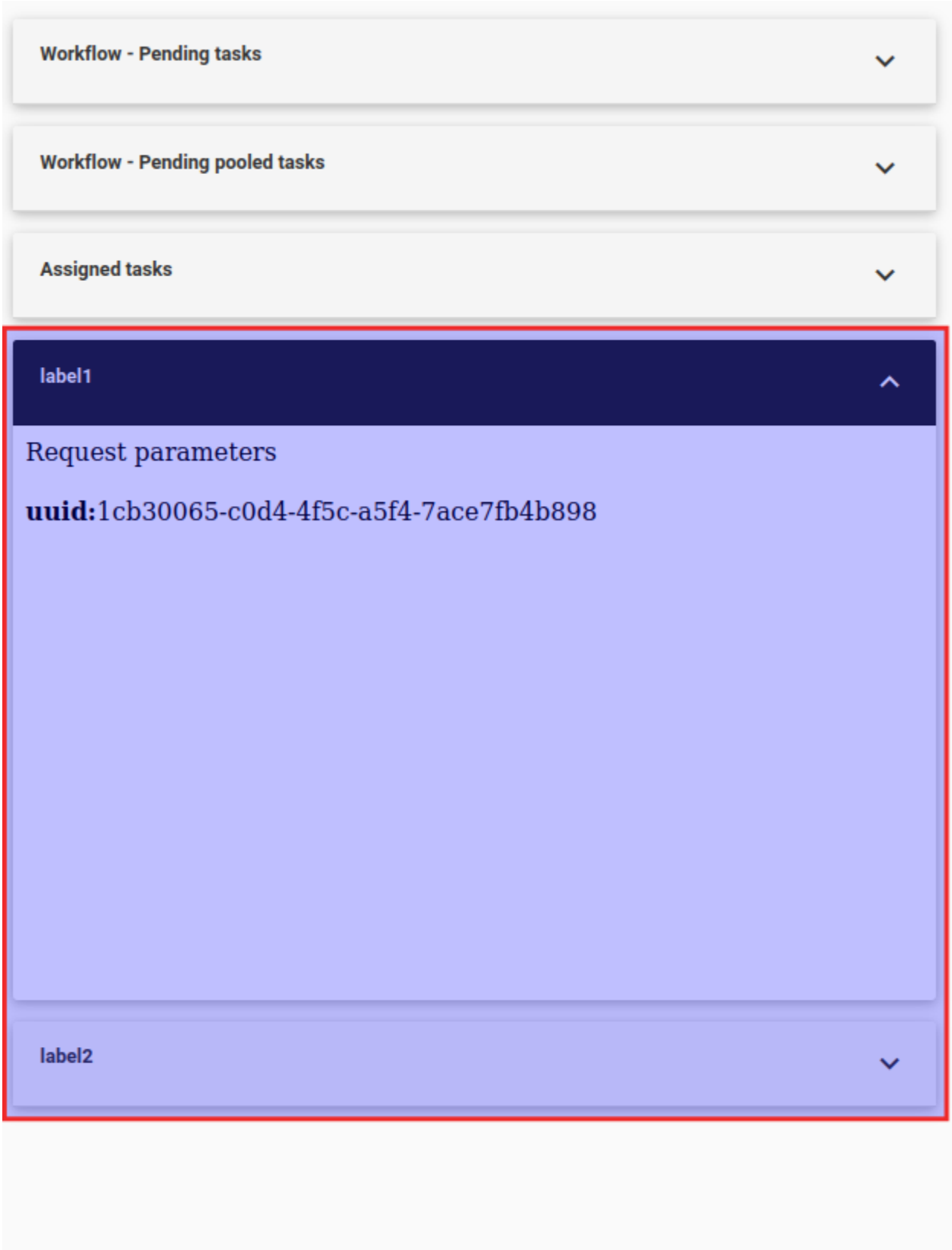
Configuration

Create file named test.jsp and copy into $TOMCAT_HOME/webapps/openkm folder.

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Show the dashboard panel***.



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

HTML editor extension

The application comes with an integrated HTML editor.

> *This option is only available if you have enabled the **HTML Editor** feature.*
>
> *You can edit several HTML documents at the same time and switch the editor between them for copying and pasting operations.*

Features

Feature	*Description*
Update	*Upload a new document version (check-in).* *When uploading a new version of a document, you can notify users about document changes. The application sends the notification by mail.*
Cancel edit	*Cancel document edition.*
Link document	*Add a link to a document.*
Link folder	*Add a link to a folder.*
Add image	*Add image.*
Download link	*Add download link.*
Code	*Add code section.*

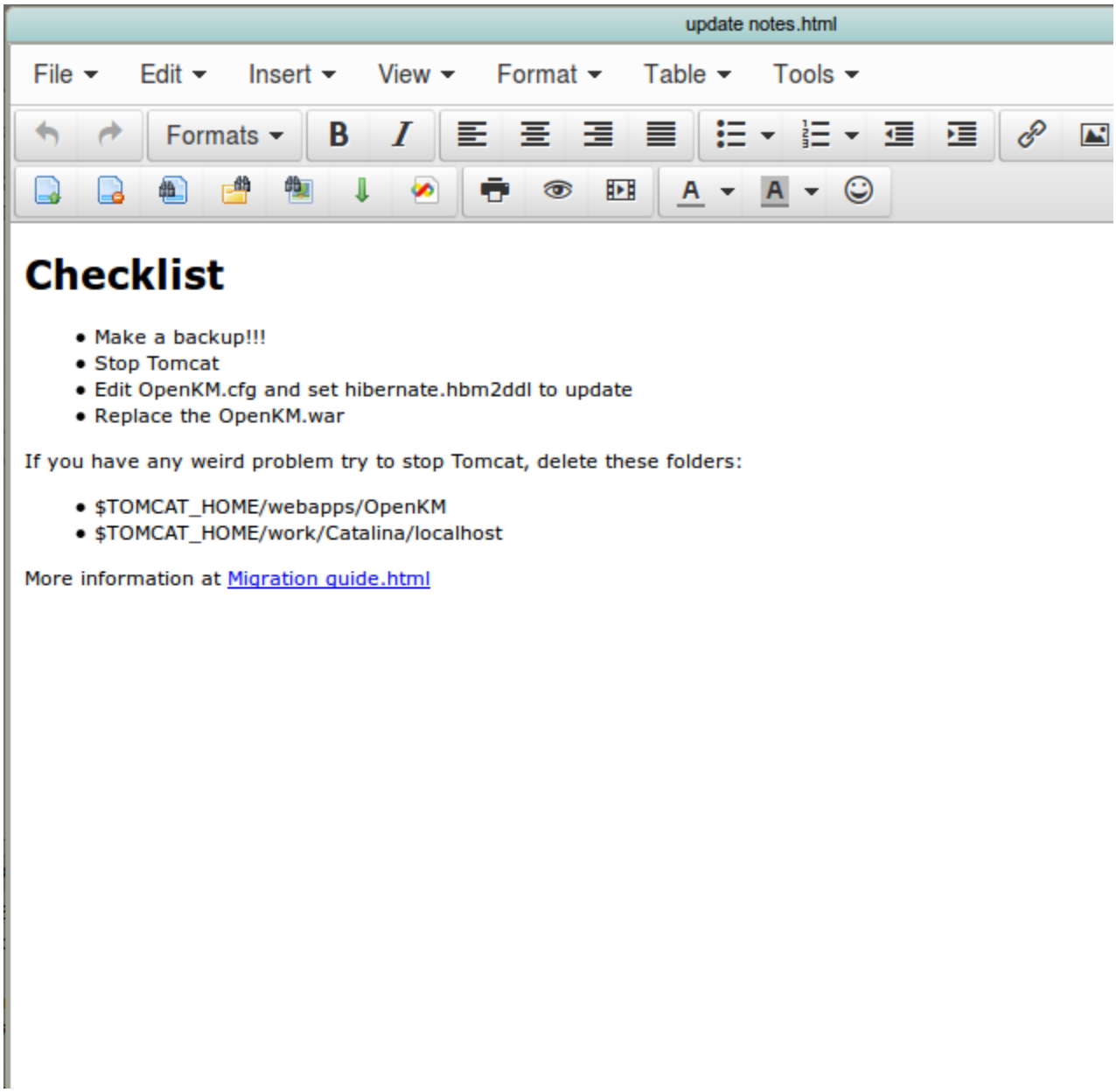
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

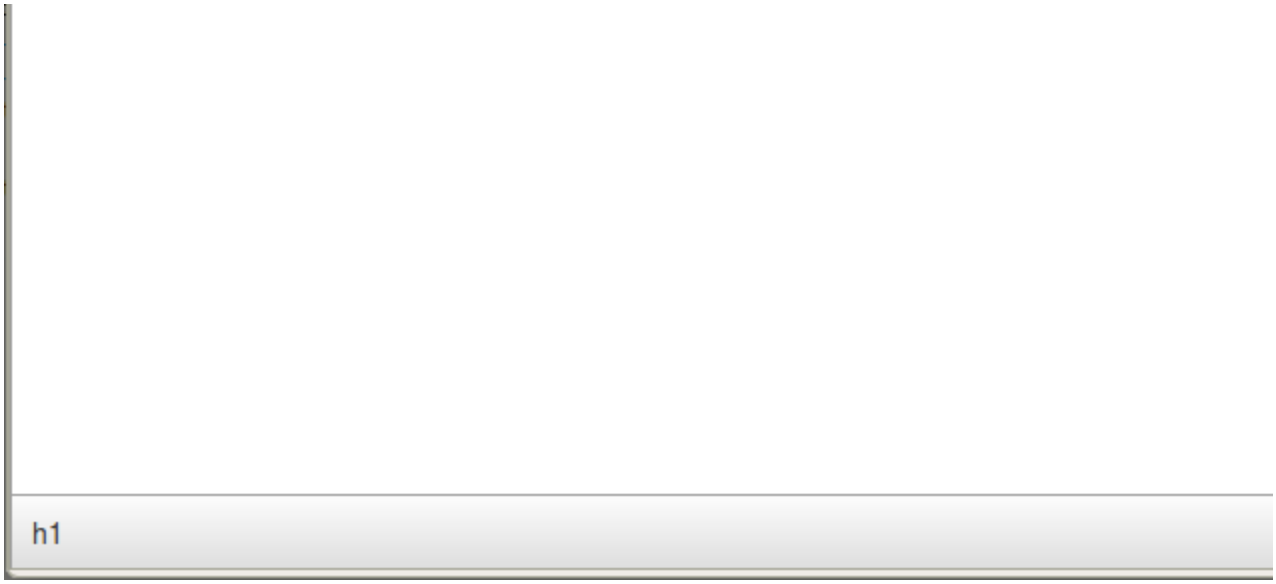
You can navigate across linked documents and folders or download documents from the Preview view.

Edit HTML

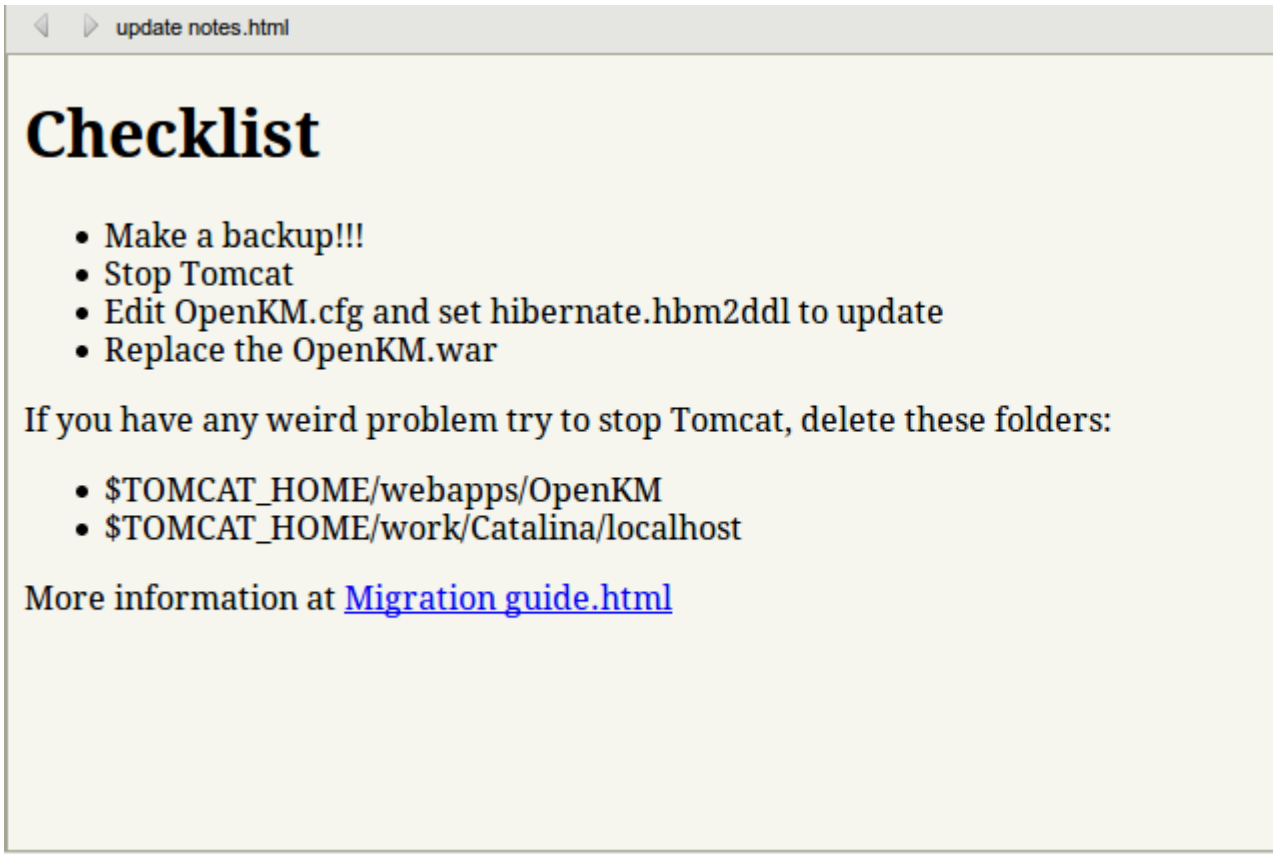
- ***Choose HTML document***.
- *Click on* ***Edit icon***.
- *Modify the content.*
- *Click on* ***Update icon***.

Editor panel





Preview panel



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Forum extension

The Forum extension helps you create discussion topics on nodes (documents, folders, mails, or records). Also, there is a general-purpose forum available from the Dashboard view.

*This option is only available if you have enabled the **Forum** feature.*

Only administrators can create, update, or delete forums from the Dashboard view.

Only administrators can view the "document discussions" forum in Dashboard view. This forum contains all the node topics.

Nobody can delete "document discussions" forum in Dashboard view.

Create new topic

- ***Choose a node*** *(document, folder, mail or record).*
- *Click on* ***New topic button***.
- ***Fill the form***.
- *Click on* ***Create button***.

Create reply

- ***Choose a topic on a node***.
- *Click on* ***Post reply button***.
- ***Fill the form***.
- *Click on* ***Create button***.

Update post

- ***Choose a topic on a node***.
- *Click on*  ***Edit icon***.
- ***Modify the form***.
- *Click on* ***Update button***.

Delete post

- ***Choose a topic on a node***.
- *Click on*  ***Delete icon***.
- *Click on* ***Accept button***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Stamp extension

The Stamp feature helps you to insert an image into a document.

*This option is only available if you have enabled the **Stamp** feature.*

You can stamp a document from image or text.

Document stamping

- ***Choose a document** (PDF or convertible to PDF).*
- *Click on*  ***Stamp icon**.*
- ***Choose the stamp type** from list.*
- *Click on **Stamp button**.*

When selected document is not PDF, it will be created a new PDF document with stamp into.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

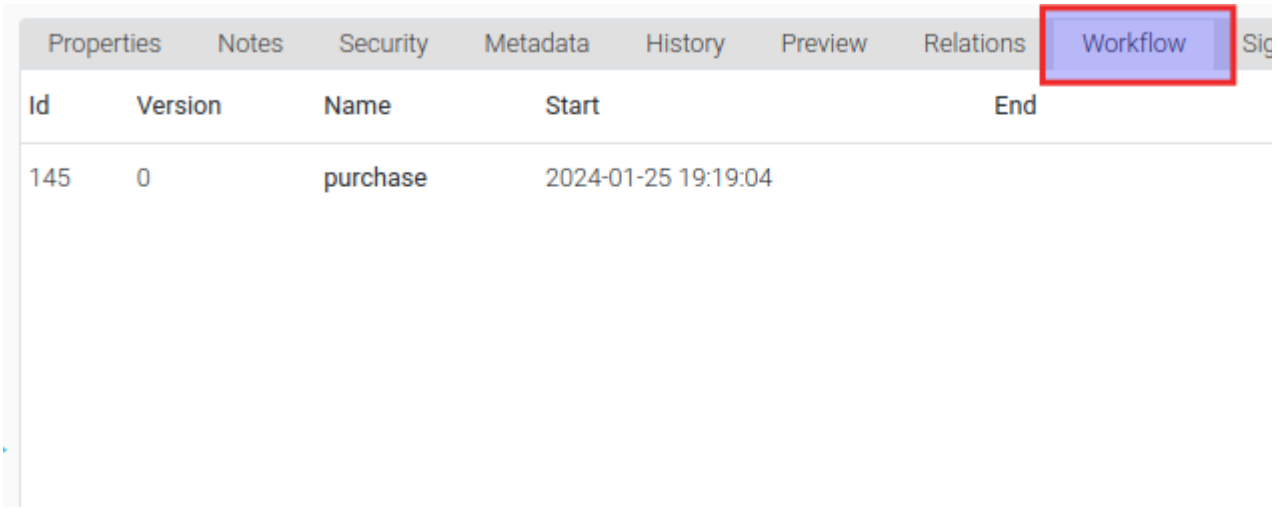
Workflow extension

The workflow extension helps to chronologically show the tasks performed and the graph of the status of the workflow.

Usage

Show workflows assigned to a node

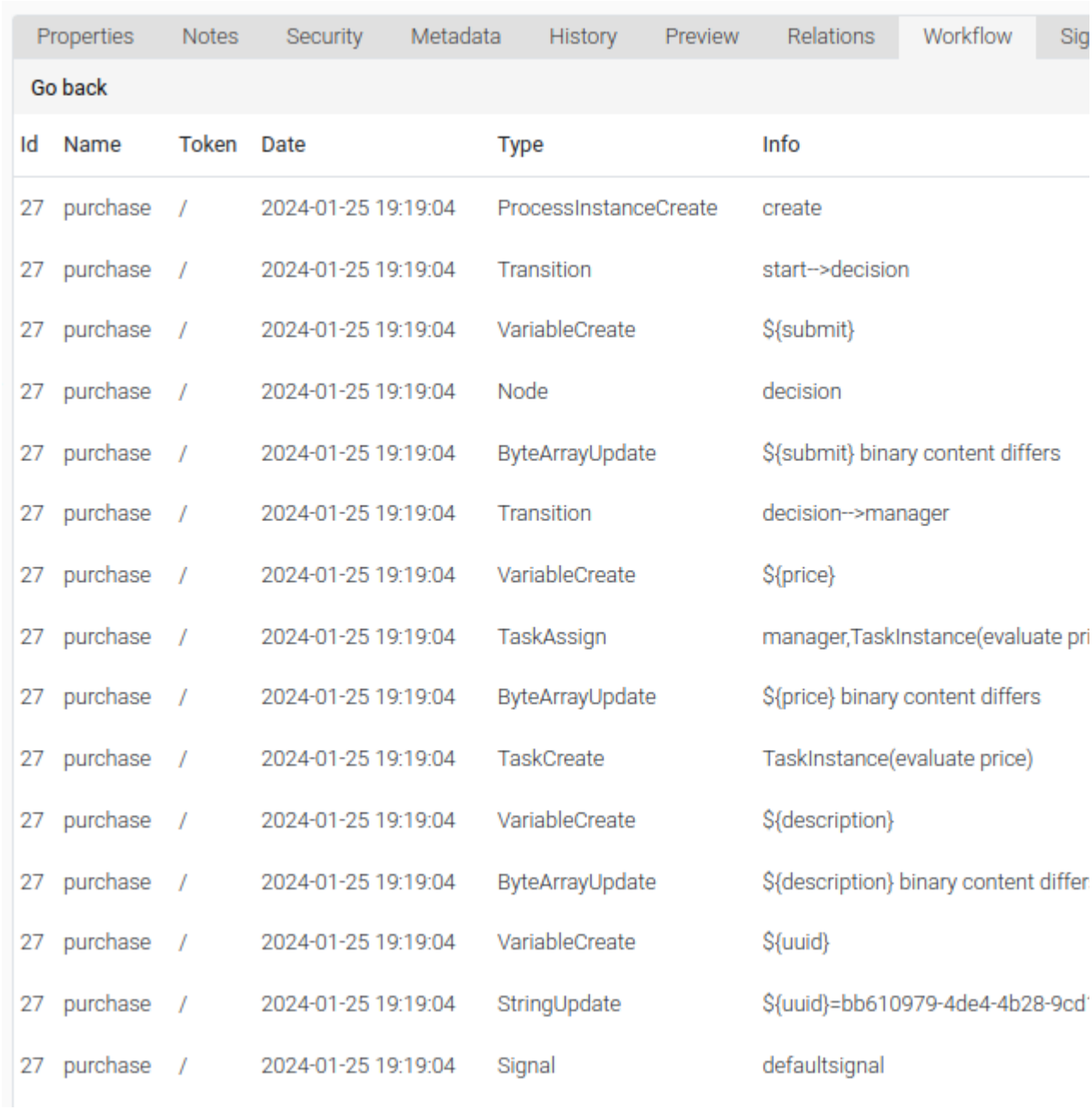
- *Choose a node.*
- *Select the Workflow tab.*



Log

- *Click on **Log** Show detailed information on the workflow.*
- *Click on **Go back**. To return to the list of assigned workflows.*

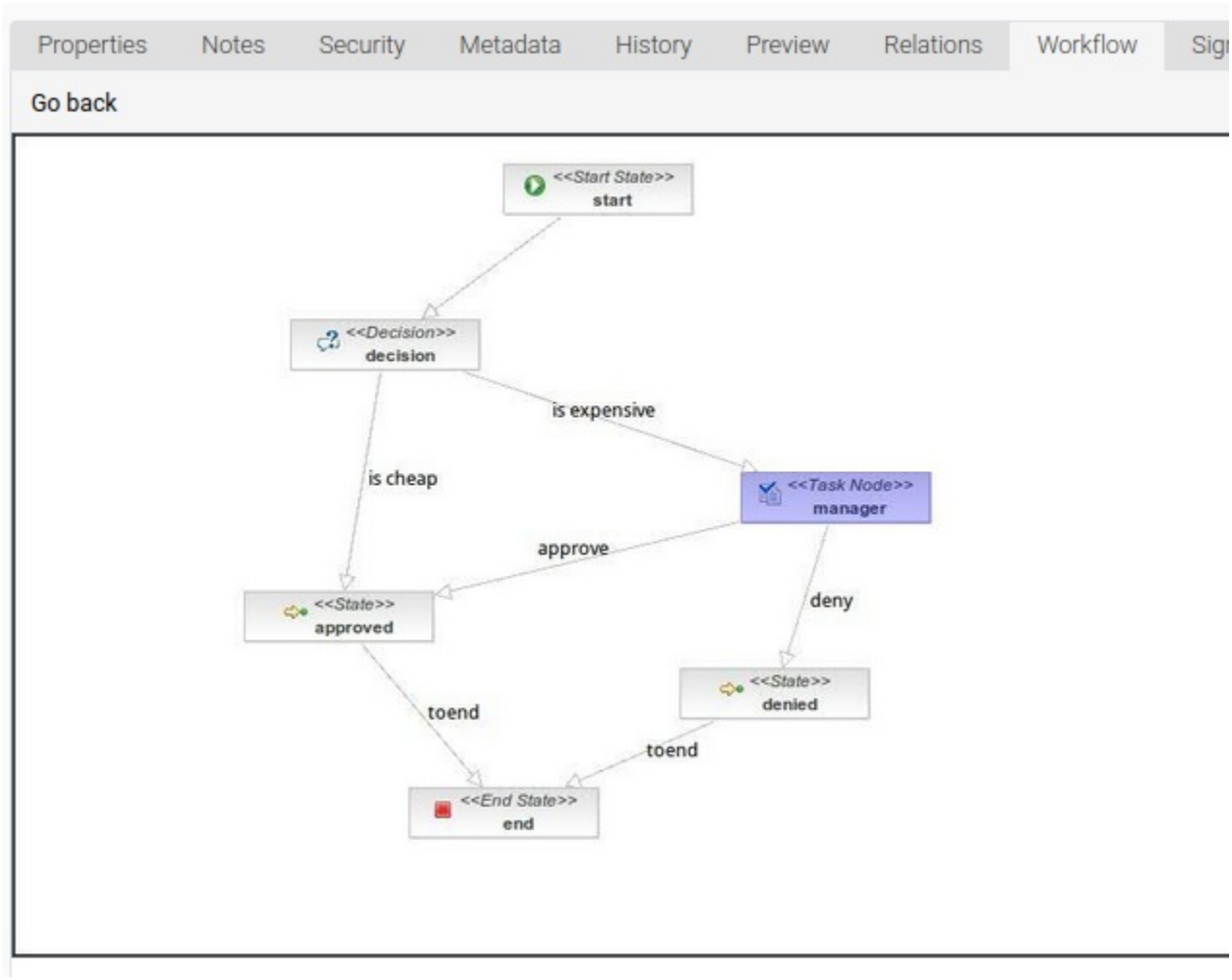
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Graph

- *Click on **Graph**. Show the workflow graph with the current workflow task highlighted.*
- *Click on **Go back**. To return to the list of assigned workflows.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Using OpenKM from external applications

There are a number of options available where you can access and use OpenKM from external applications.

Microsoft Office Add-in

You can create, edit and save files that are stored in the OpenKM repository, directly from Microsoft Office applications such as Microsoft Word, Microsoft Excel, Microsoft Power Point and Microsoft Outlook.

With OpenKM Microsoft Outlook Addin you can save mails to OpenKM repository from within Microsoft Outlook.

> *More information at Microsoft Addin.*

Digital Signature Client

Digital Signature Clients help you to quickly sign - electronic digital signature - documents to OpenKM.

> *More information at Digital Signature Tool .*

Other applications

1. *Encryption Tool*
2. *Bulk Uploader*
3. *Hot folder*
4. *Live Edit*
5. *Scan Station*
6. *Virtual printer*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Migration guide

*Download the latest "**Updater utility**" named OKMUpdater.jar*

- *Please, make a backup before any update.*
- *This utility needs access to **update.openkm.com** and **download.openkm.com** domains.*
- *In case you have customizations like workflows, automations, crontabs or other OpenKM modules keep in mind these customizations might need also a migration.*

Once downloaded, put at $TOMCAT_HOME directory and execute:

```
$ java -jar OKMUpdater.jar -l
```

Follow the wizard steps.

More information at Updater utility.

If you access the Internet using a proxy, just configure it when executing the application:

```
$ java -Dhttp.proxyHost=192.168.0.1 -Dhttp.proxyPort=3128 -jar OKMUpdater.jar -l
```

*You may also need **https.proxyHost** and **https.proxyPort** or even **http.proxyUser**, **http.proxyPassword**, **https.proxyUser** and **https.proxyPassword** parameters.*

It is recommended to clear browser cache (prevents "The response could not be deserialized" errors).

In case you have a Cluster configuration be careful because all the OpenKM instances share the same database, so once you have updated a node the others should be updated without executing the actions so they can be reviewed and executed manually. Ask for support if unsure about which actions should be executed in slave nodes.

```
$ java -jar OKMUpdater.jar -l -m
```

Troubleshooting

In case of having problems when you have configured HTTPS in Tomcat, follow these steps:

- *Change the Tomcat config back to HTTP port 8080*
- *Execute the OKMUpdate utility*
- *Change the Tomcat config to HTTPS port 8443 again (change only server.xml and web.xml)*

In early OpenKM installations, the upgrade process may need some previous steps. If you see an error like this one, you have this problem:



To be able to use the updater utility you have to follow these steps:

- *Download **the same version of your current OpenKM instance** from http://download.openkm.com.*
- *Stop OpenKM.*
- *Replace the $TOMCAT_HOME/webapps/openkm.war file with the downloaded one.*
- *Delete the $TOMCAT_HOME/webapps/openkm directory.*
- *Start OpenKM.*

Once OpenKM is running you can use the updater utility again.

If the problem persists, please open a support ticket at http://support.openkm.com and attach OKMUpdater.log and $TOMCAT_HOME/logs/catalina.log files.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Changelog

v8.1.16

- *Release: ?*
-

v8.1.15

- *Release: 2025-04-14*
- *Removed deprecated methods and outdated CIFS support to streamline the codebase.*
- *Resolved authentication problems with multiple sources in openkm.xml.*
- *Corrected metadata migration errors between MySQL and PostgreSQL.*
- *Corrected Lucene query error when searching filenames with spaces.*
- *Improved handling of text extraction and previews for Visio files.*
- *Fixed export issue when dealing with “Record” folders.*
- *Allowed metadata fields to be used in Lucene queries.*
- *Added provider support for local Ollama integration.*
- *Fixed UI plugin rendering issue for AI components.*
- *Added DTD for `okmflow` to support validation.*
- *Addressed plugin collision handling issues.*
- *Improved administration license info.*
- *Enabled execution of subreports.*
- *Improvements and fixes in KCenter:*
    - *Fixed pagination issues where sorting fields from plugins weren't being properly transmitted.*
    - *Added metadata field support to Lucene queries for more powerful search capabilities.*
    - *Corrected display problems in custom pagination where values weren't showing.*
    - *Extended video preview expiration time to enable longer viewing sessions.*
    - *Resolved an issue where AI plugins weren't refreshing the wizard properly*
    - *Fixed Lucene query button activation when search plugins are enabled.*
    - *Replaced the current video previewer with a more robust solution.*
    - *Added protection to prevent style removal during HTML code editing.*
    - *Implemented a workflow diagram preview for the new workflow engine.*
    - *Implemented wizard functionality for improved user guidance.*
- *Improvements and fixes in mobile:*
    - *AI plugin actions and copy functionality*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Document AI Plugin Implementation*
- *TinyMCE editor improvements*

v8.1.14

- *Release: 2025-02-17*
- *Now you can configure extra tab workspace extension by tenant.*
- *New integration with ChatGPT, Gemini and DeepSeek.*
- *New CSV report from SQL.*
- *The shard has been added to the profile configuration.*
- *Improved repository checker behavior.*
- *Improved repository import behavior.*
- *Improved rebuild indexes behavior.*
- *Fixed issue with user password validation.*
- *Fixed issue with recursive mail import.*
- *Fixed several security issues.*
- *Fixed issues with sharding.*
- *The issue where the LOGIN_FAILED event was being recorded twice has been fixed.*
- *The configuration parameter chat.gpt.text.extraction.url has been removed.*
- *Improvements and fixes in KCenter:*
  - *From the workflows tab, a link to the administration URL is included.*
  - *UI settings are now stored within the application.*
  - *Fixed issue with image URL in HTML editor.*
  - *Fixed bug in websockets management.*
  - *Sorting of fields is now allowed in pagination plugins.*
- *Improvements and fixes in mobile:*
  - *Now you can compact versions and view metadata of a specific version.*
  - *Version management is now allowed in history panel.*

v8.1.13

- *Release: 2024-12-05*
- *Now automation validation can be negated.*
- *Fixed issue with OCRZonal integration.*
- *Fixed issue with some MSG mail import.*
- *Fixed issue with POP3 mail server check.*
- *Fixed issue with tenant favicon modification.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Fixed issue with node record export and import.*
- *Fixed issue with PDF optimization in stamp REST API.*
- *Fixed issue with PdfNewTextExtractor and some document images.*
- *PDF text extractors now try to preserve document structure.*
- *New improved method for document thumbnail generation.*
- *New "type" attribute in TextArea (choose text or html)*
- *More improvements in user and role manement.*
- *Authentication tokens can now be created for applications.*
- *Improved method to load DTD.*
- *Improvements and fixes in KCenter:*
    - *New message to encourage users to enable two-factor authentication.*
    - *New HTML type in TextArea form fields.*
    - *Improved login process performance.*
    - *Fixed issue in Stamp page navigation.*
    - *Fixed issue with "check all" button in PDF Tools.*
- *Improvements and fixes in mobile:*
    - *Improvements in search menu.*
    - *Many-to-many relationship management is now supported.*
    - *Node security management is now supported.*
    - *The design of notes has been improved.*
    - *The appearance of thumbnails on the details screen has been improved.*
    - *Managing the title and description in the details view is now possible.*
    - *It is now possible to manage categories in the details view.*
    - *It is now possible to switch between tenants.*
    - *The creation and editing of tasks is allowed.*
    - *Trash management is now enabled.*
    - *Fixes an error when displaying the date range values in a stored query.*

v8.1.12

- *Release: 2024-09-09*
- *Simplified LDAP and AD integration (See **IMPORTANT CHANGES**)*
- *Fixed issue with Collabora and MS Office documents.*
- *Fixed issue with 2FA and AD integration.*
- *Improvements and fixes in KCenter:*
    - *New postpone a wizard option.*

- *Added "optimize PDF" filemanager option.*
- *Added "send" field in email properties tab,*
- *New option to jump directly to a node using its "UUID".*
- *Fix: Stored searches sometimes did not load correctly.*
- *Fix: The tab link now redirects to the right URL.*
- *Fix: The "order" field was not considered when creating a filtering condition in an email account.*
- *Fix: Error in generating the authentication token shared with the dashboard widget extension.*

- *Improvements and fixes in mobile:*
  - *New bulk actions menu imeplemeted.*
  - *QR code generation has been added in the node properties.*
  - *Individual link to access the details of a task directly.*
  - *In the details view, several tabs have been integrated to minimize menu size.*
  - *Improvements in application navigation.*
  - *Notification icon added in the top right corner.*
  - *The task calendar has been implemented.*
  - *External preview has been implemented.*
  - *Fix: An error when canceling photo insertion has been fixed.*
  - *Fix: An error when exiting the application has been fixed.*

IMPORTANT CHANGES

- *The configuration parameter **principal.adapter** no longer exists.*



*We recommend that **users with LDAP integration** or **custom authentication open a support ticket** on the customer support website **before upgrading** to this version.*

- *The configuration parameter **notification.url** must be updated.*

For example the the old URL with value:

Field / Property	*Type*	*Description*
notification.url	***String***	*http://localhost/openkm/kcenter*

Should be updated to:

Field / Property	*Type*	*Description*
notification.url	***String***	*http://localhost/openkm/open*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

v8.1.11

- *Release: 2024-06-18*
- *Fixed issue with Greek reports.*
- *Fixed issue with Arabic mail preview.*
- *Fixed issues with Collabora online editor.*
- *Fixed issue with MySQL recursive queries.*
- *Fixed issue with PostgreSQL recursive queries.*
- *Improved PostgreSQL compatibility.*
- *New experimental ChatGPT text extraction.*
- *Swagger annotation in more REST methods.*
- *Now you can search by empty metadata fields.*
- *Several improvements in document conversion.*
- *Check for overlapping conversion and extraction plugins.*
- *Updated several dependencies to avoid security issues.*

IMPORTANT CHANGES

- *The hash has been removed from the kcenter URL. The* ***notification.url*** *configuration parameter must be updated removing the hash.*

For example the the old URL with value:

Field / Property	*Type*	*Description*
notification.url	***String***	*http://localhost/openkm/kcenter/#*

Should be updated to:

Field / Property	*Type*	*Description*
notification.url	***String***	*http://localhost/openkm/kcenter*

- *The WebSocket URL has been changed. If you have a reverse proxy in front of the application, you should update the path removing the "frontend".*

For example:

```
ProxyPass /openkm/frontend/webSocket ws://127.0.0.1:8080/openkm/frontend/webSocket
```

It should be updated to

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ProxyPass /openkm/webSocket ws://127.0.0.1:8080/openkm/webSocket

- *OKMSearch*
  - *Deleted method*
    - *List<QueryResult>* ***find****(String token, QueryParams params, String sortField, boolean sortReverse, String propertiesPlugin)*
    - *ResultSet* ***findPaginated****(String token, QueryParams params, int offset, int limit, String propertiesPlugin)*
    - *ResultSet* ***findPaginated****(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int limit)*
    - *ResultSet* ***findPaginated****(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)*
    - *List<QueryResult>* ***findWithMetadata****(String token, QueryParams params, String sortField, boolean sortReverse, String propertiesPlugin, List<String> groups)*
    - *ResultSet* ***findWithMetadataPaginated****(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin, List<String> groups)*
    - *SimpleNodeBaseResultSet* ***findSimpleNodeBasePaginated****(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int limit)*
- *Improvements and fixes in KCenter:*
  - *New feature:* ***change in the format of the kcenter URL. The hashtag has been removed from the URL*** *.*
  - *New feature: ask for confirmation in the delete saved search action.*
  - *New feature: the fields "content" and "to" have been added to the search by email panel.*
  - *New feature: added field "send date" in the properties tab of the nodes of type email.*
  - *Improvement: the textarea is now shown as an input in the search panel.*
  - *Fixed issue in the parameters shared in the dashboard external widget.*
  - *Fixed issue when creating a new task with nodes. The nodes now are assigned to the task.*
  - *Fixed issue, now the new password is checked with the password validation pattern.*
  - *Fixed the issue with the custom styles not loading in the UI.*
  - *Fixed issue with email ordering in the email panel view. Now it uses the same criteria as in the browser panel.*
  - *Fixed error when adding a filter in an email account.*
  - *Fixed issue when importing emails in the email account panel.*
  - *Fixed issue with the "add keyword at the wizard" option at the top menu.*
- *Improvements and fixes in mobile:*
  - *New refined search by a range of dates.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

v8.1.10

- *Release: 2024-04-18*
- *Fixed issue with internal metadata map modifications.*
- *Fixed issue with administration admin permissions.*
- *Fixed issue with the user who executes the report.*
- *Fixed converte URL security configuration.*
- *Fixed issue with PathContains validation.*
- *The actuator endpoint was enabled again.*
- *New start workflow automation.*
- *Improved swagger REST API documentation.*
- *Improvements and fixes in KCenter:*
  - *New feature: preview a document in the right panel of the search view.*
  - *New feature: prevent searches by folder with high depth.*
  - *New feature: show versions in the history panel.*
  - *New feature: plugins in the dashboard.*
  - *New option in the saved searches: Now you can execute or only load the saved search.*
  - *New refined search by a range of dates. Now you can choose hours and minutes.*
  - *A new feature in the workflow to unassign a pooled task.*
  - *Fixed issue with WebSockets messages sent to the UI.*
  - *Fixed missing page range in the stamp image.*
  - *Fixed issue with link color task description.*
- *Improvements and fixes in mobile:*
  - *New feature: create from template.*

v8.1.9

- *Release: 2024-03-20*
- *Several improvements in database migration from OpenKM 7.1 and 6.4.*
- *Fixed issue when uploading a MSG archive using the REST API.*
- *Fixed issue with document version comparison.*
- *Fixed issue with plain text content extraction.*
- *Fixed issue with websocket user messages.*
- *Crontabs can be enabled or disabled from listing.*
- *Reworked many REST API methods.*
- *Improved Swagger REST documentation.*
- *Improvements and fixes in KCenter:*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Fixed UI issue in preview from massive actions: when the user chooses a tab on the left side, it continues to be chosen when previewing a new document.*
- *Fixed UI issue the search panel: when the user has chosen metadata fields for filtering was not able to use them again after clean.*
- *Major change in many REST API methods (mainly PUT and POST types).*
- *Node link is opened in a new tab.*

- *Improvements and fixes in mobile:*
  - *Major change in many REST API methods (mainly PUT and POST types).*
  - *Updated sdk4nodejs dependency to v2.0.0.*

v8.1.8

- *Release: 2024-02-21*
- *Fixed issue with database connection leak and metadata.*
- *Fixed issue with database connection leak and reports.*
- *Fixed issue with huge amount of imported mails.*
- *Fixed issue with node workflow performance.*
- *Fixed issue with report in non-default tenant.*
- *Fixed issue with CSV document preview.*
- *Fixed issue with activity log date format.*
- *Fixed issue with PDF preview flickering.*
- *Fixed issue with node search result sort.*
- *Fixed issue with password validator.*
- *Improved uncontrolled error handling.*
- *Improved search by literal text.*
- *Improvements and fixes in KCenter:*
  - *Fixed minor defect in UI. It was not using the profile value to show or hide encryption.*
  - *Fixed minor defect in templates panel. A scroll has been added for large forms.*
  - *Fixed an issue with date pickers which raises a parseToDate error.*

v8.1.7

- *Release: 2024-01-16*
- *Now a workflow can be assigned to several tenants.*
- *Added "barcode" support in StampDocument automation action.*
- *Fixed issue with metadata indexation in non-default tenant.*
- *Fixed issue with mail recipients address format.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Fixed issue with node workflow extension.*
- *Improvements and fixes in KCenter:*
    - *Now the user can set the text to be stamped in text stamp.*
    - *Now you can propagate security and metadata when creating a folder hierarchy from the template panel.*
    - *New "add category" in wizard panel.*
    - *New "add keyword" in wizard panel.*
    - *New "add keyword" and "add category" in the wizard from the browser panel.*
- *Improvements and fixes in mobile:*
    - *Added public URL.*
    - *Added password type in input workflow field.*
    - *Added subscription and indexable features in property panel.*
    - *Fixed issue with logo image: now the image will not be rescaled and will be displayed properly.*
    - *Fixed issue with Arabic and vertical panels.*

v8.1.6

- *Release: 2023-12-13*
- *Fix: Removed obsolete validatorClassName attribute in workflow-form*
- *Fix: Public user should not be disabled in AD synchronization.*
- *Fix: Issue error message with wrong password.*
- *Fix: Issue with one-time download token.*
- *Fix: Issue with mail MSG file import.*
- *Fix: NPE in mail preview when token was expired.*
- *Fix: NPE when sorting user names.*
- *Fix: Security issue with 2FA in logout.*
- *Fix: Security issue with error page.*
- *Improvement: a new configuration property was added to choose the PDF template date pattern.*
- *Improvement: a new administration utility was added to purge node versions.*
- *Improvement: several improvements in document conversion.*
- *Improvements in migration from OpenKM 6.4.*
- *Improvements in migration from OpenKM 7.1.*
- *Improvements in text extraction logging.*
- *Improvements and fixes in KCenter:*
    - *Improvement in thumbnails, folders, and records which can be browsed by double click.*
    - *Fixed a bug in wizard:The clean button should not be shown when the cancel option is disabled.*
    - *Fixed a bug in email forwarding.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Fixed a bug with 2FA logout.*

v8.1.5

- *Release: 2023-11-15*
- *Fixed issue with LowerCaseFieldBridge inconsistency.*
- *Fixed issue with PDF templates and metadata using vpanel and hpanel.*
- *Fixed issue related to recursive movement of node records.*
- *Fixed issue with custom actions and user profile.*
- *Fixed issue with arabic filtering in administration languages.*
- *Fixed issue with logout event and activity log.*
- *Improvements in automation execution in fileplan.*
- *Improvements in automation execution logging.*
- *Improved document stamp precision.*
- *Improvements and fixes in KCenter:*
    - *New feature: copy public URL link.*
    - *New feature: insert pages in PDF.*
    - *New feature: password metadata field type.*
    - *New feature: Custom action can be shown in the horizontal menu.*
    - *New feature: Custom action based on scripting.*
    - *New feature: add metadata in wizard.*
    - *New feature: add a workflow in wizard.*
    - *New feature: preview email attachments.*
    - *Fixed an issue the merge option was not shown in the horizontal menu.*
    - *Fixed errors when uploading a document in the workflow panel now are logged.*
    - *Fixed an issue when getting the email of a user.*
    - *Fixed an issue in workflow panel when validating the extension of the files to be uploaded.*
    - *Fixed an issue in security panel checkbox: the checkbox only must be shown when the recursive checkbox is enabled.*
    - *Fixed an issue in wizard steps that was not copying the latest data.*
    - *Fixed an issue with the location of the stamp.*
    - *Fixed a UI defect in Chrome when closing the left panel.*
- *Improvements and fixes in mobile:*
    - *Fixed an issue with Arabic due problems loading custom CSS.*
    - *Added new password metadata field type.*
    - *Added an option to close contextual menu.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Added a feature to show errors in a log panel.*
- *Added 2FA support.*

v8.1.4

- *Release: 2023-08-29*
- *First public release.*
- *New autocompleteValueClassName attribute in the property group definition.*
- *New allowedExtensions attribute in workflow upload form element.*
- *Repository related reports are now stored in OpenKM.*
- *The left menu can be hidden in the PDF preview.*
- *Improved text document preview.*
- *Fixed issue with 2FA and user edit.*
- *Fixed issue with email content search.*
- *Improvements and fixes in KCenter:*
  - *Added a new feature that allows to delete a version of a node.*
  - *Improvement in the workflow upload field: Now you can chose which document extensions are allowed.*
  - *Improvement in notifications panel: Tags have been added to link nodes, workflow tasks, and tasks.*
  - *Improvement in metadata: the user has the option to fill in automatically with autocomplete plugin.*
  - *Improvement in the preview. Now you can hide the left panel or choose the default view.*
  - *Improvement in suggestbox usability.*
  - *Improvement in error panel usability.*
  - *Fixed issue with drag & drop, when the user does not have grants to upload in the current folder, but is able to update documents because have grants for them.*
  - *Fixed issue previewing text files. The text files were previewed as PDF and now are previewed as a regular text file.*
  - *Fixed the issue with the clean button in forms with horizontal and vertical panels.*
  - *Fixed an issue in the alert panel when the row element is not a node.*
  - *Fixed an issue with the fields of type link.*
- *Improvements and fixes in mobile:*
  - *Added external tabs feature.*
  - *Added workspace tabs feature*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Updater utility

OpenKM "Updater utility" is a Java Administrator tool that helps with performing upgrades automatically. You can download it from https://download.openkm.com/okm/OKMUpdater.jar.



*This utility needs access to **update.openkm.com** and **download.openkm.com** domains.*



Updater utility requires Java 11

Command-line parameters:

Parameter	*Description*
-h	*Show help information.*
-d	*Update development version (should only be used under OpenKM technical supervision).*
-f	*Force the update.*
-l	*Remove the Tomcat logs.*
-m	*Apply actions manually.*
-n	*Monochrome mode.*
-o	*Disable post-stage actions.*
-r	*Disable pre-stage actions.*
-u	*Override detected Tomcat URL.*
-v	*Show version.*

Upgrade stages

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Check if it's located at the $TOMCAT_HOME folder, otherwise, it will raise an error message.*
- *Ask for some information from the administrator.*
    - *OpenKM URL (usually someone accessible from the server like http://localhost:8080/openkm).*
    - *The application is configured as a service [Y/N].*
    - *The OpenKM valid ROLE_ADMIN user (okmAdmin).*
    - *The administrator user password.*
- *Check the service configuration, the user connection, and the OpenKM version.*
- *If the utility is successful on these checks, it will require confirmation by the user to proceed with the upgrade [Y/N].*
- *Download the new openkm.war file (current version + 1).*
- *When finished downloading, it will execute pre-upgrade actions.*
- *Stop OpenKM.*
- *Replace the old OpenKM.war file by the newly downloaded one.*
- *Modify the parameter **hibernate.hbm2ddl** to **update** into the file $TOMCAT_HOME/openkm.properties.*
- *Start OpenKM again.*
- *Check if OpenKM is already available and then execute post-upgrade actions.*

Manual mode

*If you invoke the OKMUpdater utility with the **-m** parameter, **none of the actions will be executed, so you need to execute them manually**. Depending on the action type you need to use different utilities located at **Administration > Utility**.*



It is very important to execute the actions, and execute them in the right order, otherwise, it may cause information loss.

Types of actions:

- ***SQL**: Go to **Administration > Utilities > Database query** (Choose SQL from select)*
- ***HQL**: Go to **Administration > Utilities > Database query** (Choose HQL from select)*
- ***Script**: Go to **Administration > Utilities > Scripting**.*

Keep in mind that some actions should be executed before the upgrade and others after the upgrade. This is the upgrade process in short:

- ***Execute PRE actions***
- *Stop OpenKM*
- *Replace openkm.war*
- *Start OpenKM*
- ***Execute POST actions***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



It is important to know which actions are executed in PRE and in POST.



Sometimes maybe there are no PRE or POST actions

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Updater utility english video

Your browser does not support the video tag.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Updater utility spanish video

Your browser does not support the video tag.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Tomcat update



The upgrade procedure described below is valid for Tomcat versions:

- *9.0.76 and upper*

By default, OpenKM is installed with a specific Tomcat version. But every few weeks a new version is released and maybe you want your installation up to date. In this case, should open a ticket in the OpenKM support website to ensure this new version has been packed by the OpenKM Team. Once you have verified and downloaded the new version is available you have to follow the steps described in this section of the documentation.



Before updating Tomcat we recommend to upgrade to the latest OpenKM version and ensure you have a recent working backup.



Although you can use a pristine Tomcat we always recommend using those provided by OpenKM team because have additional files and default configurations which work best with OpenKM.

Quick method

The easiest method to update your Tomcat instance is following these steps:

- *Stop current Tomcat / OpenKM process.*
- *Download the latest Tomcat provided by support guys.*
- *Remove the TOMCAT_HOME/lib folder content and replace by the downloaded one.*
- *Start Tomcat / OpenKM againt.*

Once the process has finished you can verify the new version:

```
$ cd TOMCAT_HOME
$ ./bin/catalina.sh version
```

If you have any doubt or question, please open a support ticket at https://support.openkm.com

Linux

Preliminaries

- *Check in the **openkm.properties** file if you have a special repository path location (more information at Change repository home).*

Tomcat replacement

- ***Stop the current Tomcat***
- ***Unzip the new Tomcat***
- ***Copy** these **files and folders from the old Tomcat to the new** one:*



Remember when copying and moving data to check the security applied in the destination. Sometimes when you move data the security applied might change and what cause issues in the OpenKM startup process.

You can set them recursively:

```
$ sudo chown openkm:openkm tomcat-9.0.76 -R
```

Name	*Type*	*Comments*
tomcat-XXX/bin/setenv.sh	*File*	*Optional. Only required if you have modified this file.*
tomcat-XXX/openkm.properties	*File*	*OpenKM configuration file.* *If there's a mistake copying this file, might damage the repository. Please check the file has been correctly copied into the destination.*
tomcat-XXX/openkm.xml	*File*	*OpenKM security configuration file.* *This file might not exists. You can find this file only in some special installation.*
tomcat-XXX/logback.xml	*File*	*OpenKM log configuration file. Might not exists.*
tomcat-XXX/repository	*Folder*	*The folder where documents are stored.* *If there's a mistake copying this folder, might damage the repository. Please check the file has been correctly copied into the destination.*
tomcat-XXX/plugins	*Folder*	*Directory with OpenKM plugins.* *Might be empty. It only contains files if the OpenKM instance has customizations.*

tomcat-XXX/webapps	*Folder*	*OpenKM deployed application.*
tomcat-XXX/jbpm-console.properties	*File*	*jBPM engine configuration file.* *This file might not exists. This file exists only if jBPM engine has been installed.*

- ***Update your Tomcat service configuration**:*
  - *Change the **Tomcat path** at **/etc/init.d/openkm***
- ***Start OpenKM***
- *Go to Administration > Configuration parameters*
- ***Filter** in the top right corner **by the word "tomcat"***
- ***Replace the old Tomcat path** used in configuration parameters **by the new path**.*
- ***Restart OpenKM service.***
- ***Do a quick test.***

Suggested quick test:

- *Browse the repository.*
- *Download an existing file.*
- *Upload a new file a try to download.*

Rememeber to update the backup script because the location of the Tomcat directory would change. In case of Linux you can check it by doing (as root):

```
$ crontab -l
```

This way, you can see the location of the backup script.

Preserving the old Tomcat is a good idea, but keep on mind that the backup script usually backups all the /home/openkm directory so move it to another place.

If you want a more recent branch you can try Tomcat 9.0, but keep on mind that this new branch has not been fully tested and is not supported by OpenKM team.

Windows

Preliminaries

- *From services check the user who's executing OpenKM. Usually, the user should be "Local services", but sometimes it is used a standard user. Remember when registering the OpenKM service again to choose the right user.*
- *Check in the openkm.properties file if you have a special repository path location (more information at Change*

repository home).

Tomcat replacement

- ***Stop OpenKM from services***.
- ***Delete OpenKM from services.***

Two options to delete OpenKM from services:

First option:

- *Open a terminal with Administrator user.*
- *Go to tomcat-XXX/bin folder.*
- *Execute the next command:*

```
$:> service.bat remove
```

Second option:

- *Open a terminal with Administrator user.*
- *Execute the next command:*

```
$:> sc delete OpenKM
```

- ***Unzip de new Tomcat.***
- ***Copy*** *these* ***files and folders from the old Tomcat to the new*** *one:*

Remember when copying and moving data to check the security applied in the destination. Sometimes when you move data the security applied might change and what cause issues in the OpenKM startup process.

Name	*Type*	*Comments*
tomcat-XXX\bin\setenv.bat	*File*	*Optional. Only required if you have modified this file.*
tomcat-XXX\openkm.properties	*File*	*OpenKM configuration file.* *If there's a mistake copying this file, might damage the repository. Please check the file has been correctly copied into the destination.*
tomcat-XXX\openkm.xml	*File*	*OpenKM security configuration file.*

		This file might not exists. You can find this file only in some special installation.
tomcat-XXX\logback.xml	*File*	*OpenKM log configuration file. Might not exists.*
tomcat-XXX\repository	*Folder*	*The folder where documents are stored.* *If there's a mistake copying this folder, might damage the repository. Please check the file has been correctly copied into the destination.*
tomcat-XXX\plugins	*Folder*	*Directory with OpenKM plugins.* *Might be empty. It only contains files if the OpenKM instance has customizations.*
tomcat-XXX\extras	*Folder*	*Third-party applications used by OpenKM*
tomcat-XXX\webapps	*Folder*	*OpenKM deployed application.*
tomcat-XXX\jbpm-console.properties	*File*	*jBPM engine configuration file.* *This file might not exists. This file exisit only if jBPM engine has been installed.*

- ***Configure OpenKM as service***

- *Open a terminal with Administrator user.*
- *Go to tomcat-XXX/bin folder.*
- *Execute the next command:*

```
$:> service.bat install
```

- *Execute the next command:*

```
$:> openkm //US//OpenKM ++JvmOptions="-XX:+UseG1GC"
$:> openkm //US//OpenKM ++JvmOptions="-XX:+UseStringDeduplication"
$:> openkm //US//OpenKM ++JvmOptions="-XX:+HeapDumpOnOutOfMemoryError"
$:> openkm //US//OpenKM ++JvmOptions="-XX:HeapDumpPath=$CATALINA_HOME"
```

```
$:> openkm //US//OpenKM ++JvmOptions="-Djava.net.preferIPv4Stack=true"
$:> openkm //US//OpenKM ++JvmOptions="-Dmail.mime.ignoreunknownencoding=true"
$:> openkm //US//OpenKM ++JvmOptions="-Dfile.encoding=utf-8"
$:> openkm //US//OpenKM ++JvmOptions="-Dsun.jnu.encoding=UTF-8"
$:> openkm //US//OpenKM ++JvmOptions="-Djava.awt.headless=true"
$:> openkm //US//OpenKM ++JvmOptions="-
Dsun.java2d.cmm=sun.java2d.cmm.kcms.KcmsServiceProvider"
```

- ***Start OpenKM service***
- *Go to Administration >* *Configuration parameters*
- ***Filter*** *in the top right corner* ***by the word "tomcat"***.
- ***Replace the old Tomcat path*** *used in configuration parameters* ***by the new path***.
- ***Restart OpenKM service.***
- ***Do a quick test.***



Suggested quick test:

- *Browse the repository.*
- *Download an existing file.*
- *Upload a new file a try to download.*

Manual update



In the whole public documentation, you will not find any explanation about how doing a manual upgrade.

The whole migration section and the OpenKM Updater tool has been created with the purpose of helping users in the upgrading OpenKM process. Unfortunately the automatic upgrade tools

Unfortunately, the automatic upgrade tools are not compliance with all the companies policies and in some cases is mandatory do a manual upgrade.

We have decided not explaining here because although almost migration steps between OpenKM versions are easy, someone's needs you pay attention to them. For better service and to keep safe your system we think working together on it is the best scenario.



If this is your case, please create a ticket at our OpenKM customer support website and we will help you on it.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Oracle Java to Temurin

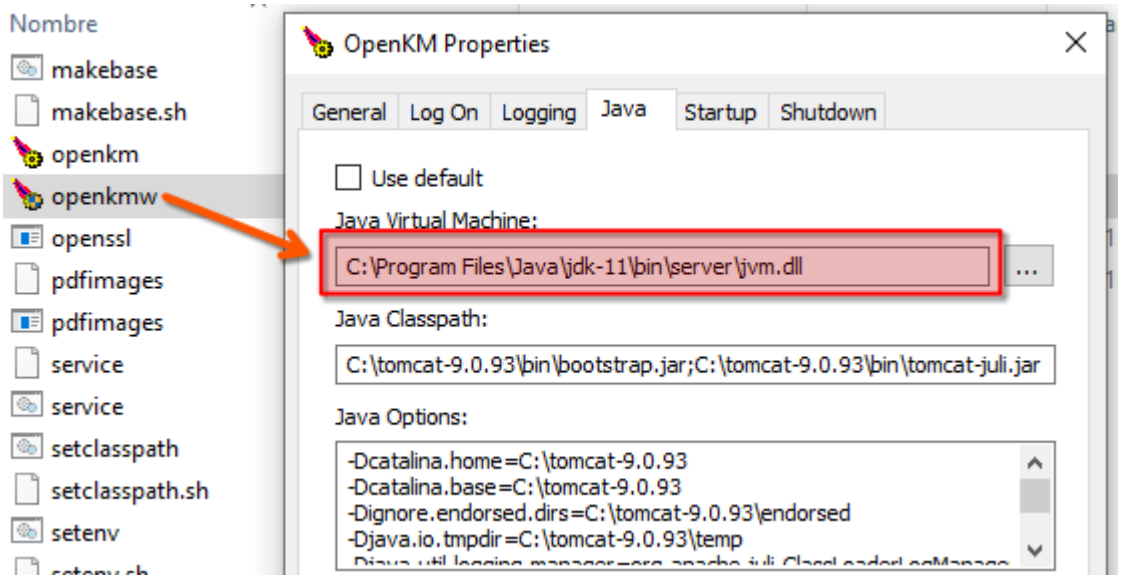
To swtich from Oracle Java to Temurin Java in Windows, follow these steps:

- *Stop OpenKM service.*
- *Uninstall Oracle Java.*
- *Download Temurin JDK from [Adoptium website (Java 11)](#).*



- *When installing, mark the "Set JAVA_HOME variable" option.*
- *Check if "JAVA_HOME" system variable is set properly.*
- *Go to OpenKM installation directory and execute "openkmw.exe" program.*
- *Go to "Java" tab and change the Java Virtual Machine path to Temurin installation directory.*
    - *It should be someting like "C:\Program Files\Eclipse Adoptium\jdk-11.0.26.4-hotspot\bin\server\jvm.dll" depending on the version.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



- *Start OpenKM service.*

Changes from version 7.1

Change

- *Java 11 support.*
- *Tomcat 9 support.*
- *Major library updates (added, updated, and removed libraries to improve the security).*
- *Added support for the Elastic Search.*
- ***SSO authentication** must now be implemented as an external module. In the developer course, an example is available on how to implement this module for OAUTH2 authentication.*
- *Changes that may affect the expected behavior in some automation events in comparison with version 7:*
  - *The **copy of a folder** and the **create folder from a template** raise a "**create folder**" event.*
  - *The **copy of a record** and the **create record from a template** raise a "**create record**" event.*
  - *The "**create document**" event is not raised in a **copy folder or record** hierarchy.*
  - *The "**create document**" event is not raised when **creating an email attachment**.*

Deprecated features from version 7.1

- ***GWT** support removed.*
- ***Classical UI** has been removed.*
- ***Thesaurus** support removed (may be implemented as a customization).*
- ***CMIS** support removed (may be implemented as an external module).*
- ***WEBDAV** support removed (may be implemented as an external module).*
- *The **Stack Metadata Adapter** plugin was removed (because the new UI no longer uses it).*
- *The **Folder style** feature was removed (because the new UI no longer uses it). The Creating your own NodePaginator plugin functionality replaces this feature.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Development

OpenKM professional portable development environment

The OpenKM Professional Portable development environment comes with all-in-one configured tools.

> *We encourage using our OpenKM portable development environment rather than configuring your own Eclipse, especially for Java beginners.*
>
> *We've tried to provide a development platform as quickly as possible to get it running with a few steps.*
>
> *More information about OpenKM professional portable development environment at Download center.*

Tools:

- *Eclipse*
- *OpenJDK 11*
- *MariaDB*
- *DBeaver*
- *iReport*
- *Tesseract*
- *LibreOffice*
- *Geany*
- *jBPM 3.3.1.GA*
- *Firefox*

Code:

- *Automation samples.*
- *Crontab samples.*
- *Form samples.*
- *OCR samples.*
- *Report samples.*
- *Workflow course.*

Suggested minimum hardware configuration:

- *2 CPU's*
- *4GB RAM*
- *64 bits architecture*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Javadoc

Each OpenKM version is created with a complete Javadoc available at http://docs.openkm.com/apidoc/.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Personal development environment

OpenKM personal development environment is distributed as a virtual machine image (OVA).

The personal development environment comes with everything you need to start quickly working:

- *OS Linux Mint distro.*
- *OpenKM application installed*
- *MariaDB database*
- *Eclipse*
  - *Code samples*
  - *Workflow samples*
- *IReport*
  - *SQL samples*
  - *Scripting samples*

Requirements

The virtual machine uses 2 core and 4GB of RAM, the host machine should have at least 4 core and 8GB of RAM.



We suggest having almost all applications stopped while executing the virtual machine.

We suggest installing it on a desktop computer either laptop computer. Although can work on a laptop with good hardware.

We suggest SSD disk to get extra performance in the virtual machine.

In some users, we observe executing the virtual machine in a secondary disk provides extra performance, seems some motherboards do not like the OS and Virtualization in the same hard disk.

How to get the personal development environment

Can download the virtual machine file from OpenKM download center.

- *Go to https://download.openkm.com*
- *Click on the button "OpenKM Professional Download Center".*
- *Login with your customer credentials.*
- *In the top menu choose tools.*
- *In the tools, ensure you have chosen OpenKM version 8.1.x*
- *Click in the Open KM-PorDev.ova file*



The download link is sent by e-mail.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

The link can be used only one time.

You have 24h from the e-mail that has been sent to use the link, passed this time the link is expired.

How to install

The OpenKM personal development environment can be installed in almost virtual machine software like Virtual Box, VMWare, Hyper-V.



Because OVA (Open Virtual Appliance) is not 100% compatible with virtualization software at least users of VMWare usually should make a few modifications in the file or extract the mkv to import the virtual machine.

If you are looking for a quick deployment we encourage using Virtual Box.

If you decide for Virtual Box, follow the steps described in the next video *https://www.youtube.com/watch?v=9ugBYf5ggu8&feature=emb_logo*

License activation



Personal development environment must have Internet connection to register the license.

Finally, the license of your personal development environment must be activated, for it follows the next steps:

1- Open a terminal (the password of the user ***openkm*** *is* ***openkm****)*

2- Execute the command

```
$> pdl-reset
```

3- Start OpenKM service

```
$> sudo service openkm start
```

4- Open an issue in the OpenKM customer support website. Share the application ID available in the home of the administration.

How to upgrade the development environment



Because virtual machine comes with OpenKM, jbpm-console, and compilation libraries the upgrade process has at least 3 steps:

- *Upgrade OpenKM*
- *Upgrade OpenKM compilation libraries*
- *Upgrade jbpm-console*

Step 1 - OpenKM upgrade

- *Close eclipse*
- *Start OpenKM as a service*

```
$> sudo su
$> service openkm start
```

- *Upgrade OpenKM tool as any standard OpenKM. The process for upgrading the development environment is exactly the same as upgrading the standard OpenKM professional edition. More information at Updater utility.*

```
$> cd /home/openkm/Development/tomcat
$> wget http://download.openkm.com/OKMUpdater.jar
$> java -jar OKMUpdater.jar
```

The latest OpenKM development environment come with update.sh script

```
$> cd /home/openkm/Development/tomcat
$> ./update.sh
```

Step 2 - OpenKM compilation libraries upgrade

- *Close eclipse*
- *Edit the download.sh script with the latest OpenKM and jbpm-console versions at the header*

```
$> cd /home/openkm/Development/portable/lib
$> nano download.sh
```

Sample script for version 7.1.22 (remember to set the right JBPM version based on the compatibility table at JBPM Console documentation section):

```
#!/bin/bash
OKM_VERSION="7.1.22"
JBPM_VERSION="1.8"

# Download OpenKM libraries
wget http://download.openkm.com/okm/portable/jar/openkm-${OKM_VERSION}-client.jar -O openkm-cli
wget http://download.openkm.com/okm/portable/jar/openkm-${OKM_VERSION}-jar-with-dependencies.j

# Download jBPM libraries
wget http://download.openkm.com/okm/pro/workflow/jbpm-${JBPM_VERSION}-client.jar -O jbpm-client.j
wget http://download.openkm.com/okm/pro/workflow/jbpm-${JBPM_VERSION}-jar-with-dependencies.ja
```

- *Execute the download.sh script*

```
$> ./download.sh
```

- *Edit the register.sh script with the latest OpenKM versions at the header*

```
$> nano register.sh
```

Sample script for version 7.1.22 (remember to set the right JBPM version based on the compatibility table at JBPM Console documentation section):

```
#!/bin/bash
VERSION="7.1.22"
JBPM_VERSION="1.8"

# Register OpenKM libraries
mvn install:install-file -DgroupId=com.openkm -DartifactId=openkm -Dversion=${VERSION} -Dclassifier=
mvn install:install-file -DgroupId=com.openkm -DartifactId=openkm -Dversion=${VERSION} -Dclassifier=

# Register Jbpm libraries
mvn install:install-file -DgroupId=com.openkm -DartifactId=jbpm-console -Dversion=${JBPM_VERSION}
```

- *Execute the register.sh script*

```
$> ./register.sh
```

Step 3 - Jbpm-console upgrade

- *First, check the right jbpm-console version based on the compatibility table at JBPM Console documentation section.*
- *Follow the steps described at Migration guide.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configuring IDE from a clean Eclipse



We recommend using our OpenKM Portable Development IDE rather than configuring your own.

You can also use IntelliJ IDEA or other development Java tools like NetBeans.

Requirements

Software:

- *Eclipse*
- *OpenJDK 11*



Your eclipse must have support for:

- *Maven*
- *Subversion (recommended)*

Hardware:

Suggested minimum hardware configuration:

- *2 CPU's*
- *4GB of ram*
- *64 bits architecture*

Download eclipse

*We suggest downloading the latest stable version of Eclipse. Eclipse comes in several versions, you should download the "**Eclipse IDE for Java EE Developers**" for your computer architecture (64 bits).*



More information at https://eclipse.org/downloads/.

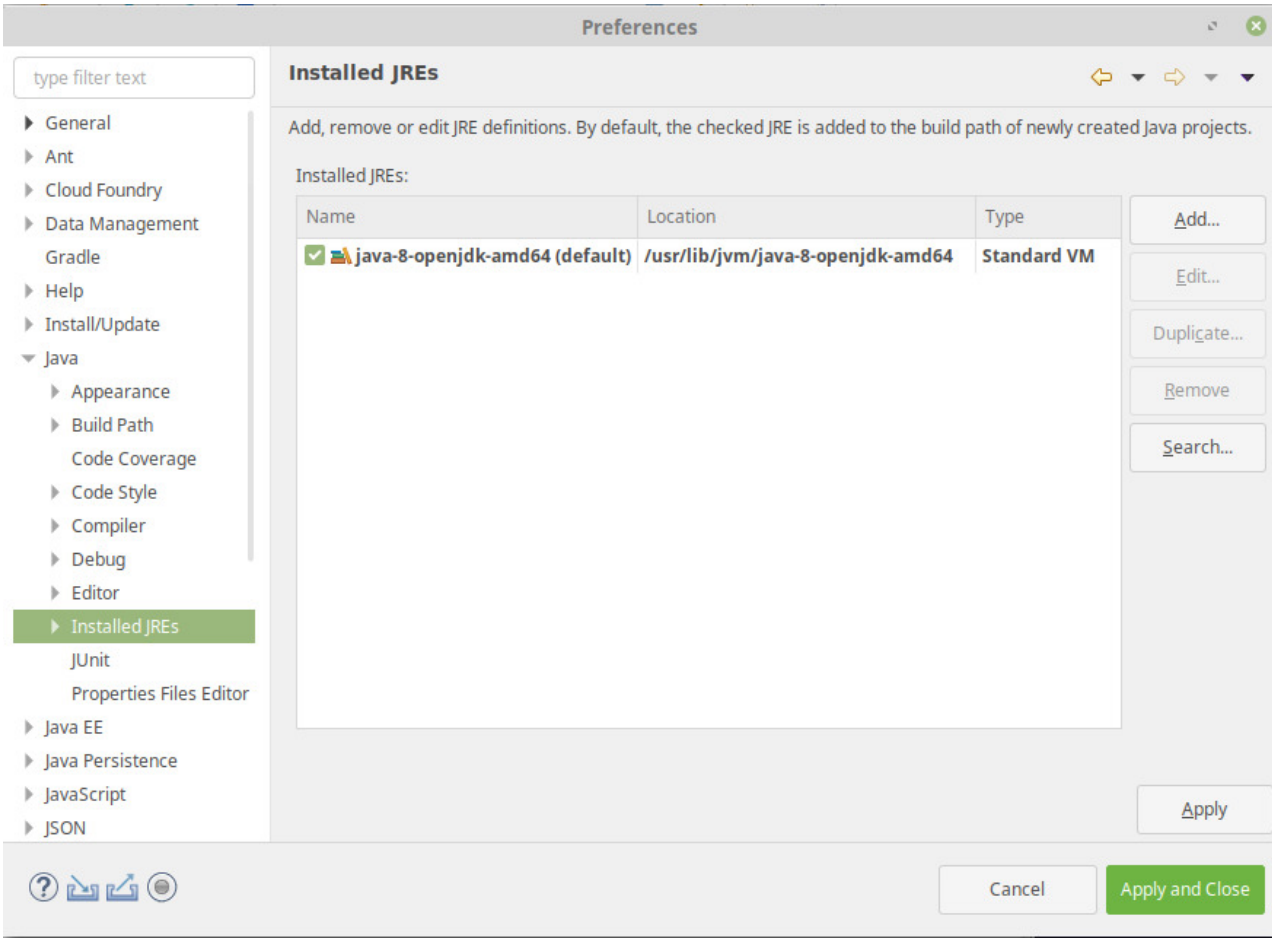
Configure Java

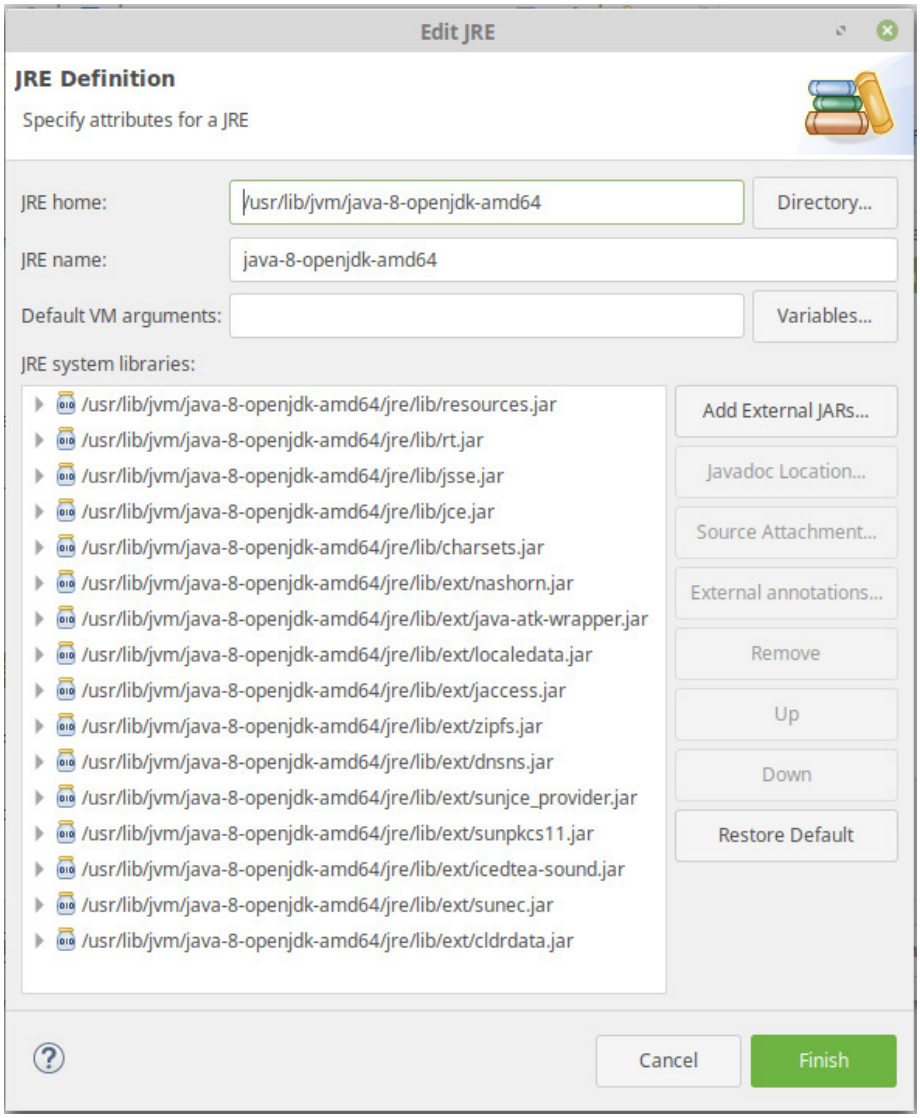


You must have Java installed on your server otherwise go to https://learn.microsoft.com/en-us/java/openjdk/download and download the latest OpenJDK 11 version according to your OS and hardware architecture. Execute the Java installer and follow the installation wizard.

*Go to **Window** / **Preferences** / **Java** / **Installed JRE**. You must have OpenJDK 11 defined here:*

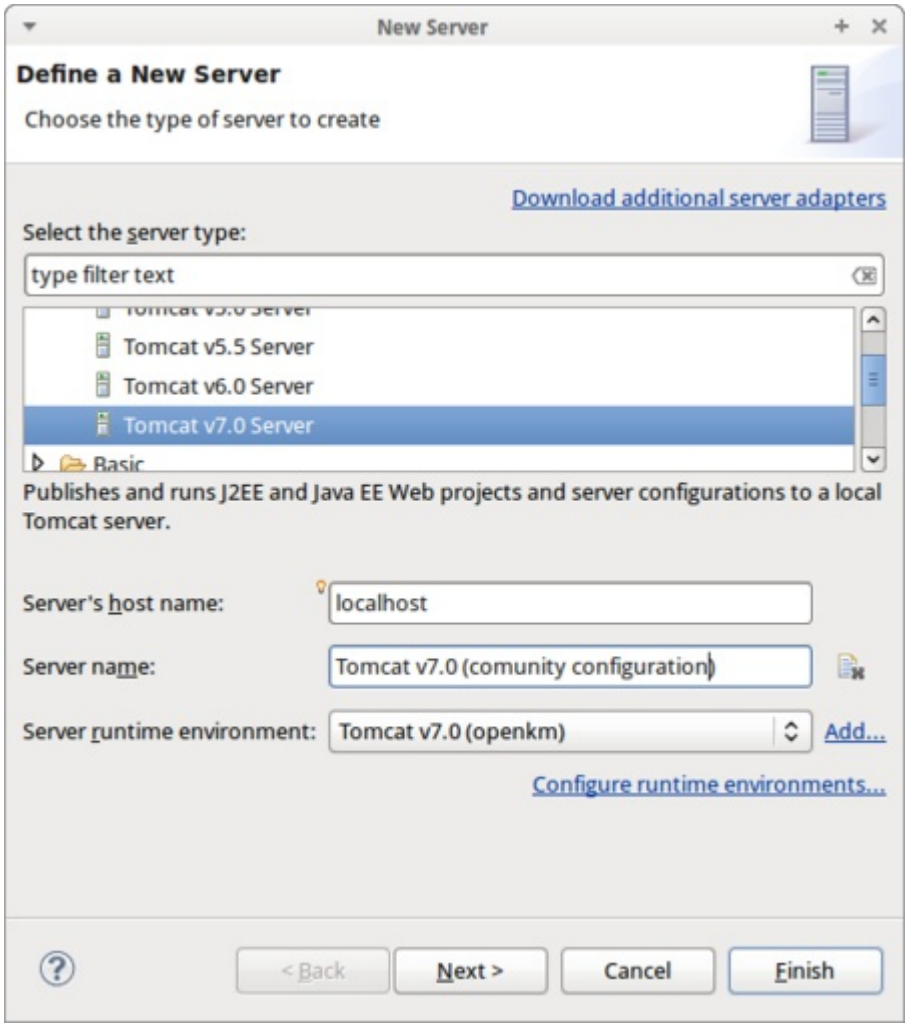
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


Preferences
type filter text
General
Ant
Cloud Foundry
Data Management
Gradle
Help
Install/Update
Java
Appearance
Build Path
Code Coverage
Code Style
Compiler
Debug
Editor
Installed JREs
JUnit
Properties Files Editor
Java EE
Java Persistence
JavaScript
JSON
Installed JREs
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.
Installed JREs:
Name
Location
Type
java-8-openjdk-amd64 (default)
/usr/lib/jvm/java-8-openjdk-amd64
Standard VM
Add...
Edit...
Duplicate...
Remove
Search...
Apply
Cancel
Apply and Close



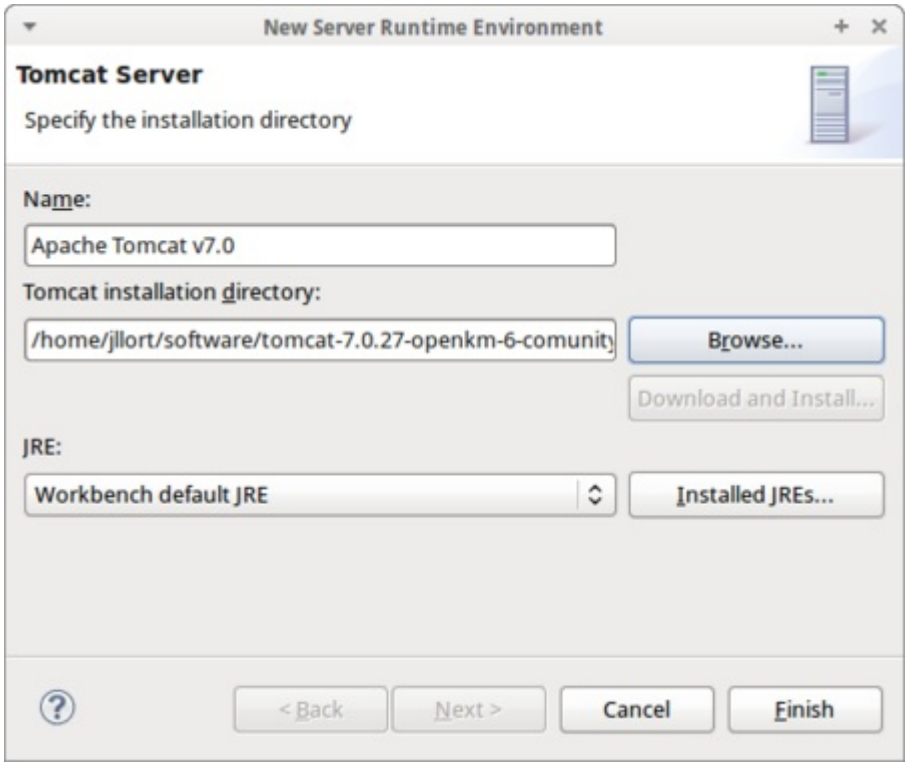
Configure tomcat server

*Go to "**servers**" tab. Click on right click mouse button and from the contextual menu choose "**Add / server**". From the list choose "**Tomcat 7 server**":*



*Configure a new **Run time server environment**:*

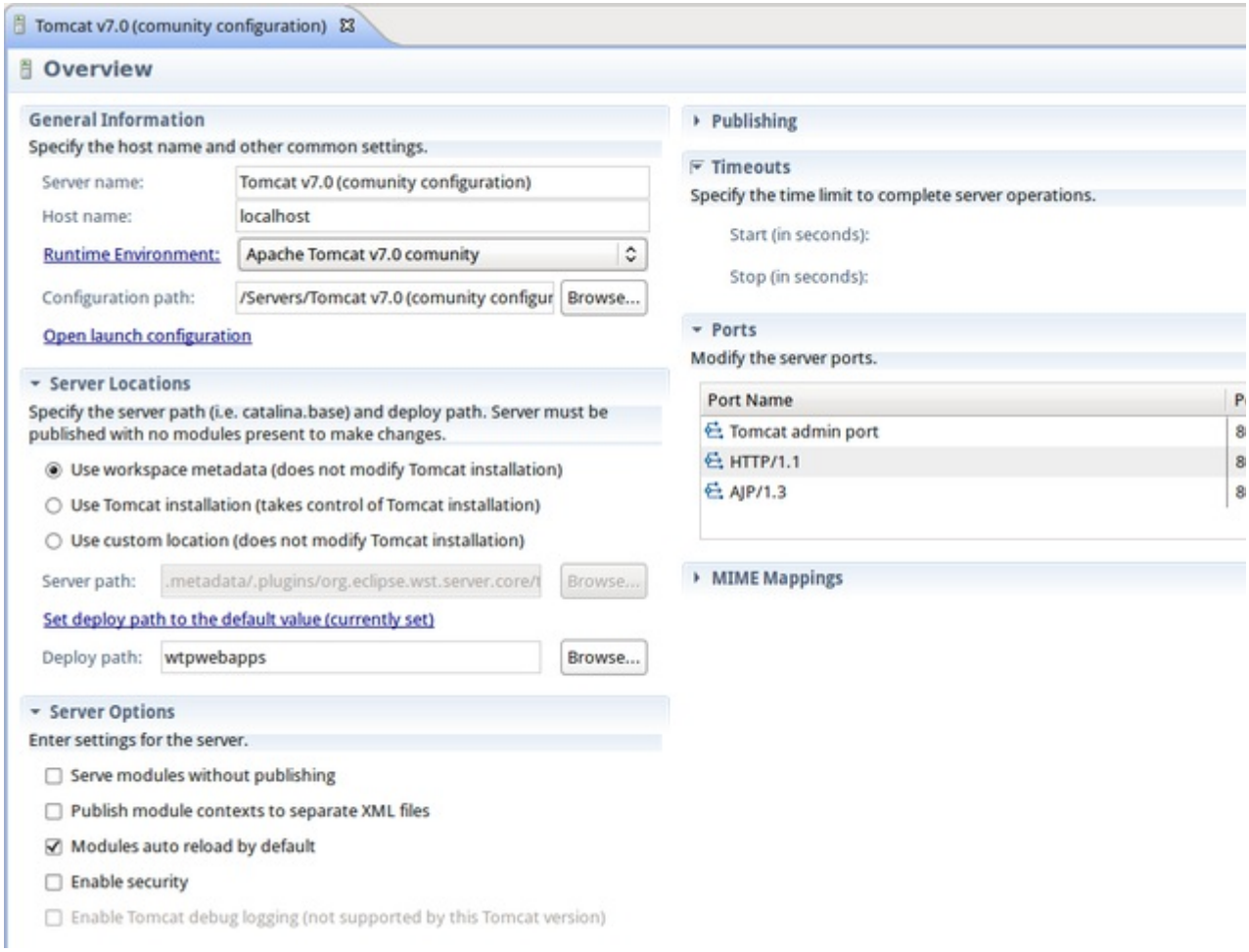
> *Choose the "Workbench default JRE" when your default JRE is "OpenJDK 11" or choose "OpenJDK 11" from JRE list of installed JREs.*



*Check box **Use Tomcat installation** (takes control of tomcat installation).*

*Increment start and stop **timeouts** (for example to 180 and 120 seconds respectively).*

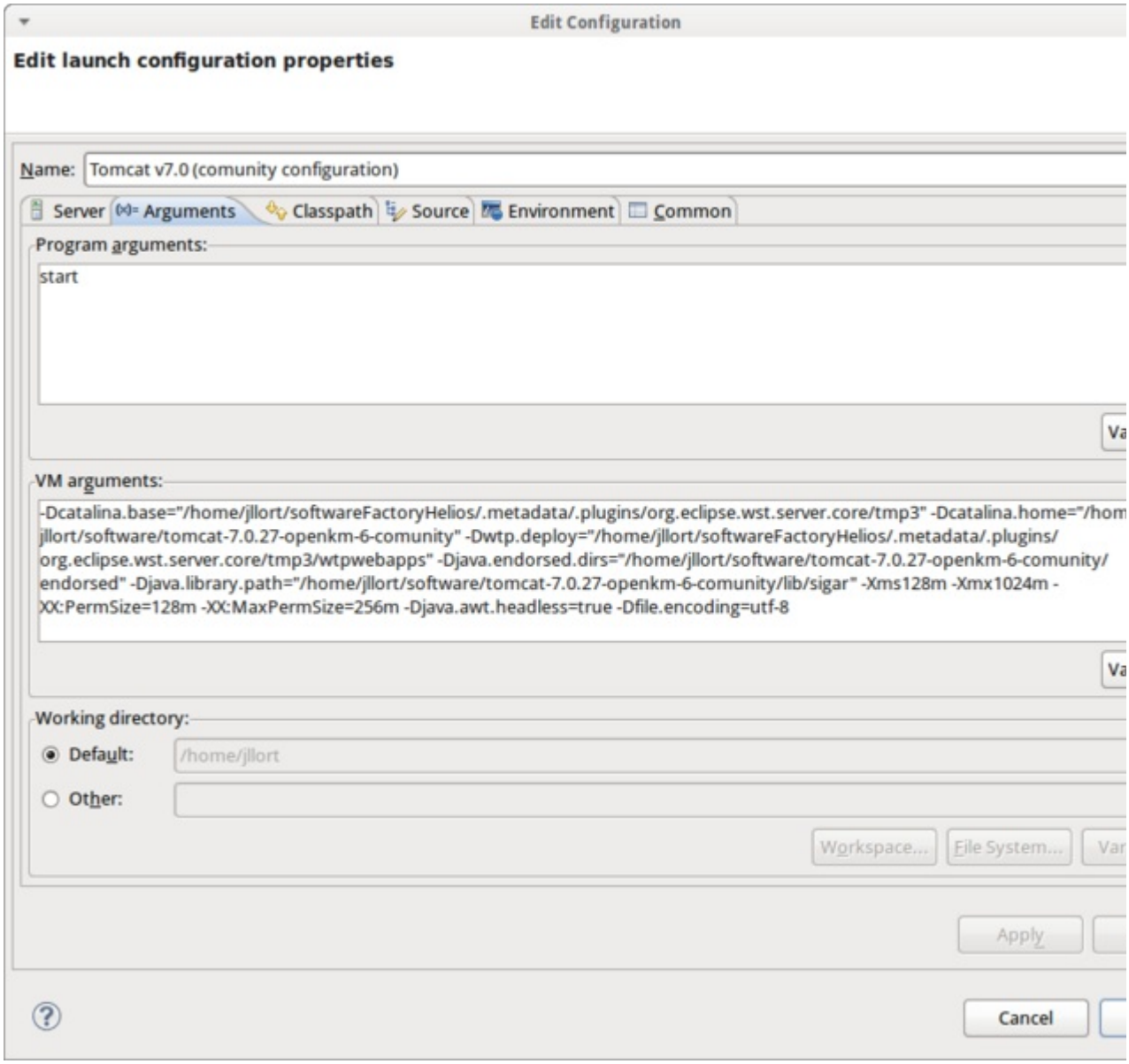
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



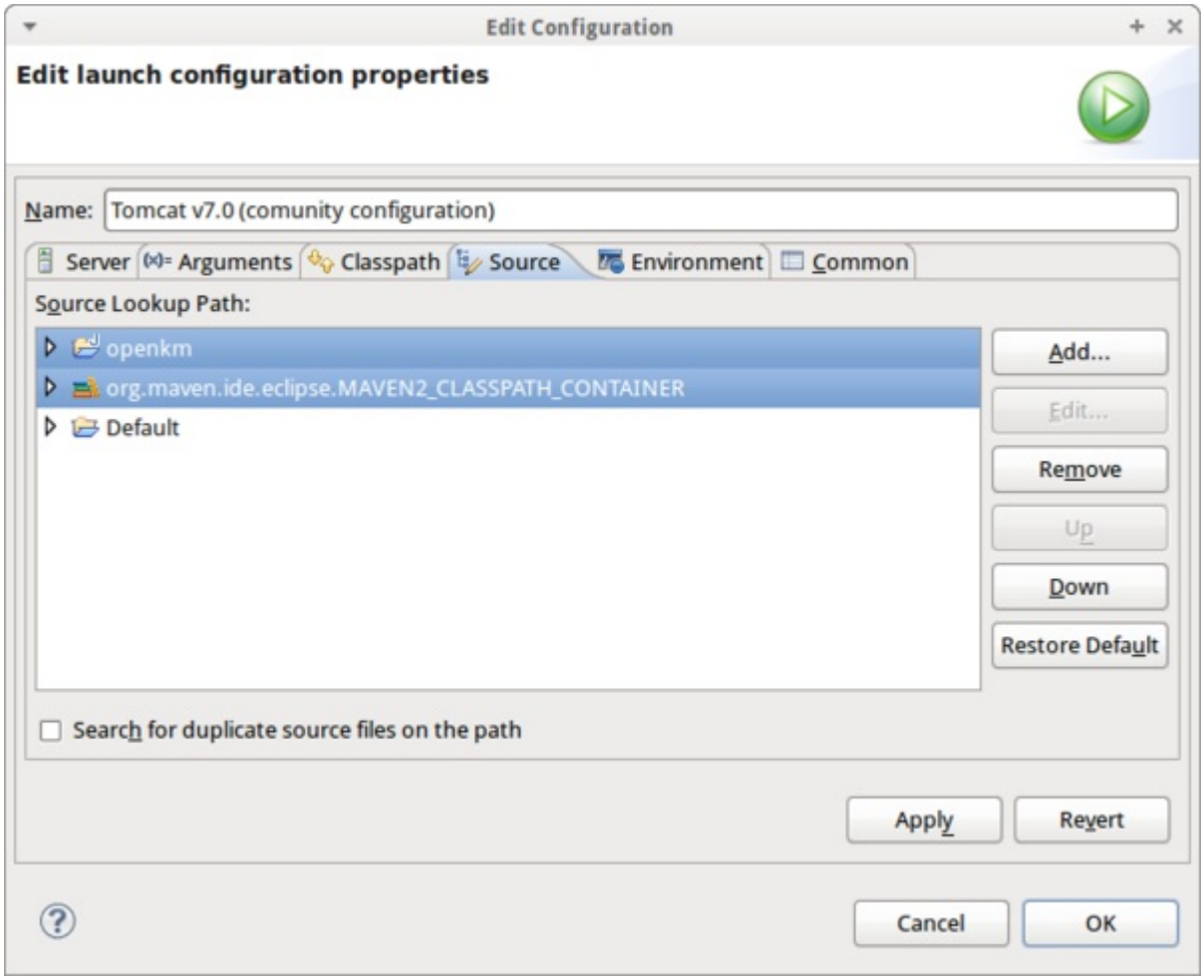
*Edit **launch configuration**: Add **memory parameters** - we suggest 4GB - and **utf-8** file encoding at VM arguments field:*

```
-Xms1024m -Xmx4096m -Djava.awt.headless=true -Dfile.encoding=utf-8
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



*Add your **OpenKM project** at source tab:*

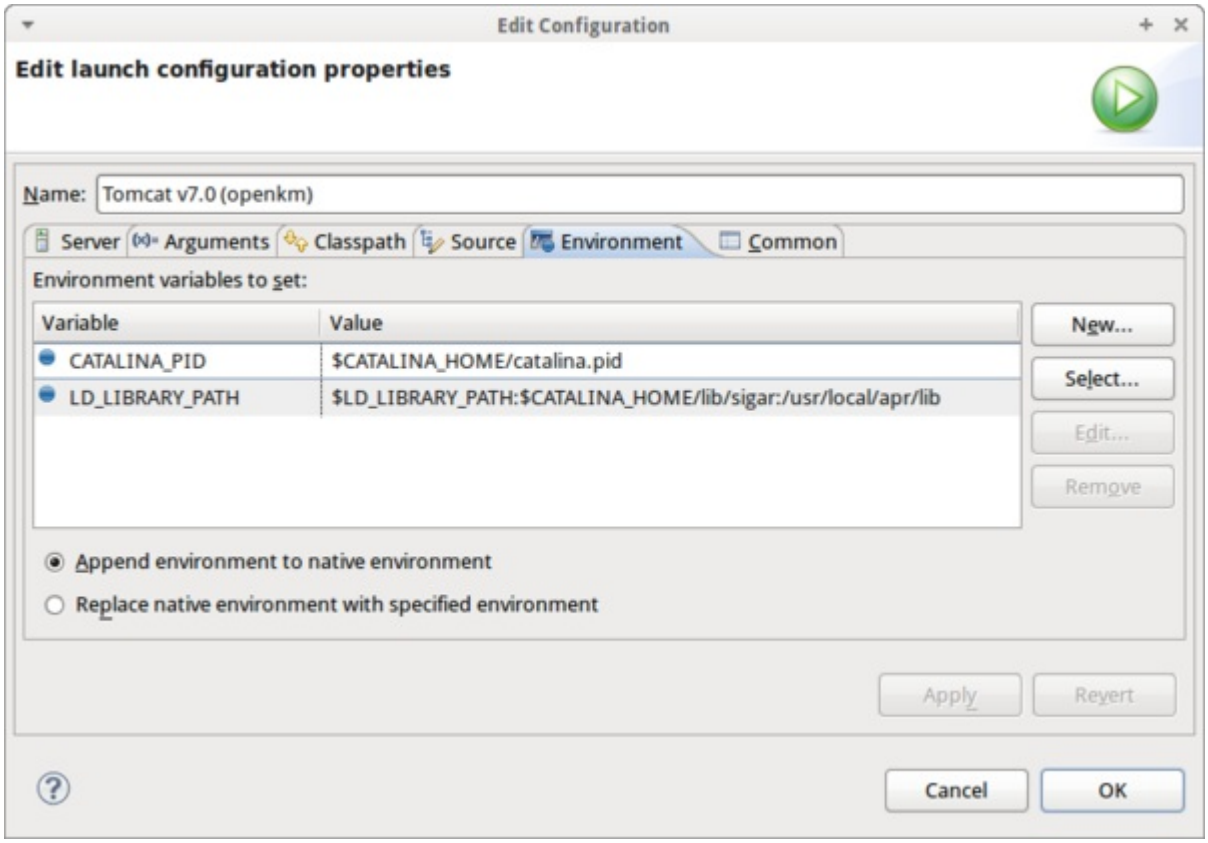


*Add **CATALINA_PID** at environment variables:*

```
CATALINA_PID=$CATALINA_HOME/catalina.pid
```

or

```
CATALINA_PID=$CATALINA_HOME/catalina.pid
```



Maven support for Eclipse

The Eclipse version comes by default with maven support plugins yet installed.

Configuring Maven in your OS

The latest Eclipse version comes with embedded Maven software already installed. We suggest using the default configuration.

Maven is a software tool for Java project management and build automation. It is similar in functionality to the Apache Ant tool, but is based on different concepts. Maven is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

Maven uses a process known as a Project Object Model (POM) to describe the software project being built, its dependencies on other external modules and components, and the build order. It comes with pre-defined targets for performing certain well defined tasks such as compilation of code and its packaging.

More information at http://maven.apache.org.

Install Maven version 3.3.x

Follow the Maven installation guide.

Maven tips

Download sources and javadoc from libraries dependency to improve the developer experience:

```
$ mvn install -DdownloadSources=true  -DdownloadJavadocs=true
```

Ignore test failures in the build process this way:

```
$ mvn install -Dmaven.test.failure.ignore=true
```

To skip tests, you can do so by executing the following command:

```
$ mvn install -Dmaven.test.skip=true
```

Memory problems

To set more memory for Maven in GNU/Linux you can add the Maven executable to the binary path editing the $HOME/.bashrc file and adding this line at the end:

```
export PATH=$PATH:/path/to/maven/installation/bin
export MAVEN_OPTS="-Xmx512m"
```

On windows set from general environment variables feature.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Configure eclipse to use local Maven

Go to ***Window > Preferences > Maven > Installation*** *and click on* ***Add button*** *to set your local Maven.*



We do not suggest you to apply this change, default embedded Maven configuration usually goes right on almost cases.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Eclipse tips

Debug tomcat

You can debug your OpenKM installation using the Tomcat logging facility. This is a useful thing when you have problems with your configuration. Default OpenKM installation tries to log important events like errors and warnings. Is possible to change this configuration editing the file $TOMCAT_HOME/conf/logging.properties.

Default Tomcat log configuration can generate a lot of messages. These files are stored at $TOMCAT_HOME/logs.

Debug remote tomcat server



When you have problems in the production environment this configuration is especially useful.

Add the following options when the JVM is started:

```
-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n
```

*Or if you are using shell scripts to start Tomcat, start it with the following command (**recommended**):*

```
$ $TOMCAT_HOME/bin/catalina.sh jpda start
```

It will start Tomcat so that a remote debugger can be connected to port 8000.



More information at:

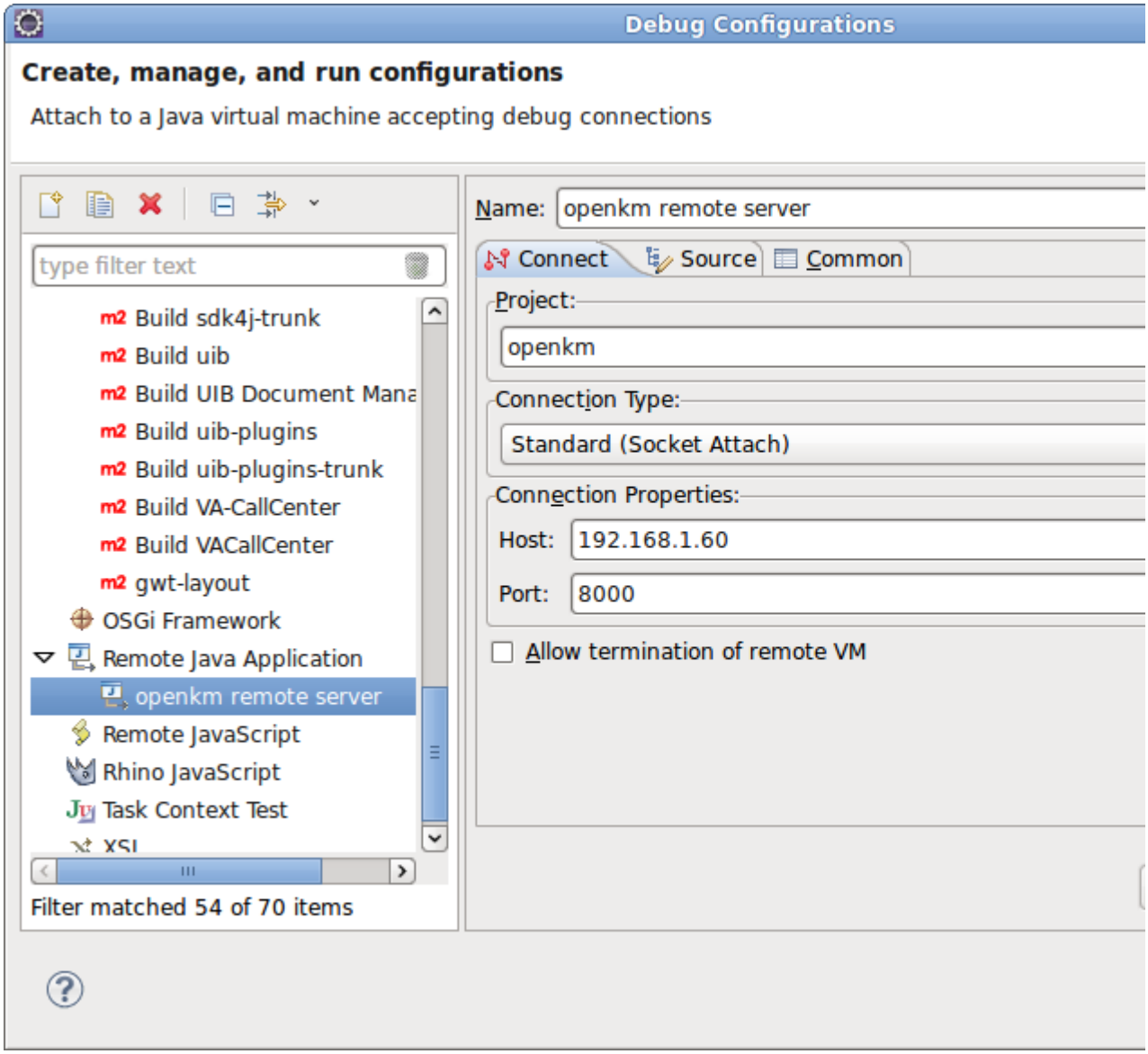
- *Tomcat developing: How do I configure Tomcat to support remote debugging?*
- *Como hacer debug remoto en Tomcat desde Eclipse .*

Eclipse configuration

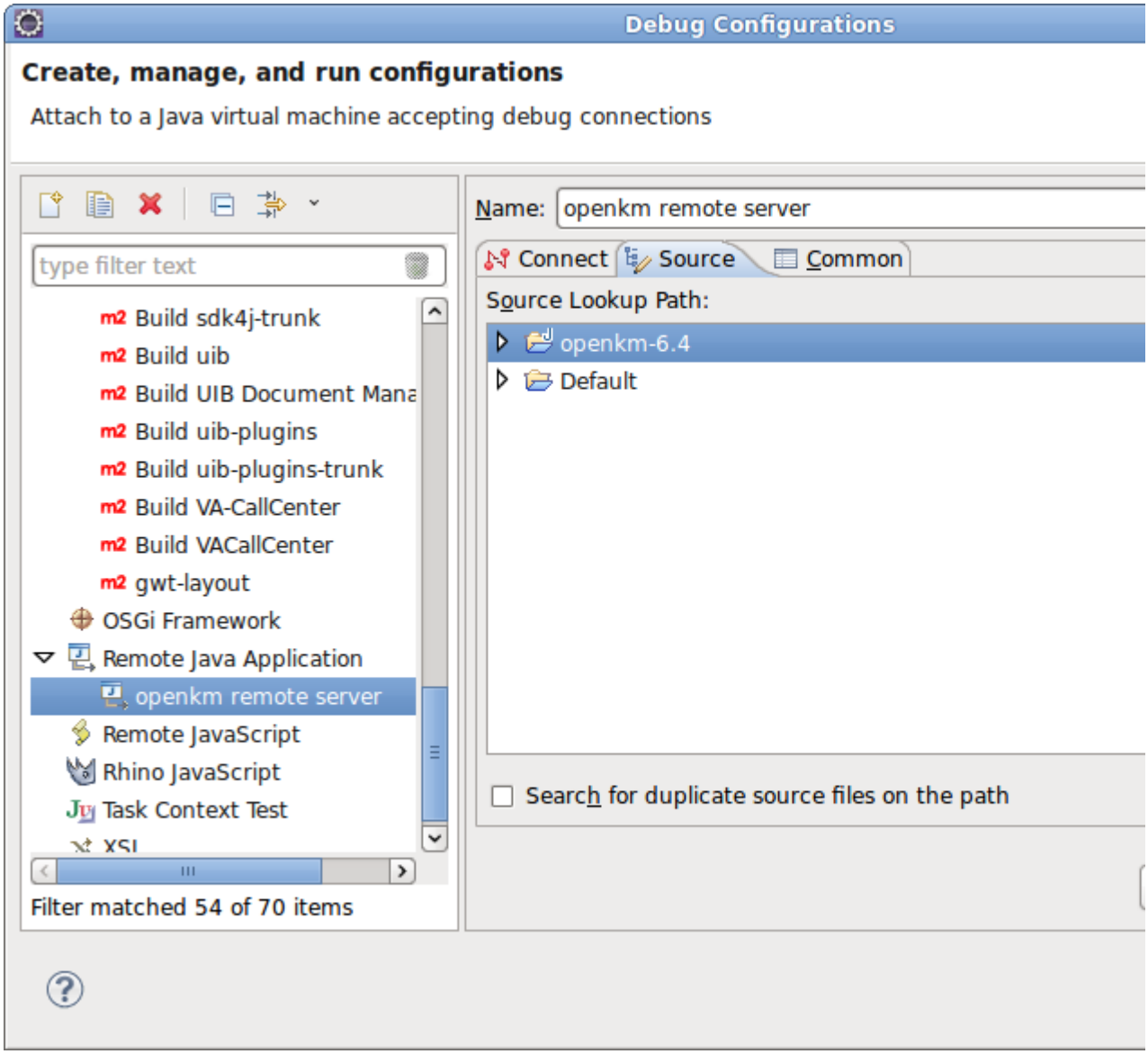
*Open eclipse "**Debug configurations**", then choose "**Remote Java Application**", right click on mouse and choose "**new**".*

*Set the "**host**" and "**port**" of the remote server:*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Include in source tab your OpenKM java project:



You can debug normally, setting breakpoints, etc.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Eclipse known troubleshooting

Maven raise error 206 the filename or extension is too long

When you build an application on windows you can get the error ***create process error 206 the filename or extension is too long*** *this is caused to a Windows OS limitation. To solve it:*

- *Move* ***.m2*** *repository to* ***c:***\
- *Go to* ***Eclipse*** *>* ***Windows/Preferences/Maven/User Settings*** *->* ***Create your own setting.xml*** *with its content:*

```
<settings>
  <localRepository>c:/.m2/repository</localRepository>
</settings>
```

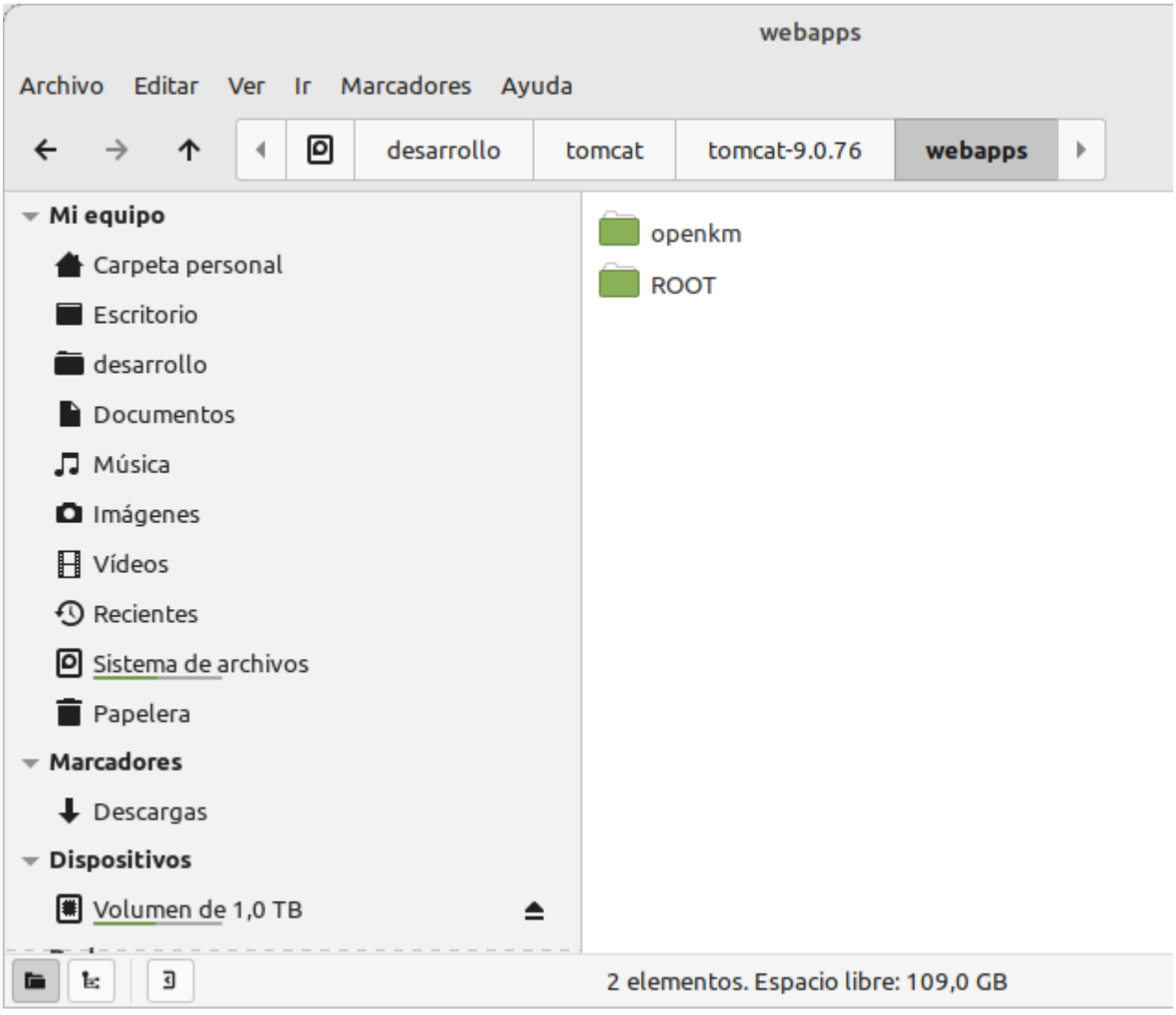
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Development tips

Tomcat application deployment

For faster development we recommend not deploying openkm.war directly, file and create in $TOMCAT_HOME/webapps a folder alias of target/openkm:

> *To create symbolic links in Windows use "**juntion.exe**" application.*
>
> *More information at http://technet.microsoft.com/en-us/sysinternals/bb896768.*



Deploying script sample

You can use a deploying script:

```
#!/bin/bash
# @author: Kenneth Walter

#Cleans and re-compiles the source files
mvn -Dmaven.test.skip=true clean install $*

#Set the TARGET_DIRECTORY to the path of your tomcat installation
TARGET_DIRECTORY=~/Downloads/tomcat-9.0.76/
FILE_TO_MOVE=target/openkm.war
#This will only attempt to replace the existing WAR if the new WAR exists
if [ -f $FILE_TO_MOVE ]
then
  echo 'Deploying WAR to Tomcat Directory'
  cp -v $FILE_TO_MOVE $TARGET_DIRECTORY/webapps/.
  echo 'Done'
fi
```

Profiling application

You can profile local and remote Java applications using Java VisualVM.



For the local application is easy because it detects automatically and show them in the listing. For the remote application you need to do a little work.

Edit Tomcat startup scripts

*Edit the **$TOMCAT_HOME/bin/setenv.*** files (sentenv.sh for linux and setenv.bat for Windows).*



*The **java.rmi.server.hostname** should match the IP of the server, usually the **127.0.0.1** value by the right IP.*

```
CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote=true"
CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.port=9090"
CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.ssl=false"
CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.authenticate=false"
CATALINA_OPTS="$CATALINA_OPTS -Djava.rmi.server.hostname=127.0.0.1"
```



To connect to this remote Tomcat instance:

- *Go to **File > Add JMX Connection...** and fill the hostname and port.*
- *Additionally can change the port is 9090 using the **com.sun.management.jmxremote.port** property.*

Authentication

*To enhance security, enable authentication. Edit the **$TOMCAT_HOME/bin/setenv.*** files (sentenv.sh for linux and setenv.bat for Windows).*

```
CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.authenticate=true"
CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.password.file=$CATALINA_HOME/co
CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.access.file=$CATALINA_HOME/conf/
```

*Edit the access allow file **$CATALINA_HOME/conf/jmxremote.access**:*

```
monitorRole readonly
controlRole readwrite
```



*To show remote threads you need the **readwrite** permission.*

*Edit the password file **$CATALINA_BASE/conf/jmxremote.password**:*

```
monitorRole tomcat
controlRole tomcat
```

Restrict access to these files:

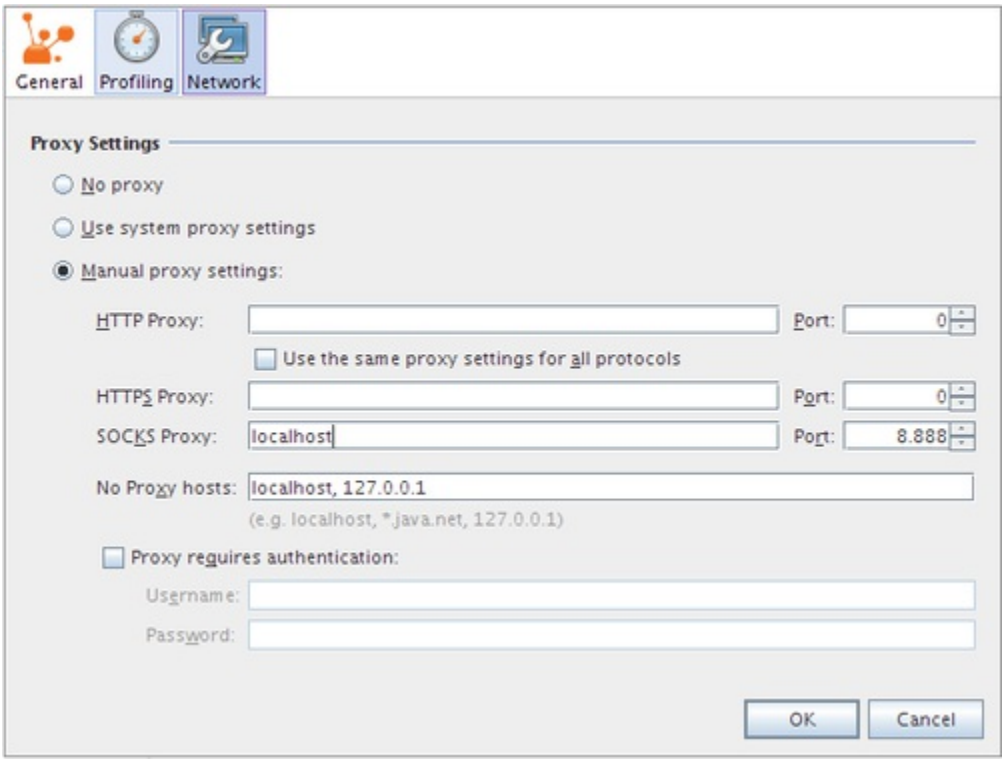
```
$ chmod 600 $CATALINA_HOME/conf/jmxremote.access
$ chmod 600 $CATALINA_HOME/conf/jmxremote.password
```

Monitoring Remote JVM Over SSH

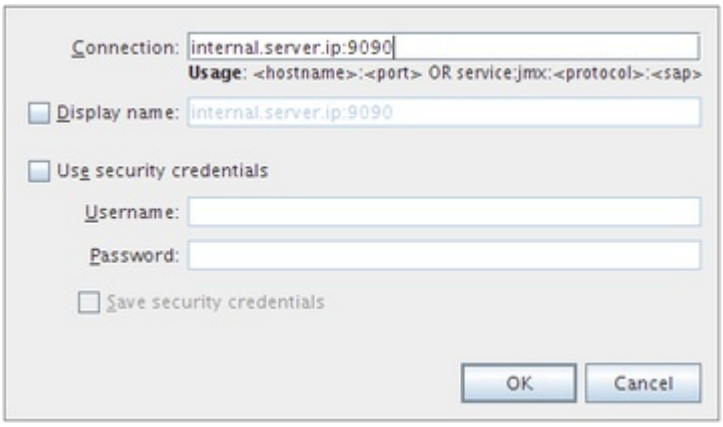
The easiest way to connect to the remote JMX is to use a SOCKS proxy.

```
$ ssh user@external.server.ip -D 8888
```

Configure VisualVM to use this proxy. Go to ***Tools*** *>* ***Options*** *>* ***Network*** *and edit* ***Manual Proxy Settings****. Configure SOCKS Proxy at localhost and port 8888.*



Now you can connect VisualVM to the remote target. Go to ***File*** *>* ***Add JMX Connection...*** *and type the IP or hostname of the remote machine and the configured JMX port (in our sample configuration is 9090).*



More information at:

- *Profiling With VisualVM, Part 1*
- *Profiling With VisualVM, Part 2*
- *Connecting Visual VM to Tomcat 7*
- *Monitoring and Managing Tomcat 7*
- *Remote Profiling of JBoss using VisualVM*
- *Troubleshooting application performance with VisualVM*
- *Monitoring of Tomcat with VisualVM and VisualGC*
- *Using VisualVM to fix live Tomcat and JVM problems*

HPROF

Java includes HPROF, a profiler which collect application runtime information. HPROF is capable of presenting CPU usage, heap allocation statistics, and monitor contention profiles.

For example, can collect CPU usage information by sampling threads.

*Edit **$TOMCAT_HOME/bin/setenv.sh** file (or **setenv.bat** if using Windows):*

```
CATALINA_OPTS="$CATALINA_OPTS -agentlib:hprof=cpu=samples"
```

*When Tomcat starts you can see a file called **java.hprof.txt**. The CPU profiling info will be dumped to this file once Tomcat process is stopped.*

More information at:

- *HPROF: A Heap/CPU Profiling Tool in J2SE 5.0*
- *HPjmeter*

Application version compatibility table

Application compatibility changes can be caused by:

- *DTD definition changes. Usually, newer application version supports older DTD versions.*
- *Libraries upgrades.*

If your OpenKM server can't access to the Internet, you can download this DTD and copy to a convenient place at your server. Remember to update your forms.xml with the DTD location.

```
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Property Groups 3.12//EN"
                                "file:///path/to/property-groups-3.12.dtd">
```

Compatibility tables:

Property Groups	OpenKM
property-groups-3.12.dtd	Since 8.1.14
property-groups-3.11.dtd	Since 7.1.41
property-groups-3.10.dtd	Since 7.1.23
property-groups-3.9.dtd	Since 7.1.22
property-groups-3.8.dtd	Since 7.1.9
property-groups-3.7.dtd	Since 7.1.5

Workflow Forms	OpenKM
workflow-forms-2.17.dtd	Since 8.1.6
workflow-forms-2.16.dtd	Since 8.1.5
workflow-forms-2.15.dtd	Since 7.1.41
workflow-forms-2.14.dtd	Since 7.1.35
workflow-forms-2.13.dtd	Since 7.1.27
workflow-forms-2.12.dtd	Since 7.1.18
workflow-forms-2.11.dtd	Since 7.1.5

Report Parameters	OpenKM
report-parameters-2.1.dtd	Since 7.1.5

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating a JAR file with Eclipse



These instructions are only a guide for creating a JAR based on a single .java file.

Take it as is, an introduction.

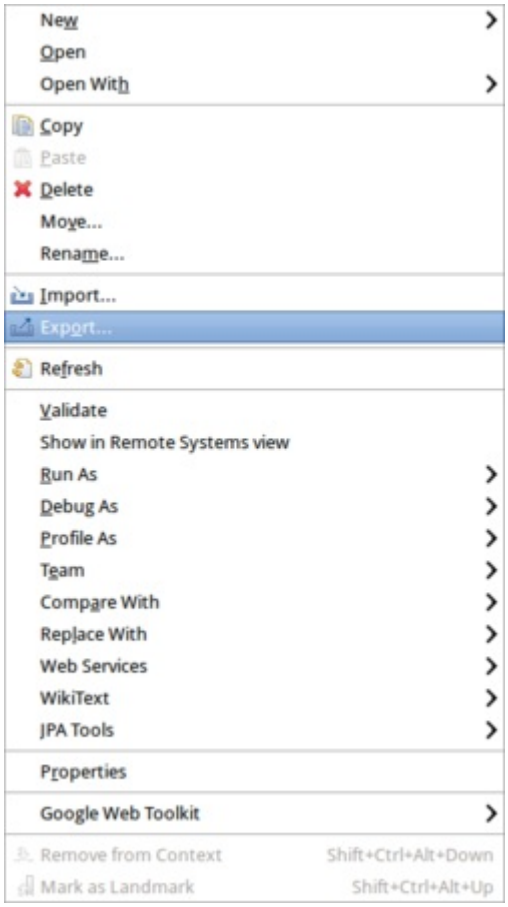
The following steps can be changed depending on the new Eclipse version changes.

1. ***Choose the Java file*** *from Project Explorer or Navigator view.*
2. ***Do right click*** *(will be shown contextual menu) and choose "export".*
3. *Choose* ***Java -> JAR file***.
4. *Select the* ***destination folder****, for example C:\openkm-dev\jar\your-class.jar.*
5. ***Click Next****.*
6. ***Click Next****.*
7. ***Optional step****. If your java class implements "**public static void main(String[] args)**" then you must choose your java class as the class of the entry point.*
8. ***Click Finish****.*

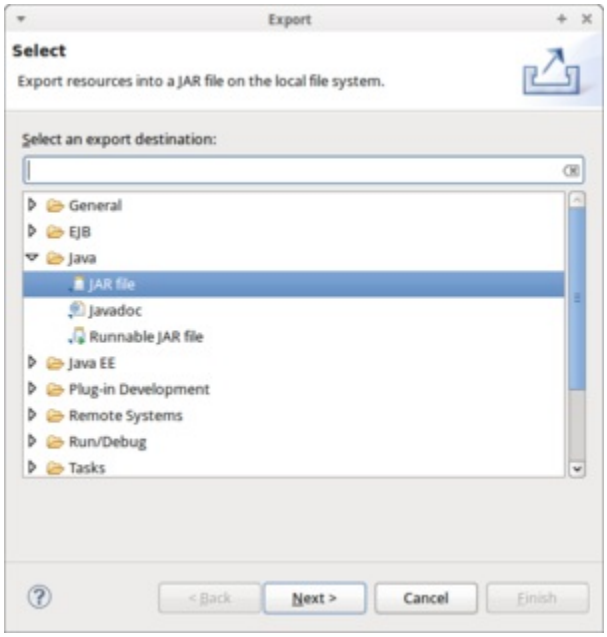
Images

Select the file to be exported as a jar. Right-click on the mouse and select " ***export****" option.*

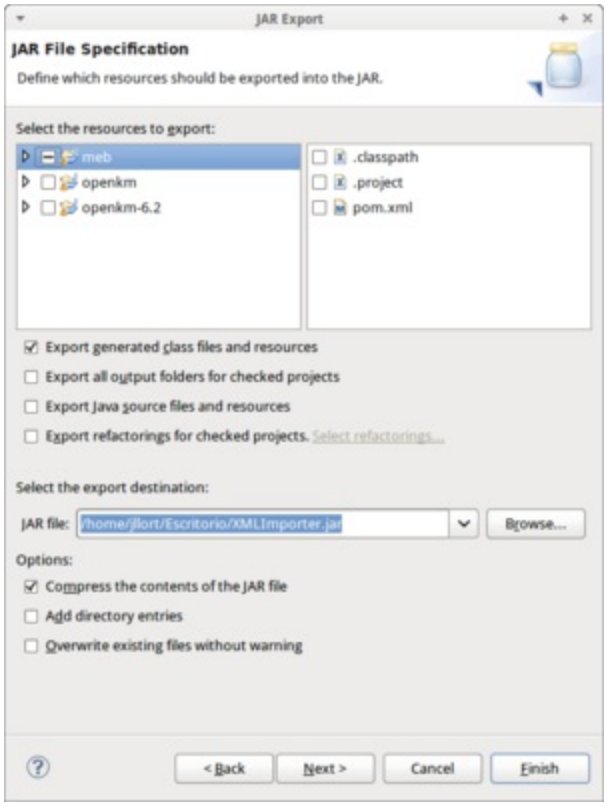
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



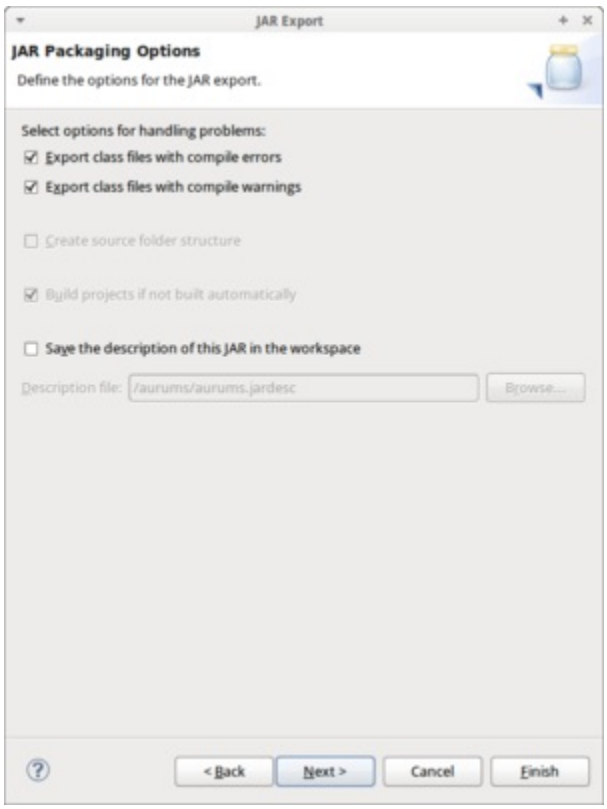
*Select "**.jar**" file option and click on "**next**" button.*



*Select the "**JAVA classes to be included**" into the jar file and click on "**next**" button.*

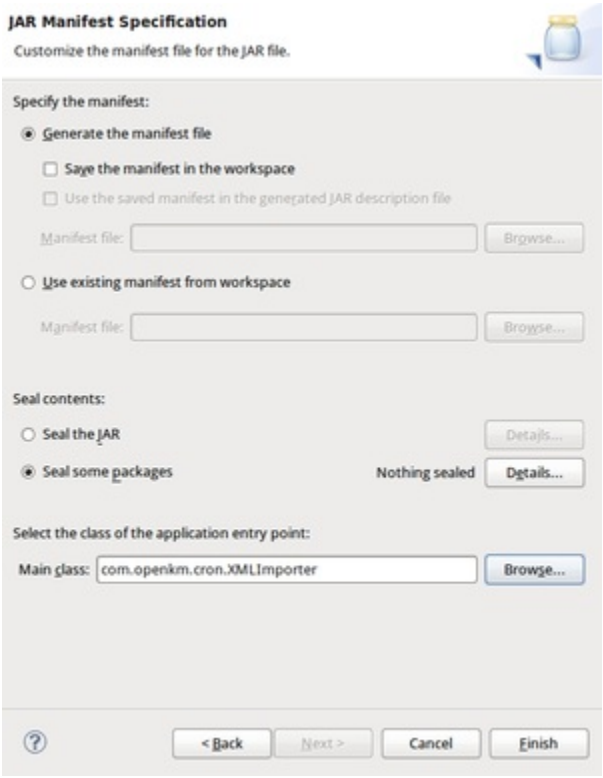


*Click on "**next**" button.*



*Select the "**Main class**" and click on "**Finish**" button.*

> *If your JAVA class implements "**public static void main(String[] args)**" then you must choose your java Main class as the class of the entry point, otherwise you can skip this step.*



Sample JAR file

Requisites:

- *Must implement the class "**public static void main(String[] args)**".*

```
package com.openkm;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Test {
    private static Logger log = LoggerFactory.getLogger(Test.class);

    public static void main(String[] args) {
        System.out.println("Test sample");
    }
}
```

Creating plugins

OpenKM plugin system helps you to add software components, increasing the application's features.



*OpenKM plugin system is based on **hot deploy** of JAR packages.*

From the OpenKM plugin system, you can:

- *Register new Antivirus analyzers.*
- *Register new Automation actions.*
- *Register new Automation validations.*
- *Register a new cron.*
- *Register new default form values.*
- *Register a new document converter.*
- *Register a new autocomplete form.*
- *Register new form validators.*
- *Register a new node paginator.*
- *Register new NodeProperties.*
- *Register new OCR validation controls (new OCR control points used to identify documents).*
- *Register new OCR data capture (new OCR parsers to retrieve data).*
- *Register new option select values (retrieve list values from third-party software).*
- *Register new principal adapters (connectors to retrieve users and roles from third-party software).*
- *Register a new REST plugin (extend the REST API).*
- *Register a new search plugin (extend the search features, for example, connecting with third-party tools)*
- *Register a new suggestion analyzer (build your own logic to suggest form values).*
- *Register a new text extractor.*
- *Registera new version number adapter (create your own document version number).*
- *Register a new security access manager*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Register a new plugin

- *To install new plugin **create a jar file** and **copy into your $TOMCAT/plugins** folder. More information at Creating a JAR file with Eclipse (you also can build your jar files with Maven).*
- *Go to **Administration > Utilities > Plugins** and click at top right the  Reload plugins button.*
- *In **plugins table will be shown** the new plugin.*



A plugin configuration needs to change Configuration parameters or restart application. For example Creating your own Principal Adapter case.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own AI prompt plugin

You can create your own AI prompt.

Conditions:

- *The new AI prompt class **must implement the "AIPrompt"** interface.*
- *The new Automation Action class **must be declared under the package "com.openkm.plugin.ai".***
- *The new Automation Action class **must be annotated with "@PluginImplementation".***
- *The new Automation Action class **must extend of "BasePlugin".***

Automation Action interface:

```
package com.openkm.plugin.ai;

import com.openkm.ws.common.util.AIResult;
import net.xeoh.plugins.base.Plugin;

public interface AIPrompt extends Plugin {
 String getName();

 AIResult executeNodePrompt(String uuid) throws AIPromptException;

 AIResult executeTextPrompt(String text) throws AIPromptException;

 String[] getContentTypes();

 boolean isManageNodePrompt();

 boolean isManageTextPrompt();
}
```

*The **new class must be loaded into the package com.openkm.plugin.ai** because the application plugins system will try to load from there.*



*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*



More information about Register a new plugin.

Methods description

Method	*Type*	*Description*
AIResult executeNodePrompt(String	*void*	*The method executes an AI prompt on a specific node.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

uuid)		
AIResult executeTextPrompt(String text)	*void*	*The method executes an AI prompt on a specific text.*
getName()	*String*	*Sets the name shown in the user interface selector list.*
getContentTypes()	*String[]*	*Returns a list of MIME types for documents in which prompt execution is allowed. If the list is empty, the prompt can be applied to any document from which content can be extracted in text format.* *This restriction applies only to AI prompts for nodes.*
isManageNodePrompt()	*boolean*	*When the value is true, it indicates that the plugin can be applied to nodes.*
isManageTextPrompt()	*boolean*	*When the value is true, it indicates that the plugin can be applied to text.*

*In the following examples, we recommend **paying attention to the calls** made to the methods of the **OpenAIUtils** class. Currently, prompts can be executed with either **text** or **images**. These are the only two allowed formats.*

Image to text sample

***SampleImageToTextAIPrompt** class:*

```
package com.openkm.plugin.ai;

import com.openkm.ai.OpenAIUtils;
import com.openkm.bean.Document;
import com.openkm.core.MimeTypeConfig;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.ws.common.util.AIResult;
import com.vladsch.flexmark.html.HtmlRenderer;
import com.vladsch.flexmark.parser.Parser;
import com.vladsch.flexmark.util.ast.Node;
import com.vladsch.flexmark.util.data.MutableDataSet;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import java.io.InputStream;

@PluginImplementation
public class SampleImageToTextAIPrompt extends BasePlugin implements AIPrompt {
 private static final Logger log = LoggerFactory.getLogger(SampleImageToTextAIPrompt.class);

 @Autowired
 private DbDocumentModule dbDocumentModule;

 @Autowired
 private OpenAIUtils openAIUtils;

 @Override
 public String getName() {
  return "Image to Text";
 }

 @Override
 public AIResult executeNodePrompt(String uuid) throws AIPromptException {
  log.debug("executeNodePrompt({})", uuid);

  try {
   Document doc = dbDocumentModule.getProperties(null, uuid);
   AIResult result = new AIResult();
   String text;

   try (InputStream is = dbDocumentModule.getContent(null, uuid, false)) {
    text = openAIUtils.extractTextFromImage(doc.getMimeType(), IOUtils.toByteArray(is));
   }

   if (StringUtils.isNotEmpty(text)) {
    result.setText(text);

    // HTML response
    MutableDataSet options = new MutableDataSet();
    Parser parser = Parser.builder(options).build();
    HtmlRenderer renderer = HtmlRenderer.builder(options).build();
    Node document = parser.parse(text);
    result.setHtml(renderer.render(document));
   } else {
    result.setError("Not being able to extract text from image");
    result.setErrorCode(AIResult.ErrorCodes.OTHER);
   }

   return result;
  } catch (Exception e) {
   throw new AIPromptException(e);
  }
 }

 @Override
 public AIResult executeTextPrompt(String text) throws AIPromptException {
  throw new AIPromptException("Not implemented");
 }

 @Override
 public String[] getContentTypes() {
  return new String[]{MimeTypeConfig.MIME_JPEG, MimeTypeConfig.MIME_PNG, MimeTypeConfig.MIME_GIF};
 }

 @Override
 public boolean isManageNodePrompt() {
  return true;
 }

 @Override
 public boolean isManageTextPrompt() {
```

```
  return false;
 }
}
```

Extract metadata sample

Invoice metadata XML definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.11//EN"
        "https://www.openkm.com/dtd/property-groups-3.11.dtd">
<property-groups>
  <property-group label="Invoice" name="okg:invoice">
    <input label="Invoice Number" name="okp:invoice.number"/>
    <input label="Invoice Date" type="date" name="okp:invoice.date"/>
    <input label="Invoice Amount" name="okp:invoice.amount"/>
    <input label="Customer Name" name="okp:invoice.customer_name"/>
    <input label="Customer CIF" name="okp:invoice.customer_cif"/>
    <textarea label="Customer Address" name="okp:invoice.customer_address"/>
  </property-group>
</property-groups>
```

SampleMetadataAIPrompt *class:*

```
package com.openkm.plugin.ai;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.openkm.ai.OpenAIUtils;
import com.openkm.core.MimeTypeConfig;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.module.db.DbPropertyGroupModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.ISO8601;
import com.openkm.ws.common.util.AIResult;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.HashMap;
import java.util.Map;

@PluginImplementation
public class SampleMetadataAIPrompt extends BasePlugin implements AIPrompt {
 private static final Logger log = LoggerFactory.getLogger(SampleMetadataAIPrompt.class);
 private static final ObjectMapper objectMapper = new ObjectMapper();
 private static final String METADATA_GROUP = "okg:invoice";

 @Autowired
 private DbPropertyGroupModule dbPropertyGroupModule;

 @Autowired
 private DbDocumentModule dbDocumentModule;

 @Autowired
 private OpenAIUtils openAIUtils;

 @Override
```

```
public String getName() {
 return "Sample Metadata AI";
}

@Override
public AIResult executeNodePrompt(String uuid) throws AIPromptException {
 log.debug("executePrompt({})", uuid);

 try {
  AIResult result = new AIResult();
  String text = dbDocumentModule.getExtractedText(null, uuid);
  String json = "";

  if (StringUtils.isNotEmpty(text)) {
   json = openAIUtils.extractMetadataFromText(text, InvoiceMetadataSimple.class);
  } else {
   result.setError("The document has no content or it has not been extracted yet");
   result.setErrorCode(AIResult.ErrorCodes.TEXT_EMPTY);
  }

  if (StringUtils.isNotEmpty(json)) {
   InvoiceMetadataSimple metadata = objectMapper.readValue(json, InvoiceMetadataSimple.class);
   log.info("JSON: {} => InvoiceMetadata: {}", json, metadata);
   StringBuilder sb = new StringBuilder();

   if (metadata.isValid()) {
    Map<String, String> props = metadata.getProperties();

    // Add or update metadata group
    if (dbPropertyGroupModule.hasGroup(null, uuid, METADATA_GROUP)) {
     dbPropertyGroupModule.setProperties(null, uuid, METADATA_GROUP, props);
     sb.append("Metadata updated for document");
    } else {
     dbPropertyGroupModule.addGroup(null, uuid, METADATA_GROUP, props);
     sb.append("Metadata added to document");
    }
   } else {
    sb.append("Invoice not detected");
   }

   // HTML response text response
   String html = "<p class=\"text-success\">" + sb + "</p>\n";
   html += metadata.toHtml();
   result.setHtml(html);

   // Plain text response
   sb.append("\n");
   sb.append(metadata.toText());
   result.setText(sb.toString());
  }

  return result;
 } catch (Exception e) {
  throw new AIPromptException(e);
 }
}

@Override
public AIResult executeTextPrompt(String text) throws AIPromptException {
 throw new AIPromptException("Not implemented");
}

@Override
public String[] getContentTypes() {
 return new String[]{MimeTypeConfig.MIME_TEXT};
}
```

```
 @Override
 public boolean isManageNodePrompt() {
  return true;
 }

 @Override
 public boolean isManageTextPrompt() {
  return false;
 }

 @JsonIgnoreProperties(ignoreUnknown = true)
 public static class InvoiceMetadataSimple {
  public String invoice_number;
  public String invoice_date;
  public String invoice_amount;
  public String customer_cif;
  public String customer_name;
  public String customer_address;

  public Map<String, String> getProperties() {
   Map<String, String> props = new HashMap<>();
   props.put("okp:invoice.number", this.invoice_number);
   props.put("okp:invoice.amount", this.invoice_amount);
   props.put("okp:invoice.customer_cif", this.customer_cif);
   props.put("okp:invoice.customer_name", this.customer_name);
   props.put("okp:invoice.customer_address", this.customer_address);

   if (ISO8601.parseBasic(this.invoice_date) != null) {
    props.put("okp:invoice.date", this.invoice_date);
   } else {
    log.warn("Wrong invoice date format: {}", this.invoice_date);
   }

   return props;
  }

  public boolean isValid() {
   return StringUtils.isNotEmpty(this.invoice_number)
     && StringUtils.isNotEmpty(this.invoice_date)
     && StringUtils.isNotEmpty(this.invoice_amount);
  }

  public String toHtml() {
   StringBuilder builder = new StringBuilder();
   builder.append("<ul>");
   builder.append("<li><b>invoice_number</b>: ").append(invoice_number).append("</li>\n");
   builder.append("<li><b>invoice_date</b>: ").append(invoice_date).append("</li>\n");
   builder.append("<li><b>invoice_amount</b>: ").append(invoice_amount).append("</li>\n");
   builder.append("<li><b>customer_cif</b>: ").append(customer_cif).append("</li>\n");
   builder.append("<li><b>customer_name</b>: ").append(customer_name).append("</li>\n");
   builder.append("<li><b>customer_address</b>: ").append(customer_address).append("</li>\n");
   builder.append("</ul>");
   return builder.toString();
  }

  public String toText() {
   StringBuilder builder = new StringBuilder();
   builder.append("invoice_number: ").append(invoice_number).append("\n");
   builder.append("invoice_date: ").append(invoice_date).append("\n");
   builder.append("invoice_amount: ").append(invoice_amount).append("\n");
   builder.append("customer_cif: ").append(customer_cif).append("\n");
   builder.append("customer_name: ").append(customer_name).append("\n");
   builder.append("customer_address: ").append(customer_address).append("\n");
   return builder.toString();
  }
 }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Basic summarization sample

***SampleTextAIPrompt** class:*

```
package com.openkm.plugin.ai;

import com.openkm.ai.OpenAIUtils;
import com.openkm.core.MimeTypeConfig;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.ws.common.util.AIResult;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

@PluginImplementation
public class SampleTextAIPrompt extends BasePlugin implements AIPrompt {
 private static final Logger log = LoggerFactory.getLogger(SampleTextAIPrompt.class);
 private static final String PROMPT = "Summarize the provided text responding in text language using the content. "
   "Returned response in plain text with a maximum of 1000 characters.";

 @Autowired
 private DbDocumentModule dbDocumentModule;

 @Autowired
 private OpenAIUtils openAIUtils;

 @Override
 public String getName() {
  return "Sample Text AI";
 }

 @Override
 public AIResult executeNodePrompt(String uuid) throws AIPromptException {
  log.debug("executeNodePrompt({})", uuid);

  try {
   AIResult result = new AIResult();
   String text = dbDocumentModule.getExtractedText(null, uuid);

   if (StringUtils.isNotEmpty(text)) {
    return generateSummary(text);
   } else {
    result.setError("The document has no content or it has not been extracted yet");
    result.setErrorCode(AIResult.ErrorCodes.TEXT_EMPTY);
   }

   return result;
  } catch (Exception e) {
   throw new AIPromptException(e);
  }
 }

 @Override
 public AIResult executeTextPrompt(String text) throws AIPromptException {
  log.debug("executeTextPrompt({})", text);

  try {
   return generateSummary(text);
  } catch (Exception e) {
```

```
    throw new AIPromptException(e);
   }
  }

  /**
   * Generate summary from text
   */
  private AIResult generateSummary(String text) throws Exception {
   AIResult result = new AIResult();
   String summary = "";

   if (StringUtils.isNotEmpty(text)) {
    summary = openAIUtils.processText(PROMPT, text);
   }

   if (StringUtils.isNotEmpty(summary)) {
    // HTML response text response
    String html = "<p>" + summary + "</p>\n";
    result.setHtml(html);

    // Plain text response
    result.setText(summary);
   }

   return result;
  }

  @Override
  public String[] getContentTypes() {
   return new String[]{MimeTypeConfig.MIME_TEXT};
  }

  @Override
  public boolean isManageNodePrompt() {
   return true;
  }

  @Override
  public boolean isManageTextPrompt() {
   return true;
  }
 }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Antivirus Analyzer plugin

You can create your own Antivirus Analyzer.

Conditions:

- *The new Antivirus Analyzer class* ***must implement the "Antivirus"*** *interface.*
- *The new Antivirus Analyzer class* ***must be declared under the package "com.openkm.plugin.antivirus".***
- *The new Antivirus Analyzer class* ***must be annotated with "@PluginImplementation"****.*
- *The new Antivirus Analyzer class* ***must extend of "BasePlugin".***

Antivirus Analyzer interface:

```
package com.openkm.plugin.antivirus;

import java.io.File;

import net.xeoh.plugins.base.Plugin;

/**
 * Antivirus
 */
public interface Antivirus extends Plugin {

 /**
  * Check for viruses in file
  */
 String detect(File file);
}
```

The ***new class must be loaded into the package com.openkm.plugin.antivirus*** *because the application plugins system will try to load it from there.*



Do not miss the tag ***@PluginImplementation*** *otherwise the application plugin system will not be able to retrieve the new class.*



More information at: Register a new plugin.

Method description

Method	*Type*	*Description*
detect(File file)	*String*	*The method check the file for viruses. In case the retun value will not be null the application will consider it has been found some virus.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example of the clamav antivirus analyzer implementation

```
package com.openkm.plugin.antivirus;

import com.openkm.core.Config;
import com.openkm.plugin.BasePlugin;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.io.IOUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.File;
import java.io.IOException;

/**
 * Default implementation of an antivirus.
 *
 */
@PluginImplementation
public class ClamavAntivirus extends BasePlugin implements Antivirus {

    private static Logger log = LoggerFactory.getLogger(ClamavAntivirus.class);

    /**
     * Check for viruses in file
     */
    @Override
    public String detect(File file) {
        try {
            // Performs virus check
            log.debug("CMD: " + Config.SYSTEM_ANTIVIR + " " + file.getPath());
            ProcessBuilder pb = new ProcessBuilder(Config.SYSTEM_ANTIVIR, "--no-summary", file.getPath());
            Process process = pb.start();
            process.waitFor();
            String info = IOUtils.toString(process.getInputStream());
            process.destroy();

            // Check return code
            if (process.exitValue() == 1) {
                log.warn(info);
                info = info.substring(info.indexOf(':') + 1);
                return info;
            } else {
                return null;
            }
        } catch (InterruptedException | IOException e) {
            log.warn("Failed to check for viruses", e);
        }

        return "Failed to check for viruses";
    }
}
```

Creating your own Automation Action

You can create your own Automation Action.

Conditions:

- *The new Automation Action class **must implement the "Action"** interface.*
- *The new Automation Action class **must be declared under the package "com.openkm.plugin.automation.action".***
- *The new Automation Action class **must be annotated with "@PluginImplementation"**.*
- *The new Automation Action class **must extend of "BasePlugin".***

Automation Action interface:

```
package com.openkm.plugin.automation;

import com.openkm.db.bean.AutomationRule.EnumEvents;
import net.xeoh.plugins.base.Plugin;

import java.util.List;
import java.util.Map;

public interface Action extends Plugin {

    void executePre(Map<String, Object> env, Object... params) throws AutomationException;

    void executePost(Map<String, Object> env, Object... params) throws AutomationException;

    String getName();

    String getParamType00();

    String getParamSrc00();

    String getParamDesc00();

    String getParamType01();

    String getParamSrc01();

    String getParamDesc01();

    String getParamType02();

    String getParamSrc02();

    String getParamDesc02();

    List<EnumEvents> getValidEventsAtPre();

    List<EnumEvents> getValidEventsAtPost();
}
```

*The **new class must be loaded into the package com.openkm.plugin.automation.action** because application plugins system will try to load from there.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*

More information about Register a new plugin.

Methods description

Method	*Type*	*Description*
executePre(Map<String, Object> env, Object... params)	*void*	*The method executed by the Automation event when the validation conditions succeed, on "pre" stage.*
executePost(Map<String, Object> env, Object... params)	*void*	*The method executed by the Automation event when the validation conditions succeed, on "post" stage.*
getName()	*String*	*Sets the name that will be shown in the administrator user interface selector list.*
getParamType00 ***getParamType01*** ***getParamType02***	*String*	*Sets the parameter type.* *Available values:* • *Automation.PARAM_TYPE_EMPTY* • *Automation.PARAM_TYPE_TEXT* • *Automation.PARAM_TYPE_INTEGER* • *Automation.PARAM_TYPE_BOOLEAN* • *Automation.PARAM_TYPE_TEXTAREA* • *Automation.PARAM_TYPE_CODE* • *Automation.PARAM_TYPE_USER* • *Automation.PARAM_TYPE_ROLE* • *Automation.PARAM_TYPE_OMR* *When value is set to Automation.PARAM_TYPE_EMPTY* *The parameters work as a logical group.* ***[getParamType00 + getParamSrc00 + getParamDesc00]***

getParamSrc00 ***getParamSrc01*** ***getParamSrc02***	*String*	*Sets the source type.* *Available values:* • *Automation.PARAM_SOURCE_EMPTY* • *Automation.PARAM_SOURCE_FOLDER* • *Automation.PARAM_SOURCE_OMR*
getParamDesc00 ***getParamDesc01*** ***getParamDesc02***	*String*	*The parameter description.*
getValidEventsAtPre()	*List<EnumEvents>*	*Return a list of events.* *The method returns a list of valid events for the action in pre.*
getValidEventsAtPost()	*List<EnumEvents>*	*Return a list of events.* *The method returns a list of valid events for the action in post.*

Understanding env variable

The env variable is present on isValid method, it is a Map of values injected by automation. These map values provide information about the node involved on the event and other related information.

> *It takes some time, until you take control of the AutomationUtils class, because they are centralized, the methods to retrieve data.*
>
> *Take a look at Variables by automation event to know what variables are available for each event.*

For example to retrieve the node that caused the event.

```
NodeBase node = automationUtils.getNodeToEvaluate(env);
```

Understanding params variable

The env variable is present on isValid method, it is an array of Objects. This array is filled by Automation event based on the number of parameters set on Action class (which will not be Automation.PARAM_TYPE_EMPTY value).

The array can have serveral Objects with a distinct type, it is necessary retrieve the parameters to the correct object type.

> *It takes some time, until you take control of AutomationUtils class, because they are centralized, the methods to retrieve parameters.*

For example to retrieve the first parameters as String object.

```
String param00 = AutomationUtils.getString(0, params);
```

Description of available ParamType values:

Property	*Description*
Automation.PARAM_TYPE_EMPTY	*Indicates an empty value.*
Automation.PARAM_TYPE_TEXT	*Indicates a text value will be required.*
Automation.PARAM_TYPE_INTEGER	*Indicates an integer value will be required.*
Automation.PARAM_TYPE_BOOLEAN	*Indicates a boolean value will be required.*
Automation.PARAM_TYPE_TEXTAREA	*Indicates a text area value will be required.*
Automation.PARAM_TYPE_CODE	*Indicates that code will be required.*
Automation.PARAM_TYPE_USER	*Indicates that a valid application user will be required.*
Automation.PARAM_TYPE_ROLE	*Indicates that a valid application role will be required.*
Automation.PARAM_TYPE_OMR	*Indicates that a valid OMR id - object mark recognition - will be required.*

Description of available Src values:

Property	*Description*

PARAM_SOURCE_EMPTY	*Indicates an empty source.*
PARAM_SOURCE_FOLDER	*Indicates that the source must be a folder.*
PARAM_SOURCE_OMR	*Indicates that the source must be a valid OMR - object mark recognition -.*

Detecting recursion

When you create your automation actions, should consider recursion case. For example if your action goes linked with "CREATE_DOCUMENT" event and into the action you create a new document you can go into an infinite loop. For it when you create a new Action should take special care of this cases.



Basic recursive samples:

- *Automation task linked with "**Create document**" event when into is set an **action** that also **creates a new document**.*
- *Automation task linked with "**Set metadata group**" event when into is set an **action** that also **changes metadata group values**.*

Complex recursive sample:

- *Automation task linked with "**Create document**" event when into is set an action that **changes metadata group values** and automation task linked with "**Set metadata group**" event when into is set an action that **creates a new document**.*

Code to detect recursive calls caused by automation action classes:

```
if (StackTraceUtils.isCallingMe(this.getClass().getName())) {
  log.info("Recursion detected");
  return;
}
```

Example

***AddKeyword** class:*

```
package com.openkm.plugin.automation.action;

import com.openkm.core.Config;
import com.openkm.db.bean.Automation;
import com.openkm.db.bean.AutomationRule.EnumEvents;
import com.openkm.db.service.NodeBaseSrv;
import com.openkm.plugin.BasePlugin;
```

```
import com.openkm.plugin.automation.Action;
import com.openkm.plugin.automation.AutomationException;
import com.openkm.plugin.automation.AutomationUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.stream.Stream;

/**
 * AddKeyword
 */
@PluginImplementation
public class AddKeyword extends BasePlugin implements Action {

    private static ArrayList<EnumEvents> EVENTS_AT_PRE = new ArrayList<>();

    private static ArrayList<EnumEvents> EVENTS_AT_POST
            = Stream.of(EnumEvents.EVENT_DOCUMENT_CREATE, EnumEvents.EVENT_DOCUMENT_UPDATE, E
                    EnumEvents.EVENT_DOCUMENT_RENAME, EnumEvents.EVENT_DOCUMENT_MOVE, EnumEven
                    EnumEvents.EVENT_DOCUMENT_DOWNLOAD_FROM_UI, EnumEvents.EVENT_DOCUMENT_DO
                    EnumEvents.EVENT_DOCUMENT_PRINT, EnumEvents.EVENT_DOCUMENT_STAMP, EnumEvents
                    EnumEvents.EVENT_RECORD_DELETE, EnumEvents.EVENT_FOLDER_CREATE, EnumEvents.EV
                    EnumEvents.EVENT_MAIL_CREATE, EnumEvents.EVENT_MAIL_DOWNLOAD_FROM_UI, EnumEv
                    EnumEvents.EVENT_TEXT_EXTRACTOR, EnumEvents.EVENT_PROPERTY_GROUP_ADD, EnumE
                    EnumEvents.EVENT_PROPERTY_GROUP_REMOVE, EnumEvents.EVENT_MAIL_IMPORT_ATTAC
            ).collect(Collectors.toCollection(ArrayList::new));

    @Autowired
    private NodeBaseSrv nodeBaseSrv;

    @Autowired
    private AutomationUtils automationUtils;

    @Override
    public void executePre(Map<String, Object> env, Object... params) throws AutomationException {
    }

    @Override
    public void executePost(Map<String, Object> env, Object... params) throws AutomationException {
        try {
            String keyword = automationUtils.getString(0, params);
            String uuid = automationUtils.getNodeToEvaluate(env).getUuid();

            if (uuid != null && keyword != null && !keyword.isEmpty()) {
                if (Config.SYSTEM_KEYWORD_LOWERCASE) {
                    keyword = keyword.toLowerCase();
                }

                nodeBaseSrv.addKeyword(uuid, keyword);
            }
        } catch (Exception e) {
            throw new AutomationException("AddKeyword exception", e);
        }
    }

    @Override
    public String getName() {
        return "AddKeyword";
    }

    @Override
    public String getParamType00() {
        return Automation.PARAM_TYPE_TEXT;
```

```
    }

    @Override
    public String getParamSrc00() {
        return Automation.PARAM_SOURCE_EMPTY;
    }

    @Override
    public String getParamDesc00() {
        return "Keyword";
    }

    @Override
    public String getParamType01() {
        return Automation.PARAM_TYPE_EMPTY;
    }

    @Override
    public String getParamSrc01() {
        return Automation.PARAM_SOURCE_EMPTY;
    }

    @Override
    public String getParamDesc01() {
        return "";
    }

    @Override
    public String getParamType02() {
        return Automation.PARAM_TYPE_EMPTY;
    }

    @Override
    public String getParamSrc02() {
        return Automation.PARAM_SOURCE_EMPTY;
    }

    @Override
    public String getParamDesc02() {
        return "";
    }

    @Override
    public List<EnumEvents> getValidEventsAtPre() {
        return EVENTS_AT_PRE;
    }

    @Override
    public List<EnumEvents> getValidEventsAtPost() {
        return EVENTS_AT_POST;
    }
}
```

Creating your own Automation Validation

You can create your own Automation Validation.

Conditions:

- *The new Automation Validation class* ***must implement the "Validation"*** *interface.*
- *The new Automation Validation class* ***must be declared under the package "com.openkm.plugin.automation.validation".***
- *The new Automation Validation class* ***must be annotated with "@PluginImplementation"***.
- *The new Automation Validation class* ***must extend of "BasePlugin".***

Automation Validation interface:

```
package com.openkm.plugin.automation;

import com.openkm.db.bean.AutomationRule.EnumEvents;
import net.xeoh.plugins.base.Plugin;

import java.util.List;
import java.util.Map;

public interface Validation extends Plugin {

    String METHOD = "isValid";

    boolean isValid(Map<String, Object> env, Object... params) throws AutomationException;

    String getName();

    String getParamType00();

    String getParamSrc00();

    String getParamDesc00();

    String getParamType01();

    String getParamSrc01();

    String getParamDesc01();

    String getParamType02();

    String getParamSrc02();

    String getParamDesc02();

    List<EnumEvents> getValidEventsAtPre();

    List<EnumEvents> getValidEventsAtPost();
}
```

The ***new class must be loaded into the package com.openkm.plugin.automation.validation*** *because application plugins system*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

will try to load from there.

*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*

More information about Register a new plugin.

Methods description

Method	*Type*	*Description*
isValid(Map<String, Object> env, Object... params)	*boolean*	*The method executed by the Automation event to evaluate if conditions are true or false.*
getName()	*String*	*Sets the name that will be shown in the administrator user interface selector list.*
getParamType00 ***getParamType01*** ***getParamType02***	*String*	*Sets the parameter type.* *Available values:* • *Automation.PARAM_TYPE_EMPTY* • *Automation.PARAM_TYPE_TEXT* • *Automation.PARAM_TYPE_INTEGER* • *Automation.PARAM_TYPE_BOOLEAN* • *Automation.PARAM_TYPE_TEXTAREA* • *Automation.PARAM_TYPE_CODE* • *Automation.PARAM_TYPE_USER* • *Automation.PARAM_TYPE_ROLE* • *Automation.PARAM_TYPE_OMR* *When the value is set to Automation.PARAM_TYPE_EMPTY* *The parameters work as a logical group.* ***[getParamType00 + getParamSrc00 + getParamDesc00]***

getParamSrc00 ***getParamSrc01*** ***getParamSrc02***	*String*	*Set the source type.* *Available values:* • *Automation.PARAM_SOURCE_EMPTY* • *Automation.PARAM_SOURCE_FOLDER* • *Automation.PARAM_SOURCE_OMR*
getParamDesc00 ***getParamDesc00*** ***getParamDesc00***	*String*	*The parameter description.*
getValidEventsAtPre()	*List<EnumEvents>*	*Return a list of events.* *The method returns a list of valid events for the validation in pre.*
getValidEventsAtPost()	*List<EnumEvents>*	*Return a list of events.* *The method returns a list of valid events for the validation in post.*

Understanding env variable

The env variable is present on isValid method, it is a Map of values injected by automation. These map values provide information about the node involved on event and other related information.

> *It takes some time, until you take control of AutomationUtils class, because they are centralized, the methods for retrieve data.*
>
> *Take a look at Variables by automation event to know what variables are available for each event.*

For example to retrieve the node that caused the event.

```
NodeBase node = automationUtils.getNodeToEvaluate(env);
```

Understanding params variable

The env variable is present on isValid method, it is an array of Objects. This array is filled by Automation event based on the number of parameters set on Validation class (which will not be Automation.PARAM_TYPE_EMPTY value).

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

The array can have serveral Objects with a distinct type, it is necessary to retrieve the parameters to the correct object type.

> *It takes some time, until you take control of AutomationUtils class, because they are centralized, the methods to retrieve parameters.*

For example to retrive the first parameters as String object.

```
String param00 = AutomationUtils.getString(0, params);
```

Description of available ParamType values:

Property	*Description*
Automation.PARAM_TYPE_EMPTY	*Indicates an empty value.*
Automation.PARAM_TYPE_TEXT	*Indicates that a text value will be required.*
Automation.PARAM_TYPE_INTEGER	*Indicates that an integer value will be required.*
Automation.PARAM_TYPE_BOOLEAN	*Indicates that a boolean value will be required.*
Automation.PARAM_TYPE_TEXTAREA	*Indicates that a text area value will be required.*
Automation.PARAM_TYPE_CODE	*Indicates that a code will be required.*
Automation.PARAM_TYPE_USER	*Indicates that a valid application user will be required.*
Automation.PARAM_TYPE_ROLE	*Indicates that a valid application role will be required.*
Automation.PARAM_TYPE_OMR	*Indicates that a valid OMR id - object mark recognition - will be required.*

Description of available Src values:

Property	*Description*
PARAM_SOURCE_EMPTY	*Indicates an empty source.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

PARAM_SOURCE_FOLDER	*Indicates that the source must be a folder.*
PARAM_SOURCE_OMR	*Indicates that the source must be a valid OMR - object mark recognition -.*

Example

PathContains *class:*

```
package com.openkm.plugin.automation.validation;

import com.openkm.api.OKMRepository;
import com.openkm.db.bean.Automation;
import com.openkm.db.bean.AutomationRule.EnumEvents;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.service.NodeBaseSrv;
import com.openkm.plugin.BasePlugin;
import com.openkm.plugin.automation.AutomationException;
import com.openkm.plugin.automation.AutomationUtils;
import com.openkm.plugin.automation.Validation;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.stream.Stream;

/**
 * Check if the current parent path contains a designed one. The only parameter
 * is a path and will test if this one is included in the actual parent.
 */
@PluginImplementation
public class PathContains extends BasePlugin implements Validation {

    private static ArrayList<EnumEvents> EVENTS_AT_PRE
            = Stream.of(EnumEvents.EVENT_DOCUMENT_CREATE, EnumEvents.EVENT_DOCUMENT_UPDATE, E
                    EnumEvents.EVENT_DOCUMENT_RENAME, EnumEvents.EVENT_DOCUMENT_MOVE, EnumEven
                    EnumEvents.EVENT_DOCUMENT_DOWNLOAD_FROM_UI, EnumEvents.EVENT_DOCUMENT_DO
                    EnumEvents.EVENT_DOCUMENT_PRINT, EnumEvents.EVENT_DOCUMENT_STAMP, EnumEvents
                    EnumEvents.EVENT_RECORD_DELETE, EnumEvents.EVENT_FOLDER_CREATE, EnumEvents.EV
                    EnumEvents.EVENT_MAIL_CREATE, EnumEvents.EVENT_MAIL_DOWNLOAD_FROM_UI, EnumEv
                    EnumEvents.EVENT_TEXT_EXTRACTOR, EnumEvents.EVENT_PROPERTY_GROUP_ADD, EnumE
                    EnumEvents.EVENT_PROPERTY_GROUP_REMOVE
            ).collect(Collectors.toCollection(ArrayList::new));

    private static ArrayList<EnumEvents> EVENTS_AT_POST
            = Stream.of(EnumEvents.EVENT_DOCUMENT_CREATE, EnumEvents.EVENT_DOCUMENT_UPDATE, E
                    EnumEvents.EVENT_DOCUMENT_RENAME, EnumEvents.EVENT_DOCUMENT_MOVE, EnumEven
                    EnumEvents.EVENT_DOCUMENT_DOWNLOAD_FROM_UI, EnumEvents.EVENT_DOCUMENT_DO
                    EnumEvents.EVENT_DOCUMENT_PRINT, EnumEvents.EVENT_DOCUMENT_STAMP, EnumEvents
                    EnumEvents.EVENT_RECORD_DELETE, EnumEvents.EVENT_FOLDER_CREATE, EnumEvents.EV
                    EnumEvents.EVENT_MAIL_CREATE, EnumEvents.EVENT_MAIL_DOWNLOAD_FROM_UI, EnumEv
                    EnumEvents.EVENT_TEXT_EXTRACTOR, EnumEvents.EVENT_PROPERTY_GROUP_ADD, EnumE
                    EnumEvents.EVENT_PROPERTY_GROUP_REMOVE, EnumEvents.EVENT_MAIL_IMPORT_ATTAC
            ).collect(Collectors.toCollection(ArrayList::new));

    @Autowired
    private AutomationUtils automationUtils;
```

```
@Autowired
private OKMRepository okmRepository;

@Autowired
private NodeBaseSrv nodeBaseSrv;

@Override
public boolean isValid(Map<String, Object> env, Object... params) throws AutomationException {
    try {
        String fldUuid = automationUtils.getString(0, params);
        NodeBase node = automationUtils.getNodeToEvaluate(env);
        String parentPath = nodeBaseSrv.getPathFromUuid(node.getParent());
        String fldPath = okmRepository.getNodePath(null, fldUuid);

        if (parentPath.startsWith(fldPath)) {
            return true;
        } else {
            return false;
        }
    } catch (Exception e) {
        throw new AutomationException("PathContains exception", e);
    }
}

@Override
public String getName() {
    return "PathContains";
}

@Override
public String getParamType00() {
    return Automation.PARAM_TYPE_TEXT;
}

@Override
public String getParamSrc00() {
    return Automation.PARAM_SOURCE_FOLDER;
}

@Override
public String getParamDesc00() {
    return "String";
}

@Override
public String getParamType01() {
    return Automation.PARAM_TYPE_EMPTY;
}

@Override
public String getParamSrc01() {
    return Automation.PARAM_SOURCE_EMPTY;
}

@Override
public String getParamDesc01() {
    return "";
}

@Override
public String getParamType02() {
    return Automation.PARAM_TYPE_EMPTY;
}

@Override
public String getParamSrc02() {
```

```
        return Automation.PARAM_SOURCE_EMPTY;
    }

    @Override
    public String getParamDesc02() {
        return "";
    }

    @Override
    public List<EnumEvents> getValidEventsAtPre() {
        return EVENTS_AT_PRE;
    }

    @Override
    public List<EnumEvents> getValidEventsAtPost() {
        return EVENTS_AT_POST;
    }
}
```

Creating your own Crontab plugin

You can create your own Antivirus Analyzer.

Conditions:

- *The new Antivirus Analyzer class* ***must implement the "CronAdapter"*** *interface.*
- *The new Antivirus Analyzer class* ***must be declared under the package "com.openkm.plugin.cron".***
- *The new Antivirus Analyzer class* ***must be annotated with "@PluginImplementation".***
- *The new Antivirus Analyzer class* ***must extend of "BaseCronPlugin".***

Antivirus Analyzer interface:

```
package com.openkm.plugin.cron;

import net.xeoh.plugins.base.Plugin;

public interface CronAdapter extends Runnable, Plugin {
 /**
  * The cron name
  */
 String getName();

 /**
  * The default cron expression what will be registered the first time
  */
 String getCronExpression();

 /**
  * Return if task is running or not
  */
 boolean isRunning();

 /**
  * Cron task execution
  */
 void execute();
}
```

The ***new class must be loaded into the package com.openkm.plugin.cron*** *because the application plugins system will try to load it from there.*



Do not miss the tag ***@PluginImplementation*** *otherwise, the application plugin system will not be able to retrieve the new class.*



More information at Register a new plugin.

Method description

Method	Type	Description
getName()	String	Return the name of the crontab.
getCronExpression	String	Return the cron expression.
isRunning	Boolean	Return true when cron task is running.
execute	Void	Execute the cron task.

Example of the disposition cron implementation

```
package com.openkm.plugin.cron;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.util.FileLogger;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;


@PluginImplementation
public class TestCron extends BaseCronPlugin implements CronAdapter {
    private static Logger log = LoggerFactory.getLogger(TestCron.class);
    private static String LOG_NAME = "TestCron";

    @Autowired
    private OKMRepository okmRepository;

    @Override
    public String getName() {
        return "Cron test";
    }

    @Override
    public String getCronExpression() {
        return "0 * * * * *";
    }

    @Override
    public void execute() {
        try {
            FileLogger.info(LOG_NAME, ">>> Started");
            // Use always system token in your cron
            String token = DbSessionManager.getInstance().getSystemToken();
            Folder rootFolder = okmRepository.getRootFolder(token);
            FileLogger.info(LOG_NAME, "Root folder uuid: " + rootFolder.getUuid());
            FileLogger.info(LOG_NAME, ">>> Finished");
        } catch (Exception e) {
            log.error(e.getMessage());
        }
    }
}
```

Understanding the crontab job execution context

Execution context:

- *Crontab job is executed by internal users named "system" (* ***the job has not a valid user session*** *).*
- *Crontab job is executed by the administrator user (* ***the job has a valid user session)*** *.*



All API methods have a variable named token.

When ***a user is logged*** *the call to the API use* ***token value to null****. That indicates the application must use the logged session.*

When ***a user is not logged****, the call to the API must use a* ***not null token****, otherwise will get an error.*

Example of use:

- *logged user ->* ***valid*** *API call -> OKMFolder.getInstance().createSimple(null, "/okm:root/test");*
- *not logged user ->* ***invalid*** *API call -> OKMFolder.getInstance().createSimple(****null****, "/okm:root/test");*

Use the system token is a good practice when calling API methods into crontab task:

```
String systemToken = DbSessionManager.getInstance().getSystemsystemToken();
OKMFolder.getInstance().createSimple(systemToken, "/okm:root/test")
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Autocomplete Form plugin

Available from the OpenKM version 7.1.41 and the dtd version 3.11

You can create your own autocomplete form.

Conditions:

- *The new Default Form Values class* ***must implement the "AutocompleteFormValues"*** *interface.*
- *The new Default Form Values class* ***must be declared under the package "com.openkm.plugin.form.values".***
- *The new Default Form Values class* ***must be annotated with "@PluginImplementation".***
- *The new Default Form Values class* ***must extend of "BasePlugin".***

Default Autocomplete Form interface:

```
package com.openkm.plugin.form.values;

import com.openkm.bean.form.FormElement;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

public interface AutocompleteFormValues extends Plugin {
 String getName();

 void autocompleteFormValues(String uuid, String grpName, List<FormElement> formElementList);
}
```

The ***new class must be loaded into the package com.openkm.plugin.form.values*** *because the application plugins system will try to load it from there. See the sample below:*

Do not miss the tag ***@PluginImplementation*** *otherwise the application plugin system will not be able to retrieve the new class.*

More information at: Register a new plugin

Method description

Method	*Type*	*Description*
autocompleteFormValues(String uuid, String grpName, List<FormElement> formElementList)	*void*	*The method autocompletes the form elements of the group.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example of the default form values implementation

```
package com.openkm.plugin.form.values;

import com.openkm.bean.form.FormElement;
import com.openkm.bean.form.HorizontalPanel;
import com.openkm.bean.form.Input;
import com.openkm.bean.form.VerticalPanel;
import com.openkm.plugin.BasePlugin;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.List;

@PluginImplementation
public class SampleAutocompleteFormValues extends BasePlugin implements AutocompleteFormValues {
 private static final Logger log = LoggerFactory.getLogger(SampleAutocompleteFormValues.class);

 @Override
 public String getName() {
  return "Sample autocomplete form values implementation";
 }

 @Override
 public void autocompleteFormValues(String uuid, String grpName, List<FormElement> formElementList) {
  log.info("autocompleteFormValues({}, {}, {})", uuid, grpName, formElementList);
  autocomplete(formElementList);
 }

 private void autocomplete(List<FormElement> formElementList) {
  for (FormElement formElement : formElementList) {
   if (formElement instanceof Input) {
    Input input = (Input) formElement;
    if (StringUtils.isEmpty(input.getValue())) {
     input.setValue("Autocomplete with name: " + formElement.getName());
    }
   } else if (formElement instanceof HorizontalPanel) {
    HorizontalPanel hPanel = (HorizontalPanel) formElement;
    autocomplete(hPanel.getElements());
   } else if (formElement instanceof VerticalPanel) {
    VerticalPanel vPanel = (VerticalPanel) formElement;
    autocomplete(vPanel.getElements());
   }
  }
 }
}
```

Metadata definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.11//EN"
    "http://www.openkm.com/dtd/property-groups-3.11.dtd">
<property-groups>
    <property-group label="Autocomple sample" name="okg:autocomplete" autocompleteValueClassName="com.op
        <input label="Input label first" name="okp:autocomplete.first" type="text"/>
        <hpanel name="okp:autocomplete.horizontal1">
            <input label="Input label 1" name="okp:autocomplete.input_h1" />
            <input label="Input label 2" name="okp:autocomplete.input_h2" />
        </hpanel>
    </property-group>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
</property-groups>
```

Metadata Extraction using AI

If a document contains extracted text and the artificial intelligence parameters are properly configured in the OpenKM application settings, metadata can be automatically extracted using the following autocomplete plugin:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.11//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.11.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting" autocompleteValueClassName="com.openkm.plugin.fo
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Default Form Values

Deprecated as of version 8.1.15, we suggest using Creating your own FormInterceptor plugin

From OpenKM version 7.1.38 the method changed by adding the uuid of the node. The old version of the method was:

```
void setDefaultValues(String grpName, List<FormElement> formElementList);
```

You can create your own Default form Values.

Conditions:

- *The new Default Form Values class* ***must implement the "FormDefaultValues"*** *interface.*
- *The new Default Form Values class* ***must be declared under the package "com.openkm.plugin.form.values".***
- *The new Default Form Values class* ***must be annotated with "@PluginImplementation".***
- *The new Default Form Values class* ***must extend of "BasePlugin".***

Default Form Values interface:

```
package com.openkm.plugin.form.values;

import com.openkm.bean.form.FormElement;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

public interface FormDefaultValues extends Plugin {

    String getName();

    void setDefaultValues(String uuid, String grpName, List<FormElement> formElementList);
}
```

The ***new class must be loaded into the package com.openkm.plugin.form.values*** *because the application plugins system will try to load it from there. See the sample below:*

Do not miss the tag ***@PluginImplementation*** *otherwise the application plugin system will not be able to retrieve the new class.*

More information at: Register a new plugin.

Method description

Method	*Type*	*Description*
setDefaultValues(String grpName, List<FormElement> formElementList)	*void*	*The method sets defaults values for the form elements of the group.*

Example of the default form values implementation

```
package com.openkm.plugin.form.values;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import com.openkm.bean.form.FormElement;
import com.openkm.bean.form.Input;
import com.openkm.plugin.BasePlugin;

import net.xeoh.plugins.base.annotations.PluginImplementation;

/**
 * Implementation of default form values
 */
@PluginImplementation
public class FormDefaultValuesDate extends BasePlugin implements FormDefaultValues {

    @Override
    public String getName() {
        return "Date Default Values";
    }

    @Override
    public void setDefaultValues(String uuid, String grpName, List<FormElement> formElementList) {
        if (grpName.equalsIgnoreCase("okg:test")) {
            for (FormElement element : formElementList) {
                if (element.getName().equals("okp:test.date")) {
                    Input input = (Input) element;
                    if (input.getValue().isEmpty()) {
                        DateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmss");
                        Date date = new Date();
                        input.setValue(dateFormat.format(date));
                    }
                }
            }
        }
    }

}
```

Metadata definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN" "http://www.openkm.com/dtd/p
<property-groups>
  <property-group label="Test" name="okg:test" defaultValueClassName="com.openkm.plugin.form.values.FormDe
    <input label="Fecha" type="date" name="okp:test.date"/>
  </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Document Converter

You can create your own Document Converter.

Conditions:

- *The new Document Converter class **must implement the "Converter"** interface.*
- *The new Document Converter class **must be declared under the package "com.openkm.plugin.conversion".***
- *The new Document Converter class **must be annotated with "@PluginImplementation"**.*
- *The new Document Converter class **must extend of "BasePlugin".***

Document Converter interface:

```
package com.openkm.plugin.conversion;

import com.openkm.core.ConversionException;
import net.xeoh.plugins.base.Plugin;

import java.io.File;
import java.io.IOException;
import java.util.List;

public interface Converter extends Plugin {

    int TO_PDF = 0;
    int TO_IMAGE = 1;

    void makeConversion(File input, String mimeType, File output) throws IOException, ConversionException;

    List<String> getSourceMimeTypes();

    int getConversionType();
}
```

*The **new class must be loaded into the package com.openkm.plugin.conversion** because application plugins system will try to load from there.*



*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*



More information at: Register a new plugin.

Methods description

Method	*Type*	*Description*
makeConversion(File input, String mimeType, File output)	*void*	*The method converts the file at input to some specific format into file output*

getSourceMimeTypes()	*List<String>*	*Return a list of valid document MIME TYPE allowed to be converted.*
getConversionType()	*int*	*Return the value of the MIME TYPE of that will be converted the input document.* *Allowed values are:* • *Converter.TO_PDF* • *Converter.TO_IMAGE* *When you choose Converter.TO_IMAGE is expected to retun an image in PNG format.*

Example of the Document converter implementation

```
package com.openkm.plugin.conversion;

import com.openkm.core.ConversionException;
import com.openkm.core.MimeTypeConfig;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.DocConverter;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

import java.io.File;
import java.io.IOException;
import java.util.Arrays;
import java.util.List;


@PluginImplementation
public class SqlToPdfConversion extends BasePlugin implements Converter {

    String[] sourceMimeTypes = new String[]{MimeTypeConfig.MIME_SQL};

    @Autowired
    private DocConverter docConverter;

    @Override
    public void makeConversion(File input, String mimeType, File output) throws IOException, ConversionException {
        docConverter.src2pdf(input, output, "sql");
    }

    @Override
    public List<String> getSourceMimeTypes() {
        return Arrays.asList(sourceMimeTypes);
    }

    @Override
    public int getConversionType() {
        return Converter.TO_PDF;
```

```
    }
  }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Form Validator plugin

Single field validation

Form validator are used to implement complex logic that must be evaluated from server side logic and are not covered by default Metadata Validator element.

You can create your own Form Validator

Conditions:

- *The new Form Validator class **must implement the "FieldValidator"** interface.*
- *The new Form Validator class **must be declared under the package "com.openkm.plugin.form.validator".***
- *The new Form Validator class **must be annotated with "@PluginImplementation".***
- *The new Form Validator class **must extend of "BasePlugin".***

Form Validator interface:

```
package com.openkm.plugin.form;

import net.xeoh.plugins.base.Plugin;

import java.util.List;

/**
 * FieldValidator
 */
public interface FieldValidator extends Plugin {

    String getName();

    String validate(String value, List<String> uuids);
}
```

*The **new class must be loaded into the package com.openkm.plugin.form.validator** because application plugins system will try to load from there.*



*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*



More information at Register a new plugin.

Methods description

Method	*Type*	*Description*

getName()	*String*	*The method returns the name of the plugin used in form list selector.*
validate(String value, List<String> uuids)	*String*	*When returned string is equals to "", application consider all goes right, otherwise text returned is the error.* *Value variable, contains the value of the metadata field and the uuids are the list of the uuid nodes affected (documents, folders, mails, records).* *In case will be updated only one node, common case, the list will only contains one uuid, in case multiple nodes selected will contains the list of all selected nodes where want to be applied the metadata changes.*

Example of the Form validator implementation

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
    <property-group label="Technology" name="okg:technology">
        <input label="Title" type="text" name="okp:technology.title"/>
        <input label="Date" type="date" name="okp:technology.date" />
        <textarea label="Description" name="okp:technology.description"/>
        <select label="Type" name="okp:technology.type" type="multiple">
            <option label="New Product" value="t1"/>
            <option label="New Feature" value="t2" />
            <option label="Customatization" value="t3" />
            <option label="Integration" value="t4" />
            <option label="Quality" value="t5" />
            <option label="Other" value="t6" />
        </select>
        <select label="Priority" name="okp:technology.priority" type="simple">
            <option label="low" value="low"/>
            <option label="Medium" value="Medium"/>
            <option label="High" value="High" />
        </select>
        <textarea label="Comment" name="okp:technology.comment">
            <validator type="plugin" parameter="com.openkm.plugin.form.validator.DuplicateDocumentNumberValidator
        </textarea>
        <input label="Link" type="link" name="okp:technology.link"/>
    </property-group>
</property-groups>
```

In this example it will not be allowed two equal comments in the metadata field named okp:tecnology.comment

```
package com.openkm.plugin.form.validator;

import java.util.ArrayList;
import java.util.List;

import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.plugin.form.FieldValidator;
```

```
import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMRelation;
import com.openkm.bean.Relation;
import com.openkm.db.service.LegacySrv;
import com.openkm.plugin.BasePlugin;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class DuplicateDocumentNumberValidator extends BasePlugin implements FieldValidator {

    @Autowired
    private LegacySrv legacySrv;

    @Autowired
    private OKMRelation okmRelation;

    @Override
    public String getName() {
        return "Duplicated document number";
    }

    @Override
    public String validate(String value, List<String> uuids) {
        String validate = "";
        String token = DbSessionManager.getInstance().getSystemToken();
        String sql = "SELECT RGT_UUID FROM OKM_PGRP_CUR_TECHNOLOGY WHERE RGT_PRO_COMMENT
        try {
            List<List<String>> result = legacySrv.executeSQL(sql);
            if (result.size() > 0) {
                if (uuids.isEmpty()) {
                    validate = "Duplicated document number";
                } else {
                    List<String> allowedUuids = new ArrayList<>();
                    for (String uuid : uuids) {
                        allowedUuids.add(uuid);
                        List<Relation> relations = okmRelation.getRelations(token, uuid);
                        for (Relation relation : relations) {
                            allowedUuids.add(relation.getNode());
                        }
                    }

                    for (List<String> resultList : result) {
                        if (!allowedUuids.contains(resultList.get(0))) {
                            validate = "Duplicated document number";
                        }
                    }
                }
            }
        } catch (Exception e) {
            validate = e.getMessage();
        }

        return validate;
    }
}
```

Full form validation



Deprecated as of version 8.1.15, we suggest using *Creating your own FormInterceptor plugin*

This method is executed when adding, updating, and checking if a node can be promoted as a record.

The application check if a node can be promoted as a record when the file plan configuration is enabled.

The variable fullValidation is set to true only when the application is checking if a node can be promoted as a record.

Form validator are used to implement complex logic that must be evaluated from server side logic and are not covered by default Metadata Validator element.

You can create your own Form Validator

Conditions:

- *The new Form Validator class **must implement the "FormValidator"** interface.*
- *The new Form Validator class **must be declared under the package "com.openkm.plugin.form.validator".***
- *The new Form Validator class **must be annotated with "@PluginImplementation".***
- *The new Form Validator class **must extend of "BasePlugin".***

Form Validator interface:

```
package com.openkm.plugin.form;

import net.xeoh.plugins.base.Plugin;

import java.util.List;

/**
 * FormValidator
 *
 */
public interface FormValidator extends Plugin {
 void validate(NodeBase node, String grpName, Map<String, String> properties, boolean fullValidation) throws Valid

 String getName();
}
```

*The **new class must be loaded into the package com.openkm.plugin.form.validator** because application plugins system will try to load from there.*

*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*

More information at Register a new plugin.

Methods description

Method	Type	Description
getName()	*String*	*The method returns the name of the plugin used in the form list selector.*
validate(NodeBase node, String grpName, Map<String, String> properties, boolean fullValidation)	*Void*	*When an error is found should return an exception of type ValidationFormException.* *The node variable is the OpenKM node where the user added or updated the metadata.* *The grpName is the name of the metadata group.* *The properties variable contains a map with all the metadata values set by the user.* *The variable fullValidation is set to true when checking if a node can be promoted as a record.*

Example of the Form validator implementation

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
    <property-group label="Full validation" name="okg:full" validatorClassName="com.openkm.plugin.form.validator.F
        <input label="Field 1" name="okp:full.number1"></input>
        <input label="Field 2" name="okp:full.number2"></input>
        <input label="Field 3" name="okp:full.number3"></input>
    </property-group>
</property-groups>
```

In this example the fields should have numeric values and field1 < field2 < field3:

```
import com.company.MetadataConstants;
import com.openkm.bean.form.ValidationError;
import com.openkm.core.ValidationFormException;
import com.openkm.db.bean.NodeBase;
import com.openkm.plugin.BasePlugin;
import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@PluginImplementation
public class FullMetadataValidationSample extends BasePlugin implements FormValidator {
```

```
  @Override
  public void validate(NodeBase nodeBase, String grpName, Map<String, String> properties, boolean fullValidation
Map<String, List<ValidationError>> errors = new HashMap<>();
List<ValidationError> validationErrors = new ArrayList<>();
String field1 = properties.get(MetadataConstants.LOCATION_PROPERTY_NUMBER1_NAME);
String field2 = properties.get(MetadataConstants.LOCATION_PROPERTY_NUMBER2_NAME);
String field3 = properties.get(MetadataConstants.LOCATION_PROPERTY_NUMBER3_NAME);
int field1Values = 0;
int field2Values = 0;
int field3Values = 0;

// Parse integer
try {
 field1Values = Integer.parseInt(field1);
} catch (Exception e) {
 ValidationError validationError = new ValidationError();
 validationError.setErrorMessage("Field 1 seems does not contain an integer");
 validationError.setPropertyName(MetadataConstants.LOCATION_PROPERTY_NUMBER1_NAME);
 validationError.setErrorCode("ERROR-003345");
 validationError.setPropertyLabel("First number");
 validationError.setUuid(nodeBase.getUuid());
 validationErrors.add(validationError);
}

try {
 field2Values = Integer.parseInt(field2);
} catch (Exception e) {
 ValidationError validationError = new ValidationError();
 validationError.setErrorMessage("Field 2 seems does not contain an integer");
 validationError.setPropertyName(MetadataConstants.LOCATION_PROPERTY_NUMBER2_NAME);
 validationError.setErrorCode("ERROR-003345");
 validationError.setPropertyLabel("First number");
 validationError.setUuid(nodeBase.getUuid());
 validationErrors.add(validationError);
}

try {
 field3Values = Integer.parseInt(field3);
} catch (Exception e) {
 ValidationError validationError = new ValidationError();
 validationError.setErrorMessage("Field 3 seems does not contain an integer");
 validationError.setPropertyName(MetadataConstants.LOCATION_PROPERTY_NUMBER2_NAME);
 validationError.setErrorCode("ERROR-003345");
 validationError.setPropertyLabel("First number");
 validationError.setUuid(nodeBase.getUuid());
 validationErrors.add(validationError);
}

// Field2 value greather than field1
if (field2Values < field1Values) {
 ValidationError validationError = new ValidationError();
 validationError.setErrorMessage("Field 1 is greater than field2");
 validationError.setPropertyName(MetadataConstants.LOCATION_PROPERTY_NUMBER2_NAME);
 validationError.setErrorCode("ERROR-003345");
 validationError.setPropertyLabel("Second number");
 validationError.setUuid(nodeBase.getUuid());
 validationErrors.add(validationError);
}

// Field2 value greather than field1
if (field3Values < field2Values) {
 ValidationError validationError = new ValidationError();
 validationError.setErrorMessage("Field 2 is greater than field3");
 validationError.setPropertyName(MetadataConstants.LOCATION_PROPERTY_NUMBER3_NAME);
 validationError.setErrorCode("ERROR-003345");
 validationError.setPropertyLabel("Second number");
```

```
    validationError.setUuid(nodeBase.getUuid());
    validationErrors.add(validationError);
   }

   if (validationErrors.size() > 0) {
    errors.put("Validation was not successfully", validationErrors);
    throw new ValidationFormException(errors);
   }
     }

    @Override
    public String getName() {
        return "Full metadata validation sample";
    }
}
```

Creating your own FormInterceptor plugin

Available since OpenKM version 8.1.15

You can create your own form interceptor plugin to intercept all the add, set and get methods.

Description of common features allowed to be implemented by this plugin type:

1. *Intercept the values of the properties before creation (add) what allows to:*
    1. *Update values*
    2. *Validate values*
    3. *Autocomplete values*
2. *Intercept get properties or form elements to:*
    1. *Remove values (fields)*
    2. *Change UI behaviour of form elements (set fields as read-only, hide fields, etc)*
    3. *Create form elements on the fly what will not be keep in the database (requires filtering in add and set stages)*
    4. *Apply mask to values to be displayed (protect critical values)*
    5. *Change the order of the form elements*
3. *intercept the values of the properties before update (set) to:*
    1. *Update values*
    2. *Validate values*
    3. *Autocomplete values*

Conditions:

- *The new Form Validator class* ***must implement the "FormInterceptor"*** *interface.*
- *The new Form Validator class* ***must be declared under the package "com.openkm.plugin.form.interceptor".***
- *The new Form Validator class* ***must be annotated with "@PluginImplementation"****.*
- *The new Form Validator class* ***must extend of "BasePlugin".***

Form Validator interface:

```
package com.openkm.plugin.form.interceptor;

import com.openkm.bean.form.FormElement;
import com.openkm.core.ValidationFormException;
import net.xeoh.plugins.base.Plugin;

import java.util.List;
import java.util.Map;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
public interface FormInterceptor extends Plugin {
 String getName();

 void add(String uuid, String grpName, Map<String, String> properties) throws ValidationFormException;

 void get(String uuid, String grpName, Map<String, String> properties, List<FormElement> formElements, boolean u

 void set(String uuid, String grpName, Map<String, String> properties) throws ValidationFormException;
}
```

*The **new class must be loaded into the package com.openkm.plugin.form.interceptor** because the application plugins system will try to load from there.*



*Do not miss the tag **@PluginImplementation**; otherwise, the application plugin system cannot retrieve the new class.*



More information at Register a new plugin.

Methods description

Method	*Type*	*Description*
getName()	*String*	*The method returns the name of the plugin*
add(String uuid, String grpName, Map<String, String> properties)	*void*	*Intercept the action of adding metadata.*
get(String uuid, String grpName, Map<String, String> properties, List<FormElement> formElements, boolean useDefaultValue, boolean autocomplete)	*void*	*Intercept the action of getting metadata.*
set(String uuid, String grpName, Map<String, String> properties)	*void*	*Intercept the action of set metadata.*

Example of the Form validator implementation

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

In the following example plugin, different modifications are applied to the metadata only during the add, get, and set actions for a user with the ID "test." We suggest testing the example with this user and a different one.

```
package com.openkm.plugin.form.interceptor;

import com.openkm.bean.form.FormElement;
import com.openkm.bean.form.Input;
import com.openkm.bean.form.ValidationError;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.ValidationFormException;
import com.openkm.db.service.NodeBasePropertiesSrv;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@PluginImplementation
public class SampleFormInterceptor extends BasePlugin implements FormInterceptor {
 private static final Logger log = LoggerFactory.getLogger(SampleFormInterceptor.class);
 private final String TEST_USER = "test";

 @Autowired
 private NodeBasePropertiesSrv nodeBasePropertiesSrv;

 @Override
 public String getName() {
  return "SampleFormInterceptor";
 }

 /**
  * Description of common features allowed to be implemented by this plugin type:
  * 1. intercept the values of the properties before creation (add) what allows to:
  * - update values
  * - validate values
  * - autocomplete values
  * <p>
  * 2. Intercept get properties or form elements to:
  * - remove values ( fields )
  * - change UI behaviour of form elements ( set fields as read-only, hide fields, etc )
  * - create form elements on the fly what will not be keep in the database ( requires filtering in add and set stages )
  * - apply mask to values to be displayed ( protect critical values )
```

```
 * - change the order of the form elements
 * <p>
 * 3. intercept the values of the properties before update (set) to:
 * - update values
 * - validate values
 * - autocomplete values
 **/
@Override
public void add(String uuid, String grpName, Map<String, String> properties) throws ValidationFormException {
 log.info("add({}, {}, {})", uuid, grpName, properties);
 String userId = PrincipalUtils.getUser();

 if (TEST_USER.equals(userId)) {
  // Iterate through the properties map and log each property individually
  log.info("Properties in group '{}' for node '{}':", grpName, uuid);

  for (Map.Entry<String, String> entry : properties.entrySet()) {
   log.info("  Property - Key: '{}', Value: '{}'", entry.getKey(), entry.getValue());

   // update the value of the property with the key "okp:consulting.input1"
   if ("okp:consulting.input1".equals(entry.getKey())) {
    properties.put(entry.getKey(), entry.getValue() + " ( Updated by SampleFormInterceptor at add )");
   } else if ("okp:consulting.input2".equals(entry.getKey())) {
    String valueInput2 = entry.getValue();

    if (StringUtils.isEmpty(valueInput2) || valueInput2.length() < 4) {
     // Show error in UI
     Map<String, List<ValidationError>> errors = new HashMap<>();
     List<ValidationError> validationErrors = new ArrayList<>();
     ValidationError validationError = new ValidationError();
     validationError.setErrorMessage("Field input2 is mandatory and must have at least 4 characters");
     validationError.setPropertyName("okp:consulting.input2");
     validationError.setErrorCode("ERROR-003345");
     validationError.setPropertyLabel("Input label 2");
     validationError.setUuid(uuid);
     validationErrors.add(validationError);
     errors.put("Validation was not successfully", validationErrors);
     throw new ValidationFormException(errors);
    }
   }
  }
 }
}

@Override
public void get(String uuid, String grpName, Map<String, String> properties, List<FormElement> formElements, boo
    boolean autocomplete) {
 // Use default value is true when showing form elements before adding the metadata group
 String userId = PrincipalUtils.getUser();

 if (TEST_USER.equals(userId)) {
  if (properties != null) {
   properties.remove("okp:consulting.input1");
   properties.put("okp:consulting.input3", "On the fly field");
  }

  if (CollectionUtils.isNotEmpty(formElements)) {
   formElements.removeIf(formElement -> "okp:consulting.input1".equals(formElement.getName()));

   // Add form element on the fly
   Input input = new Input();
   input.setName("okp:consulting.input3");
   input.setLabel("Input label 3");
   input.setType("text");
   input.setValue("On the fly field");
   input.setReadonly(true);
   formElements.add(input);
```

```
    }
   }
  }

  @Override
  public void set(String uuid, String grpName, Map<String, String> properties) throws ValidationFormException {
   log.info("set({}, {}, {})", uuid, grpName, properties);
   String userId = PrincipalUtils.getUser();

   if (TEST_USER.equals(userId)) {
    // Because the parameter "okp:consulting.input1" is not in the properties map, it will be added to the map
    // to keep the properties map consistent with the form elements in the database
    Map<String, String> currentProperties;

    try {
     currentProperties = nodeBasePropertiesSrv.getProperties(uuid, grpName, true);
    } catch (PathNotFoundException | DatabaseException e) {
     throw new RuntimeException(e);
    }

    if (currentProperties.containsKey("okp:consulting.input1")) {
     properties.put("okp:consulting.input1", properties.get("okp:consulting.input1"));
    }

    if (properties.containsKey("okp:consulting.input2")) {
     properties.put("okp:consulting.input2", properties.get("okp:consulting.input2"));
    }

    if (properties.containsKey("okp:consulting.input2")) {
     String valueInput2 = properties.get("okp:consulting.input2");

     if (StringUtils.isEmpty(valueInput2) || valueInput2.length() < 4) {
      // Show error in UI
      Map<String, List<ValidationError>> errors = new HashMap<>();
      List<ValidationError> validationErrors = new ArrayList<>();
      ValidationError validationError = new ValidationError();
      validationError.setErrorMessage("Field input2 is mandatory and must have at least 4 characters");
      validationError.setPropertyName("okp:consulting.input2");
      validationError.setErrorCode("ERROR-003345");
      validationError.setPropertyLabel("Input label 2");
      validationError.setUuid(uuid);
      validationErrors.add(validationError);
      errors.put("Validation was not successfully", validationErrors);
      throw new ValidationFormException(errors);
     }
    }
   }
  }
 }
```

Creating your own NodePaginator plugin

NodePaginator is used to customize the file browser columns shown, including metadata columns.

*Starting from version **8.1.15**, the content of the columns supports **HTML**. This functionality replaces the deprecated **Folder Styles** feature.*

You can create your own NodePaginator.

Conditions:

- *The new NodePaginator class **must implement the "NodePaginator"** interface.*
- *The new NodePaginator class **must be declared under the package "com.openkm.plugin.paginate".***
- *The new NodePaginator class **must be annotated with "@PluginImplementation"**.*
- *The new NodePaginator class **must extend of "BasePlugin".***

NodePaginator interface:

```
package com.openkm.plugin.paginate;

import com.openkm.bean.NodePaginatorConfig;
import com.openkm.bean.PageInfo;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.db.bean.NodeBase;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

/**
 * Node Paginator
 */
public interface NodePaginator extends Plugin {

 PageInfo getChildrenPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc,
 List<Class<? extends NodeBase>> filteredByNodeTypeList) throws AccessDeniedException, PathNotFoundExcep
 DatabaseException;

 NodePaginatorConfig getConfig() throws DatabaseException;

 PageInfo getChildrenByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField,
 boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList) throws AccessDeniedException,
 PathNotFoundException, DatabaseException;

 String getName();
}
```

*The **new class must be loaded into the package com.openkm.plugin.paginate** because the application plugins system will try to load it from there.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*Do not miss the tag **@PluginImplementation** otherwise, the application plugin system will not be able to retrieve the new class.*

More information about Register a new plugin.

Method description

Method	*Type*	*Description*
getChildrenPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList)	***PageInfo***	*Get a paginated list of nodes from a parent (uuid). The object PageInfo contains the list of values for each column including metadata columns.*
getChildrenByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList)	***PageInfo***	*Get a paginated list of nodes by category from a parent (uuid). The object PageInfo contains the list of values for each column including metadata columns.*
getConfig()	***NodePaginatorConfig***	*Used to set visible columns and their names in case of metadata columns.* *When the label of a column is not empty the columns are considered visible.*
getName()	***String***	*The name of the plugin.*

Example

***Metadata XML definition**:*

More information at Metadata.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.7//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.7.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label 1" name="okp:consulting.input1" dbColumnSize="256"/>
    <input label="Input label 2" name="okp:consulting.input2" dbColumnSize="512"/>
  </property-group>
</property-groups>
```

TestNodePaginator *class:*

```
package com.openkm.plugin.paginate;

import com.openkm.bean.NodePaginatorConfig;
import com.openkm.bean.PageInfo;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.Config;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.Profile;
import com.openkm.db.service.LegacySrv;
import com.openkm.db.service.NodeBaseSrv;
import com.openkm.db.service.ProfileSrv;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalUtils;
import com.openkm.ws.rest.util.SimpleNodeBase;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@PluginImplementation
public class TestNodePaginator extends BasePlugin implements NodePaginator {
 private static final Logger log = LoggerFactory.getLogger(TestNodePaginator.class);
 private static final String PGRP_COLUMN_LABEL_01 = "Input 1";
 private static final String PGRP_COLUMN_LABEL_02 = "Input 2";
 private static final String PGRP_COLUMN_LABEL_03 = "Status";
 private static final String PGRP_COLUMN_LABEL_04 = "Icon";

 @Autowired
 private NodeBaseSrv nodeBaseSrv;

 @Autowired
 private ProfileSrv profileSrv;
```

```
@Autowired
private LegacySrv legacySrv;

@Autowired
private Config cfg;

@Override
public PageInfo getChildrenPaginated(String uuid, int offset, int limit, String filter, String orderByField,
  boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList) throws AccessDeniedException,
  PathNotFoundException, DatabaseException {
 log.debug("getChildrenPaginated({}, {}, {}, {}, {}, {}, {})", uuid, offset, limit, filter, orderByField, orderAsc, filteredByNo
 PageInfo pageInfo = nodeBaseSrv.getCommonChildrenPaginated(uuid, offset, limit, filter, orderByField, orderAsc,

 // Only fill the metadata values when results size greater than 0
 if (!pageInfo.getObjects().isEmpty()) {
  setMetaDataColumns((List<SimpleNodeBase>) pageInfo.getObjects());
 }

 return pageInfo;

}

@Override
public PageInfo getChildrenByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField,
  boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList) throws AccessDeniedException,
  PathNotFoundException, DatabaseException {
 PageInfo pageInfo = nodeBaseSrv.getCommonChildrenByCategoryPaginated(uuid, offset, limit, filter, orderByFieldo
   orderAsc, filteredByNodeTypeList);

 // Only fill the metadata values when results size greater than 0
 if (!pageInfo.getObjects().isEmpty()) {
  setMetaDataColumns((List<SimpleNodeBase>) pageInfo.getObjects());
 }

 return pageInfo;
}

@Override
public String getName() {
 return "Consulting metadata sample";
}

private void setMetaDataColumns(List<SimpleNodeBase> nodes) throws DatabaseException {
 Map<String, List<String>> rowValues = new HashMap<>();
 StringBuilder query = new StringBuilder("select RGT_UUID, RGT_PRO_INPUT1, RGT_PRO_INPUT2 "
   + "from OKM_PGRP_CUR_CONSULTING where RGT_UUID in (");

 // Create inFilter
 StringBuilder inFilter = new StringBuilder();
 for (SimpleNodeBase snb : nodes) {
  if (inFilter.length() > 0) {
   inFilter.append(",");
  }
  inFilter.append("'");
  inFilter.append(snb.getUuid());
  inFilter.append("'");
 }
 query.append(inFilter);
 query.append(")");
 log.info("Paginator query: {}", query);

 // Convert list of values to map of list
 for (List<String> row : legacySrv.executeSQL(query.toString())) {
  List<String> cols = new ArrayList<>();

  String uuid = row.get(0);
```

```
    for (int i = 1; i < row.size(); i++) {
     cols.add(row.get(i));
    }

    rowValues.put(uuid, cols);
   }

   // Set metadata field values
   for (SimpleNodeBase node : nodes) {
    if (rowValues.get(node.getUuid()) != null) {
     node.setPgrpColumnValue1(rowValues.get(node.getUuid()).get(0));
     node.setPgrpColumnValue2(rowValues.get(node.getUuid()).get(1));
     node.setPgrpColumnValue3("<div class=\"bg-success text-white px-1\">found</div>");
     node.setPgrpColumnValue4("<i v-if=\"account.item.active\" class=\"text-success mdi-1_5x mdi mdi-check\"></i>")
    } else {
     node.setPgrpColumnValue3("<div class=\"bg-danger text-white px-1\">lost</div>");
     node.setPgrpColumnValue4("<i class=\"text-danger mdi-1_5x mdi mdi-clear\"></i>");
    }
   }
  }

  public NodePaginatorConfig getConfig() throws DatabaseException {
   NodePaginatorConfig config = new NodePaginatorConfig();
   config.setPgrpColumnLabel01(PGRP_COLUMN_LABEL_01);
   config.setPgrpColumnLabel02(PGRP_COLUMN_LABEL_02);
   config.setPgrpColumnLabel03(PGRP_COLUMN_LABEL_03);
   config.setPgrpColumnLabel04(PGRP_COLUMN_LABEL_04);

   // Use current profile configuration for other parameters except in case of system user
   String user = PrincipalUtils.getUser();

   if (!cfg.DEFAULT_SYSTEM_USER.equals(user)) {
    Profile profile = profileSrv.findByUser(user);
    config.setColumnMassiveVisible(profile.getPrfFileBrowser().getMassiveVisible());
    config.setColumnIconVisible(profile.getPrfFileBrowser().getIconVisible());
    config.setColumnNameVisible(profile.getPrfFileBrowser().getNameVisible());
    config.setColumnTitleVisible(profile.getPrfFileBrowser().getTitleVisible());
    config.setColumnLanguageVisible(profile.getPrfFileBrowser().getLanguageVisible());
    config.setColumnSizeVisible(profile.getPrfFileBrowser().getSizeVisible());
    config.setColumnLastModifiedVisible(profile.getPrfFileBrowser().getLastModifiedVisible());
    config.setColumnAuthorVisible(profile.getPrfFileBrowser().getAuthorVisible());
    config.setColumnVersionVisible(profile.getPrfFileBrowser().getVersionVisible());
   }

   return config;
  }
 }
```

*Once the plugin has been deployed, remember to enable in **Profile** > **General** > **Pagination** > **Pagination plugins**. After that ypu will be see an icon in filebrowser which can be used to choose between the default paginator and the new one.*

Creating your own NodeProperties plugin (customize search results objects)



NodeProperties is usually used from Webservices scenario. Using them you can customize the results objects from a query search. Usually as a result of some search query you only need to use few variables of the returned Objects rather than complete ones. With large query results, returning few data you get best perfomance (less time for filling data at server side and less time for marshalling and unmarshalling webservices procedure).

You can create your own NodeProperties.

Conditions:

- *The new NodeProperties class* ***must implement the "NodeProperties"*** *interface.*
- *The new NodeProperties class* ***must be declared under the package "com.openkm.plugin.properties".***
- *The new NodeProperties class* ***must be annotated with "@PluginImplementation".***
- *The new NodeProperties class* ***must extend of "BasePlugin".***

NodeProperties interface:

```
package com.openkm.plugin.properties;

import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.bean.Mail;
import com.openkm.bean.Record;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.RepositoryException;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.bean.NodeFolder;
import com.openkm.db.bean.NodeMail;
import com.openkm.db.bean.NodeRecord;
import net.xeoh.plugins.base.Plugin;

/**
 * NodeProperties
 */
public interface NodeProperties extends Plugin {

    /**
     * Get document properties
     */
    Document getProperties(String user, NodeDocument nDocument, String linkTarget) throws PathNotFoundExcept

    /**
     * Get folder properties
     */
    Folder getProperties(String user, NodeFolder nFolder) throws PathNotFoundException, DatabaseException, Rep

    /**
     * Get mail properties
     */
    Mail getProperties(String user, NodeMail nMail) throws PathNotFoundException, DatabaseException, Repository

    /**
     * Get record properties
```

```
    */
    Record getProperties(String user, NodeRecord nRecord) throws PathNotFoundException, DatabaseException, R
}
```

*The **new class must be loaded into the package com.openkm.plugin.properties** because the application plugins system will try to load it from there.*



*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*



More information at: Register a new plugin.

Method description

Method	*Type*	*Description*
getProperties(String user, NodeDocument nDocument, String linkTarget)	***Document***	*Define which values are loaded in Document Object returned by search engine.*
getProperties(String user, NodeFolder nFolder)	***Return***	*Define which values are loaded in Folder Object returned by search engine.*
getProperties(String user, NodeMail nMail)	***Mail***	*Define which values are loaded in Mail Object returned by search engine.*
getProperties(String user, NodeRecord nRecord)	***Record***	*Define which values are loaded in Record Object returned by search engine.*

Example of NodeProperties implementation

```
package com.openkm.plugin.properties;

import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.bean.Mail;
import com.openkm.bean.Record;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.RepositoryException;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.bean.NodeFolder;
import com.openkm.db.bean.NodeMail;
import com.openkm.db.bean.NodeRecord;
import com.openkm.plugin.BasePlugin;
```

```
import net.xeoh.plugins.base.annotations.PluginImplementation;

/**
 * UuidNodeProperties
 */
@PluginImplementation
public class UuidNodeProperties extends BasePlugin implements NodeProperties {

    /**
     * Get document properties
     */
    @Override
    public Document getProperties(String user, NodeDocument nDocument, String linkTarget) throws PathNotFound
        Document node = new Document();
        node.setUuid(nDocument.getUuid());
        return node;
    }

    /**
     * Get folder properties
     */
    @Override
    public Folder getProperties(String user, NodeFolder nFolder) throws PathNotFoundException, DatabaseExceptio
        Folder node = new Folder();
        node.setUuid(nFolder.getUuid());
        return node;
    }

    /**
     * Get mail properties
     */
    @Override
    public Mail getProperties(String user, NodeMail nMail) throws PathNotFoundException, DatabaseException, Rep
        Mail node = new Mail();
        node.setUuid(nMail.getUuid());
        return node;
    }

    /**
     * Get record properties
     */
    @Override
    public Record getProperties(String user, NodeRecord nRecord) throws PathNotFoundException, DatabaseExcep
        Record node = new Record();
        node.setUuid(nRecord.getUuid());
        return node;
    }

}
```

Creating your own OCR Validation Control plugin

You can create your own OCR Validation plugin.

Conditions:

- *The new OCR Validation Control class* ***must implement the "OCRTemplateControlParser"*** *interface.*
- *The new OCR Validation Control class* ***must be declared under the package "com.openkm.plugin.ocr.template.controlparser".***
- *The new OCR Validation Control class* ***must be annotated with "@PluginImplementation"***.
- *The new OCR Validation Control class* ***must extend of "BasePlugin".***

OCR Validation Control interface:

```
package com.openkm.plugin.ocr.template;

import com.openkm.db.bean.OCRTemplateControlField;
import net.xeoh.plugins.base.Plugin;

/**
 * OCRTemplateControlParser
 */
public interface OCRTemplateControlParser extends Plugin {

    boolean parse(OCRTemplateControlField otcf, String text) throws OCRTemplateException, OCRParserEmptyVal

    String getName();

    boolean isPatternRequired();

    String info();
}
```

The ***new class must be loaded into the package com.openkm.plugin.ocr.template.controlparser*** *because application plugins system will try to load from there.*



Do not miss the tag ***@PluginImplementation*** *otherwise the application plugin system will not be able to retrieve the new class.*



More information at: Register a new plugin.

Methods description

Method	Type	Description
parse(OCRTemplateControlField	boolean	When returns true indicates that the field complies with the conditions.

otcf, String text)		*More information about OCR Template stages at: OCR templates.*
getName()	*String*	*Sets the name that will be shown in the administrator user interface selector list.*
isPatternRequired()	*boolean*	*When returns true indicates that the pattern field is required.* *It will show an error - missing pattern - on the administrator user interface if a pattern is required and the user is not set.*
info()	*String*	*Description of the implemented logic.* *This description is shown in the administrator user interface.*

Example:

StringSimilarParser *class:*

```
package com.openkm.plugin.ocr.template.controlparser;

import com.openkm.core.DatabaseException;
import com.openkm.db.bean.OCRTemplateControlField;
import com.openkm.db.service.ConfigSrv;
import com.openkm.plugin.BasePlugin;
import com.openkm.plugin.ocr.template.OCRParserEmptyValueException;
import com.openkm.plugin.ocr.template.OCRTemplateControlParser;
import com.openkm.plugin.ocr.template.OCRTemplateException;
import com.openkm.util.AlgorithmUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * StringParser
 */
@PluginImplementation
public class StringSimilarParser extends BasePlugin implements OCRTemplateControlParser {

    @Autowired
    private ConfigSrv configSrv;

    @Override
    public boolean parse(OCRTemplateControlField otf, String text) throws OCRTemplateException, OCRParserEmp
        if (text == null || text.equals("")) {
```

```
            return false;
        } else {
            if (otf.getPattern() == null || otf.getPattern().equals("")) {
                int distance = AlgorithmUtils.computeLevenshteinDistance(text, otf.getValue());
                try {
                    return distance < configSrv.getInteger("ocr.control.similar.distance", 5);
                } catch (DatabaseException e) {
                    throw new OCRTemplateException(e);
                }
            } else {
                Pattern pattern = Pattern.compile("(" + otf.getPattern() + ")", Pattern.UNICODE_CASE);
                Matcher matcher = pattern.matcher(text);
                if (matcher.find() && matcher.groupCount() == 1) {
                    if (otf.getValue().equals("")) {
                        return true;
                    } else {
                        int distance = AlgorithmUtils.computeLevenshteinDistance(matcher.group(), otf.getValue());
                        try {
                            return distance < configSrv.getInteger("ocr.control.similar.distance", 5);
                        } catch (DatabaseException e) {
                            throw new OCRTemplateException(e);
                        }
                    }
                } else {
                    return false;
                }
            }
        }
    }

    @Override
    public String getName() {
        return "Similar String";
    }

    @Override
    public boolean isPatternRequired() {
        return false;
    }

    @Override
    public String info() {
        StringBuilder sb = new StringBuilder();
        sb.append("Empty pattern case: true if extracted text similar field value.\n");
        sb.append("Pattern with value field empty: true if pattern found.\n");
        sb.append("Pattern with value field: true if pattern found similar value.\n");
        return null;
    }
}
```

Creating your own OCR Data Capture plugin

You can create your own OCR Data Capture plugin.

Conditions:

- *The new OCR Data Capture class* ***must implement the "OCRTemplateParser"*** *interface.*
- *The new OCR Data Capture class* ***must be declared under the package "com.openkm.plugin.ocr.template.parser".***
- *The new OCR Data Capture class* ***must be annotated with "@PluginImplementation".***
- *The new OCR Data Capture class* ***must extend of "BasePlugin".***

OCR Data Capture interface:

```
package com.openkm.plugin.ocr.template;

import com.openkm.db.bean.OCRTemplateField;
import net.xeoh.plugins.base.Plugin;

/**
 * OCRTemplateParser
 */
public interface OCRTemplateParser extends Plugin {

    Object parse(OCRTemplateField otf, String text) throws OCRTemplateException, OCRParserEmptyValueExcepti

    String getName();

    boolean isPatternRequired();
}
```

The ***new class must be loaded into the package com.openkm.plugin.ocr.template.parser*** *because application plugins system will try to load from there.*



Do not miss the tag ***@PluginImplementation*** *otherwise the application plugin system will not be able to retrieve the new class.*



More information at Register a new plugin.

Methods description

Method	Type	Description
parse(OCRTemplateField otf, String text)	*Object*	*Returns the value of the data captured into an Object.* *Allowed Object classes:*

		• *String* • *Calendar (will be converted to ISO8601 String)* • *Other (will be converted to string with String.valueOf(object).*
getName()	*String*	*Sets the name that will be shown in the administrator user interface selector list.*
isPatternRequired()	*boolean*	*When returns true indicates that the pattern field is required.* *Will show an error - missing pattern - on the administrator user interface if pattern is required and the user is not set.*

Example

StringParser *class:*

```
package com.openkm.plugin.ocr.template.parser;

import com.openkm.db.bean.OCRTemplateField;
import com.openkm.plugin.BasePlugin;
import com.openkm.plugin.ocr.template.OCRParserEmptyValueException;
import com.openkm.plugin.ocr.template.OCRTemplateException;
import com.openkm.plugin.ocr.template.OCRTemplateParser;
import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * StringParser
 */
@PluginImplementation
public class StringParser extends BasePlugin implements OCRTemplateParser {

    @Override
    public Object parse(OCRTemplateField otf, String text) throws OCRTemplateException, OCRParserEmptyValueE
        if (text == null || text.equals("")) {
            throw new OCRParserEmptyValueException("Empty value");
        }

        if (otf.getPattern() == null || otf.getPattern().equals("")) {
            return text != null ? text.trim() : null;
        } else {
            Pattern pattern = Pattern.compile("(" + otf.getPattern() + ")", Pattern.UNICODE_CASE);
            Matcher matcher = pattern.matcher(text);

            if (matcher.find() && matcher.groupCount() == 1) {
                return matcher.group();
            } else {
                throw new OCRTemplateException("Bad format, parse exception");
            }
        }
    }
```

```
    @Override
    public String getName() {
        return "String";
    }

    @Override
    public boolean isPatternRequired() {
        return false;
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Option Select Values plugin

You can create your own Option Select Values.

Conditions:

- *The new Option Select Values class* ***must implement the "OptionSelectValues"*** *interface.*
- *The new Option Select Values class* ***must be declared under the package "com.openkm.plugin.form.values".***
- *The new Option Select Values class* ***must be annotated with "@PluginImplementation".***
- *The new Option Select Values class* ***must extend of "BasePlugin".***

Option Select Values interface:

```
package com.openkm.plugin.form.values;

import com.openkm.bean.form.Option;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

/**
 * OptionSelectValues
 */
public interface OptionSelectValues extends Plugin {

    String getName();

    List<Option> getOptions() throws OptionSelectException;
}
```

The ***new class must be loaded into the package com.openkm.plugin.form.values*** *because application plugin system will try to load from there.*



Do not miss the tag ***@PluginImplementation*** *otherwise the application plugin system will not be able to retrieve the new class.*



More information at: Register a new plugin.

Methods description

Method	Type	Description
getName()	String	Return the name what will be shown at plugins table.
getOptions()	List<Option>	Return a list with the options.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example of the Option select values implementation

```
package com.openkm.plugin.form.values;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
import com.openkm.bean.form.Option;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalAdapterException;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;

@PluginImplementation
public class OptionSelectUserList extends BasePlugin implements OptionSelectValues {

    @Autowired
    private OKMAuth okmAuth;

    @Override
    public String getName() {
        return "User list";
    }

    @Override
    public List<Option> getOptions() throws OptionSelectException {
        List<Option> options = new ArrayList<>();

        try {
            List<CommonUser> userList = okmAuth.getUsers(null);
            for (CommonUser user : userList) {
                Option option = new Option();
                option.setLabel(user.getName());
                option.setValue(user.getId());
                options.add(option);
            }
        } catch (PrincipalAdapterException e) {
            throw new OptionSelectException(e);
        }

        return options;
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Principal Adapter



This section is deprecated from version 8.1.12

Principal adapters are used by authentication module to retrieve users and roles information. For example retrieve users and roles from some LDAP, or external database that stores all the company users and roles.

You can create your own Principal Adapter

Conditions:

- *The new Principal Adapter class* ***must implement the "PrincipalAdapter"*** *interface.*
- *The new Principal Adapter class* ***must be declared under the package "com.openkm.plugin.principal".***
- *The new Principal Adapter class* ***must be annotated with "@PluginImplementation".***
- *The new Principal Adapter class* ***must extend of "BasePlugin".***

Principal Adapter interface:

```
package com.openkm.plugin.principal;

import com.openkm.bean.CommonUser;
import com.openkm.db.bean.Profile;
import com.openkm.principal.PrincipalAdapterException;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

public interface PrincipalAdapter extends Plugin {

    List<CommonUser> getUsers() throws PrincipalAdapterException;

    List<String> getRoles() throws PrincipalAdapterException;

    CommonUser getUser(String userId) throws PrincipalAdapterException;

    List<CommonUser> getUsersByRole(String role) throws PrincipalAdapterException;

    List<String> getRolesByUser(String user) throws PrincipalAdapterException;

    String getPassword(String user) throws PrincipalAdapterException;

    /*
     * ----------------------------------------------------------------
     * These methods only works if using the OpenKM user database.
     * ----------------------------------------------------------------
     */

    CommonUser createUser(CommonUser user) throws PrincipalAdapterException;

    void deleteUser(String user) throws PrincipalAdapterException;

    CommonUser updateUser(CommonUser user) throws PrincipalAdapterException;

    void createRole(String role, boolean active) throws PrincipalAdapterException;
```

```
    void deleteRole(String role) throws PrincipalAdapterException;
    void updateRole(String role, boolean active) throws PrincipalAdapterException;
    void assignRole(String user, String role) throws PrincipalAdapterException;
    void removeRole(String user, String role) throws PrincipalAdapterException;
    List<Profile> getProfiles(boolean filterByActive) throws PrincipalAdapterException;
    Profile getUserProfile(String userId) throws PrincipalAdapterException;
    void setUserProfile(String userId, long profileId) throws PrincipalAdapterException;
    boolean isManageUsers();
    boolean isManageRoles();
}
```

*The **new class must be loaded into the package com.openkm.plugin.principal** because application plugins system will try to load from there.*



*Do not miss the tag **@PluginImplementation** otherwise, the application plugin system will not be able to retrieve the new class.*



More information at Register a new plugin.

*To **enable the new Principal adapter** go to **Administration > Configuration parameters** > find the parameter named **principal.adapter** and modify the value with **com.openkm.plugin.adapter.OwnPrincipalAdapter***

*To take effect this change we need to **restart the application**.*

Methods description

Method	*Type*	*Description*
getUsers()	*List<CommonUser>*	*Return the list of all the users.*
getRoles()	*List<String>*	*Return the list of all the roles.*
getUser(String userId)	*CommonUser*	*Return all user data*
getUsersByRole(String role)	*List<CommonUser>*	*Return the list of all the users who have assigned a role.*
getRolesByUser(String user)	*List<String>*	*Return the list of all the roles assigned to a user.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

String getPassword(String user)	*String*	*Return the password associated to a specific user.*
createUser(CommonUser user)	*void*	*Create a new user.*
deleteUser(String user)	*void*	*Delete a user.*
updateUser(CommonUser user)	*void*	*Update a user.* *Password parameter can be null or empty.*
createRole(String role, boolean active)	*void*	*Create a new role.*
deleteRole(String role)	*void*	*Delete a role.*
updateRole(String role, boolean active)	*void*	*Update a role.*
assignRole(String user, String role)	*void*	*Grant user with some role.*
removeRole(String user, String role)	*void*	*Revoke a role to some user.*
getProfiles(boolean filterByActive)	*List<Profile>*	*Return the list of all profiles.*
getUserProfile(String userId)	*Profile*	*Return the profile assigned to a user.*
setUserProfile(String userId, long profileId)	*void*	*Change the assigned profile to a user.*
isManageUsers()	*boolean*	*Returns a boolean that indicates if the user is on manager or not.*
		Returns a boolean that indicates if the role is on

isManageRoles()	*boolean*	*manager or not.*

Example of the Principal adapter implementation

The example retrieves users, roles and mails based on properties files.

```
package com.openkm.plugin.principal;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.List;
import java.util.Properties;
import java.util.StringTokenizer;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
import com.openkm.core.Config;
import com.openkm.db.bean.Profile;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalAdapterException;

public class UsersRolesPrincipalAdapter extends BasePlugin implements PrincipalAdapter {

    private static Logger log = LoggerFactory.getLogger(UsersRolesPrincipalAdapter.class);

    @Autowired
    private OKMAuth okmAuth;

    @Override
    public List<CommonUser> getUsers() throws PrincipalAdapterException {
        log.debug("getUsers()");
        List<CommonUser> list = new ArrayList<>();
        Properties prop = new Properties();

        try {
            prop.load(new FileInputStream(Config.HOME_DIR + "/server/default/conf/props/openkm-users.properties"));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        for (Enumeration<Object> e = prop.keys(); e.hasMoreElements();) {
            String user = (String) e.nextElement();
            if (!Config.SYSTEM_USER.equals(user)) {
                CommonUser commonUser = okmAuth.getUser(null, user);
                list.add(commonUser);
            }
        }

        log.debug("getUsers: {}", list);
        return list;
    }

    @Override
```

```
public List<String> getRoles() throws PrincipalAdapterException {
    log.debug("getRoles()");
    List<String> list = new ArrayList<String>();
    Properties prop = new Properties();

    try {
        prop.load(new FileInputStream(Config.HOME_DIR + "/server/default/conf/props/openkm-roles.properties"));
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    for (Enumeration<Object> e = prop.elements(); e.hasMoreElements();) {
        for (StringTokenizer st = new StringTokenizer((String) e.nextElement(), ","); st.hasMoreTokens();) {
            String role = st.nextToken();

            if (!Config.DEFAULT_ADMIN_ROLE.equals(role) && !list.contains(role)) {
                list.add(role);
            }
        }
    }

    log.debug("getRoles: {}", list);
    return list;
}

@Override
public CommonUser getUser(String userId) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("getUser");
}

@Override
public List<CommonUser> getUsersByRole(String role) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("getUsersByRole");
}

@Override
public List<String> getRolesByUser(String user) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("getRolesByUser");
}

@Override
public String getPassword(String user) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("getPassword");
}

@Override
public CommonUser createUser(CommonUser user) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("createUser");
}

@Override
public void deleteUser(String user) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("deleteUser");
}

@Override
public CommonUser updateUser(CommonUser user) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("updateUser");
}

@Override
public void createRole(String role, boolean active) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("createRole");
}
```

```
    @Override
    public void deleteRole(String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("deleteRole");
    }

    @Override
    public void updateRole(String role, boolean active) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("updateRole");
    }

    @Override
    public void assignRole(String user, String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("assignRole");
    }

    @Override
    public void removeRole(String user, String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("removeRole");
    }

    @Override
    public List<Profile> getProfiles(boolean filterByActive) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getProfiles");
    }

    @Override
    public Profile getUserProfile(String userId) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getUserProfile");
    }

    @Override
    public void setUserProfile(String userId, long profileId) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("setUserProfile");
    }

    @Override
    public boolean isManageUsers() {
        throw new UnsupportedOperationException("isManageUsers");
    }

    @Override
    public boolean isManageRoles() {
        throw new UnsupportedOperationException("isManageRoles");
    }

}
```

Creating your own Report plugin

Report plugin are used to generate your own reports to be rendered in the OpenKM UI.

You can create your own Report plugin.

Conditions:

- *The new Report Plugin class **must implement the "Report"** interface.*
- *The new Report Plugin class **must be declared under the package "com.openkm.plugin.report".***
- *The new Report Plugin class **must be annotated with "@PluginImplementation".***
- *The new Report Plugin class **must extend of "BasePlugin".***

Report interface:

```
package com.openkm.plugin.report;

import com.openkm.bean.PageInfo;
import com.openkm.bean.form.FormElement;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.RepositoryException;
import net.xeoh.plugins.base.Plugin;

import java.util.List;
import java.util.Map;

/**
 * Report plugin
 */
public interface Report extends Plugin {

 List<FormElement> getFormElements();

 PageInfo execute(Map<String, String> params) throws AccessDeniedException, PathNotFoundException, Reposit

 String getName();
}
```

*The **new class must be loaded into the package com.openkm.plugin.report** because the application plugins system will try to load from there.*

*Do not miss the tag **@PluginImplementation** otherwise, the application plugin system will not be able to retrieve the new class.*

More information at Register a new plugin.

Methods description

Method	*Type*	*Description*
getFormElements()	*List<FormElement>*	*The method returns a list of FormElement objects what will be rendered in the UI asking for report filtering values.*
execute(Map<String, String> params)	*PageInfo*	*Returns a list of elements related to the report.* *The params variables contain the values of the form elements set from the UI by the user.*
getName()	*String*	*Return the name of the report.*

Example of the Report plugin implementation

In this example, we return a list of documents filtered by document name as an input value, and also with a limit of results as input value (if it is empty by default the limit is 100). The list only shows 3 fields related to the document (Author, Path, Date)

```
package com.openkm.plugin.report;

import com.openkm.bean.Folder;
import com.openkm.bean.PageInfo;
import com.openkm.bean.form.FormElement;
import com.openkm.bean.form.Input;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.RepositoryException;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.service.NodeBaseSrv;
import com.openkm.module.db.DbRepositoryModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.FormatUtil;
import com.openkm.ws.rest.util.SimpleNodeBase;
import com.openkm.ws.rest.util.SimpleRowReport;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

@PluginImplementation
public class TestReport extends BasePlugin implements Report {
 private static final Logger log = LoggerFactory.getLogger(TestReport.class);

 private static final String REPORTS_DEFAULT_LIMIT = "100";
```

```
private static final String REPORT_INPUT_FILTER = "filter";
private static final String REPORT_INPUT_LIMIT = "limit";
private static final String REPORT_COLUMN_LABEL_1 = "Author";
private static final String REPORT_COLUMN_LABEL_2 = "Path";
private static final String REPORT_COLUMN_LABEL_3 = "Date";

@Autowired
private DbRepositoryModule dbRepositoryModule;

@Autowired
private NodeBaseSrv nodeBaseSrv;

@Override
public String getName() {
 return "Consulting documents sample";
}

@Override
public List getFormElements() {
 List formElements = new ArrayList<>();

 FormElement feInputFilter = new Input();
 feInputFilter.setName(REPORT_INPUT_FILTER);
 feInputFilter.setLabel("Filter");

 FormElement feInputLimit = new Input();
 feInputLimit.setName(REPORT_INPUT_LIMIT);
 feInputLimit.setLabel("Limit");

 formElements.add(feInputFilter);
 formElements.add(feInputLimit);

 return formElements;
}

@Override
public PageInfo execute(Map<String, String> params) throws AccessDeniedException, PathNotFoundException,
  RepositoryException, DatabaseException {
 PageInfo reportPageInfo = new PageInfo();
 List simpleRowReportList = new ArrayList<>();
 Folder rootFolder = dbRepositoryModule.getRootFolder(null);

 String filter = params.get(REPORT_INPUT_FILTER);
 String limit = StringUtils.isNotEmpty(params.get(REPORT_INPUT_LIMIT)) ? params.get(REPORT_INPUT_LIMIT)

 List<Class<? extends NodeBase>> filteredByNodeTypeList = new ArrayList<>();
 filteredByNodeTypeList.add(NodeDocument.class);

 PageInfo pageInfo = nodeBaseSrv.getCommonChildrenPaginated(rootFolder.getUuid(), 0, Integer.parseInt(limit), f
  true, filteredByNodeTypeList);

 // Only fill the metadata values when results size greater than 0
 if (pageInfo.getObjects().size() > 0) {
  for (SimpleNodeBase snb : (List) pageInfo.getObjects()) {
   SimpleRowReport simpleRowReportPlugin = new SimpleRowReport();
   simpleRowReportPlugin.setUuid(snb.getUuid());
   simpleRowReportPlugin.setName(snb.getName());
   simpleRowReportPlugin.setReportColumnLabel1(REPORT_COLUMN_LABEL_1);
   simpleRowReportPlugin.setReportColumnLabel2(REPORT_COLUMN_LABEL_2);
   simpleRowReportPlugin.setReportColumnLabel3(REPORT_COLUMN_LABEL_3);
   simpleRowReportPlugin.setReportColumnValue1(snb.getAuthor());
   simpleRowReportPlugin.setReportColumnValue2(snb.getPath());
   simpleRowReportPlugin.setReportColumnValue3(FormatUtil.formatDate(snb.getVersionCreated())); // formatted t

   simpleRowReportList.add(simpleRowReportPlugin);
  }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    reportPageInfo.setTotalElements(pageInfo.getObjects().size());
    reportPageInfo.setObjects(simpleRowReportList);
   }

   return reportPageInfo;
  }
 }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Rest Plugin (extending REST API)

The Rest Plugin feature helps you to extending the REST API.

You can create your own Form Validator

Conditions:

- *The new Rest class **must implement the "RestPlugin"** interface.*
- *The new Rest class **must be declared under the package "com.openkm.plugin.rest".***
- *The new Rest class **must be annotated with "@PluginImplementation".***
- *The new Rest class **must extend of "BasePlugin".***

Rest interface:

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.Plugin;

import java.io.InputStream;
import java.util.Map;

/**
 * RestPlugin
 */
public interface RestPlugin extends Plugin {

    Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception;

}
```

*The **new class must be located into the package com.openkm.plugin.rest** because the application plugins system will try to load it from there.*

*Do not miss the tag **@PluginImplementation** otherwise the application plugin system won't be able to use the class.*

More information at Register a new plugin.

Method description

Method	*Type*	*Description*
executePlugin	***String***	*The method executes a REST plugin an return an object.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example of REST plugin implementation

```
package com.openkm.plugin.rest;

import com.openkm.plugin.BasePlugin;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.InputStream;
import java.util.Map;

/**
 * Sample rest plugin
 */
@PluginImplementation
public class TestRestPlugin extends BasePlugin implements RestPlugin {

    private static Logger log = LoggerFactory.getLogger(TestRestPlugin.class);

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        sb.append("className=").append(this.getClass().getCanonicalName());
        for (String key : parameters.keySet()) {
            sb.append(", " + key + "=").append(parameters.get(key));
        }
        return sb.toString();
    }
}
```

Example of REST plugin implementation what returns file



The sample below goes in combination of REST webservice "/rest/plugin/executeGetPluginReturnFile" method.

```
package com.openkm.plugin.rest;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import java.io.InputStream;
import java.util.Map;

/**
 * Sample rest plugin
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlugin {
```

```
    private static Logger log = LoggerFactory.getLogger(TestGetDocumentRestPlugin.class);

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, false);

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"" + fileName + "\"");
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
        }

        responseHeaders.setContentLength(doc.getActualVersion().getSize());
        InputStreamResource inputStreamResource = new InputStreamResource(isContent);
        log.debug("TestGetDocumentRestPlugin: [BINARY]");
        return new ResponseEntity<>(inputStreamResource, responseHeaders, HttpStatus.OK);
    }
}
```

Objects declaration

The method executionPlugin returns an Object. This object will be processed - marshall - by OpenKM. Basic types like String, Integer, etc. can be directly used, but for complex objects you need to use annotation on classes.

Marshall and unmarshall strategies:

- *The result of the executed method is always an String. Internally the method converts the complex objects to JSON.*
- *The result of the executed mehod is a complex Object.*

String scenario

From the plugin side - server side - you must convert the Object to JSON:

```
return new Gson().toJson(someObject)
```

From the cliend side you must convert JSON String to Object:

```
// value contains the returned String value by the REST call
SomeObject someObject = new Gson().fromJson(value, SomeObject.class);
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Complex Object scenario

You can use existing OpenKM classes like Document, Folder, Record, Mail among others, because theses clases have already the annotation.

Use the anotation (in the sample below the object will be marshalled with name "mail"):

@XmlRootElement(name = "mail")

Example of complex classes:

Return a list of Activities:

```
package com.openkm.ws.rest.util;

import com.openkm.db.bean.Activity;

import java.util.ArrayList;
import java.util.List;

public class ActivityList {

    List<Activity> activities = new ArrayList<>();

    private int totalElements;

    public List<Activity> getActivities() {
        return activities;
    }

    public void setActivities(List<Activity> activities) {
        this.activities = activities;
    }

    public int getTotalElements() {
        return totalElements;
    }

    public void setTotalElements(int totalElements) {
        this.totalElements = totalElements;
    }
}
```

Complex object:

```
package com.openkm.bean;

public class Record extends Parent {

    private static final long serialVersionUID = 1L;
    public static final String TYPE = "okm:record";
    public static final String NAME = "okm:name";

    private boolean locked;
    private LockInfo lockInfo;
    private String title;
```

```
    private boolean hasLinks;

    public boolean isLocked() {
        return locked;
    }

    public void setLocked(boolean locked) {
        this.locked = locked;
    }

    public LockInfo getLockInfo() {
        return lockInfo;
    }

    public void setLockInfo(LockInfo lockInfo) {
        this.lockInfo = lockInfo;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public boolean hasLinks() {
        return hasLinks;
    }

    public void setHasLinks(boolean hasLinks) {
        this.hasLinks = hasLinks;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        sb.append("uuid=").append(uuid);
        sb.append(", parent=").append(parent);
        sb.append(", path=").append(path);
        sb.append(", permissions=").append(permissions);
        sb.append(", created=").append(created == null ? null : created.getTime());
        sb.append(", subscribed=").append(subscribed);
        sb.append(", subscriptors=").append(subscriptors);
        sb.append(", keywords=").append(keywords);
        sb.append(", categories=").append(categories);
        sb.append(", notes=").append(notes);
        sb.append(", locked=").append(locked);
        sb.append(", lockInfo=").append(lockInfo);
        sb.append(", title=").append(title);
        sb.append(", nodeClass=").append(getNodeClass());
        sb.append(", nodeClassChildren=").append(getNodeClassChildren());
        sb.append(", dispositionCurrentStage=").append(getDispositionCurrentStage());
        sb.append("}");
        return sb.toString();
    }
}
```

Advanced REST sample



Because this sample uses low-level API, you should always ask OpenKM IT staff before using it on production for understanding the benefits and restrictions of this kind of implementation.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

The next sample is a project that uses low-level API implementation for quick upload and set metadata at the same time. A sample comes with plugin implementation and client plugin test.

If you have a huge uploading process and are looking for performance this is a good approach.

- *Download the sample project sample-rest.zip.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Search plugin

NodePaginator is used to customize the file browser columns shown, including metadata columns.

You can create your own NodePaginator.

Conditions:

- *The new NodePaginator class **must implement the "NodeSearch"** interface.*
- *The new NodePaginator class **must be created under the package "com.openkm.plugin.search".***
- *The new NodePaginator class **must be annotated with "@PluginImplementation".***
- *The new NodePaginator class **must extend "BasePlugin".***

NodePaginator interface:

```
package com.openkm.plugin.search;

import com.openkm.bean.NodeSearchConfig;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.ParseException;
import com.openkm.core.RepositoryException;
import com.openkm.db.bean.QueryParams;
import com.openkm.ws.rest.util.SimpleNodeBaseResultSet;
import net.xeoh.plugins.base.Plugin;

import java.io.IOException;

/**
 * Node Search
 */
public interface NodeSearch extends Plugin {
 SimpleNodeBaseResultSet findSimpleNodeBasePaginated(QueryParams params, int offset, int limit) throws Acces
   RepositoryException, IOException, ParseException, DatabaseException;

 NodeSearchConfig getConfig() throws DatabaseException;

 String getName();
}
```

*The **new class must be created into com.openkm.plugin.search package** because the plugin system will try to load it from there.*

*Do not miss the tag **@PluginImplementation** otherwise, the plugin system will not be able to detect the new class.*

More information about Register a new plugin.

Method description

Method	*Type*	*Description*
findSimpleNodeBasePaginated(QueryParams params, int offset, int limit)	***SimpleNodeBaseResultSet***	*Perform a search based on the query parameters. The offset and limit values set the range of the results.*
getConfig()	***NodeSearchConfig***	*Used to set visible search fields.*  *By default, all the search fields are hidden.*
getName()	***String***	*The name of the plugin.*

Example

TestNodeSearch *class:*

```
ackage com.openkm.plugin.search;

import com.openkm.bean.NodeSearchConfig;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.ParseException;
import com.openkm.core.RepositoryException;
import com.openkm.db.bean.QueryParams;
import com.openkm.module.db.DbSearchModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.ws.rest.util.SimpleNodeBaseResultSet;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.io.IOException;

@PluginImplementation
public class TestNodeSearch extends BasePlugin implements NodeSearch {
 private static final Logger log = LoggerFactory.getLogger(TestNodeSearch.class);

 @Autowired
 private DbSearchModule dbSearchModule;

 @Override
 public SimpleNodeBaseResultSet findSimpleNodeBasePaginated(QueryParams params, int offset, int limit)
   throws AccessDeniedException, RepositoryException, IOException, ParseException, DatabaseException {
  return dbSearchModule.findSimpleNodeBasePaginated(null, params, offset, limit);
 }

 @Override
 public NodeSearchConfig getConfig() throws DatabaseException {
```

```
  NodeSearchConfig nsc = new NodeSearchConfig();
  nsc.setTabBasicVisible(true);
  nsc.setSaveSearchVisible(true);
  nsc.setContextVisible(true);
  nsc.setContentVisible(true);
  nsc.setNameVisible(true);
  nsc.setDateVisible(true);
  return nsc;
 }

 @Override
 public String getName() {
  return "Test search sample";
 }
}
```

Creating your own security access manager

The security access manager is used by the OpenKM to check the user security access level of the nodes.

Conditions:

- *The new Principal Adapter class* ***must implement the "DbAccessManager"*** *interface.*
- *The new Principal Adapter class* ***must be declared under the package "com.openkm.plugin.access".***
- *The new Principal Adapter class* ***must be annotated with*** *"****@PluginImplementation****".*
- *The new Principal Adapter class* ***must extend of "BasePlugin".***

Security access manager interface:

```
package com.openkm.plugin.access;

import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.db.bean.NodeBase;
import com.openkm.principal.PrincipalAdapterException;
import net.xeoh.plugins.base.Plugin;

/**
 * Check user permissions on documents and folders.
 *
 * @author pavila
 */
public interface DbAccessManager extends Plugin {

 void checkPermission(NodeBase node, int permissions) throws AccessDeniedException, PathNotFoundException,

 boolean isGranted(NodeBase node, int permissions) throws DatabaseException, PathNotFoundException;

 boolean isGranted(NodeBase node, String user, int permissions) throws PrincipalAdapterException, DatabaseExce

 boolean isGranted(NodeBase node, long nodeClass, int permissions) throws DatabaseException;
}
```

The ***new class must be loaded into the package com.openkm.plugin.access*** *because application plugins system will try to load from there.*



Do not miss the tag ***@PluginImplementation*** *otherwise, the application plugin system will not be able to retrieve the new class.*



More information at Register a new plugin.

To ***enable the new Security access manager*** *go to* ***Administration > Configuration parameters*** *> find the parameter named* ***security.access.manager*** *and modify the value with* ***com.openkm.plugin.access.OwnSecurityManager***

To take effect this change we need to ***restart the application****.*

Methods description

In the next table the values of permissions are (more information in the class Permission.java):

0 NONE
1 READ
2 WRITE
4 DELETE
8 SECURITY
16 MOVE
1024 DOWNLOAD
2048 START WORKFLOW
4096 COMPACT HISTORY
8192 METADATA

Method	Type	Description
checkPermission(NodeBase node, int permissions)	***void***	*Description*
isGranted(NodeBase node, int permissions)	***Boolean***	*Return true when the user in the session has the grant.*
isGranted(NodeBase node, String user, int permissions)	***Boolean***	*Return true when the user has the grant.*
isGranted(NodeBase node, long nodeClass, int permissions)	***Boolean***	*Return true when the user in the session has the grant.* *This method only has sense when file plan is enabled.*

Example of Security access manager implementation

```
package com.openkm.plugin.access;

import com.openkm.core.AccessDeniedException;
import com.openkm.core.Config;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.NodeClass;
import com.openkm.db.service.NodeBaseSrv;
import com.openkm.db.service.NodeClassSrv;
import com.openkm.module.common.CommonAuthModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.plugin.fileplan.security.FilePlanAccessManager;
import com.openkm.principal.PrincipalAdapterException;
import com.openkm.principal.PrincipalUtils;
import com.openkm.util.StackTraceUtils;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

/**
 * Check user permissions on documents and folders.
 *
 * @see com.openkm.plugin.access.DbRecursiveAccessManager
 */
@PluginImplementation
public class DbSimpleAccessManager extends BasePlugin implements DbAccessManager {
 private static final Logger log = LoggerFactory.getLogger(DbSimpleAccessManager.class);

 @Autowired
 private CommonAuthModule commonAuthModule;

 @Autowired
 private NodeClassSrv nodeClassSrv;

 @Autowired
 private NodeBaseSrv nodeBaseSrv;

 /**
  * Check for permissions.
  */
 public void checkPermission(NodeBase node, int permissions) throws AccessDeniedException, PathNotFoundExce
  if (!isGranted(node, permissions)) {
   String nodePath = nodeBaseSrv.getPathFromUuid(node.getUuid());
   throw new AccessDeniedException(node.getUuid() + " : " + nodePath);
  }
 }

 /**
  * Check for permissions.
  */
 @Override
 public boolean isGranted(NodeBase node, int permissions) throws DatabaseException {
  String userId = PrincipalUtils.getUser();
  Set<String> roles = PrincipalUtils.getRoles();
  return isGranted(node, userId, roles, permissions);
 }

 /**
  * Check for permissions.
  */
 @Override
 public boolean isGranted(NodeBase node, String user, int permissions) throws PrincipalAdapterException, Databas
  List<String> roles = commonAuthModule.getRolesByUser(user);
  return isGranted(node, user, new HashSet<>(roles), permissions);
 }

 @Override
 public boolean isGranted(NodeBase node, long nodeClass, int permissions) throws DatabaseException {
  NodeClass nc = nodeClassSrv.findByPk(nodeClass);
  FilePlanAccessManager accessManager = nodeClassSrv.findAccessManagerByClassName(nc.getAccessManage
  return accessManager.isGranted(node, nodeClass, PrincipalUtils.getUser(), PrincipalUtils.getRoles(), permissions)
 }

 /**
  * Check for permissions.
  */
```

```
private boolean isGranted(NodeBase node, String user, Set<String> roles, int permissions) {
 log.debug("isGranted({}, {})", node.getUuid(), permissions);
 boolean access;

 if (user != null) {
  if (Config.SYSTEM_USER.equals(user) || Config.ADMIN_USER.equals(user)) {
   // An okmAdmin user has total access
   access = true;
  } else {
   if (roles.contains(Config.DEFAULT_ADMIN_ROLE)) {
    // An user with AdminRole has total access
    access = true;
   } else {
    access = checkProperties(node.getUserPermissions(), node.getRolePermissions(), user, roles, permissions);
   }
  }
 } else {
  access = true;

  log.info("*******************************************************");
  StackTraceUtils.logTrace(log);
  log.info("*******************************************************");
 }

 log.debug("isGranted: {}", access);
 return access;
}

/**
 * Check access properties
 */
private boolean checkProperties(Map<String, Integer> usersPerms, Map<String, Integer> rolesPerms, String user,
   Set<String> roles, int perms) {
 log.debug("checkProperties({}, {}, {}, {})", usersPerms, rolesPerms, roles, perms);
 boolean access = false;

 // Fist try with user permissions
 Integer userPerms = usersPerms.get(user);

 if (userPerms != null && (perms & userPerms) != 0) {
  log.debug("checkProperties: {}", true);
  return true;
 }

 // If there is no user specific access, try with roles permissions
 for (String role : roles) {
  Integer rolePerms = rolesPerms.get(role);

  if (rolePerms != null && (perms & rolePerms) != 0) {
   log.debug("checkProperties: {}", true);
   return true;
  }
 }

 log.debug("checkProperties: {}", access);
 return access;
}
}
```

Creating your own Stack Metadata Adapter (deprecated)

Feature deprecated and removed at version 8.1.11

The StackMetadata class implements the logic that takes control on what is shown into metadata stack - tree nodes on left panel - and their objects (documents, nodes, folders, records) on right panel (browser).

You can create your own Stack Metadata Adapter.

Conditions:

- *The new Stack Metadata Adapter class* ***must implement the "StackMetadataAdapter"*** *interface.*
- *The new Stack Metadata Adapter class* ***must be declared under the package "com.openkm.plugin.stack.metadata".***
- *The new Stack Metadata Adapter class* ***must be annotated with "@PluginImplementation".***
- *The new Stack Metadata Adapter class* ***must extend of "BasePlugin".***

Stack Metadata Adapter interface:

```
package com.openkm.plugin.stack;

import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.bean.Mail;
import com.openkm.bean.Record;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

public interface StackMetadataAdapter extends Plugin {

    String getName();

    boolean isPropertyGroupHidden(String pgName);

    Folder getProperties(String fldPath);

    List<Folder> getChildren(String fldPath);

    List<Document> getDocumentMetadataChildren(String fldPath);

    List<Record> getRecordMetadataChildren(String fldPath);

    List<Mail> getMailMetadataChildren(String fldPath);

    List<Folder> getFolderMetadataChildren(String fldPath);
}
```

The ***new class must be loaded into the package com.openkm.plugin.stack.metadata*** *because application plugins system will try to load from there.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*



More information at Register a new plugin.

Methods description

Method	*Type*	*Description*
getName()	*String*	*Return the name shown at plugins table.*
isPropertyGroupHidden(String pgName)	*boolean*	*Return if metadata fields are hidden.*
getProperties(String fldPath)	*Folder*	*Return folder properties.*
getChildren(String fldPath)	*List<Folder>*	*Return a list of folders.* *When folder node is chosen from tree (left panel) this method is called to retrieve children folders. These folders are displayed on tree as child nodes.*
getDocumentMetadataChildren(String fldPath)	*List<Document>*	*Return a list of documents.* *When the folder node is chosen from tree (left panel) this method is called to retrieve children documents. These documents are displayed on browser explorer, right panel.*
getRecordMetadataChildren(String fldPath)	*List<Record>*	*Return a list of records.* *When folder node is chosen from tree (left panel) this method is called to retrieve children records. These records are displayed on browser explorer, right panel.*
getMailMetadataChildren(String fldPath)	*List<Mail>*	*Return a list of mails.*

		When folder node is chosen from tree (left panel) this method is called to retrieve children mails. These mails are displayed on browser explorer, right panel.
getFolderMetadataChildren(String fldPath)	*List<Folder>*	*Return a list of folders.* *When folder node is chosen from tree (left panel) this method is called to retrieve children folders. These folders are displayed on browser explorer, right panel.*

Example of the Stack Metadata implementation

```
package com.openkm.plugin.stack.metadata;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.api.OKMRepository;
import com.openkm.api.OKMSearch;
import com.openkm.bean.*;
import com.openkm.bean.form.*;
import com.openkm.core.Config;
import com.openkm.db.bean.KeyValue;
import com.openkm.db.service.KeyValueSrv;
import com.openkm.plugin.BasePlugin;
import com.openkm.plugin.stack.StackMetadataAdapter;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;

@PluginImplementation
public class ParentChildStackMetadataAdapter extends BasePlugin implements StackMetadataAdapter {

    private static Logger log = LoggerFactory.getLogger(ParentChildStackMetadataAdapter.class);

    @Autowired
    private OKMPropertyGroup okmPropertyGroup;

    @Autowired
    private OKMRepository okmRepository;

    @Autowired
    private KeyValueSrv keyValueSrv;

    @Autowired
    private OKMSearch okmSearch;

    @Override
    public String getName() {
        return "Parent Child Metadata Adapter";
    }

    @Override
    public List<Folder> getFolderMetadataChildren(String fldPath) {
```

```
    List<Folder> folderList = new ArrayList<>();

    try {
        // Metadata value level
        if (fldPath.startsWith("/" + Repository.METADATA) && (fldPath.split("/").length - 1) >= 4) {
            String subFolder[] = fldPath.split("/");
            String group = subFolder[2];
            String property = subFolder[subFolder.length - 2];
            String value = subFolder[subFolder.length - 1];

            // In this case we are always in a select with parent Element
            // We need to find de property name, because in property we have the parent value
            if (subFolder.length - 1 > 4) {
                for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, group)) {
                    if (formElement instanceof Select) {
                        Select select = (Select) formElement;
                        if (select.getParentElement() != null && !select.getParentElement().isEmpty()) {
                            for (Option option : ((Select) formElement).getOptions()) {
                                if (option.getValue().equalsIgnoreCase(value)) {
                                    property = select.getName();
                                    break;
                                }
                            }
                        }
                    }
                }
            }

            folderList = okmSearch.getFoldersByPropertyValue(null, group, property, value);
        }
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    }

    return folderList;
}

@Override
public List<Document> getDocumentMetadataChildren(String fldPath) {
    List<Document> docList = new ArrayList<>();

    try {
        // Case metadata at value level
        if (fldPath.split("/").length - 1 >= 4) {
            String subFolder[] = fldPath.split("/");
            String group = subFolder[2];
            String property = subFolder[subFolder.length - 2];
            String value = subFolder[subFolder.length - 1];

            // In this case we are always in a select with parent Element
            // We need to find de property name, because we have the parent value
            if (subFolder.length - 1 > 4) {
                for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, group)) {
                    if (formElement instanceof Select) {
                        Select select = (Select) formElement;
                        if (select.getParentElement() != null && !select.getParentElement().isEmpty()) {
                            for (Option option : ((Select) formElement).getOptions()) {
                                if (option.getValue().equalsIgnoreCase(value)) {
                                    property = select.getName();
                                    break;
                                }
                            }
                        }
                    }
                }
            }
            docList = okmSearch.getDocumentsByPropertyValue(null, group, property, value);
```

```
        }

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    }

    return docList;
}

@Override
public List<Record> getRecordMetadataChildren(String fldPath) {
    List<Record> recordList = new ArrayList<>();

    try {
        // Case metadata at value level
        if (fldPath.split("/").length - 1 >= 4) {
            String subFolder[] = fldPath.split("/");
            String group = subFolder[2];
            String property = subFolder[subFolder.length - 2];
            String value = subFolder[subFolder.length - 1];

            // In this case we are always in a select with parent Element
            // We need to find de property name, because we have the parent value
            if (subFolder.length - 1 > 4) {
                for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, group)) {
                    if (formElement instanceof Select) {
                        Select select = (Select) formElement;
                        if (select.getParentElement() != null && !select.getParentElement().isEmpty()) {
                            for (Option option : ((Select) formElement).getOptions()) {
                                if (option.getValue().equalsIgnoreCase(value)) {
                                    property = select.getName();
                                    break;
                                }
                            }
                        }
                    }
                }
            }

            recordList = okmSearch.getRecordsByPropertyValue(null, group, property, value);
        }

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    }

    return recordList;
}

@Override
public List<Mail> getMailMetadataChildren(String fldPath) {
    List<Mail> mailList = new ArrayList<>();

    try {
        // Case metadata at value level
        if (fldPath.split("/").length - 1 >= 4) {
            String subFolder[] = fldPath.split("/");
            String group = subFolder[2];
            String property = subFolder[subFolder.length - 2];
            String value = subFolder[subFolder.length - 1];

            // In this case we are always in a select with parent Element
            // We need to find de property name, because we have the parent value
            if (subFolder.length - 1 > 4) {
                for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, group)) {
                    if (formElement instanceof Select) {
                        Select select = (Select) formElement;
```

```
                    if (select.getParentElement() != null && !select.getParentElement().isEmpty()) {
                        for (Option option : ((Select) formElement).getOptions()) {
                            if (option.getValue().equalsIgnoreCase(value)) {
                                property = select.getName();
                                break;
                            }
                        }
                    }
                }
            }

            mailList = okmSearch.getMailsByPropertyValue(null, group, property, value);
        }

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    }

    return mailList;
}

@Override
public List<Folder> getChildren(String fldPath) {
    List<Folder> folderList = new ArrayList<>();

    try {
        switch (fldPath.split("/").length - 1) {
            case 1: // getting group level
                for (PropertyGroup pg : okmPropertyGroup.getAllGroups(null)) {
                    // Only represent metadata groups which have some representable object ( select or suggestbox )
                    boolean hasValidChildren = false;

                    for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, pg.getName())) {
                        if (!isPropertyGroupHidden(pg.getName())) {
                            if (!isPropertyGroupHidden(formElement.getName()) && (formElement instanceof Select || formI
                                hasValidChildren = true;
                                break;
                            }
                        }
                    }

                    if (hasValidChildren) {
                        Folder folder = new Folder();
                        String path = fldPath + "/" + pg.getName();
                        folder.initMetadata(path, pg.getLabel(),
                                okmPropertyGroup.getPropertyGroupForm(null, pg.getName()).size() > 0);
                        folderList.add(folder);
                    }
                }
                break;

            case 2: // getting property level
                String grpName = fldPath.substring(fldPath.lastIndexOf("/") + 1);
                for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, grpName)) {
                    if (!isPropertyGroupHidden(formElement.getName())) {
                        if (formElement instanceof Select) {
                            Select select = (Select) formElement;
                            // Only draw select with no parent
                            if (select.getParentElement() == null || select.getParentElement().isEmpty()) {
                                Folder folder = new Folder();
                                String path = fldPath + "/" + formElement.getName();
                                folder.initMetadata(path, formElement.getLabel(), false);
                                folderList.add(folder);
                            }
                        } else if (formElement instanceof SuggestBox || formElement instanceof CheckBox) {
                            Folder folder = new Folder();
```

```
                String path = fldPath + "/" + formElement.getName();
                folder.initMetadata(path, formElement.getLabel(), false);
                folderList.add(folder);
            }
        }
    }
    break;

case 3: // getting property value first level
    String subFolder[] = fldPath.split("/");
    grpName = subFolder[2];
    String propertyName = subFolder[3];

    for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, grpName)) {
        if (formElement.getName().equals(propertyName)) {
            if (formElement instanceof Select) {
                for (Option option : ((Select) formElement).getOptions()) {
                    Folder folder = new Folder();
                    String path = fldPath + "/" + option.getValue();
                    folder.initMetadata(path, option.getLabel(), false);
                    folderList.add(folder);
                }
            } else if (formElement instanceof SuggestBox) {
                SuggestBox sb = ((SuggestBox) formElement);
                String sqlFilter = sb.getFilterQuery();

                // replacing filter parameter, if exist
                sqlFilter = sqlFilter.replaceAll("\\{0\\}", "");

                for (KeyValue keyValue : keyValueSrv.getKeyValues(sb.getTable(), sqlFilter)) {
                    Folder folder = new Folder();
                    String path = fldPath + "/" + keyValue.getKey();
                    folder.initMetadata(path, keyValue.getValue(), false);
                    folderList.add(folder);
                }
            } else if (formElement instanceof CheckBox) {
                Folder folder = new Folder();
                String path = fldPath + "/true";
                folder.initMetadata(path, "true", false);
                folderList.add(folder);
                folder = new Folder();
                path = fldPath + "/false";
                folder.initMetadata(path, "false", false);
                folderList.add(folder);
            }
            break;
        }
    }
    break;

default: // getting property value for select with parent elements
    subFolder = fldPath.split("/");
    grpName = subFolder[2];
    String pgValue = subFolder[subFolder.length - 1];

    for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, grpName)) {
        if (formElement instanceof Select) {
            Select select = (Select) formElement;
            if (select.getParentElement() != null && !select.getParentElement().isEmpty()) {
                for (Option option : select.getOptions()) {
                    if (option.getParentValue().equalsIgnoreCase(pgValue)) {
                        Folder folder = new Folder();
                        String path = fldPath + "/" + option.getValue();
                        folder.initMetadata(path, option.getLabel(), false);
                        folderList.add(folder);
                    }
                }
```

```
                    }
                }
            }
            break;
        }

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    }

    return folderList;
}

/**
 * If if a given property or property group is hidden.
 *
 * @param pgName The name of the property or property group.
 * @return Is property group is hidden or not by Config.
 */
@Override
public boolean isPropertyGroupHidden(String pgName) {
    boolean isHidden = false;

    for (String metadata : Config.STACK_METADATA_HIDDEN_PROPERTIES) {
        if (pgName.equalsIgnoreCase(metadata)) {
            isHidden = true;
            break;
        }
    }

    return isHidden;
}

@Override
public Folder getProperties(String fldPath) {
    Folder folder = new Folder();

    try {
        String grpName = "";
        String propertyName = "";
        String subFolder[] = fldPath.split("/");

        switch (fldPath.split("/").length - 1) {
            case 1:
                folder = okmRepository.getMetadataFolder(null);
                break;

            case 2: // group level
                grpName = fldPath.substring(fldPath.lastIndexOf("/") + 1);
                String label = "";

                for (PropertyGroup pg : okmPropertyGroup.getAllGroups(null)) {
                    if (pg.getName().equals(grpName)) {
                        label = pg.getLabel();
                        break;
                    }
                }

                List<FormElement> props = okmPropertyGroup.getPropertyGroupForm(null, grpName);
                folder.initMetadata(fldPath, label, props.size() > 0);
                break;

            case 3: // property level
                grpName = subFolder[2];
                propertyName = subFolder[3];

                for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, grpName)) {
```

```
                if (formElement.getName().equals(propertyName)) {
                    folder.initMetadata(fldPath, formElement.getLabel(), false);
                    break;
                }
            }
            break;

        case 4: // value level
            grpName = subFolder[2];
            propertyName = subFolder[3];
            String value = subFolder[4];

            for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, grpName)) {
                if (formElement.getName().equals(propertyName)) {
                    if (formElement instanceof Select) {
                        Select select = (Select) formElement;
                        // Only select with no parent
                        if (select.getParentElement() == null || select.getParentElement().isEmpty()) {
                            for (Option option : ((Select) formElement).getOptions()) {
                                if (option.getValue().equals(value)) {
                                    folder.initMetadata(fldPath, option.getLabel(), false);
                                    break;
                                }
                            }
                        }
                    } else if (formElement instanceof SuggestBox) {
                        SuggestBox sb = ((SuggestBox) formElement);
                        String sqlFilter = sb.getFilterQuery();
                        sqlFilter = sqlFilter.replaceAll("\\{0\\}", ""); // replacing filter parameter, if exist

                        for (KeyValue keyValue : keyValueSrv.getKeyValues(sb.getTable(), sqlFilter)) {
                            if (keyValue.getKey().equals(value)) {
                                folder.initMetadata(fldPath, keyValue.getValue(), false);
                                break;
                            }
                        }
                    } else if (formElement instanceof CheckBox) {
                        folder.initMetadata(fldPath, value, false);
                    }
                    break;
                }
            }
            break;

        default: // value level for select with parent
            grpName = subFolder[2];
            String pgValue = subFolder[subFolder.length - 1];

            for (FormElement formElement : okmPropertyGroup.getPropertyGroupForm(null, grpName)) {
                if (formElement instanceof Select) {
                    Select select = (Select) formElement;
                    // Only select with no parents
                    if (select.getParentElement() != null && !select.getParentElement().isEmpty()) {
                        for (Option option : ((Select) formElement).getOptions()) {
                            if (option.getValue().equalsIgnoreCase(pgValue)) {
                                folder.initMetadata(fldPath, option.getLabel(), false);
                                break;
                            }
                        }
                    }
                }
            }
            break;
    }
} catch (Exception e) {
    log.error(e.getMessage(), e);
}
```

```
        return folder;
    }

}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own Suggestion Analyzer

You can create your own Suggestion Analyzer.

Conditions:

- *The new Suggestion Analyzer class **must implement the "Suggestion"** interface.*
- *The new Suggestion Analyzer class **must be declared under the package "com.openkm.plugin.form.suggestion".***
- *The new Suggestion Analyzer class **must be annotated with "@PluginImplementation".***
- *The new Suggestion Analyzer class **must extend of "BasePlugin".***

Suggestion Analyzer interface:

```
package com.openkm.plugin.form.suggestion;

import com.openkm.bean.form.Select;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

public interface Suggestion extends Plugin {
    List<String> getSuggestions(String nodeUuid, String nodePath, Select sel) throws PathNotFoundException, Sugg
        DatabaseException;
}
```

*The **new class must be loaded into the package com.openkm.plugin.form.suggestion** because application plugins system will try to load from there.*

*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*

More information at Register a new plugin.

Methods description

Method	*Type*	*Description*
getSuggestions(String nodeUuid, String nodePath, Select sel)	*List<String>*	*Return a list of suggested words to tag the document.* *The method need the document uuid (nodeUuid) and the document path (nodePath).* *Select can be used as a collection of possible*

		values (enclosed dictionary).

Example of Suggestion Analyzer

```
package com.openkm.plugin.form.suggestion;

import com.openkm.bean.form.Option;
import com.openkm.bean.form.Select;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.db.bean.NodeDocumentVersion;
import com.openkm.db.service.NodeDocumentSrv;
import com.openkm.db.service.NodeDocumentUtilsSrv;
import com.openkm.db.service.NodeDocumentVersionSrv;
import com.openkm.extractor.TextExtractorWork;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalUtils;
import com.openkm.search.lucene.LuceneUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

@PluginImplementation
public class DocumentTermsSuggestion extends BasePlugin implements Suggestion {
    private static final Logger log = LoggerFactory.getLogger(DocumentTermsSuggestion.class);

    @Autowired
    private NodeDocumentVersionSrv nodeDocumentVersionSrv;

    @Autowired
    private NodeDocumentUtilsSrv nodeDocumentUtilsSrv;

    @Autowired
    private NodeDocumentSrv nodeDocumentSrv;

    @Autowired
    private LuceneUtils luceneUtils;

    @Override
    public List<String> getSuggestions(String nodeUuid, String nodePath, Select sel) throws PathNotFoundExceptior
   DatabaseException {
        log.debug("getSuggestions({}, {}, {})", nodeUuid, nodePath, sel);
        List<String> list = new ArrayList<>();

        try {
            if (nodeDocumentSrv.itemExists(nodeUuid)) {
                // Get text extraction
                if (!nodeDocumentUtilsSrv.isTextExtracted(nodeUuid)) {
                    // Force text extraction
                    NodeDocumentVersion nDocVer = nodeDocumentVersionSrv.findCurrentVersion(nodeUuid);
                    TextExtractorWork tew = new TextExtractorWork();
                    tew.setDocUuid(nodeUuid);
                    tew.setDocPath(nodePath);
                    tew.setDocVerUuid(nDocVer.getUuid());
                    tew.setTenant(PrincipalUtils.getTenantId());

                    // Execute extractor
                    nodeDocumentUtilsSrv.textExtractorHelper(tew);
```

```
                }

                for (String term : luceneUtils.getTerms(nodeUuid)) {
                    for (Option opt : sel.getOptions()) {
                        if (term.equals(opt.getLabel().toLowerCase())) {
                            if (!list.contains(opt.getValue())) {
                                list.add(opt.getValue());
                                break;
                            }
                        }
                    }
                }
            }
        } catch (IOException e) {
            throw new SuggestionException(e.getMessage(), e);
        }

        return list;
    }
}
```

Creating your own Suggestbox plugin

You can create your own Suggestbox

Conditions:

- *The new Suggestbox class **must implement the "SuggestBoxValues"** interface.*
- *The new Suggestbox class **must be declared under the package "com.openkm.plugin.form.values".***
- *The new Suggestbox class **must be annotated with "@PluginImplementation".***
- *The new Suggestbox class **must extend of "BasePlugin".***

Suggestbox interface:

```
package com.openkm.plugin.form.values;

import com.openkm.db.bean.KeyValue;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

public interface SuggestBoxValues extends Plugin {

    String getName();

    List<KeyValue> getSuggestBoxValuesByFilter(String filterValue) throws SuggestBoxException;

    KeyValue getExactKeyValueByClassName(String value) throws SuggestBoxException;

}
```

*The **new class must be loaded into the package com.openkm.plugin.form.values** because application plugin system will try to load from there.*



*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*



More information at: Register a new plugin.

Methods description

Method	*Type*	*Description*
getName()	*String*	*Return the name what will be shown at plugins table.*

getSuggestBoxValuesByFilter(String filterValue)	*List<KeyValue>*	*Return a list with the object KeyValue.*
getExactKeyValueByClassName(String value)	*KeyValue*	*Return a object KeyValue.*

Example of implementation

```
package com.openkm.plugin.form.values;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
import com.openkm.db.bean.KeyValue;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalAdapterException;

@PluginImplementation
public class SuggestBoxUserList extends BasePlugin implements SuggestBoxValues {

    @Autowired
    private OKMAuth okmAuth;

    @Override
    public String getName() {
        return "User list suggest box";
    }

    @Override
    public List<KeyValue> getSuggestBoxValuesByFilter(String filterValue) throws SuggestBoxException {
        List<KeyValue> list = new ArrayList<>();

        try {
            for (CommonUser user : okmAuth.getUsers(null)) {
                if (user.getName().toLowerCase().contains(filterValue.toLowerCase())) {
                    KeyValue value = new KeyValue();
                    value.setKey(user.getId());
                    value.setValue(user.getName());
                    list.add(value);
                }
            }
        } catch (PrincipalAdapterException e) {
            throw new SuggestBoxException(e);
        }

        return list;
    }

    @Override
    public KeyValue getExactKeyValueByClassName(String value) throws SuggestBoxException {
        KeyValue keyValue = null;

        try {
            for (CommonUser user : okmAuth.getUsers(null)) {
                if (user.equals(value)) {
```

```
                keyValue = new KeyValue();
                keyValue.setKey(user.getId());
                keyValue.setValue(user.getName());
                break;
            }
          }
        } catch (PrincipalAdapterException e) {
          throw new SuggestBoxException(e);
        }

        return keyValue;
    }
}
```

Creating your own Text extractor

You can create your own Text Extractor

Conditions:

- *The new Text Extractor class* ***must implement the "FieldValidator"*** *interface.*
- *The new Text Extractor class* ***must be declared under the package "com.openkm.plugin.extractor".***
- *The new Text Extractor class* ***must be annotated with "@PluginImplementation"****.*
- *The new Text Extractor class* ***must extend of "AbstractTextExtractor".***

Text Extractor interface:

```
package com.openkm.plugin.extractor;

import net.xeoh.plugins.base.Plugin;

import java.io.IOException;
import java.io.InputStream;

public interface TextExtractor extends Plugin {

    String[] getContentTypes();

    String extractText(InputStream stream, String type, String encoding) throws IOException;
}
```

The ***new class must be loaded into the package com.openkm.plugin.extractor*** *because application plugins system will try to load from there.*

> *Do not miss the tag* ***@PluginImplementation*** *otherwise the application plugin system will not be able to retrieve the new class.*

> *More information at Register a new plugin.*

Methods description

Method	Type	Description
getContentTypes()	*String[]*	*Returns the MIME types supported by this extractor. The returned strings must be in lower case, and the returned array must not be empty.*
extractText(InputStream stream, String type, String	*String*	*Returns a reader for the text content of the given binary document. The content type and character encoding (if available and applicable) are given*

encoding)		*as arguments.*

Example of the Text extractor implementation

```
package com.openkm.plugin.extractor;

import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.io.IOUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;

/**
 * Text extractor for plain text.
 */
@PluginImplementation
public class PlainTextExtractor extends AbstractTextExtractor {

    private static final Logger log = LoggerFactory.getLogger(PlainTextExtractor.class);

    /**
     * Creates a new <code>PlainTextExtractor</code> instance.
     */
    public PlainTextExtractor() {
        super(new String[]{"text/plain"});
    }

    // ------------------------------------------------------------< TextExtractor >
    /**
     * Wraps the given input stream to an {@link InputStreamReader} using the
     * given encoding, or the platform default encoding if the encoding is not
     * given or is unsupported. Closes the stream and returns an empty reader if
     * the given encoding is not supported.
     *
     * @param stream binary stream
     * @param type ignored
     * @param encoding character encoding, optional
     * @return reader for the plain text content
     * @throws IOException if the binary stream can not be closed in case of an
     * encoding issue
     */
    @Override
    public String extractText(InputStream stream, String type, String encoding) throws IOException {
        log.debug("extractText({}, {}, {})", stream, type, encoding);

        try {
            if (encoding != null) {
                return IOUtils.toString(stream, encoding);
            }
        } catch (UnsupportedEncodingException e) {
            log.warn("Unsupported encoding '{}', using default ({}) instead.", encoding, System.getProperty("file.encoding
        }

        return IOUtils.toString(stream, StandardCharsets.UTF_8);
    }
}
```

Creating your own Version Number Adapter

The version number adapter is used by the application to set the logic that takes control of document version number each time the document is updated.



Keep in mind changing version number adapter when application already has document version it means sometimes you must migrate older version numbers to the new format or a new implementation must take in consideration older ones.

You can create your own Version Number Adapter

Conditions:

- *The new Form Validator class **must implement the "VersionNumerationAdapter"** interface.*
- *The new Form Validator class **must be declared under the package "com.openkm.plugin.vernum".***
- *The new Form Validator class **must be annotated with "@PluginImplementation".***
- *The new Form Validator class **must extend of "BasePlugin".***

Version Number Adapter interface:

```
package com.openkm.plugin.vernum;

import com.openkm.db.bean.NodeVersion;
import net.xeoh.plugins.base.Plugin;

public interface VersionNumerationAdapter extends Plugin {

    int INCREASE_DEFAULT = 0;
    int INCREASE_MINOR = 1;
    int INCREASE_MAJOR = 2;
    int INCREASE_MAJOR_MINOR = 3;

    String qs = "from NodeVersion nv where nv.parent=:parent and nv.name=:name";

    String getInitialVersionNumber();

    int getVersionNumberIncrement();

    String getNextVersionNumber(NodeVersion nVer, int increment);
}
```

*The **new class must be loaded into the package com.openkm.plugin.vernum** because application plugin system will try to load from there.*



*Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.*



More information at Register a new plugin.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*To **enable the new Version number adapter** go to **Administration > Configuration parameters** > find the parameter named **version.numeration.adapter** and modify the value with **com.openkm.vernum.OwnNumerationAdapter***

*To take effect this change need to **restating application**.*

Methods description

Method	*Type*	*Description*
getInitialVersionNumber()	*String*	*Return the first version of a new document.* *First version usually will be "1.0".*
getVersionNumberIncrement()	*int*	*Return a number that indicate the available increase options.* *Available values:* • *VersionNumerationAdapter.INCREASE_DEFAULT* • *VersionNumerationAdapter.INCREASE_MINOR* • *VersionNumerationAdapter.INCREASE_MAJOR* • *VersionNumerationAdapter.INCREASE_MAJOR_MINOR* *INCREASE_NONE: Not showing any option, increase by default.* *INCREASE_MINOR: Only shows the option of increase minor version, if not selected, increase by default.* *INCREASE_MAYOR: Only shows the option of increase mayor version, if not selected, increase by default.* *INCREASE_MAJOR_MINOR: Shows the options of increase major and minor version, if not selected, increase by default.*
String getNextVersionNumber(Session session, NodeDocument nDoc, NodeDocumentVersion nDocVer, int increment)()	*String*	*Calculate the next version number from a given one*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example of the Version number adapter implementation

```
package com.openkm.plugin.vernum;

import com.openkm.core.Config;
import com.openkm.db.bean.NodeVersion;
import com.openkm.db.repository.NodeVersionRepo;
import com.openkm.plugin.BasePlugin;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

@PluginImplementation
public class PlainVersionNumerationAdapter extends BasePlugin implements VersionNumerationAdapter {

    @Autowired
    private NodeVersionRepo nodeVersionRepo;

    @Override
    public String getInitialVersionNumber() {
        return String.format(Config.VERSION_NUMERATION_FORMAT, 1);
    }

    @Override
    public int getVersionNumberIncrement() {
        return INCREASE_DEFAULT;
    }

    @Override
    public String getNextVersionNumber(NodeVersion nVer, int increment) {
        String versionNumber = nVer.getName();
        int nextVerNumber = Integer.parseInt(versionNumber);
        NodeVersion nv;

        do {
            nextVerNumber++;
            String name = String.valueOf(nextVerNumber);
            nv = nodeVersionRepo.getNodeVersionByNodeAndName(nVer.getParent(), name);
        } while (nv != null);

        return String.format(Config.VERSION_NUMERATION_FORMAT, nextVerNumber);
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating templates

You can create new documents based on Templates. The application can handle 3 types of template documents:

- *PDF*
- *LibreOffice*
- *Text (This includes other text-based like HTML, for example).*

In order to generate the final document from the template, we need to make an association with Metadata . The final document will be assigned the corresponding Metadata with the user input values, so you can use them for a future search.

The template syntax, which is used by the FreeMarker Template Engine. This engine can be used from making simply variable substitution to loops and other control structure.



In the samples we use only the basic ones, but you can learn more at FreeMarker Manual.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating LibreOffice templates

Advantages of using LibreOffice templates:

- *Can produce odt file.*
- *Less time expended in the creation of the document.*

Disadvantages in comparison with Text templates:

- *Produce bigger documents.*
- *More difficulties in maintaining the templates in time.*

Preliminaries

Register the Metadata group definition at ***Administration*** *>* *Metadata*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Sample Template" name="okg:tpl">
    <input label="Name" name="okp:tpl.name"/>
    <input label="Birth Date" name="okp:tpl.birth_date" type="date" />
    <select label="Language" name="okp:tpl.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
  </property-group>
</property-groups>
```

In order to make a relation between the template form field and the property group, you have to name them following a naming convention: replace ":" and "." by "_". Here you can see the matching between both fields:

Metadata field name	*Template field name*	*Template field value*
okp:tpl.name	*okp_tpl_name*	*${okp_tpl_name!}*
okp:tpl.birth_date	*okp_tpl_birth_date*	*${okp_tpl_birth_date!}*
okp:tpl.language	*okp_tpl_language*	*${okp_tpl_language!}*

Creating the template

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

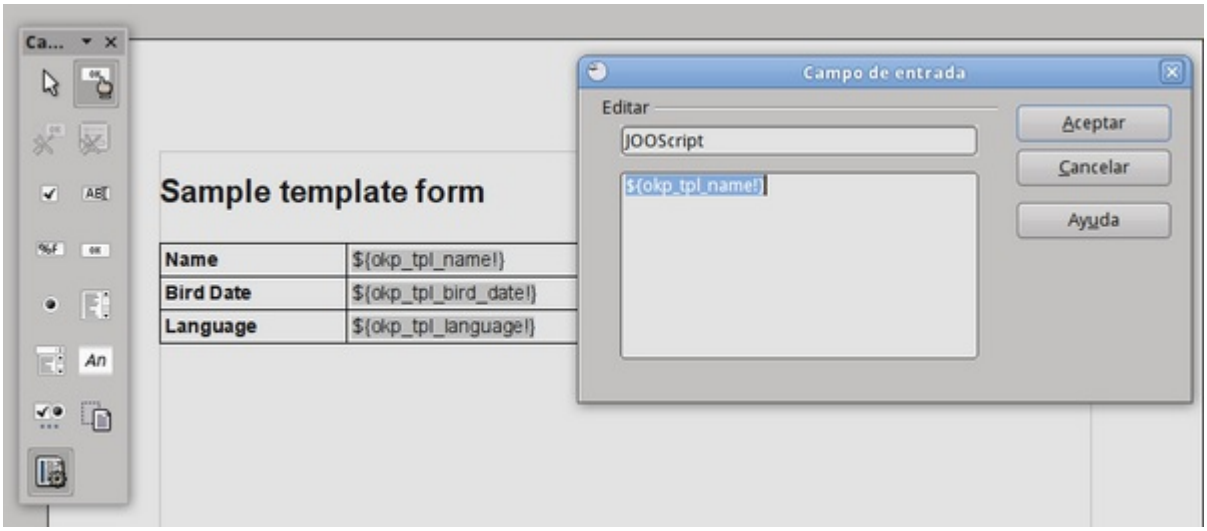
To create a template in LibreOffice go to ***View*** > ***Toolbars*** > ***Form Controls***.

A new dialog box will appear and you will be able to add form elements to the document. This dialogue box contains stuff such as Labels, Text Boxes, Check Boxes, etc.

The field that will be replaced with the value of a variable passed can be specified using the standard FreeMarker expression notation by typing the variable name into the document as follows:

```
Hi ${name}!
```

However, mixing processing instructions and normal text into the document may become confusing and clutter the layout. So JODReports provides an alternative way of inserting fields. You can insert a visual field in LibreOffice Writer from the menu ***Insert*** / ***Fields*** / ***Other...*** *(or just press* ***Ctrl+F2****), then click on the* ***Functions tab*** *and select* ***Input field***. *Change the field Reference to* ***JOOScript***. *In the following field Content enter the variable name (e.g. "$name"). This field will typically be displayed by Writer as a grayed rectangle with the variable name, while moving the mouse over it will reveal the field reference.*



This is the LibreOffice document used in this sample *tpl.odt*.

Read about LibreOffice templates at *JODReports*.

Date format

JODReports uses FreeMarker inside, so you can take advantage of the FreeMarker formatting flexibilities. For example you can set date format by this way:

```
${okp_tpl_birth_date?string("yyyy-MM-dd")}
```

If date may be null, then you need to check this (in case of odt templates should use #if between "[" and "]"):

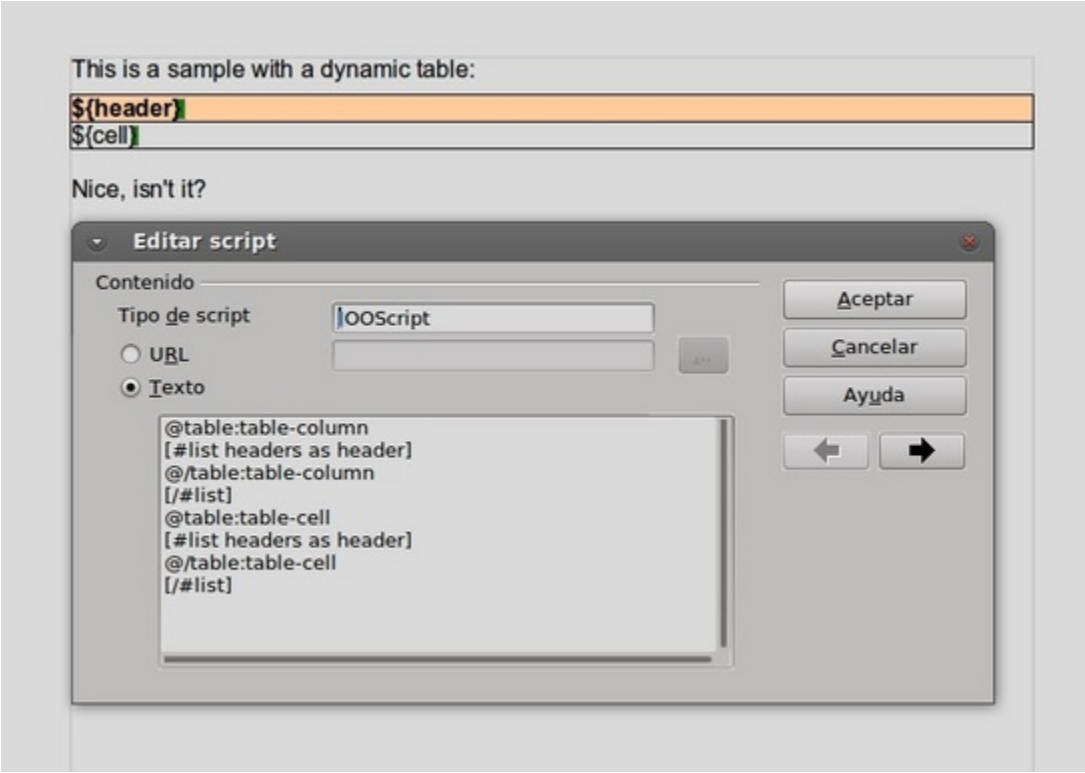
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
[#if okp_tpl_birth_date??]${okp_tpl_birth_date?string("yyyy-MM-dd")}[/#if]
```

Learn more about FreeMarker formatting at Built-in Reference.

Dynamic tables

You can even use the template feature to generate dynamic tables. Lets see an example.



In the screen shot you can see a document with a table. The header has a colored background and also is bold. The script shows the options to generate the table headers.

This is the LibreOffice document used in this sample: dynamic_table.odt

Learn more about this feature at:

- *JODReports: Inserting Directives at Special Locations.*
- *Dynamic table column count*

Configuring sample

- ***Upload tpl.odt** file at **/okm:templates/documents***
- ***Register metadata** definition **okg:tpl** at **Administration** > Metadata*
- ***Enable metadata** from **profiles** at **Administration** > Profiles*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- ***Add metadata*** *group okg:tpl to tpl.odt file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating PDF template

Advantages of using PDF templates:

- *Can produce PDF files.*

Disadvantages in comparison with Text templates:

- *Produce bigger documents.*
- *More difficulties in maintaining the templates in time.*
- *More time needed in the creation of the document (especially when the PDF optimization configuration parameter is enabled)*

You can create PDF template forms with several applications:

- *Acrobat Pro*
- *Master PDF Editor*

Every PDF form field name needs to be unique, you can't set the same property value to several form fields.

Preliminaries

Register the Metadata group definition at ***Administration*** *>* *Metadata*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Sample Template" name="okg:tpl">
    <input label="Name" name="okp:tpl.name"/>
    <input label="Birth Date" name="okp:tpl.birth_date" type="date"/>
    <select label="Language" name="okp:tpl.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <checkbox label="Important" name="okp:tpl.important"/>
  </property-group>
</property-groups>
```

In order to make a relation between the template form field and the property group, you have to name them following a naming convention: replace ":" and "." by "_". Here you can see the matching between both fields:

Metadata field name	*Template field name*	*Template field value*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

okp:tpl.name	*okp_tpl_name*	*${okp_tpl_name!}*
okp:tpl.birth_date	*okp_tpl_birth_date*	*${okp_tpl_birth_date!}*
okp:tpl.language	*okp_tpl_language*	*${okp_tpl_language!}*
okp:tpl.important	*okp_tpl_important*	*${okp_tpl_important!}*

Creating the template



We encourage using Acrobat Pro or Master PDF Editor to avoid problems.

This is a sample template created with Master PDF Editor:

- *template.pdf*

Date format

OpenKM uses FreeMarker inside, so you can take advantage of the FreeMarker formatting flexibilities. For example you can set date format by this way:

```
${okp_tpl_birth_date?string("yyyy-MM-dd")}
```

Learn more about FreeMarker formatting at Built-in Reference.

Configuring sample

- ***Upload tpl.pdf** file at **/okm:templates/documents***
- ***Register metadata** definition **okg:tpl** at **Administration** > Metadata*
- ***Enable metadata** from **profiles** at **Administration** > Profiles*
- ***Add metadata** group okg:tpl to tpl.pdf file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating Text templates

Advantages of using Text templates:

- *Can produce HTML or PDF files.*
- *The small size of the produced document.*
- *Easily maintain the templates in time.*
- *Less time expended in the creation of the document.*

Preliminaries

Register the Metadata group definition at ***Administration*** *>* *Metadata*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Sample Template" name="okg:tpl">
    <input label="Name" name="okp:tpl.name"/>
    <input label="Birth Date" name="okp:tpl.birth_date" type="date" />
    <select label="Language" name="okp:tpl.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
  </property-group>
</property-groups>
```

In order to make a relation between the template form field and the property group, you have to name them following a naming convention: replace ":" and "." by "_". Here you can see the matching between both fields:

Metadata field name	*Template field name*	*Template field value*
okp:tpl.name	*okp_tpl_name*	*${okp_tpl_name!}*
okp:tpl.birth_date	*okp_tpl_birth_date*	*${okp_tpl_birth_date!}*
okp:tpl.language	*okp_tpl_language*	*${okp_tpl_language!}*

Creating the template

For this kind of template we will create an HTML one. This is the template source code:

```
<html>
  <body>
    <h1>Sample template</h1>
    <table>
      <tr>
        <td><b>Name</b></td>
        <td>${okp_tpl_name!}</td>
      </tr>
      <tr>
        <td><b>Birth Date</b></td>
        <td>${okp_tpl_birth_date?string("yyyy-MM-dd")!}</td>
      </tr>
      <tr>
        <td><b>Language</b></td>
        <td>${okp_tpl_language!}</td>
      </tr>
    </table>
  </body>
</html>
```

*The tags **${okp_tpl_name!}**, **${okp_tpl_birth_date!}** and **${okp_tpl_language!}** will be replaced by the user input values. An error will occur and abort the template processing if you try to access a missing variable, but the ! operator handles this situation. In addition, you can put a default value in case of a mission one with this expression:*

```
${missing_value!"The default one"}
```

The result is an HTML file with the client input requested by OpenKM.

Date format

OpenKM uses FreeMarker inside, so you can take advantage of the FreeMarker formatting flexibilities. For example, you can set the date format in this way:

```
${okp_tpl_birth_date?string("yyyy-MM-dd")!}
```

Learn more about FreeMarker formatting at Built-in Reference.

Configuring sample

- ***Upload tpl.html** file at **/okm:templates/documents** with the contents described before.*
- ***Register metadata** definition **okg:tpl** at **Administration** > Metadata*
- ***Enable metadata** from **profiles** at **Administration** > Profiles*
- ***Add metadata** group okg:tpl to tpl.html file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own login module

The class must implement PrincipalAdapter interface.

```
package com.openkm.plugin.principal;

import com.openkm.bean.CommonUser;
import com.openkm.db.bean.Profile;
import com.openkm.principal.PrincipalAdapterException;
import net.xeoh.plugins.base.Plugin;

import java.util.List;

public interface PrincipalAdapter extends Plugin {

    /**
     * Method to retrieve all users from a authentication source.
     *
     * @return A Collection with all the users.
     * @throws PrincipalAdapterException If any error occurs.
     */
    List<CommonUser> getUsers() throws PrincipalAdapterException;

    /**
     * Method to retrieve all roles from a authentication source.
     *
     * @return A Collection with all the roles.
     * @throws PrincipalAdapterException If any error occurs.
     */
    List<String> getRoles() throws PrincipalAdapterException;

    /**
     * Method to retrieve user data
     *
     * @return the user data
     * @throws PrincipalAdapterException If any error occurs.
     */
    CommonUser getUser(String userId) throws PrincipalAdapterException;

    /**
     * Method to retrieve all users from a role.
     *
     * @return A Collection with all the users within a role.
     * @throws PrincipalAdapterException If any error occurs.
     */
    List<CommonUser> getUsersByRole(String role) throws PrincipalAdapterException;

    /**
     * Method to retrieve all roles from a user.
     *
     * @return A Collection with all the roles of the user.
     * @throws PrincipalAdapterException If any error occurs.
     */
    List<String> getRolesByUser(String user) throws PrincipalAdapterException;

    /**
     * Method to retrieve the user password
     *
     * @param user A user id.
     * @return The password of the user.
     * @throws PrincipalAdapterException If any error occurs.
     */
```

```
    String getPassword(String user) throws PrincipalAdapterException;

    /*
   * -------------------------------------------------------------------
   * These methods only works if using the OpenKM user database.
   * -------------------------------------------------------------------
    */
    /**
     * Method to create a new user
     *
     * @param user User data
     * @return User
     * @throws PrincipalAdapterException If any error occurs.
     */
    CommonUser createUser(CommonUser user) throws PrincipalAdapterException;

    /**
     * Method to create a delete a user
     *
     * @param user A user id.
     * @throws PrincipalAdapterException If any error occurs.
     */
    void deleteUser(String user) throws PrincipalAdapterException;

    /**
     * Update user information
     *
     * @param user User data
     * @return User
     * @throws PrincipalAdapterException If any error occurs.
     */
    CommonUser updateUser(CommonUser user) throws PrincipalAdapterException;

    /**
     * Method to create a new role
     *
     * @param role A role id.
     * @throws PrincipalAdapterException If any error occurs.
     */
    void createRole(String role, boolean active) throws PrincipalAdapterException;

    /**
     * Method to create a delete a role
     *
     * @param role A role id.
     * @throws PrincipalAdapterException If any error occurs.
     */
    void deleteRole(String role) throws PrincipalAdapterException;

    /**
     * Update role information
     *
     * @param role A role id..
     * @throws PrincipalAdapterException If any error occurs.
     */
    void updateRole(String role, boolean active) throws PrincipalAdapterException;

    /**
     * Method to assign a role
     *
     * @param user A user id.
     * @param role A role id.
     * @throws PrincipalAdapterException If any error occurs.
     */
    void assignRole(String user, String role) throws PrincipalAdapterException;

    /**
```

```
     * Method to remove a role
     *
     * @param user A user id.
     * @param role A role id.
     * @throws PrincipalAdapterException If any error occurs.
     */
    void removeRole(String user, String role) throws PrincipalAdapterException;

    /**
     * Get all profiles
     *
     * @param filterByActive
     * @return A list with all the registered profiles
     * @throws PrincipalAdapterException
     */
    List<Profile> getProfiles(boolean filterByActive) throws PrincipalAdapterException;

    /**
     * Get user profile
     *
     * @return The profile assigned to this user
     * @throws PrincipalAdapterException
     */
    Profile getUserProfile(String userId) throws PrincipalAdapterException;

    /**
     * Set the user profile
     *
     * @param userId The user id.
     * @param profileId The profile to be assigned.
     * @throws PrincipalAdapterException
     */
    void setUserProfile(String userId, long profileId) throws PrincipalAdapterException;

    boolean isManageUsers();

    boolean isManageRoles();
}
```

PrincipalAdapter sample based on properties

```
package com.openkm.plugin.principal;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.List;
import java.util.Properties;
import java.util.StringTokenizer;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
import com.openkm.core.Config;
import com.openkm.db.bean.Profile;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalAdapterException;

public class UsersRolesPrincipalAdapter extends BasePlugin implements PrincipalAdapter {
```

```
private static Logger log = LoggerFactory.getLogger(UsersRolesPrincipalAdapter.class);

@Autowired
private OKMAuth okmAuth;

@Override
public List<CommonUser> getUsers() throws PrincipalAdapterException {
    log.debug("getUsers()");
    List<CommonUser> list = new ArrayList<>();
    Properties prop = new Properties();

    try {
        prop.load(new FileInputStream(Config.HOME_DIR + "/server/default/conf/props/openkm-users.properties"));
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    for (Enumeration<Object> e = prop.keys(); e.hasMoreElements();) {
        String user = (String) e.nextElement();
        if (!Config.SYSTEM_USER.equals(user)) {
            CommonUser commonUser = okmAuth.getUser(null, user);
            list.add(commonUser);
        }
    }

    log.debug("getUsers: {}", list);
    return list;
}

@Override
public List<String> getRoles() throws PrincipalAdapterException {
    log.debug("getRoles()");
    List<String> list = new ArrayList<String>();
    Properties prop = new Properties();

    try {
        prop.load(new FileInputStream(Config.HOME_DIR + "/server/default/conf/props/openkm-roles.properties"));
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    for (Enumeration<Object> e = prop.elements(); e.hasMoreElements();) {
        for (StringTokenizer st = new StringTokenizer((String) e.nextElement(), ","); st.hasMoreTokens();) {
            String role = st.nextToken();

            if (!Config.DEFAULT_ADMIN_ROLE.equals(role) && !list.contains(role)) {
                list.add(role);
            }
        }
    }

    log.debug("getRoles: {}", list);
    return list;
}

@Override
public CommonUser getUser(String userId) throws PrincipalAdapterException {
    throw new UnsupportedOperationException("getUser");
}

@Override
public List<CommonUser> getUsersByRole(String role) throws PrincipalAdapterException {
```

```
        throw new UnsupportedOperationException("getUsersByRole");
    }

    @Override
    public List<String> getRolesByUser(String user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getRolesByUser");
    }

    @Override
    public String getPassword(String user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getPassword");
    }

    @Override
    public CommonUser createUser(CommonUser user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("createUser");
    }

    @Override
    public void deleteUser(String user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("deleteUser");
    }

    @Override
    public CommonUser updateUser(CommonUser user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("updateUser");
    }

    @Override
    public void createRole(String role, boolean active) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("createRole");
    }

    @Override
    public void deleteRole(String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("deleteRole");
    }

    @Override
    public void updateRole(String role, boolean active) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("updateRole");
    }

    @Override
    public void assignRole(String user, String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("assignRole");
    }

    @Override
    public void removeRole(String user, String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("removeRole");
    }

    @Override
    public List<Profile> getProfiles(boolean filterByActive) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getProfiles");
    }

    @Override
    public Profile getUserProfile(String userId) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getUserProfile");
    }

    @Override
    public void setUserProfile(String userId, long profileId) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("setUserProfile");
    }
```

```
    @Override
    public boolean isManageUsers() {
        throw new UnsupportedOperationException("isManageUsers");
    }

    @Override
    public boolean isManageRoles() {
        throw new UnsupportedOperationException("isManageRoles");
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own parameters

You can take advantage of the application configuration parameters feature to store your own parameters.

To create a new configuration parameter:

- *Go to **Administration** > **Utilities** > **Database query**.*
- *Select **JDBC option** from the list at the bottom.*
- *Write a query like this*

```
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('your.property.name', 'strin
```

- *Click at **Execute button** at top right.*



*The **type value** - CFG_VALUE column - **must be written in lower case** !.*

Sample

You can access it from source code to the properties.



When the property not exists will be initialized with default value (on sample code below "initial value").

```
import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
// If the parameter not exists, by default is set to "initial value"
String value = configSrv.getString("your.property.name", "initial value");
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating your own password validator

You can create your own password validator.

Conditions:

- *The new password validator class **must implement the "PasswordValidator"** interface.*
- *The new password validator class **must be declared under the package "com.openkm.validator.password".***

Password validator interface:

```
package com.openkm.validator.password;

import com.openkm.validator.ValidatorException;

public interface PasswordValidator {

    /**
     * Validate
     *
     * @param password
     * @throws ValidatorException
     */
    void Validate(String password) throws ValidatorException;

}
```

From password validator class you can use several configuration parameters:

- *Config.PROPERTY_VALIDATOR_PASSWORD to return value of parameter "validator.password".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_LENGTH to return value of parameter "validator.password.min.length".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MAX_LENGTH to return value of parameter "validator.password.max.length".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_LOWERCASE to return value of parameter "validator.password.min.lowercase".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_UPPERCASE to return value of parameter "validator.password.min.uppercase".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_DIGITS to return value of parameter "validator.password.min.digits".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_SPECIAL to return value of parameter "validator.password.mini.special".*

More information at Password validation configuration.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Sample

***CompletePasswordValidator** class:*

```
package com.openkm.validator.password;

import com.openkm.core.Config;
import com.openkm.validator.ValidatorException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Complex password validator
 */
public class CompletePasswordValidator implements PasswordValidator {

    @SuppressWarnings("unused")
    private static Logger log = LoggerFactory.getLogger(CompletePasswordValidator.class);

    @Override
    public void Validate(String password) throws ValidatorException {
        validateLength(password);
        checkLowerCase(password);
        checkUpperCase(password);
        checkDigits(password);
        checkSpecial(password);
    }

    /**
     * Validate length
     */
    private void validateLength(String password) throws ValidatorException {
        if (Config.VALIDATOR_PASSWORD_MIN_LENGTH > 0
                && password.length() < Config.VALIDATOR_PASSWORD_MIN_LENGTH) {
            throw new ValidatorException(Config.VALIDATOR_PASSWORD_ERROR_MIN_LENGTH);
        }

        if (Config.VALIDATOR_PASSWORD_MAX_LENGTH > 0
                && password.length() > Config.VALIDATOR_PASSWORD_MAX_LENGTH) {
            throw new ValidatorException(Config.VALIDATOR_PASSWORD_ERROR_MAX_LENGTH);
        }
    }

    /**
     * Validate lowercase characters
     */
    private void checkLowerCase(String password) throws ValidatorException {
        int count = 0;

        if (Config.VALIDATOR_PASSWORD_MIN_LOWERCASE > 0) {
            for (int i = 0; i < password.length(); i++) {
                if (Character.isLowerCase(password.charAt(i))) {
                    count++;
                }
            }

            if (Config.VALIDATOR_PASSWORD_MIN_LOWERCASE > count) {
                throw new ValidatorException(Config.VALIDATOR_PASSWORD_ERROR_MIN_LOWERCASE);
            }
        }
    }

    /**
     * Validate uppercase characters
```

```
     */
    private void checkUpperCase(String password) throws ValidatorException {
        int count = 0;

        if (Config.VALIDATOR_PASSWORD_MIN_UPPERCASE > 0) {
            for (int i = 0; i < password.length(); i++) {
                if (Character.isUpperCase(password.charAt(i))) {
                    count++;
                }
            }

            if (Config.VALIDATOR_PASSWORD_MIN_UPPERCASE > count) {
                throw new ValidatorException(Config.VALIDATOR_PASSWORD_ERROR_MIN_UPPERCASE);
            }
        }
    }

    /**
     * Validate digits
     */
    private void checkDigits(String password) throws ValidatorException {
        int count = 0;

        if (Config.VALIDATOR_PASSWORD_MIN_DIGITS > 0) {
            for (int i = 0; i < password.length(); i++) {
                if (Character.isDigit(password.charAt(i))) {
                    count++;
                }
            }

            if (Config.VALIDATOR_PASSWORD_MIN_DIGITS > count) {
                throw new ValidatorException(Config.VALIDATOR_PASSWORD_ERROR_MIN_DIGITS);
            }
        }
    }

    /**
     * Validate special characters
     */
    private void checkSpecial(String password) throws ValidatorException {
        int count = 0;

        if (Config.VALIDATOR_PASSWORD_MIN_SPECIAL > 0) {
            for (int i = 0; i < password.length(); i++) {
                if (!Character.isLetterOrDigit(password.charAt(i))
                        && !Character.isWhitespace(password.charAt(i))) {
                    count++;
                }
            }

            if (Config.VALIDATOR_PASSWORD_MIN_SPECIAL > count) {
                throw new ValidatorException(Config.VALIDATOR_PASSWORD_ERROR_MIN_SPECIAL);
            }
        }
    }
}
```

Creating your own reports

OpenKM use JasperReports library for generating reports. For building reports you have a tool named iReport.

*The **JasperReports Library** is the world's most popular Open Source reporting engine. It is entirely written in Java and it is able to use data coming from any kind of data source and produce pixel-perfect documents that can be viewed, printed or exported in a variety of document formats including HTML, PDF, Excel, LibreOffice and Word. More information at http://community.jaspersoft.com/.*

***iReport** is the free, Open Source report designer for JasperReports and JasperReports Server. Create very sophisticated layouts containing charts, images, subreports, crosstabs and much more. Access your data through JDBC, TableModels, JavaBeans, XML, Hibernate, CSV, and custom sources. Then publish your reports as PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX, or LibreOffice. More information at http://community.jaspersoft.com/project/ireport-designer.*

In this section you will find an introduction for creating reports with JasperReports and iReports tool. Consider it as a quick introduction, not a complete guide. You must follow some tutorials, read books or papers to illustrate on it.

There are some available books and guides you should read:

- *The Definitive Guide to iReport by Giulio Toffoli.*
- *JasperReports Ultimate Guide.*
- *Ireport Designer Tutorials & Help.*

With OpenKM you can use reports based on SQL or reports based on Scripting. For beginners we suggest starting with reports based on SQL.

- *Introducing to reports based on SQL.*
- *Introducing to reports based on Java.*
- *Introducing to reports with filtering.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Introducing to reports based on SQL

- *Install **iReport** tool.*
- ***Execute iReport** tool.*



OpenKM comes with an specific JasperReports Library engine what supports older version but now newer. At Application version compatibility table you have correspondence list between OpenKM and JasperReports Library engine versions.

You can set specific output version from iReport tool:

- *Click on **Tools menu** > **options**.*
- *From **iReport** > **General tab** click on **Compatibility tab**.*
- *Set the **Jasper version compatibility**.*
- *Click on **OK button**.*

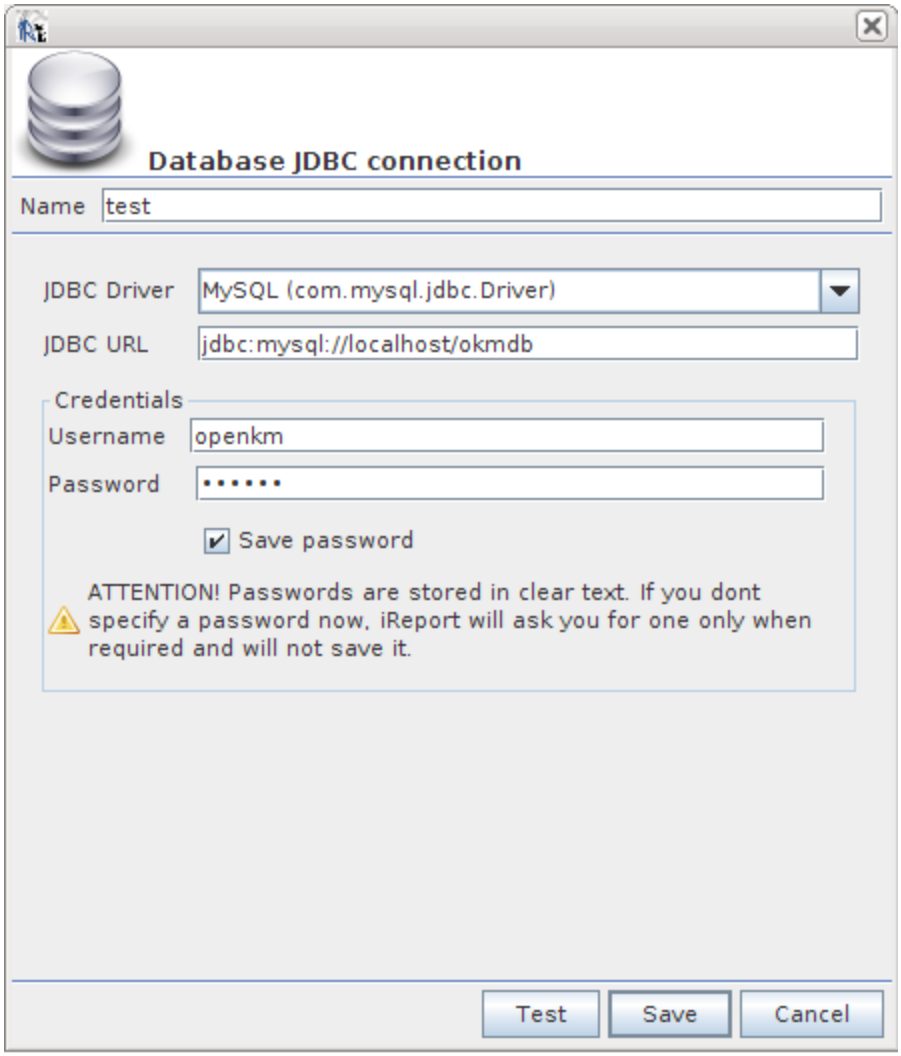
Create datasource

- *Click on **Report datasources button**.*



- *Click on **New button**.*
- *Choose "**Database JDBC connection**" option.*
- *Click on **Next button**.*
- *Set the **name**.*
- *Choose the **JDBC driver**.*
- *Click on **Text button** to check the connection parameters.*
- *Click on **Save button**.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

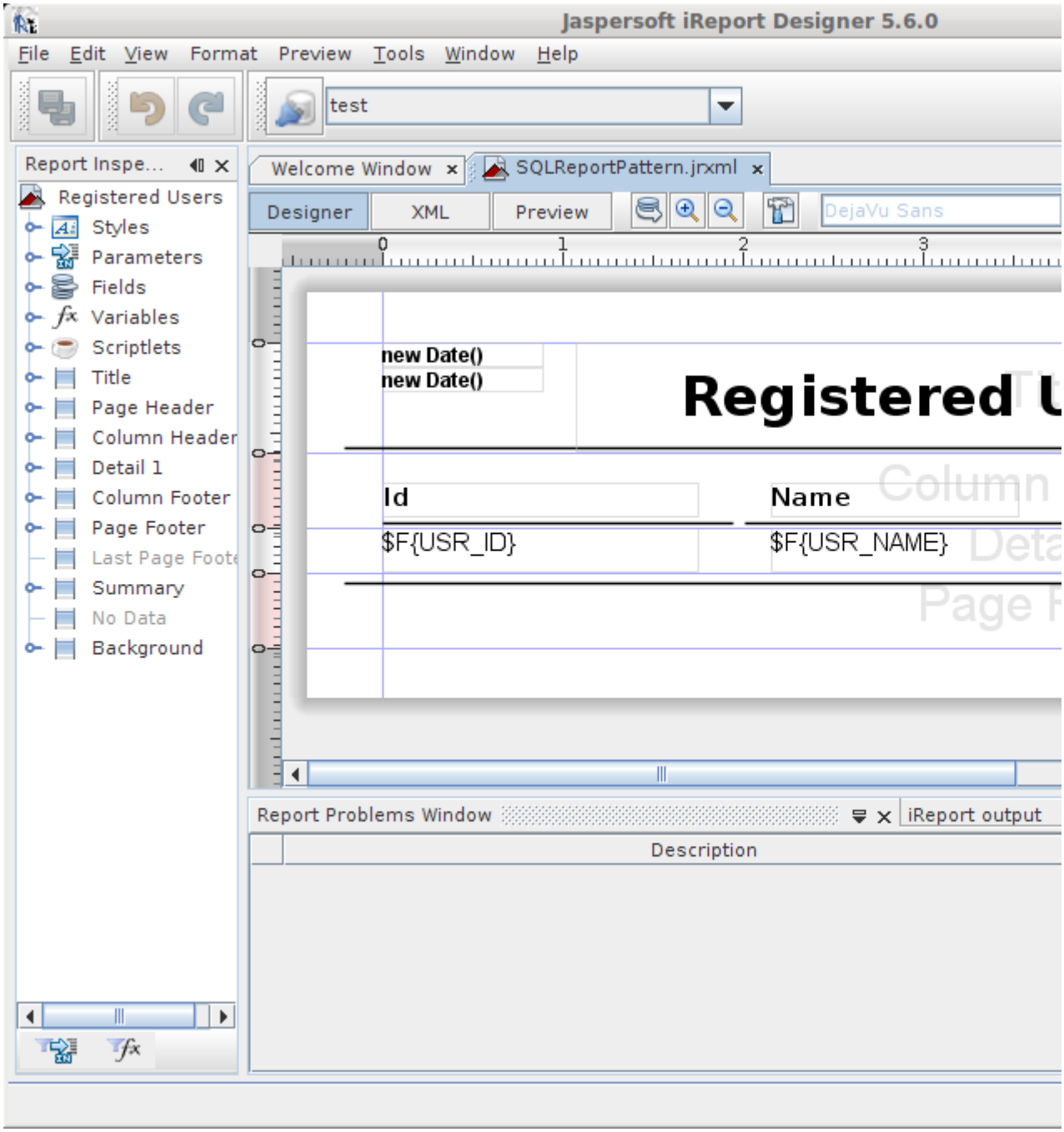


- ***Check as default checkbox***.
- *Click on* ***Close button***.

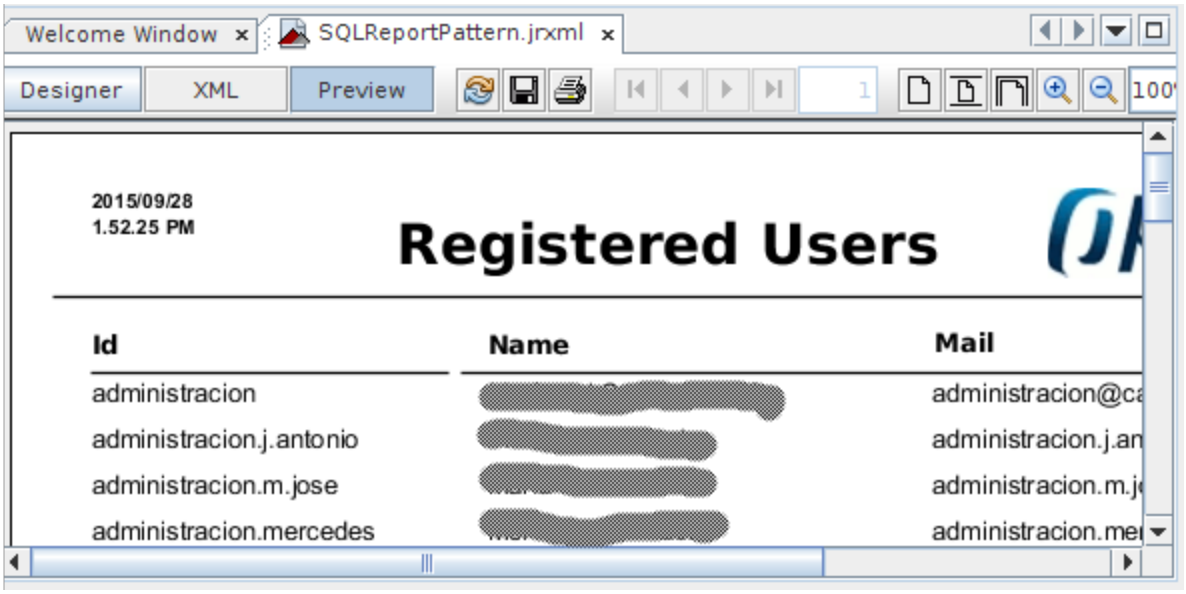
Testing SQL sample report

> *For testing the SQL Report sample you must have an OpenKM running.*

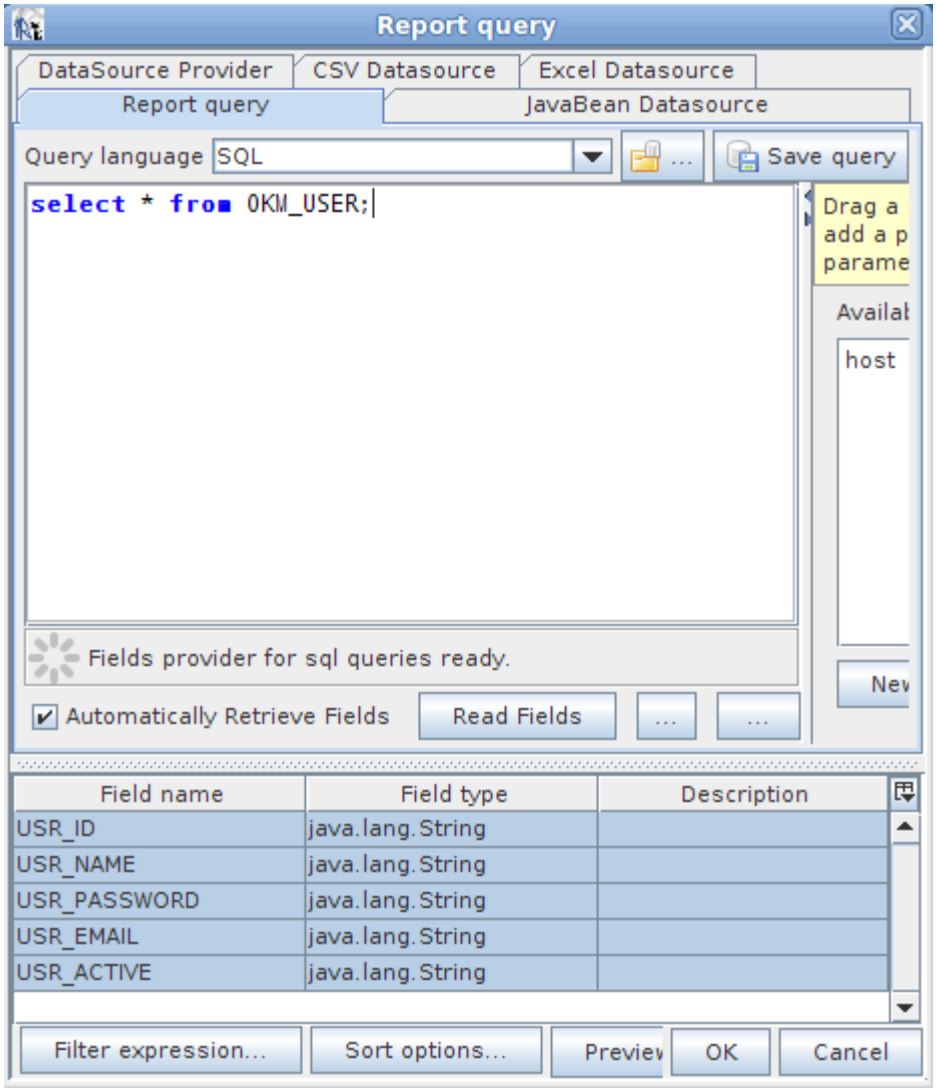
- *Download the SQLReportPattern.jrxml file.*
- *Open the file.*



- *Click on **Preview tab**.*
- *Set the **host** value (for example http://localhost:8080/OpenKM).*



- *Click on* ***Designer tab***.
- *Click on*  ***report query button***.



From this point you can add your jrxml file and register to your OpenKM at ***Administration*** *>* *Reports*.

Closer look into JRXML file

The iReport tool is creating for you the jasper report file in XML format.

Although the iReport tool is doing all work for you in friendly user interface, is a good idea for you to take a look - at least for curiousity - about what kind of file is being created.

Query section

```
<queryString>
  <![CDATA[select * from OKM_USER;]]>
</queryString>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Fields section

```
<field name="USR_ID" class="java.lang.String"/>
<field name="USR_NAME" class="java.lang.String"/>
<field name="USR_PASSWORD" class="java.lang.String"/>
<field name="USR_EMAIL" class="java.lang.String"/>
<field name="USR_ACTIVE" class="java.lang.String"/>
```

Remote image section

```
<imageExpression><![CDATA[$P{host}+"/logo/report"]]></imageExpression>
```

Using fields in report

```
<textField isBlankWhenNull="false">
  <reportElement key="textField" x="329" y="0" width="206" height="18" uuid="5e3e2df8-a99b-4ce2-aab9-9adb97bc
    <textElement>
      <font fontName="Arial"/>
    </textElement>
  <textFieldExpression><![CDATA[$F{USR_EMAIL}]]></textFieldExpression>
</textField>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Introducing to reports based on Java

- *Install the **iReport** tool.*
- ***Execute iReport** tool.*



OpenKM comes with an specific JasperReports Library engine that supports older versions but now a newer one can be found. At Application version compatibility table you have correspondence list between OpenKM and JasperReports Library engine versions.

You can set specific output version from iReport tool:

- *Click on **Tools menu** > **options**.*
- *From **iReport** > **General tab** click on **Compatibility tab**.*
- *Set the **Jasper version compatibility**.*
- *Click on **OK button**.*

Testing Java sample report

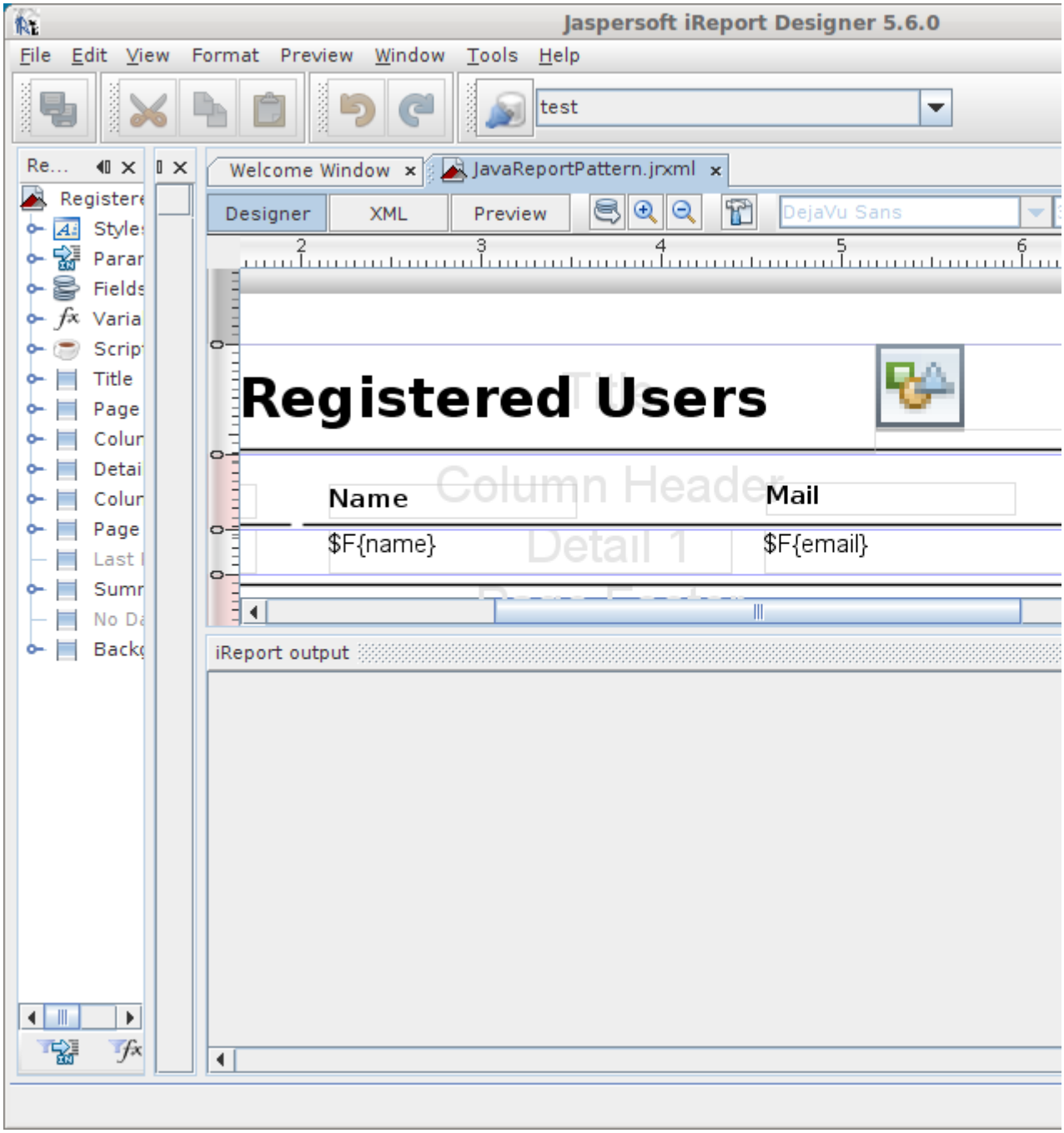


***Reports based on Java can not be previewed directly from iReport tool.** Must be checked from an OpenKM.*

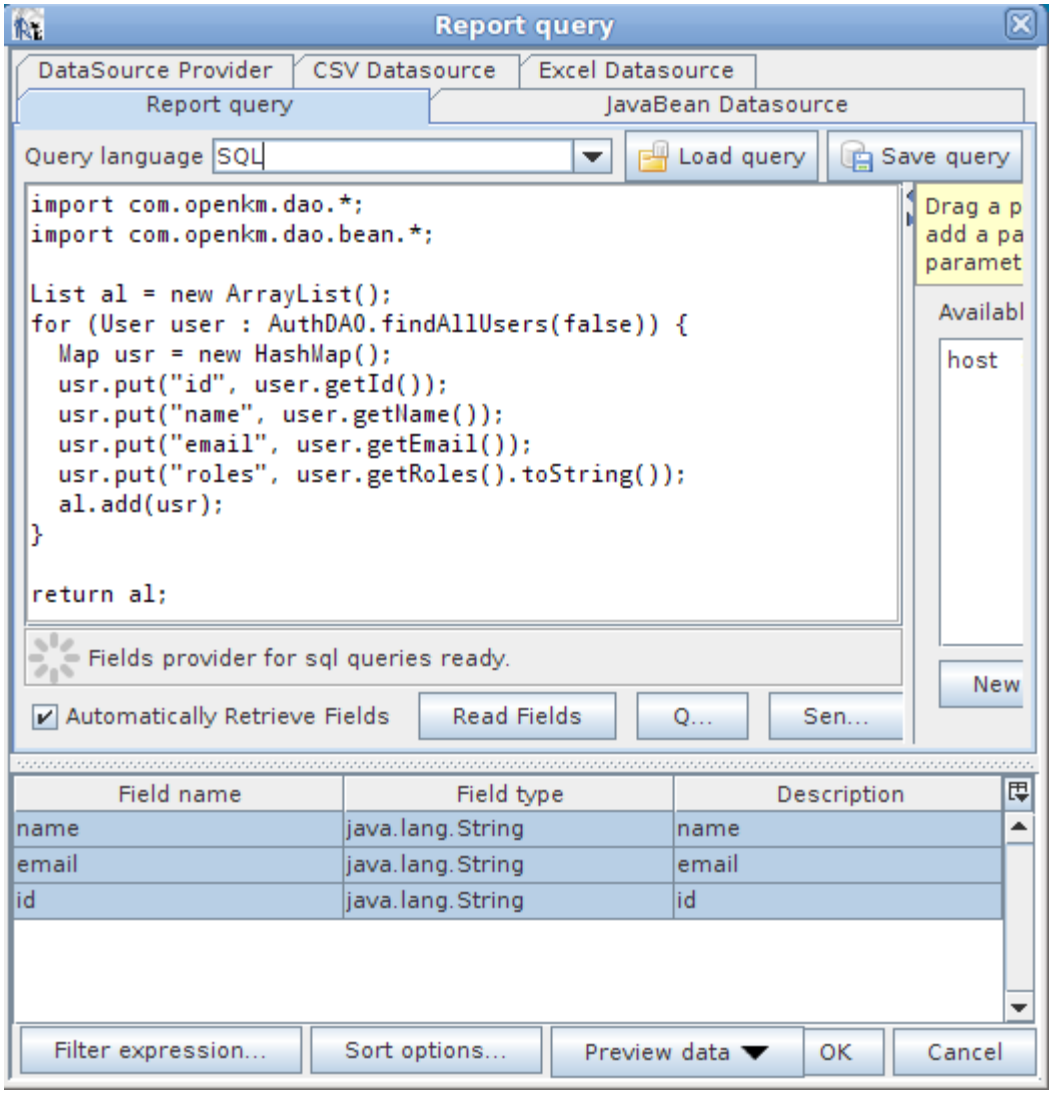
For testing SQL Report sample you must have an OpenKM running.

- *Download the JavaReportPattern.jrxml file.*
- *Open the file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



- *Click on* ***Designer tab***.
- *Click on*  ***report query button***.



In this case in place of SQL query you can set a Java Scripting (see also Scripting samples).

The script must return a List<Map> object where each value into the map corresponds to a column value.

From this point you can add your jrxml file and register to your OpenKM at ***Administration*** *>* Reports.

Understanding the code sample

- *Create an empty list.*
- *Iterate across all users calling the API method named AuthDAO.findAllUsers(false).*
- *Create a Map for each User object where each value corresponds to a column.*
- *Add each map into the list, where each map corresponds to a row.*

```
import com.openkm.db.service.*;
import com.openkm.bean.*;
import com.openkm.util.ContextWrapper;
import org.springframework.web.context.WebApplicationContext;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
WebApplicationContext cc = (WebApplicationContext) ContextWrapper.getContext();
AuthSrv authSrv = cc.getBean(AuthSrv.class);

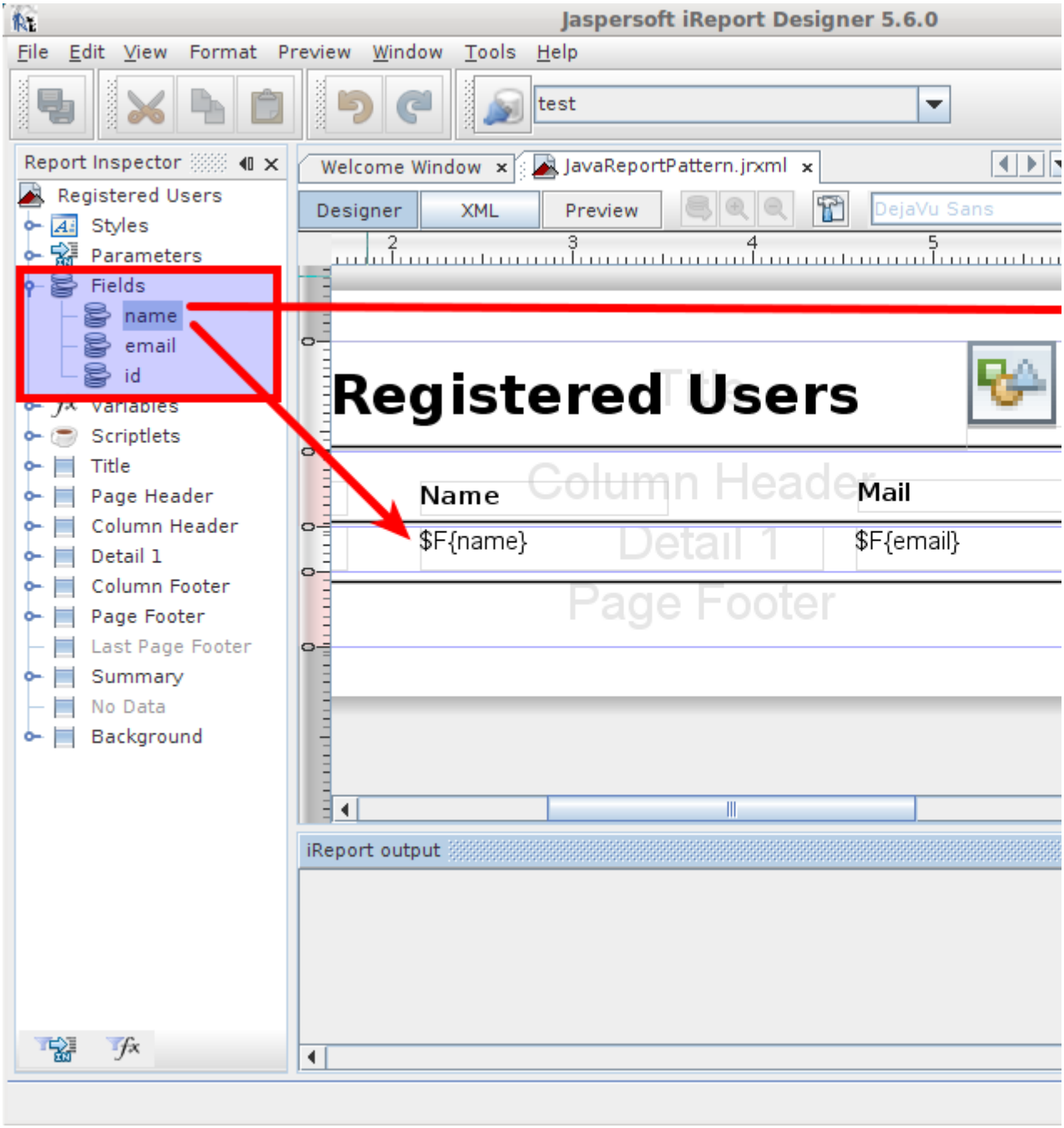
List al = new ArrayList();
for (CommonUser user : authSrv.findAllUsers(false)) {
 Map usr = new HashMap();
 usr.put("id", user.getId());
 usr.put("name", user.getName());
 usr.put("email", user.getEmail());
 usr.put("roles", user.getRoles().toString());
 al.add(usr);
}

return al;
```

Using in report

- *For each key in map create a field.*
- *Use the field in report*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


Jaspersoft iReport Designer 5.6.0
File Edit View Format Preview Window Tools Help
test
Report Inspector
Registered Users
Styles
Parameters
Fields
name
email
id
Variables
Scriptlets
Title
Page Header
Column Header
Detail 1
Column Footer
Page Footer
Last Page Footer
Summary
No Data
Background
Welcome Window
JavaReportPattern.jrxml
Designer
XML
Preview
DejaVu Sans
Registered Users
Name
Mail
$F{name}
$F{email}
iReport output

Introducing to reports with filtering

When you execute a report from OpenKM the view might be shown a form requiring the user to introduce values for filtering purposes. The most common case is filtering by a range of dates.

*The reports with form filtering end with the extension "**.rep**". Really is a ZIP file renamed as "**.rep**".*

These reports contain into two files:

- *JRXML file*
- *File named params.xml*

Download the UsersLogin.zip and take a look into (this is the original UsersLogin.rep renamed to ZIP).

Understanding filtering

To understand the form filter we will use the UsersLogin.rep report (download the UsersLogin.zip).

Params XML file

In the sample has been set two filtering fields:

Field	Type	Description
from_date	**Input**	*Field used to filter from date.*
to_date	**Input**	*Field used to filtering to date.*

The params.xml file:

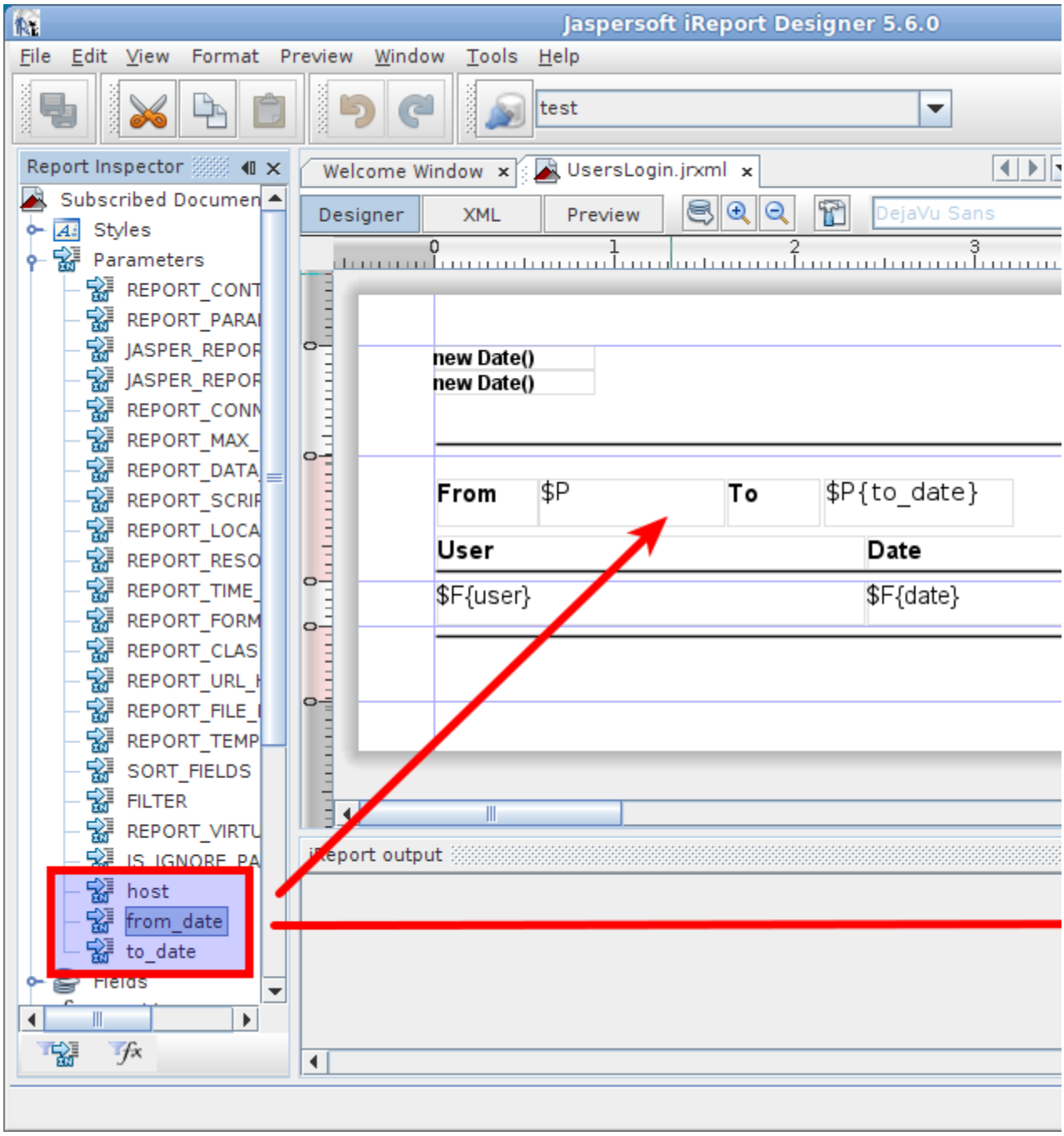
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <input label="From" name="from_date" type="date">
    <validator type="req"/>
  </input>
  <input label="To" name="to_date" type="date">
    <validator type="req"/>
  </input>
</report-parameters>
```

Using fields in report

In the report are created two parameters:

Parameter	Class	Description

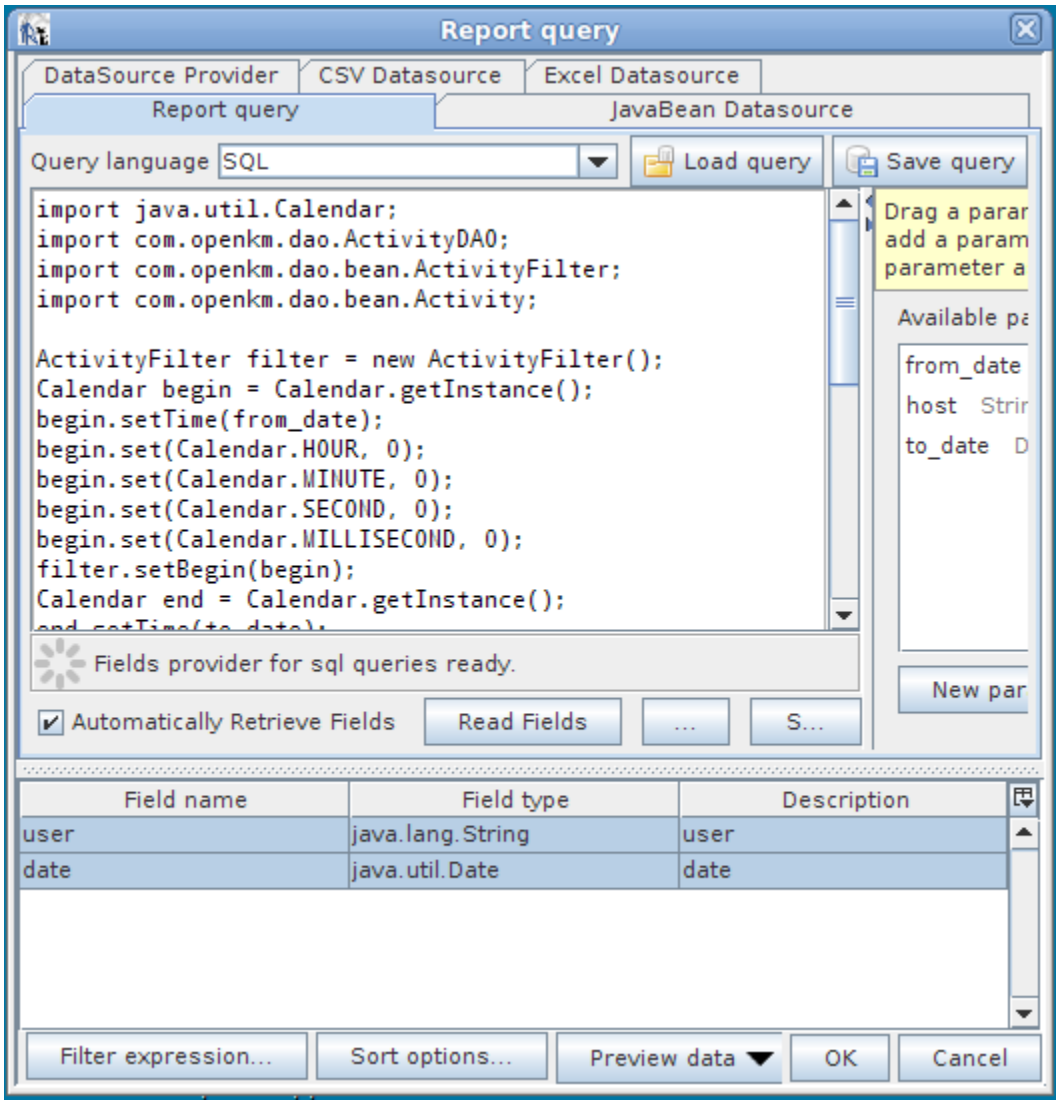
from_date	***java.util.Date***	*From date.*
to_date	***java.util.Date***	*To date.*



The parameters are also used in SQL or Java reports.

Java sample

- *Click on* ***report query button***.



The ***from_date*** *and* ***to_date*** *variables are injected in code execution and can be used directly as a sample below.*

```
import java.util.Calendar;
import com.openkm.db.service.ActivitySrv;
import com.openkm.db.bean.ActivityFilter;
import com.openkm.db.bean.Activity;
import com.openkm.util.ContextWrapper;
import org.springframework.web.context.WebApplicationContext;

WebApplicationContext cc = (WebApplicationContext) ContextWrapper.getContext();
ActivitySrv activitySrv = cc.getBean(ActivitySrv.class);
ActivityFilter filter = new ActivityFilter();
Calendar begin = Calendar.getInstance();
begin.setTime(from_date);
begin.set(Calendar.HOUR, 0);
begin.set(Calendar.MINUTE, 0);
begin.set(Calendar.SECOND, 0);
begin.set(Calendar.MILLISECOND, 0);
```

```
filter.setBegin(begin);
Calendar end = Calendar.getInstance();
end.setTime(to_date);
end.set(Calendar.HOUR, 0);
end.set(Calendar.MINUTE, 0);
end.set(Calendar.SECOND, 0);
end.set(Calendar.MILLISECOND, 0);
filter.setEnd(end);
filter.setAction("LOGIN");
List al = new ArrayList();
for (Iterator it = activitySrv.findByFilter(filter).iterator(); it.hasNext();){
    Activity act = (Activity) it.next();
    Map actMap = new HashMap();
    actMap.put("user", act.getUser());
    actMap.put("date", act.getDate().getTime());
    al.add(actMap);
}

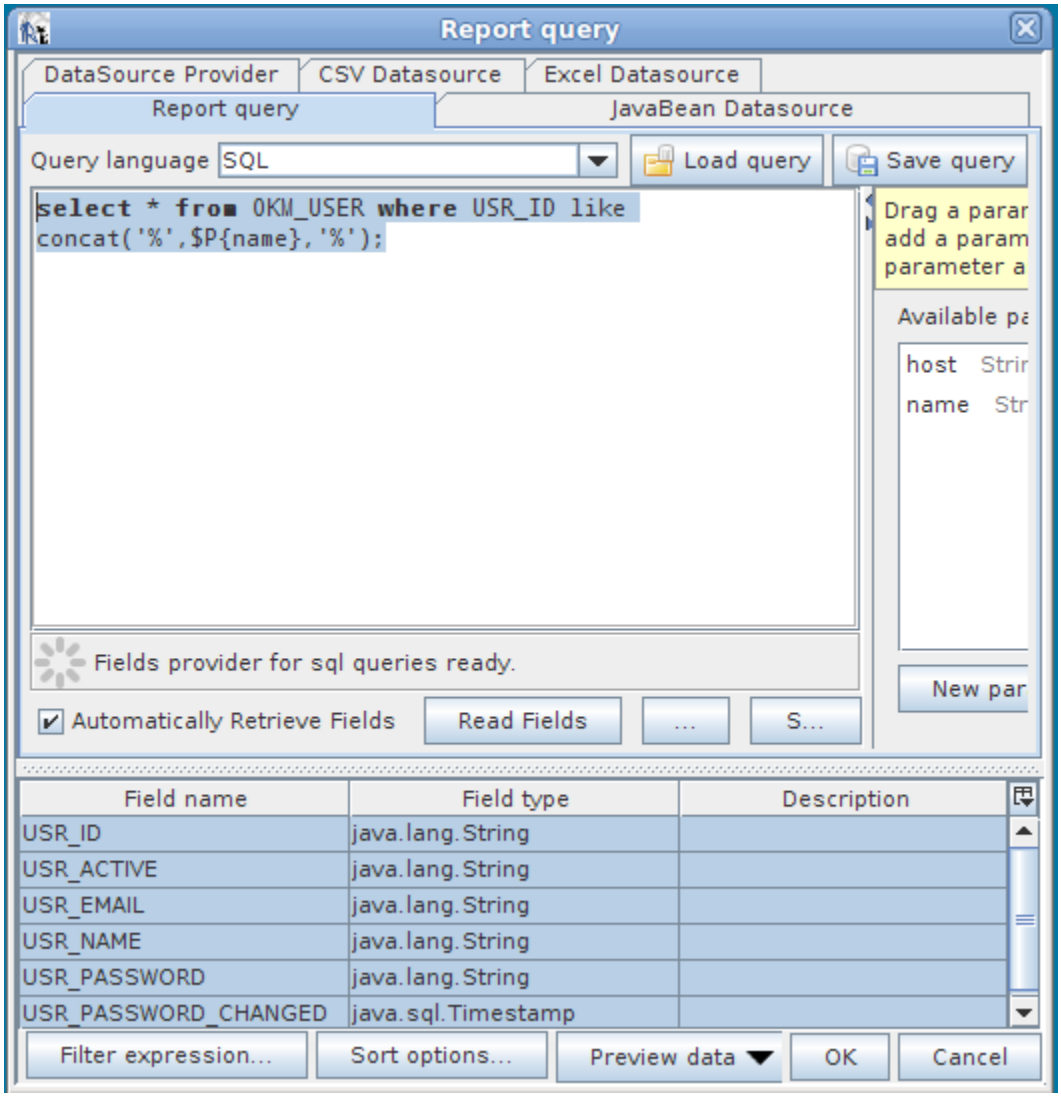
return al
```

SQL sample



The sample below is not based on the UsersLogin.rep report.

Click on  ***report query button***.



*In the sample there is a parameter named "**name**" of class "**java.lang.String**" used for filtering the users list.*

```
select * from OKM_USER where USR_ID like concat('%',$P{name},'%')
```

The engine replaces "$P{name}" tag with "String" value. For example, if the parameter value is "okm" the query executed will be:

```
select * from OKM_USER where USR_ID like concat('%','okm','%')
```

The contact function merges several Strings, the final result is:

```
select * from OKM_USER where USR_ID like '%okm%')
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params form fields

Params forms are based on formal XML definition. The DTD (Document Type Definition) defines the structure and the legal elements and attributes of an XML document.

> *Can see latest DTD in our website http://www.openkm.com/dtd/report-parameters-{major}.{minor}.dtd where **{major}** and **{minor}** values are two numbers (major release and minor release). For example the location for version 2.1 of the dtd is http://www.openkm.com/dtd/report-parameters-2.1.dtd*

> *Each application version supports only a couple of DTD definitions, read Application version compatibility table to see which are supported by your application.*

Basic XML skeleton

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                          "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>

</report-parameters>
```

- *The **DOCTYPE** is a formal definition of some DTD. In the previous example the **report-parameters-2.1.dtd**.*
- *The XML begins with **<report-parameters> tag** and ends with **</report-parameters>**. Into this hicheracy are defined the metada groups.*

Basic example

Available field types:

- *Checkbox*
- *Input*
- *Select*
- *Separator*
- *Suggestbox*
- *Text*
- *TextArea*

More information about field type elements at Params fields

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                          "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
  <input label="From" name="from_date" type="date">
    <validator type="req"/>
  </input>
  <input label="To" name="to_date" type="date">
    <validator type="req"/>
  </input>
</report-parameters>
```

Using file system DTD

If the server has no access to the Internet, then the DTD file must be accessible in some way, for example can store a copy in your application server.



The XML definition must be changed to refer the new DTD location, otherwise you will get and application error while trying to processing DTD file.

Can be file system (example below) or a valid intranet URL.

```
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.1//EN"
                                 "file:///home/openkm/report-parameters-2.1.dtd">
```

- *In the example,* ***/home/openkm/report-parameters-2.1.dtd*** *is the file system path to the file.*

Params fields

These are the available field types:

Field type	*Description*
Checkbox	*Params Checkbox field*
Input	*Params Input field*
Select	*Params Select field*
Separator	*Params Separator field*
Suggestbox	*Params Suggestbox field*
Text	*Params Text field*
TextArea	*Params TextArea field*

Some field types can have children elements, validator or option elements. This hierarchy is permitted:

- *params form*
  - *Checkbox*
    - *Validator*
  - *Input*
    - *Validator*
  - *Select*
    - *Option*
    - *Validator*
  - *Separator*
  - *Suggestbox*
    - *Validator*
  - *Text*
  - *TextArea*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Validator*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params Validator element

The goal of the Validator element is to ensure that the user provided necessary and properly formatted information needed to successfully complete an operation.

Validator can only be used in combination with other fields:

- *Params Input field.*
- *Params Suggestbox field.*
- *Params Checkbox field.*
- *Params TextArea field.*
- *Params Select field.*

Available validators:

name	*Description*	*Parameter required*
req	*Required value.*	*No*
alpha	*Alphabetic value.*	*No*
alphanum	*Alphanumeric value.*	*No*
dec	*Decimal value.*	*No*
num	*Numeric value.*	*No*
email	*E-mail value.*	*No*
url	*URL value.*	*No*
maxlen	*Set the maximum length allowed.*	*Yes*
minlen	*Set the minimum length required.*	*Yes*
lt	*Less than.*	*Yes*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

gt	*Greater than.*	*Yes*
min	*Minimum value.*	*Yes*
max	*Maximum value.*	*Yes*
regexp	*Regular expression.*	*Yes*
plugin	*Create your own form validator.* *More information at Creating your own Form Validator plugin.*	*Yes*

Parent:

- *Params Input field.*
- *Params Suggestbox field.*
- *Params Checkbox field.*
- *Params TextArea field.*
- *Params Select field.*

Basic example

- *Value required "**req**".*
- *Value must be decimal "**dec**".*
- *The maximum lenght of the decimal value is 6 digits "**maxlen**".*

```
<input label="Decimal value" name="decimal">
   <validator type="dec"/>
   <validator type="req"/>
   <validator type="maxlen" parameter="6"/>
</input>
```

Regular expression

If you want a validation which is not included in these prefefined, you can use a regular expression. For example, imagine you want to restrict an input to lowecase ASCII values:

```
<input label="Lowercase value" name="regExp">
    <validator type="regexp" parameter="[a-z]+" />
</input>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params Input field

in most cases, is used to enter free text form, although it can be used to save HTTP links, date data, or refer to the application folder (as an internal link to this folder).

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
type	*Type value. Allowed types:* • *text.* • *date.* • *link.* • *folder.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

Child elements:

- *Validator. For more information, see the Params Validator element.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
                                "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <input label="From" name="from_date" type="date">
    <validator type="req" />
  </input>
</report-parameters>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params Suggestbox field

This is an implementation of the typical AJAX suggest box component:

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* ⚠ *Two fields can't have the same name in the same task definition. The name **must be unique**.*	*true*
table	*Metadata table name.* *The application can store values in a metadata table that can be used in this query.*	*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*
filterQuery	*A Hibernate query to filter the data. Use {0} to place the user input.*	*
valueQuery	*A Hibernate query to obtain the value of an identifier. Use {0} to place the identifier.*	*
className	*A class used to get the selected options.*	*
filterMinLen	*The minimum chars in input to begin to filter results.*	*true*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

dialogTitle	*The title of the HTML dialog.*	*true*

Child elements:

- *Validator. For more information, see Params Validator element.*

*** There're three options to get suggested box values (using **one of them is mandatory**):*

- *table + filterQuery + valueQuery*
- *filterQuery + valueQuery*
- *className*

No other combination of them is allowed.

Based on the table, filterQuery, and valueQuery

Suggestbox values are defined in metadata tables.

- *The application uses two tables named OKM_DB_METADATA_TYPE AND OKM_DB_METADATA_VALUE to help store options values used in selects. That's very useful when you got a large number of values.*
- *OKM_DB_METADATA_TYPE has stored metadata table definition (table name, column name, and column type) in the table.*
- *In the table, OKM_DB_METADATA_VALUE have stored record values.*

Create metadata table definition.

*The **table** will be called **country**, with two columns, **country_id** and **country_name**.*

*The **country_id** column is an **integer** type, and **country_name** is a **text** type.*

*The **country_id** is set as **column 0 with col00** value.*

*The **country_name** is set as **column 1 with col01** value.*

- *Go to **Administration** > **Utilities** > **Database query**.*
- *At the bottom right, select **jdbc** from the list.*
- ***Paste the SQL** in the box.*

```
DELETE FROM OKM_DB_METADATA_TYPE WHERE DMT_TABLE='country';
INSERT INTO OKM_DB_METADATA_TYPE (DMT_TABLE, DMT_REAL_COLUMN, DMT_TYPE, DMT_VIRTUAL_
INSERT INTO OKM_DB_METADATA_TYPE (DMT_TABLE, DMT_REAL_COLUMN, DMT_TYPE, DMT_VIRTUAL_
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *Click at the bottom right the **Execute button**.*

Insert metadata table values:

- *Go to **Administration** > **Utilities** > **Database query**.*
- *At the bottom right select **jdbc** from list.*
- ***Paste the SQL** in the box*

```
DELETE FROM OKM_DB_METADATA_VALUE WHERE DMV_TABLE='country';
INSERT INTO OKM_DB_METADATA_VALUE (DMV_TABLE, DMV_COL00, DMV_COL01) VALUES ('country', '00
INSERT INTO OKM_DB_METADATA_VALUE (DMV_TABLE, DMV_COL00, DMV_COL01) VALUES ('country', '00
```

- *Click at the bottom right the **Execute button**.*



The insert queries depending on your database, can not be matched, for example, in Oracle:

*INSERT INTO OKM_DB_METADATA_TYPE (**DMT_ID**, DMT_TABLE, DMT_REAL_COLUMN, DMT_TYPE, DMT_VIRTUAL_COLUMN) VALUES (**HIBERNATE_SEQUENCE.NEXTVAL**, 'country', 'col00', 'integer', 'country_id');*

Finally, the XML definition:

- *In optionsQuery, the values that begin with character $ are identified as column names. See **$country_id and $country_name** below.*
- *You can use normal SQL clauses like "**lower**", "**like**", "**order by**", etc.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                    "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
   <suggestbox label="country" name="suggestbox" width="200px"
               table="country" dialogTitle="Choose country" filterMinLen="3"
               filterQuery="select $country_id, $country_name from DatabaseMetadataValue dmv where dmv.table='c
               valueQuery="select $country_id, $country_name from DatabaseMetadataValue dmv where dmv.table=
</report-parameters>
```

Based on filterQuery and valueQuery (without table attribute)

*Now **filterQuery** and **valueQuery** could be regular SQL queries.*



You should only use {0} as the parameter the user enters to filter the values from the table.

Create COUNTRY table:

```
create table COUNTRY (
  CT_ID varchar(2),
```

```
  CT_NAME varchar(32),
  primary key(CT_ID)
);

insert into COUNTRY (CT_ID, CT_NAME) values ('es', 'Spain');
insert into COUNTRY (CT_ID, CT_NAME) values ('pt', 'Portugal');
insert into COUNTRY (CT_ID, CT_NAME) values ('it', 'Italy');
```

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                        "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
   <suggestbox label="country" name="okp:consulting.suggestbox"
               width="200px" dialogTitle="Choose country" filterMinLen="3"
               filterQuery="select ct_id, ct_name from country where ct_name like '%{0}%' order by ct_name"
               valueQuery="select ct_id, ct_name from country where ct_id='{0}'" />
</report-parameters>
```

Based on className

To help get a list from external applications or other purposes, your class can be built to retrieve the option values from those systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                        "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
   <suggestbox label="country" name="okp:consulting.suggestbox"
               width="200px" dialogTitle="Choose country" filterMinLen="3"
               className="com.openkm.plugin.form.values.SuggestBoxUserList" />
</report-parameters>
```

For more information, read Creating your own Suggestbox plugin.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params Checkbox field

Checkbox

Best used to represent boolean data:

Attribute	Description	Required
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

Child elements:

- *Validator. For more information, see the Params Validator element.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <checkbox label="Checkbox label" name="checkbox" />
</report-parameters>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params TextArea field

Print a text that can contain HTML tags.

Attribute	Description	Required
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition.* ***The name must be unique****.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

Child elements:

- *Validator. For more information, see the Params Validator element.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <textarea label="textarea label" name="textarea" width="200px" />
</report-parameters>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params Select field

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
type	*Type value. Allowed types:* • *Simple (single choice allowed).* • *Multiple (multiple selections allowed).*	*true*
table	*Metadata table name.* *The application can store values in a metadata table that can be used in this query.*	*
width	*The width of the HTML element.* *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*
optionsQuery	*A metadata query or SQL query is used to get the select options.*	*
Child elements: • *Option. Couples of XML <**option label="label text" value="unique identifier value**"/> values or <**option***		

***label="label text" value="unique identifier value" parentValue="parent unique identifier value"/>** in the case is set some hierarchical relationship.*

- *Validator. For more information, see [Params Validator element](#).*

*** There're two options to get option values:*

- *Directly defined in XML.*
- *Metadata table (table + optionsQuery).*



One of these options must be defined. Otherwise, you will get an error. A combination of several cannot be done, only one can be chosen.

Basic select example

- *Type is equal to **simple**.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                  "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <select label="select label" name="select" type="simple">
    <option label="one" value="001" />
    <option label="two" value="002" />
    <option label="three" value="003" />
  </select>
</report-parameters>
```

Allow multiple choice

- *Type is equal to **multiple**.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                  "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <select label="select label" name="select" type="multiple">
    <option label="one" value="001" />
    <option label="two" value="002" />
    <option label="three" value="003" />
  </select>
</report-parameters>
```

Hierarchical relationship

- *The field options values of "**childrenSelect**" have parent relation with "**parentSelect.**" It's defined by **parentElement="parentSelect"**.*

- *The option value "**benefits**" of the field "**childrenSelect**" has a parent value "**directives**" from the area "**parentSelect.**"*

Tree relationships:

- *directives*
  - *benefits*
  - *eligibility*
  - *services*
- *handbook*
  - *basic*
  - *normal*
  - *advanced*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                          "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <select label="Parent" name="parentSelect" type="simple">
    <option value="directives" label="Directives" />
    <option value="handbook" label="Handbook Paragraphs" />
  </select>
  <select label="Children" name="childrenSelect" type="simple" parentElement="parentSelect">
    <option value="benefits" label="Benefits" parentValue="directives" />
    <option value="elegibility" label="Eligibility" parentValue="directives" />
    <option value="services" label="Services" parentValue="directives" />
    <option value="basic" label="Basic" parentValue="handbook" />
    <option value="normal" label="Advanced" parentValue="handbook" />
    <option value="advanced" label="Normal" parentValue="handbook" />
  </select>
</report-parameters
```

Based on SQL query

We can also get these option values using an SQL query. This SQL sentence should be put on the ***optionsQuery*** *attribute, but the* ***table*** *attribute should be empty (otherwise, this query is a metadata query). Note that the first element returned in every row is the option value, and the second is the option label.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                          "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <select label="country" name="country" type="simple"
        optionsQuery="select CT_ID, CT_NAME from COUNTRY order by CT_NAME" />
</report-parameters>
```

The table used in this sample is defined as follows:

```
create table COUNTRY (
    CT_ID varchar(2),
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    CT_NAME varchar(32),
    primary key(CT_ID)
);

insert into COUNTRY (CT_ID, CT_NAME) values ('es', 'Spain');
insert into COUNTRY (CT_ID, CT_NAME) values ('pt', 'Portugal');
insert into COUNTRY (CT_ID, CT_NAME) values ('it', 'Italy');
```

Based on className

To help get a list from external applications or other purposes, your class can be built to retrieve the option values from those systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                          "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
    <select label="select label" name="okp:consulting.select" type="simple"
          className="com.openkm.plugin.form.values.OptionSelectUserList" />
</report-parameters>
```

For more information, read Creating your own Option Select Values plugin.

Suggested options

Based on the content analyzer, the application can suggest some options values. At the user interface, the recommended options will be shown in green.

The application has some implementations that can easily be extended.

- ***com.openkm.form.suggestion.DocumentContentContainsSuggestion**: Get suggestions based on document content (use String. contains)*
- ***com.openkm.form.suggestion.DocumentContentTokenizerSuggestion**: Get suggestions based on document content (Use StringTokenizer).*
- ***com.openkm.form.suggestion.DocumentTermsSuggestion**: Get suggestions based on document term vectors.*

For more information, read Creating your own Suggestion Analyzer.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                          "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
    <select label="select label"  name="okp:consulting.select" type="multiple"
          suggestion="com.openkm.form.suggestion.DocumentContentContainsSuggestion">
        <option label="one" value="001" />
        <option label="two" value="002" />
        <option label="three" value="003" />
    </select>
</report-parameters>
```

Params Separator field

Draw a separator between fields.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will works* • *width="100px" will be ignored in the KCenter UI.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <input label="Input label" name="input1" />
  <separator label="Separator label" name="separator" />
  <input label="Input label" name="input2" />
</report-parameters>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Params Text field

Print a text that can contain HTML tags.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the exact task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, UI must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	*false*
height	*The height of the HTML element.*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <text label="text label" name="consulting_text" width="200px"/>
</report-parameters>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Database stuff

There are a couple of interesting configuration properties to be set at $TOMCAT_HOME/openkm.properties file.

The parameter hibernate.statistics helps developer enabling the Hibernate statistics. By default this parameter is set to false.

```
hibernate.statistics=true
```

Use these parameters meaningful only for development and fine tuning.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Webservices

OpenKM has API exposed via Webservices with:

- *RESTful*.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

RESTful

OpenKM has a complete API exposed via REST. You can call any API method from any programming language, like Java, PHP, or Python. This feature makes it possible to create a custom client or integrate with third-party applications like a CRM or a CMS.

These URLs are protected by BASIC authentication, so you must provide a user and password to access them.

You can extend the REST API—more information at Creating your own Rest Plugin (extending REST API).

We encourage using our SDKs for accessing OpenKM across REST API rather than you building your client; more information:

- *The SDK for Java of SDK for JAVA.*
- *The SDK for .NET of SDK for .NET.*

*On almost samples, you'll see parameters named "**docId**," "**fldId**" or "**nodeId**." The value of this parameter must be a valid node **UUID**.*

Example of nodeId:

- *Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";*

Almost all samples use the command curl, more information at the curl tutorial with examples of usage.

Sample usage

*You can use an HTTP Client library or any REST client to try these API methods, which eases this process. Or you can use the **curl** command-line application. For example, you can list the children's folders:*

```
$ curl -u okmAdmin: admin -H "Accept: application/json" \
  http://localhost:8080/openkm/rest/folders/de7b07c2-e707-4390-b5cc-cae340c8c3e5/children
```

The result is:


```
[
  {
    "@type": "folder",
    "created": 1568408949000,
    "lastModified": 1568409037000,
    "path": "/okm:root/test/test",
    "author": "okmAdmin",
    "permissions": 31,
```


Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    "uuid": "0fd9f008-27e8-4980-a22b-d71c9a34060a",
    "parent": "de7b07c2-e707-4390-b5cc-cae340c8c3e5",
    "subscribed": false,
    "nodeClass": 0,
    "dispositionCurrentStage": 0,
    "subscriptors": [],
    "keywords": [],
    "categories": [],
    "notes": [],
    "linkTarget": null,
    "actualVersion": {
      "name": "1.1",
      "created": 1568409037000,
      "size": 0,
      "author": "okmAdmin",
      "actual": true,
      "comment": null,
      "checksum": null,
      "action": "PROPERTY_GROUP_ADDED",
      "params": "okg:consulting",
      "uuid": null,
      "parent": "0fd9f008-27e8-4980-a22b-d71c9a34060a"
    },
    "promotedAsRecord": false,
    "hasRelations": false,
    "description": null,
    "restricted": false,
    "nodeClassChildren": [],
    "metadataLabel": null,
    "hasChildren": false,
    "style": 0
  }
]
```

In this case, you can see the result in JSON format.

This is a Java client for the same call:

```
package com.openkm;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Authenticator;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.PasswordAuthentication;
import java.net.URL;

public class JavaRestClient {

    public static void main(String[] args) throws Exception {
        try {
            String fldUuid = "de7b07c2-e707-4390-b5cc-cae340c8c3e5";
            URL url = new URL("http://localhost:8080/openkm/rest/folders/" + fldUuid + "/children");
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.setRequestProperty("Accept", "application/json");

            Authenticator.setDefault(new Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication("okmAdmin", "admin".toCharArray());
                }
```

```
            });

            if (conn.getResponseCode() == 200) {
                BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputStream())));
                System.out.println("Output from Server .... \n");
                String output;

                while ((output = br.readLine()) != null) {
                    System.out.println(output);
                }
            } else {
                System.err.println("Failed : HTTP error code : " + conn.getResponseCode());
            }

            conn.disconnect();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Folder samples

Create a new folder:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" \
 -X POST -H "Content-Type: application/json" -d 'newfolder' \
 http://localhost:8080/openkm/rest/folders/parents/fe239ae2-4d25-4d87-88b2-924d03b87faa
```

Rename a folder:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" \
 -X PUT -H "Content-Type: application/x-www-form-urlencoded" \
 -d 'newName=test2' \
http://localhost:8080/openkm/rest/folders/fe239ae2-4d25-4d87-88b2-924d03b87faa/name
```

Document samples

To create a document, we need to provide the document binary data:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" \
 -X POST -F content=newDoc.txt -F content=@/home/openkm/doc1.txt \
 http://localhost:8080/openkm/rest/documents/parents/fe239ae2-4d25-4d87-88b2-924d03b87faa
```

Or also from an HTML form:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document Upload</title>
</head>
<body>
    <form method="POST" enctype="multipart/form-data"
        action="http://localhost:8080/openkm/rest/documents/parents/fe239ae2-4d25-4d87-88b2-924d03b87faa">
      Select file: <input type="file" name="content" size="45"/><br/>
      <input type="submit" value="Upload" />
    </form>
</body>
</html>
```

And now download it:

```
$ curl -u okmAdmin:admin \
  http://localhost:8080/openkm/rest/documents/dab830ea-2fa8-439f-837e-de15409d88b9/content
```

Search samples

Search with parameters:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/search?keyword=test&name=doc*&mimeType=application/pdf
```

Search with metadata:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/search?keyword=test&property='okp:consulting.name=alfa'
```

Security samples

Show granted users:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/auth/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/permissions/users
```

Show granted roles:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/auth/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/permissions/roles
```

Add a grant:

```
$ curl -v -u okmAdmin:admin -X POST -H "Content-Type: application/x-www-form-urlencoded" \
  -d userId=sochoa -d permissions=15 -d recursive=false \
  http://localhost:8080/openkm/rest/auth/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/permissions/us
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Revoke a grant:

```
$ curl -v -u okmAdmin:admin -X DELETE -H "Content-Type: application/x-www-form-urlencoded" \
 -d permissions=15 -d recursive=false \
 http://localhost:8080/openkm/rest/auth/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/permissions/users/sochoa
```

Metadata samples

The metadata group definition used in the samples:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name" >
      <validator type="req"/>
    </input>
    <textarea label="Comment" name="okp:consulting.comment" >
      <validator type="req"/>
    </textarea>
  </property-group>
</property-groups>
```

Add a metadata group:

```
$ curl -u okmAdmin:admin -H "Accept: application/json"  \
 -X POST -H "Content-type: application/json"
 -d '{"properties":[{"name":"okp:consulting.comment","value":"new comment"},
{"name":"okp:consulting.name","value":"new name"},
{"name":"okp:consulting.date","value":"20190917163517"}],"grpName":"okg:consulting"}' \
 http://localhost:8080/openkm/rest/propertygroups/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/groups
```

Set metadata group values:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" \
 -X PUT -H "Content-Type: application/json" \
 -d '[{"name":"okp:consulting.comment","value":"set comment"},{"name":"okp:consulting.name","value":"set name"},
{"name":"okp:consulting.date","value":"20190917164717"}]' \
 http://localhost:8080/openkm/rest/propertygroups/nodes/fe239ae2-4d25-4d87-88b2-
924d03b87faa/groups/okg:consulting/properties
```

Notes samples

Create a note:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" \
  -X POST -H "Content-Type: application/x-www-form-urlencoded" -d 'text=Hello, world!' \
  http://localhost:8080/openkm/rest/note/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa
```

Conversion samples

With the conversion feature from the OpenKM API, you can convert, for example, a document from MS Office Word format to PDF, among other conversion types available.

Convert a doc file to pdf:

```
$ curl -u okmAdmin:admin -F content=@sample.doc -o sample.pdf \
  http://localhost:8080/openkm/rest/conversion/doc/pdf
```

Or also from an HTML form:

```
<html>
  <body>
    <form method="POST" enctype="multipart/form-data"
          action="http://localhost:8080/openkm/services/rest/conversion/doc2pdf">
      Select file: <input type="file" name="content" size="45"/><br/>
      <input type="submit" value="Convert" />
    </form>
  </body>
</html>
```

Convert a doc file to txt:

```
$ curl -u okmAdmin:admin -F content=@sample.doc http://localhost:8080/openkm/rest/conversion/doc/txt
```

Convert an image file to txt:

```
$ curl -u okmAdmin:admin -F content=@sample.tif http://localhost:8080/rest/rest/conversion/img/txt
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Swagger

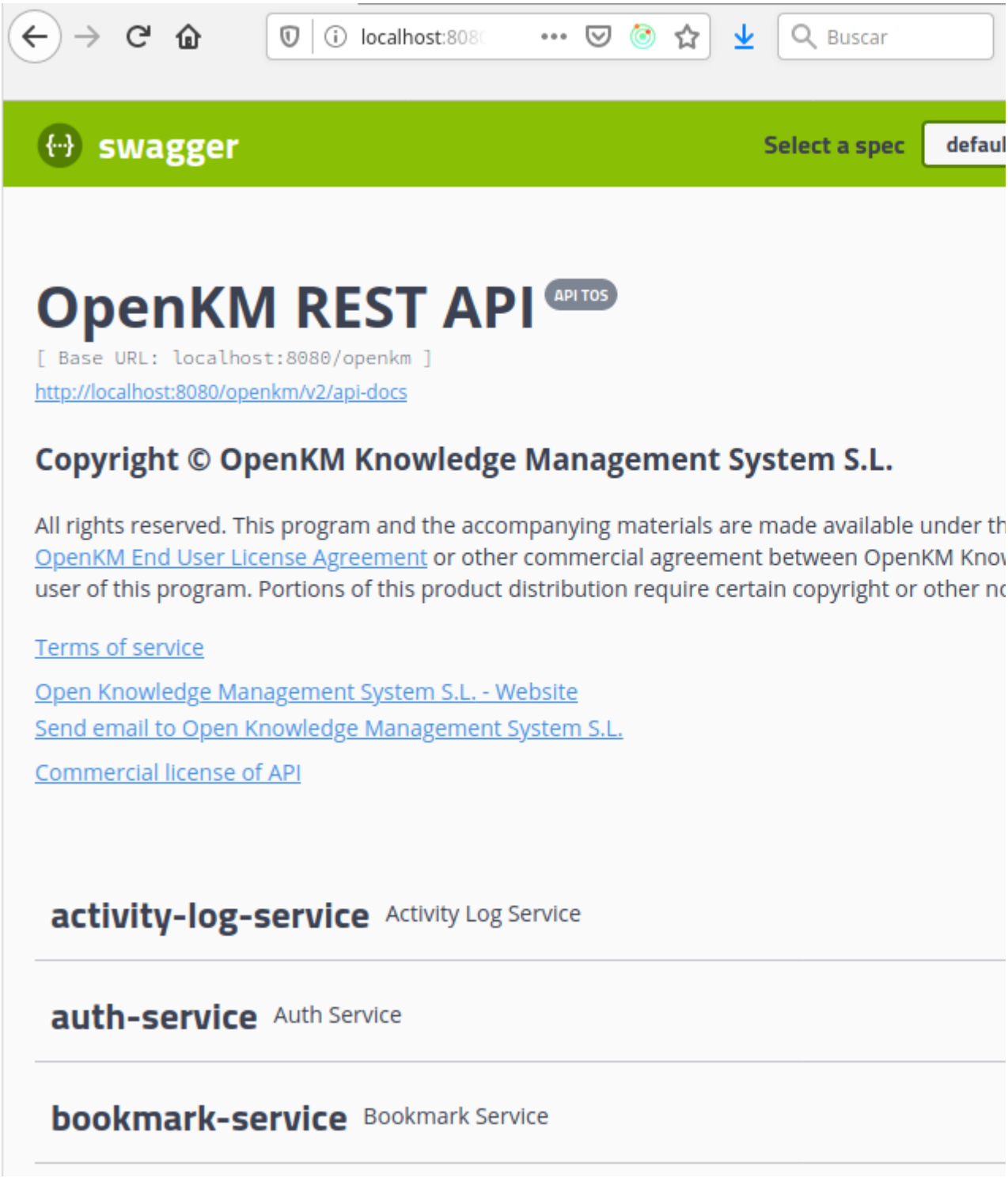
***Swagger** is an open-source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful Web services. While most users identify Swagger by the Swagger UI tool, the Swagger toolset includes support for automated documentation, code generation, and test case generation.*

How to access Swagger

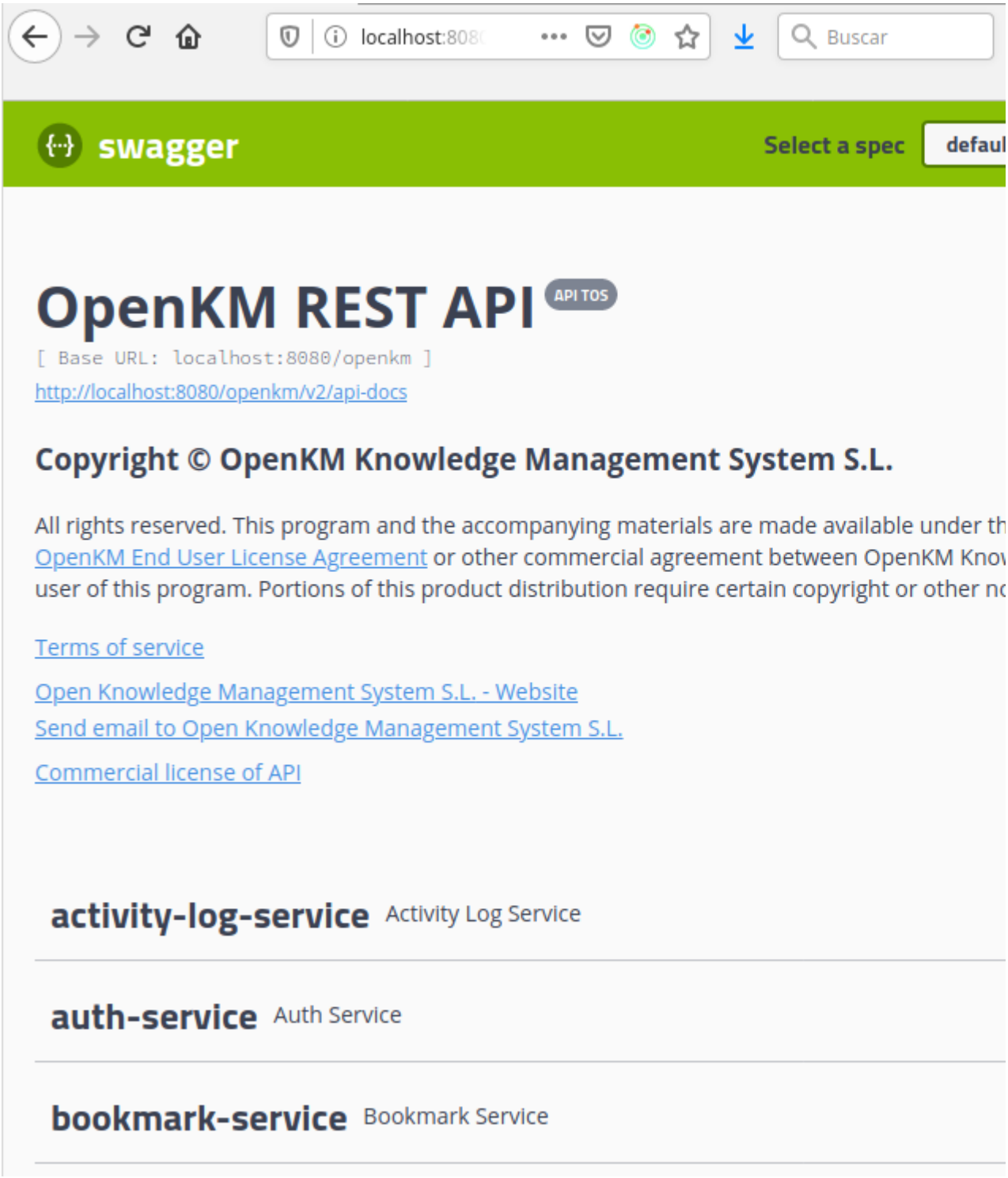
*In the **sample below, we use our OpenKM demo URL**. You **should use your OpenKM URL**.*

For accessing Swagger documentation, click in the URL http://localhost:8080/openkm/swagger-ui.html ;it will be shown and screen like shown below asking for authorized user credentials:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Then will be shown the Swagger tool:



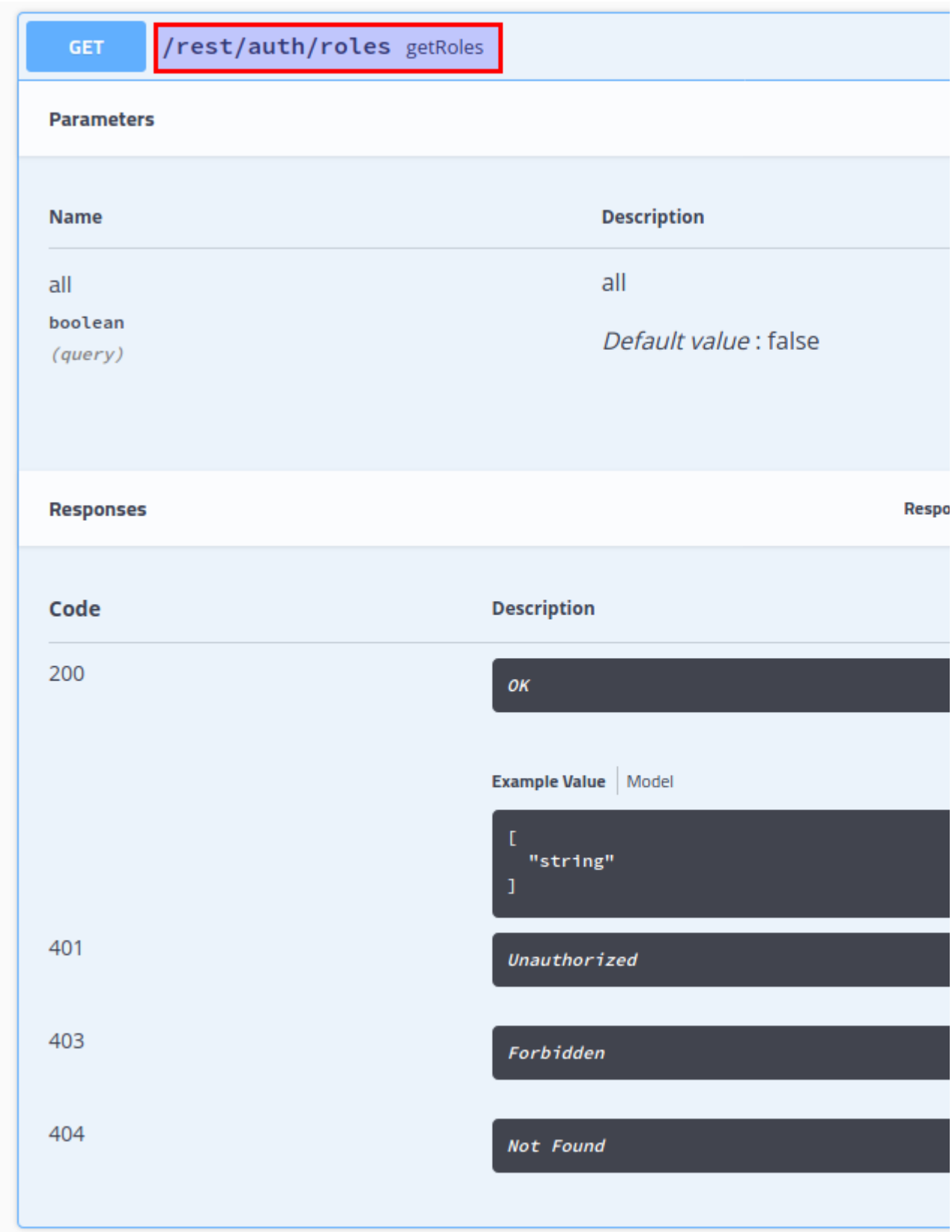
Playing with Swagger

*Let's do a **quick test** of the tool **checking the getRoles** method **from auth-service**.*

- ***Open the auth-service***



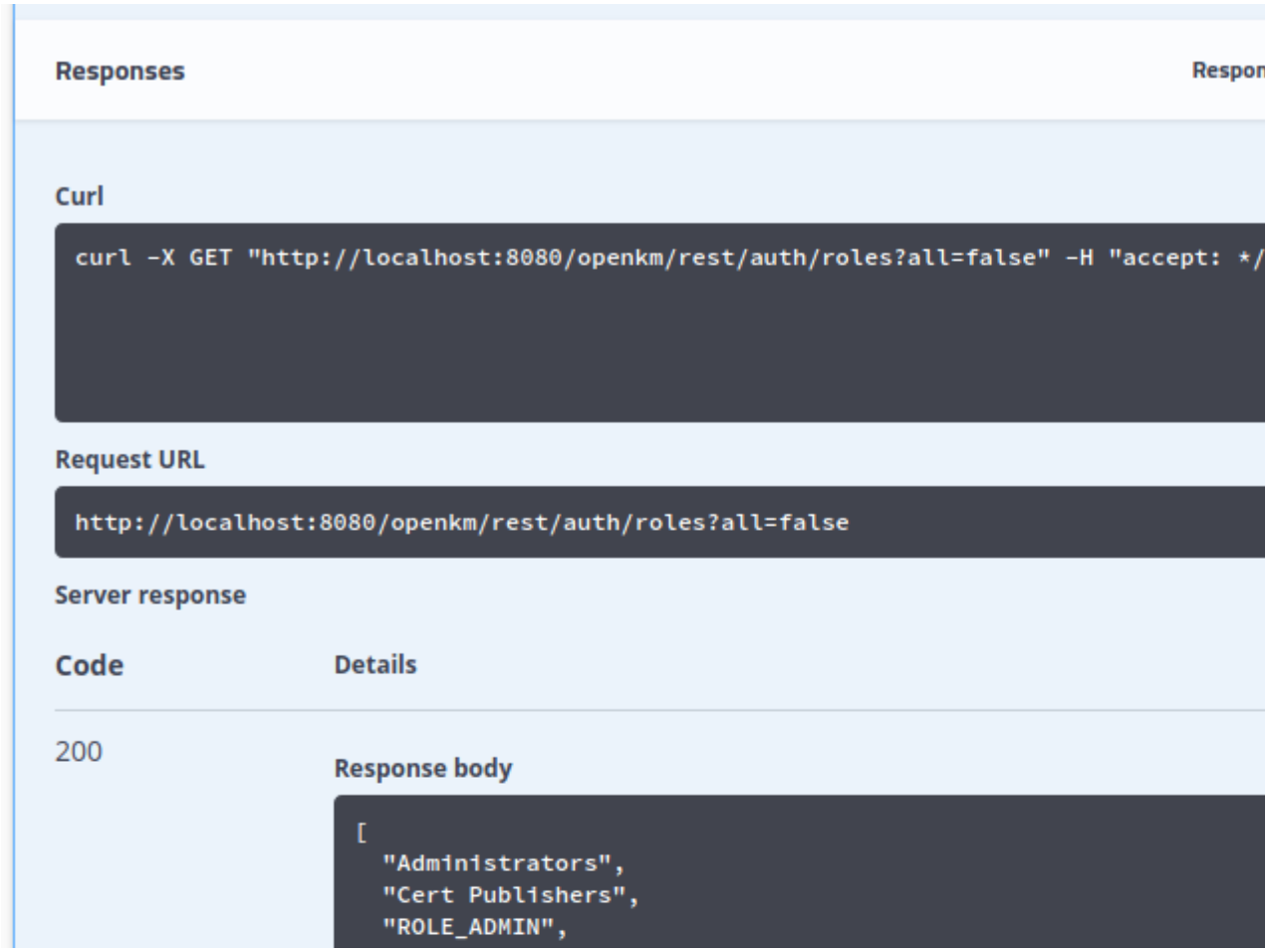
- ***Open the "roles"*** *method and* ***click on the button "Try it out"*** *:*



- ***Click on the "Execute" button;***



- *Finally will be shown the results*



```
  "ROLE_ADMINISTRACION",
  "ROLE_LICENSE",
  "ROLE_MARKETING",
  "ROLE_PROYECTO",
  "ROLE_SECURITY_RECURSIVE",
  "ROLE_TASK_MANAGER_ADMIN",
  "ROLE_TECNICO",
  "ROLE_USER"
]
```


Response headers


```
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json;charset=UTF-8
date: Wed, 12 Feb 2020 13:04:33 GMT
expires: 0
pragma: no-cache
transfer-encoding: chunked
x-application-context: application:none
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block
```


Responses

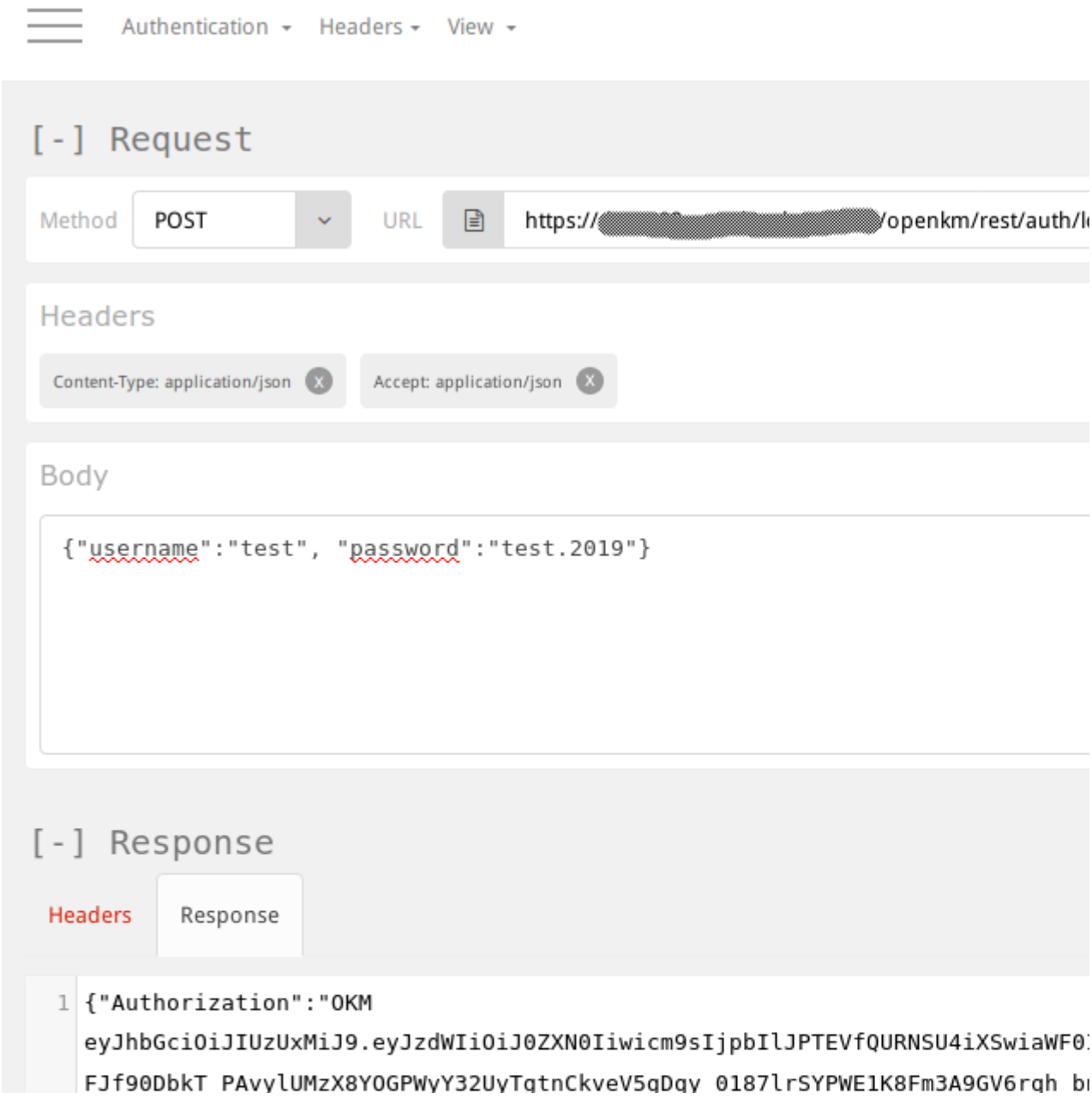
Code	Description
200	*OK* **Example Value** | Model `[ "string" ]`
401	*Unauthorized*
403	*Forbidden*
404	*Not Found*

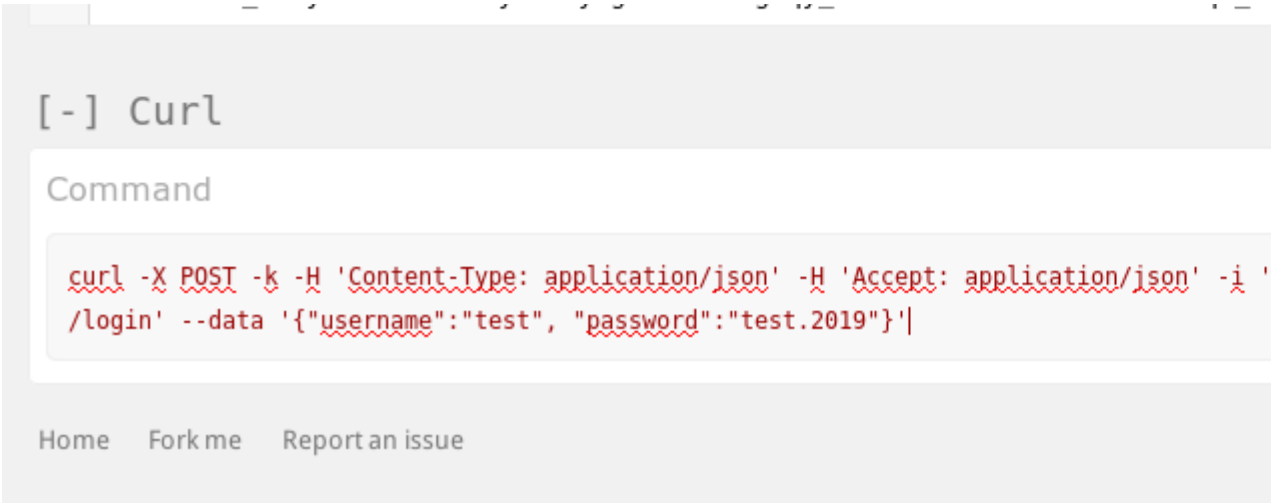
RestClient sample

RESTClient addon for Mozilla is a debugger for RESTful web services.

Login

- *First, log in to get the authorization token used in the other calls.*
- *The **username** and **password** are **sent in the body**.*
- *The **content type** is set to "**application/json**" in the header.*
- *The **accept** is set to "**application/json**" in the header.*

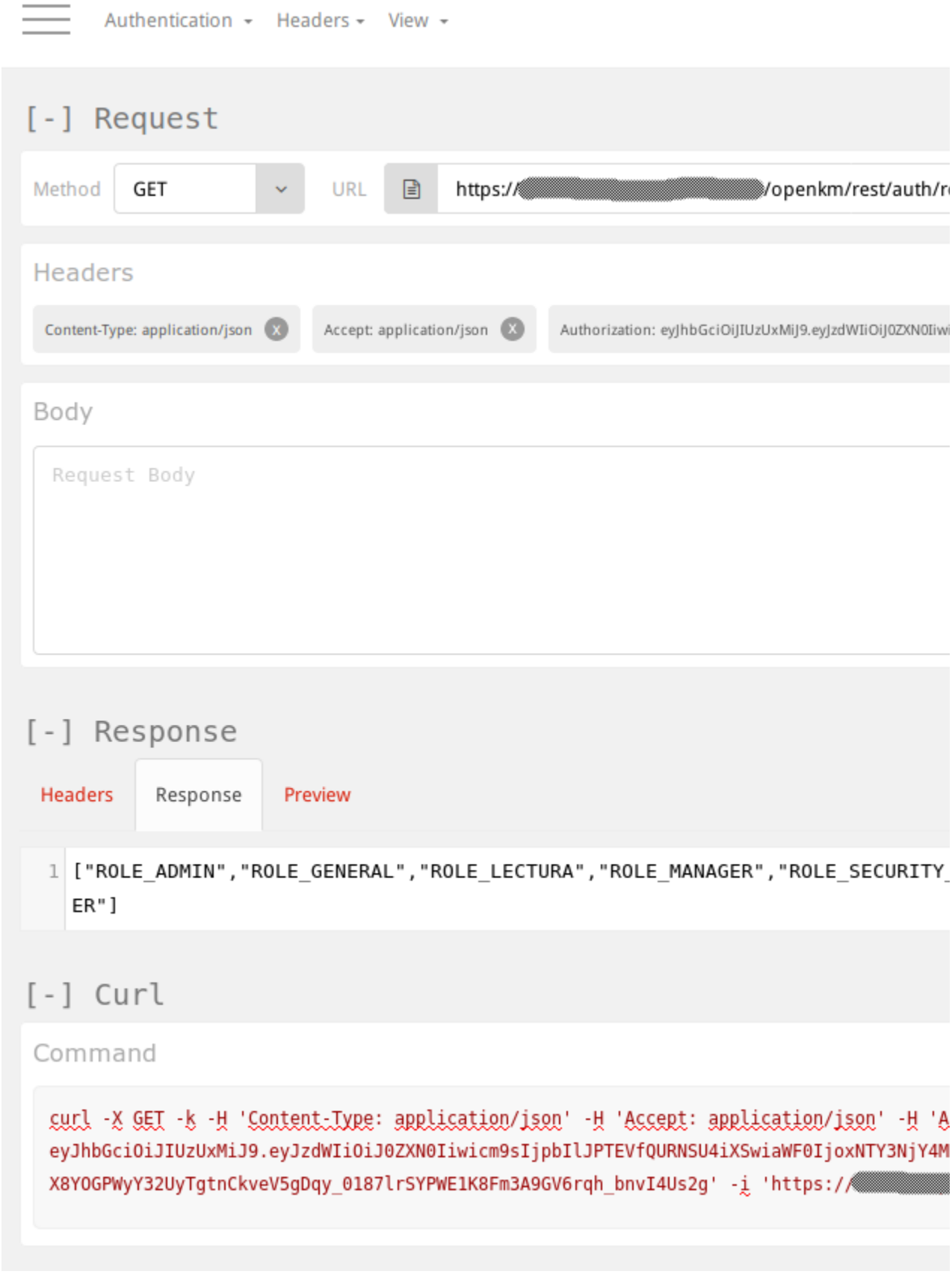




Get roles

- *The **authorization** token is set in the header.*
- *The **content-type** is set to "**application**/json" in the header.*
- *The **accept** is set to "**application**/json" in the header.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

PHP Samples

Download sdk4php

The sdk4php sample below includes the most common methods used by users when integrating their PHP applications.

> *The OpenKM has more than 400 methods in the API, the samples below cover the most common methods used when integrating PHP applications with OpenKM.*
>
> *If you have issues trying to use methods not covered in this sample, talk with the OpenKM support service.*

Professional	PHP version	Status	Download
Version 8.1.4 and upper.	7.x	Active	sdk4php-3.0.2.zip

Preview sample

The sample below describes how to integrate OpenKM preview feature in your PHP application.

Professional	PHP version	Status	Download
Version 8.1.4 and upper.	7.x	Active	okmPreview-3.0.1.zip

Login

First, you must create the webservice object:

```
$this->ws = OKMWebServicesFactory::build(self::HOST);
```

*Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:*

```
$this->ws->login(self::USER, self::PASSWORD);
```

> *Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method*

*At this point you can use all the Folder methods from "**folder**" class as is shown below:*

```
$folder = $this->ws->folder->getFolderProperties("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5");
```

```
<?php
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $folder = $this->ws->folder->getFolderProperties("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5");
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$$test->test();
?>
```

Document samples

createDocument

Description:

Method	*Return values*	*Description*
createDocument($uuid, $name, $content)	***Document***	*Creates a new document and returns as a result an object Document with the properties of the created document.*

Parameters:

$$uuid *string type is the destination uuid of the Folder or Record*

$name *string type is the name document.*

$content *string type is recommend using file_get_contents — Reads entire file into a string*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http:///localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $fileName = dirname(__FILE__) . '/files/test.pdf';
            $document = $this->ws->document->createDocument('f93ac8df-e83b-487f-9a14-0af9e910a818', 'test.pdf', fi
            var_dump($document);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

deleteDocument

Description:

Method	*Return values*	*Description*
deleteDocument($uuid)	***void***	*Deletes a document.*

Parameters:

$uuid *string type is the uuid of the Document*

> *When a document is deleted, it is automatically moved to /okm:trash/userId folder.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http:///localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->document->deleteDocument('5b37c508-b2be-4429-ba5e-acc96089c6a9');
            echo 'deleted';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getDocumentProperties

Description:

Method	*Return values*	*Description*
getDocumentProperties($uuid)	***Document***	*Returns the document properties.*
Parameters: ***$uuid*** *string type is the uuid of the Document*		

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
```

```
class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $document = $this->ws->document->getDocumentProperties('7e12a9ff-95b2-4945-bf5f-1bb709d89c14');
            var_dump($document);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getContent

Description:

Method	*Return values*	*Description*
getContent($uuid)	***string***	*Retrieves a document content - binary data - of the actual document version.*

Parameters:

$uuid *string type is the uuid of the Document*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;
```

```
    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        $method = 1;
        $this->ws->login(self::USER, self::PASSWORD);
        $content = $this->ws->document->getContent('b02877cc-bc71-4be9-9bfb-2e50b1732fd3');
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/logo_download.png', 'w+');
                fwrite($file, $content);
                fclose($file);
                echo 'download correct';
                break;
            case 2:
                ob_clean();
                $document = $this->ws->document->getDocumentProperties('b02877cc-bc71-4be9-9bfb-2e50b1732fd3');
                $name = PathUtils::getName($document->getPath());
                header("Cache-Control:  maxage=1");
                header("Pragma: public");
                header("Content-Type: " . $document->getMimeType());
                header('Content-Disposition: attachment; filename="' . $name . '"');
                header("Content-Length: " . strlen($content));
                echo $content;
                break;
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getDocumentChildren

Description:

Method	*Return values*	*Description*
getDocumentChildren($uuid)	***array***	*Returns a list of all documents which their parent is fldId.*

Parameters:

$fldId *string type is the uuid or path of the Document or a record node.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
```

```
class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $documents = $this->ws->document->getDocumentChildren('f93ac8df-e83b-487f-9a14-0af9e910a818');
            foreach ($documents as $document) {
                var_dump($document);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

moveDocument

Description:

Method	*Return values*	*Description*
moveDocument($uuid, $dstId)	***void***	*Move a document into some folder or record.*

Parameters:

$uuid *string type is the uuid of the Document*

$dstId *string type is the uuid of the Folder or Record*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {
```

```
    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->document->moveDocument('b02877cc-bc71-4be9-9bfb-2e50b1732fd3', '55102c84-e4f5-4cbc-a3
            echo 'moveDocument';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

copyDocument

Description:

Method	*Return values*	*Description*
copyDocument($uuid, $dstId, $newName)	***void***	*Copies a document to a folder or record.*

Parameters:

$uuid *string type is the uuid of the Document*

$dstId *string type is the uuid of the Folder or Record*

$newName *string type is the new name for the Document*

When parameter newName value is null, document will preservate the same name.



Only the binary data and the security grants are copied to destionation, the metadata, keywords, etc. of the document are not copied.

*See "**extendedDocumentCopy**" method for this feature.*

Example:

```
<?php
```

```
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $document = $this->ws->document->copyDocument('b02877cc-bc71-4be9-9bfb-2e50b1732fd3', '55102c84-
            var_dump($document);
            echo 'copyDocument';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

Folder sample

createFolder

Description:

Method	*Return values*	*Description*
createFolder($uuid, $name)	***Folder***	*Creates a new folder and return it as a result of an object Folder.*

Parameters:

$uuid *string type is the destination uuid of the Folder or Record*

Example:

```
<?php

include '../src/openkm/OpenKM.php';
```

```
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $folder = $this->ws->folder->createFolder('f93ac8df-e83b-487f-9a14-0af9e910a818', 'test');
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getFolderProperties

Description:

Method	*Return values*	*Description*
getFolderProperties($uuid)	***Folder***	*Returns the folder properties.*

Parameters:

$uuid *string type is the uuid of the Folder*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    // const HOST = "http://localhost:8080/openkm/";
    const HOST = "http://192.168.1.11:8888/openkm/";
```

```
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $folder = $this->ws->folder->getFolderProperties("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5");
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

deleteFolder

Description:

Method	*Return values*	*Description*
deleteFolder($uuid)	***void***	*Deletes a folder.*
Parameters: ***$uuid*** *string type is the uuid of the Folder*		

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    // const HOST = "http://localhost:8080/openkm/";
    const HOST = "http://192.168.1.11:8888/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
```

```
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->folder->deleteFolder("56c4d33b-daad-4389-bfdc-ee8dd5273899");
            echo 'delete folder';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

renameFolder

Description:

Method	*Return values*	*Description*
renameFolder($uuid, $newName)	***void***	*Renames a folder.*

Parameters:

$uuid *string type is the uuid of the Folder*

$newName *string type is the new name for the Folder*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
```

```
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            // Exists folder /okm:root/sdk/test
            $folder = $this->ws->folder->renameFolder("0b89e4ca-0019-4837-82a4-05bc05d83353", "renamedFolder");
            // Folder has renamed to /okm:root/sdk/renamedFolder
            echo 'rename Folder';
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

moveFolder

Description:

Method	*Return values*	*Description*
moveFolder($uuid, $dstId)	***void***	*Moves a folder into other folder or record.*

Parameters:

$uuid *string type is the uuid of the Folder*

$dstId *string type is the uuid of the Folder or record*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
```

```
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            // Exists folder /okm:root/sdk/test
            $this->ws->folder->moveFolder("0b89e4ca-0019-4837-82a4-05bc05d83353", "55102c84-e4f5-4cbc-a3d1-cf
            // Folder has moved to /okm:root/move/test
            echo 'move Folder';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getFolderChildren

Description:

Method	*Return values*	*Description*
getFolderChildren($uuid)	***array***	*Returns an array of all folders which their parent is fldId.*

Parameters:

$uuid *string type is the uuid of the Folder or Record node*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $folders = $this->ws->folder->getFolderChildren('f93ac8df-e83b-487f-9a14-0af9e910a818');
            foreach ($folders as $folder) {
                var_dump($folder);
```

```
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

copyFolder

Description:

Method	*Return values*	*Description*
copyFolder($uuid, $dstId, $newName)	***void***	*Copies a folder into other folder or record.*

Parameters:

$uuid *string type is the uuid of the Folder*

$dstId *string type is the uuid of the Folder or Record*

$newName *string type is the new name for the Folder. Value is null, folder will preservates the same name.*



Only the security grants are copied to the destination, the metadata, keywords, etc. of the folder are not copied.

*See "**extendedFolderCopy**" method for this feature.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }
```

```
    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->folder->copyFolder('4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5', '55102c84-e4f5-4cbc-a3d1-cf189
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

extendedFolderCopy

Description:

Method	*Return values*	*Description*
extendedFolderCopy($uuid, $dstId, $categories, $keywords, $propertyGroups, $notes, $security)	***void***	*Copies a folder with associated data into other folder or record.*

Parameters:

$uuid *string type is the uuid of the Folder*

$dstId *string type is the uuid of the Folder or Record*

$categories *boolean type*

$keywords *boolean type*

$propertyGroups *boolean type*

$notes *boolean type*

$security *boolean type*

By default only the binary data and the security grants, the metadata, keywords, etc. of the folder are not copied.

Additionally:

- *When category parameter is true the original values of the categories will be copied.*
- *When keywords parameter is true the original values of the keywords will be copied.*
- *When propertyGroups parameter is true the original values of the metadata groups will be copied.*

- *When notes parameter is true the original values of the notes will be copied.*
- *When security parameter is true the original values of the security will be copied.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $folder = $this->ws->folder->extendedFolderCopy('4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5', '55102c84-e4f
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getContentInfo

Description:

Method	*Return values*	*Description*
getContentInfo($uuid)	***ContentInfo***	*Return and object ContentInfo with information about folder.*

Parameters:

$uuid *string type is the uuid or path of the Folder.*

The ContentInfo object retrives information about:

- *The number of folders into.*
- *The number of documents into.*
- *The number of records into.*
- *The number of mails into.*
- *The size in bytes of all objects into the folder.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            var_dump($this->ws->folder->getContentInfo('4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5'));
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

purgeFolder

Description:

Method	*Return values*	*Description*
purgeFolder($uuid)	***void***	*Folder is definitely removed from repository.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Parameters:

$uuid *string type is the uuid or path of the Folder.*

Usually you will purge folders into /okm:trash/userId - the personal trash user locations - but is possible to directly purge any folder from the whole repository.



When a folder is purged only will be able to be restored from a previously repository backup. The purge action remove the folder definitely from the repository.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->folder->purgeFolder('d1faccb3-9e64-4d2d-865f-1c057e3a279b');
            echo 'purge folder';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

setStyle

Description:

Method	*Return values*	*Description*
setStyle($uuid, $styleId)	***void***	*Set the folder style.*

Parameters:

$uuid *string type is the uuid or path of the Folder.*

$styleId *long type is the id of the style.*



More information at: Folder style.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http:///localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->folder->setStyle('0b89e4ca-0019-4837-82a4-05bc05d83353', 1);
            echo 'set style';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

createMissingFolders

Description:

Method	*Return values*	*Description*
createMissingFolders($fldPath)	***void***	*Create missing folders.*

Parameters:

$fldPath *string type is the path of the Folder.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->folder->createMissingFolders("/okm:root/sdk/missingfld2/missingfld3");
            var_dump('create missing folders');
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

createFolderFromTemplate

Description:

Method	*Return values*	*Description*
createFolderFromTemplate($uuid, $dstPath, $language, $categories, $keywords, $propertyGroups, $notes, $properties)	***Folder***	*Creates a new folder from the template and returns an object Folder.*

*The $**uuid** parameter is the UUID value of the template file.*

*The $**dstPath** value is the folder destination path.*

*The $**language** paramater is optional.*

When the template uses metadata groups to fill in fields, then these values are mandatory and must be set into properties parameter.

Additional:

- *When category parameter is true the original values of the categories will be copied.*
- *When keywords parameter is true the original values of the keywords will be copied.*
- *When property Groups parameter is true the original values of the metadata groups will be copied.*
- *When notes parameter is true the original values of the notes will be copied.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    // const HOST = "http://localhost:8080/openkm/";
    const HOST = "http://192.168.1.11:8888/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);

            $properties = [];
            $properties["okp:consulting.name"] = "update value";
            $properties["okp:consulting.important"] = "false";

            $folder = $this->ws->folder->createFolderFromTemplate('47c70047-2924-483c-bc42-23d0cbef6b17', '/okm:r
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
```

```
}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

PropertyGroup samples

addPropertyGroup

Description:

Method	*Return values*	*Description*
addPropertyGroup($uuid, $grpName, $properties)	***void***	*Adds an empty metadata group to a node.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $properties = [];
            $properties["okp:consulting.name"] = "new value";
            $properties["okp:consulting.important"] = "true";
            $this->ws->propertyGroup->addPropertyGroup('7e12a9ff-95b2-4945-bf5f-1bb709d89c14', 'okg:consulting', $
            echo 'Add Group';
        } catch (Exception $e) {
            var_dump($e);
```

```
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

removePropertyGroup

Description:

Method	*Return values*	*Description*
removePropertyGroup($uuid, $grpName)	***void***	*Removes a metadata group of a node.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->propertyGroup->removePropertyGroup('7e12a9ff-95b2-4945-bf5f-1bb709d89c14', 'okg:consultin
            echo 'Remove Group';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getPropertyGroups

Description:

Method	*Return values*	*Description*
getPropertyGroups($uuid)	***array***	*Retrieves a list of metadata groups assigned to a node.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $propertyGroups = $this->ws->propertyGroup->getPropertyGroups('7e12a9ff-95b2-4945-bf5f-1bb709d89c1
            foreach ($propertyGroups as $propertyGroup) {
                var_dump($propertyGroup);
                echo '<br>';
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
```

```
$test = new Test();
$test->test();
?>
```

getAllPropertyGroups

Description:

Method	*Return values*	*Description*
getAllPropertyGroups()	***array***	*Retrieves a list of all metadata groups set into the application.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $propertyGroups = $this->ws->propertyGroup->getAllPropertyGroups();
            foreach ($propertyGroups as $propertyGroup) {
                var_dump($propertyGroup);
                echo '<br>';
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getPropertyGroupForm

Description:

Method	Return values	Description
getPropertyGroupForm($grpName)	**array**	*Retrieves a list of all metadata group elements definition.*

Parameters:

$grpName *string type is the grpName should be a valid Metadata group name.*

The method is usually used to display empty form elements for creating new metadata values.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetPropertyGroupForm() {
        try {
            $formElements = $this->ws->getPropertyGroupForm('okg:consulting');
            foreach ($formElements as $formElement) {
                var_dump($formElement);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testGetPropertyGroupPropertiesSimple();
?>
```

getPropertyGroupFormByNode

Description:

Method	Return values	Description

getPropertyGroupFormByNode($uuid, $grpName)	***array***	*Retrieves a list of all metadata group elements definition by node*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

The method is usually used to display empty form elements for creating new metadata values.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $formElements = $this->ws->propertyGroup->getPropertyGroupFormByNode('7e12a9ff-95b2-4945-bf5f-1bb
            foreach ($formElements as $formElement) {
                var_dump($formElement);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

setPropertyGroupProperties

Description:

Method	*Return values*	*Description*
setPropertyGroupProperties($uuid, $grpName, $properties)	***void***	*Changes the metadata group values of a node.*

Parameters:

$uuid *string type is the uuid or path of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

$properties *array type is an array*

*Before changing metadata, you **should have the group added in the node** (see addGroup method) otherwise will be raised and error.*

Is not mandatory set into properties parameter all fields values, is enough with the fields you wish to change its values.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>
```

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
```

```
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $properties = [];
            $properties["okp:consulting.name"] = "update value";
            $properties["okp:consulting.important"] = "false";

            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->propertyGroup->setPropertyGroupProperties('7e12a9ff-95b2-4945-bf5f-1bb709d89c14', 'okg:cor
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

hasPropertyGroup

Description:

Method	*Return values*	*Description*
hasPropertyGroup($uuid, $grpName)	***bool***	*Returns a boolean that indicate if the node has or not a metadata group.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
```

```
class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            echo 'Have metadata group: ' . $this->ws->propertyGroup->hasPropertyGroup('7e12a9ff-95b2-4945-bf5f-1bl
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getRegisteredPropertyGroupDefinition

Description:

Method	*Return values*	*Description*
getRegisteredPropertyGroupDefinition()	***string***	*Return the XML Metada groups definition.*

> *The method only can be executed by super user grants (ROLE_ADMIN member user).*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
```

```
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            var_dump($this->ws->propertyGroup->getRegisteredPropertyGroupDefinition());
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getPropertyGroupSuggestions

Description:

Method	*Return values*	*Description*
getPropertyGroupSuggestions($uuid, $grpName, $propName)	***array***	*Retrieves a list of a suggested metadata field values.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

$propName *string type is the propName parameter should be a Metadata Select field type.*



The propName parameter should be a Metadata Select field type.



More information at Creating your own Suggestion Analyzer and Metadata Select field.



The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
```

```
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $suggestions = $this->ws->propertyGroup->getPropertyGroupSuggestions('055b5206-35d0-4351-ba92-c30
            foreach ($suggestions as $suggestion) {
                var_dump($suggestion);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

registerPropertyGroupDefinition

Description:

Method	*Return values*	*Description*
registerPropertyGroupDefinition($content, $name)	***void***	*Sets the XML Metada groups definition into the repository.*

Parameters:

$content *string type is recommend using file_get_contents — Reads entire file into a string*

$name *string type is the name propertygroup*



The method only can be executed by super user grants (ROLE_ADMIN member user).

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $fileName = dirname(__FILE__) . '/propertygroup/propertyGroups.xml';
            $this->ws->propertyGroup->registerPropertyGroupDefinition(file_get_contents($fileName), 'sdk4php');
            var_dump('Register Definition correct');
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getPropertyGroupProperties

Description:

Method	*Return values*	*Description*
getPropertyGroupProperties($uuid, $grpName)	***array***	*Retrieves a list of all metadata group elements and its values of a node.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

The method is usually used to display form elements with its values to be shown or changed by user.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $properties = $this->ws->propertyGroup->getPropertyGroupProperties('7e12a9ff-95b2-4945-bf5f-1bb709d8
            foreach ($properties as $key => $value) {
                echo 'key: ' . $key . ' value: ' . $value;
                echo '<br>';
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}
```

```
$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getPropertyGroupPropertiesByVersion

Description:

Method	*Return values*	*Description*
getPropertyGroupPropertiesByVersion($uuid, $grpName, $versionName)	***array***	*Retrieves a list of all metadata group elements and its values of a node.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type is the grpName should be a valid Metadata group name.*

$versionName *string type is the version of the document, folder, mail or record.*

The method is usually used to display form elements with its values to be shown or changed by user.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http:///localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $properties = $this->ws->propertyGroup->getPropertyGroupPropertiesByVersion('7e12a9ff-95b2-4945-bf5f-
            foreach ($properties as $key => $value) {
```

```
            echo 'key: ' . $key . ' value: ' . $value;
            echo '<br>';
        }
      } catch (Exception $e) {
        var_dump($e);
      }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

Search samples

find

Description:

Method	Return values	Description
find(QueryParams $queryParams, $propertiesPlugin)	**array**	Returns a list of QueryResults filtered by the values of the queryParams parameter.

Parameters:

$queryParams *QueryParams type*

$propertiesPlugin *string type, must be the canonical class name of the class which implements the NodeProperties interface.*

> *Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";
```

```
    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $queryParams = new QueryParams();
            $queryParams->setDomain(QueryParams::DOCUMENT + QueryParams::FOLDER);
            $queryParams->setFolder("f93ac8df-e83b-487f-9a14-0af9e910a818");
            $queryParams->setFolderRecursive(true);
            $queryParams->setLastModifiedFrom(20150628000000);
            $queryParams->setLastModifiedTo(date('Ymdhis'));
            $queryResults = $this->ws->search->find($queryParams, null);
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

findPaginated

Description:

Method	*Return values*	*Description*
findPaginated(QueryParams $queryParams, $offset, $limit, $propertiesPlugin)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

Parameters:

$queryParams *QueryParams type*

$offset *int type*

$limit *int type*

$propertiesPlugin *string type, must be the canonical class name of the class which implements the NodeProperties interface.*



The parameter "limit" and "offset" allow you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query have 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $queryParams = new QueryParams();
            $queryParams->setDomain(QueryParams::DOCUMENT + QueryParams::FOLDER);
            $queryParams->setFolder("f93ac8df-e83b-487f-9a14-0af9e910a818");
            $queryParams->setFolderRecursive(true);
            $queryParams->setLastModifiedFrom(20150628000000);
            $queryParams->setLastModifiedTo(date('Ymdhis'));
            $resultSet = $this->ws->search->findPaginated($queryParams, 0, 10, null);
            echo "Total results:" . $resultSet->getTotal();
            foreach ($resultSet->getResults() as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
```

```
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Integrate preview with other web applications

You can integrate the OpenKM web preview module into other web applications quickly.

Advantatges:

- *Quickly integration into an iframe in your own web application.*
- *All supported OpenKM MIME types from preview modules.*
- *Can be used without being logged into OpenKM.*
- *Can preview documents which are not into OpenKM.*

*If your OpenKM is available from http://your_domain/openkm then, the preview OpenKM url module should be your **http://your_domain/openkm/Preview**?docUrl=http://localhost/test.pdf*

*The **supported MIME types depends on your OpenKM configuration and OpenKM conversion service**.*

This components previes most common file types:

- ***PDF, Microsoft Office** and **LibreOffice***
- ***Images like JPG, PNG, TIFF, MP4, MP3***
- ***Source code files like Java, PHP, HTML, SQL***

Preview parameters:

Parameter	*Type*	*Description*
docUrl	***String***	*The full url of a document.* *We recommend using it for all the file types.*
swfUrl	***String***	*The full url of a .swf file.*
dxfUrl	***String***	*The full url of a .dxf file.*
dwfUrl	***String***	*The full url of a .dwf file.*
htmUrl	***String***	*The full url of a .html file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

codUrl	***String***	*The full url of a file with code into, for example .java, .php, etc. files.*
mpgUrl	***String***	*The full url of a .mp4 and .mp3 file.*
pdfUrl	***String***	*The full url of a pdf document. Required by PDF.js previewer.*
file	***String***	*The full url of a pdf document. Required by PDF.js previewer.*
previewEngine	***String***	*The type of preview to be used by server. By default is PDF.js but with this parameter you can force to Flexpaper* *OpenKM support Flexpaper and PDF.js as pdf previewers*

Code sample

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Document Preview</title>
</head>

<body>
   <div align="left">
      Left panel
   </div>
   <div align="right">
      Preview panel<br />
      <iframe src="http://localhost:8080/openkm/preview?docUrl=http://localhost/test.pdf"
         style="width:800px; height:600px" frameborder="0" marginwidth="0" marginheight="0" />
   </div>
</body>

</html>
```

*If you **create your own downloading service** take in mind to set into the response the "Access-Control-Allow-Origin" in the header.*

The java code section below can provide you some ideas about it:

```
/**
 * Prepare to send the file.
 */
public static void prepareSendFile(HttpServletRequest request, HttpServletResponse response, String fi
                String mimeType, boolean inline) throws UnsupportedEncodingException {
```

```
    String userAgent = getHeader(request, "user-agent").toLowerCase();
    fileName = decodeEntities(fileName);

    // Disable browser cache
    response.setHeader("Expires", "Sat, 6 May 1971 12:00:00 GMT");
    response.setHeader("Cache-Control", "must-revalidate");
    response.addHeader("Cache-Control", "no-store, no-cache, must-revalidate");
    response.addHeader("Cache-Control", "post-check=0, pre-check=0");
    response.setHeader("Pragma", "no-cache"); */

    // Allow to use files in different domains
    response.setHeader("Access-Control-Allow-Origin", "*");

    // Set MIME type
    response.setContentType(mimeType);

    if (userAgent.contains("msie") || userAgent.contains("trident")) {
        log.debug("Agent: Explorer ({})", request.getServerPort());
        fileName = URLEncoder.encode(fileName, "UTF-8").replaceAll("\\+", " ");

        if (request.getServerPort() == 443) {
            log.debug("HTTPS detected! Apply IE workaround...");
            response.setHeader("Cache-Control", "max-age=1");
            response.setHeader("Pragma", "public");
        }
    } else if (userAgent.contains("iphone") || userAgent.contains("ipad")) {
        log.debug("Agent: iPhone - iPad");
        // Do nothing
    } else if (userAgent.contains("android")) {
        log.debug("Agent: Android");
        fileName = URLEncoder.encode(fileName, "UTF-8").replaceAll("\\+", " ");
    } else if (userAgent.contains("mozilla")) {
        log.debug("Agent: Mozilla");
        fileName = MimeUtility.encodeText(fileName, "UTF-8", "B");
    } else {
        log.debug("Agent: Unknown");
    }

    if (inline) {
        response.setHeader("Content-disposition", "inline; filename=\"" + fileName + "\"");
    } else {
        response.setHeader("Content-disposition", "attachment; filename=\"" + fileName + "\"");
    }
}

/**
 * Get HTTP header.
 */
private static String getHeader(HttpServletRequest request, String name) {
    String value = request.getHeader(name);

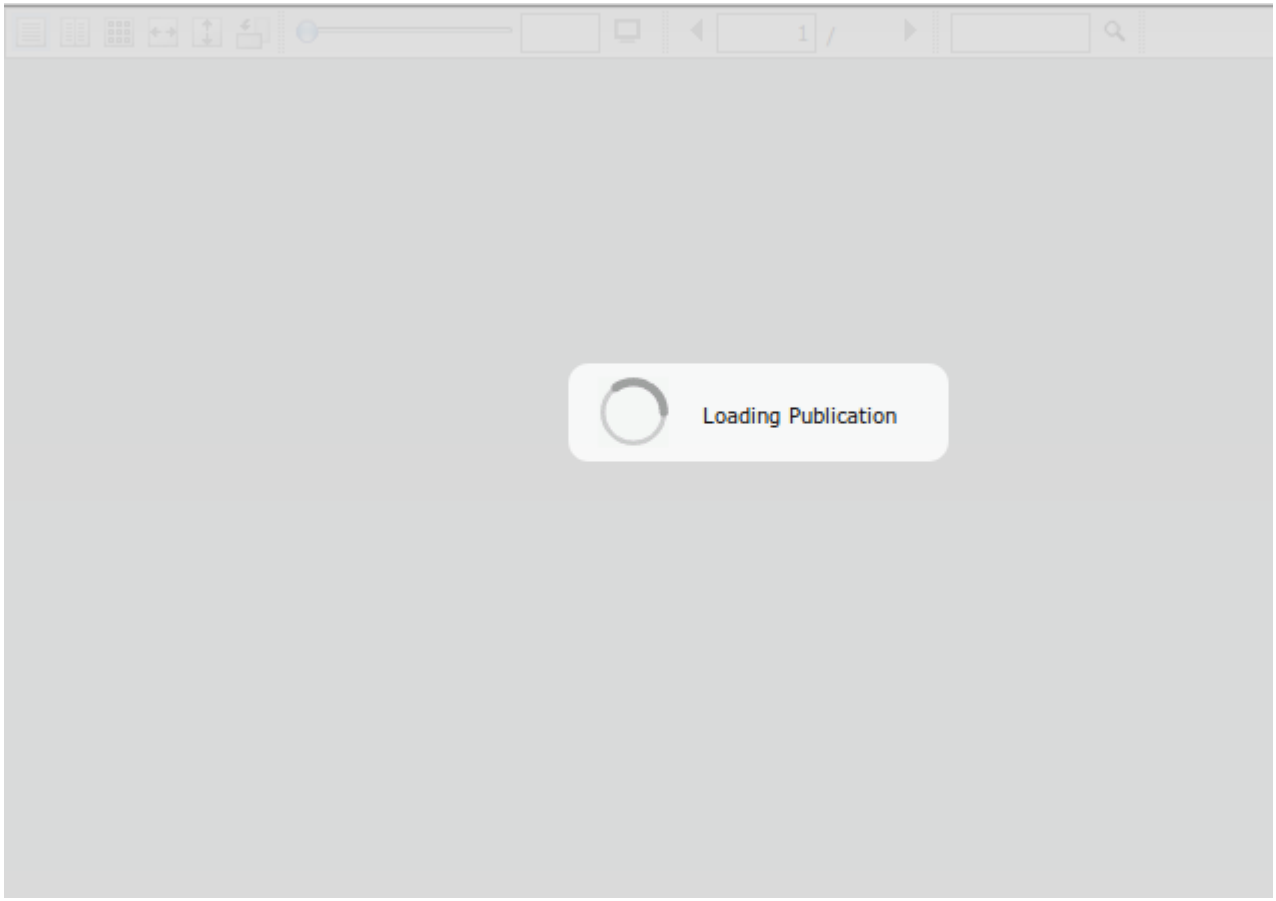
    if (value != null) {
        return value;
    } else {
        return EMPTY_STRING;
    }
}

/**
 * decodeEntities
 */
private static String decodeEntities(String fileName) {
    String tmp = fileName.replaceAll("&amp;", "&");
    tmp = tmp.replaceAll("&lt;", "<");
    tmp = tmp.replaceAll("&gt;", ">");
    tmp = tmp.replaceAll("&#34;", "\"");
    tmp = tmp.replaceAll("&#39;", "'");
```

```
    return tmp;
}
```

Troubleshooting

There are several reasons why the document is not able for previewing and you get a screen like this:



Check if you see some error in your OpenKM log.

*Take a look in the javascript console, if it's shown an error like "**missing header CORS 'Access-Control-Allow-Origin')**" then you must change your web server configuration to allowing it.*

*At http://enable-cors.org/server_apache.html you have information about how to enable **Access-Control-Allow-Origin** in the Apache web server.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Samples

Sample	*Description*
Scripting samples	*A collection of samples that use Scripting feature.*
Crontab samples	*A collection of crontab samples.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Scripting samples

Script	*Description*
Script - Basic samples	• *Updates the repository statistics.* • *Creates folders and sets property group.* • *Shows the number of documents, folders and size from a given path.* • *Gets path by UUID.* • *Shows all documents used in every running workflow.*
Script - Compact documents with size equals to 0	*Finds all documents with size=0 and compact history of versions to latest.*
Script - Convert TIF to PDF	*Transform all the TIF files in the repository to PDF.*
Script - Count all children subfolder	*Counts all children folders - and sub folders to, recursion - into first level folder (okm:root children folders).*
Script - Exporter	*The repository exports execution.*
Script - Export repository folder to server as a ZIP file	*Exports the folder contents to the server application as a ZIP file.*
Script - Folders deep	*Finds the folders with some deep.*
Script - Import data from CSV file	*Imports metadata group values from csv file.*
Script - Purge all folders into some parent	*Purges all folders into some parent, without previously moving files to trash.*
Script - Purge all users trashes	*Example of how to clean all users trashes.*
Script - Read PDF form fields	*How to read PDF form fields.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Recursive character renaming	*Renames a character in all folders or documents from some initial path.*
Script - Recursive repository traversal	*Navigates across all repository nodes recursivelly.*
Script - Recursive change security	*Change the security.*
Script - Security change at low level	*Changes the security at database level.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Basic samples

Update repository statistics

Refresh the Statistics.

```
import com.openkm.core.RepositoryInfo;
import com.openkm.util.ContextWrapper;

RepositoryInfo repositoryInfo = ContextWrapper.getContext().getBean(RepositoryInfo.class);
repositoryInfo.run();
```

Create some folders and set property group

```
import java.util.HashMap;
import java.util.Map;

import com.openkm.api.OKMFolder;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class);

for (int i = 0; i < 10; i++) {
    String path = "/okm:root/fld_" + i;
    okmFolder.createSimple(null, path);

    Map<String, String> properties = new HashMap<>();
    properties.put("okp:consulting.name", "new name");
    properties.put("okp:consulting.comment", "new comment");
    okmPropertyGroup.addGroup(null, path, "okg:consulting", properties);
}
```

Show number of documents, folders and size from a given path

```
import com.openkm.api.OKMFolder;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.ContentInfo;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FormatUtil;

OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
ContentInfo ci = okmFolder.getContentInfo(null, "151f3a54-f370-47d6-801a-d20faecec180");

print("Folders: " + ci.getFolders() + "<br/>");
print("Documents: " + ci.getDocuments() + "<br/>");
print("Size: " + FormatUtil.formatSize(ci.getSize()) + "<br/>");
```

Get path by UUID

```
import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
String path = okmRepository.getNodePath(null, "162735a1-661c-4555-bf53-4e9115f38c62");
print(path);
```

Show all documents used in every running workflow

```
import com.openkm.api.OKMRepository;
import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.ProcessDefinition;
import com.openkm.bean.workflow.ProcessInstance;
import com.openkm.core.Config;
import com.openkm.util.ContextWrapper;

OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);
for (ProcessDefinition procDef : okmWorkflow.findAllProcessDefinitions(null)) {
    for (ProcessInstance procIns : okmWorkflow.findProcessInstances(null, procDef.getId())) {
        if (procIns.getEnd() == null) {
            String uuid = (String) procIns.getVariables().get(Config.WORKFLOW_PROCESS_INSTANCE_VARIABLE_U
            print("Doc UUID: " + uuid + ", Process Definition: " + procDef.getId() + ", Process Instance: "
                    + procIns.getId());
        }
    }
}
```

Script - Compact documents with size equals to 0

Find all documents with size=0 and compact history of versions to latest.

```
import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

import com.openkm.api.OKMDocument;
import com.openkm.api.OKMRepository;
import com.openkm.db.bean.NodeDocumentVersion;
import com.openkm.util.ContextWrapper;

SessionFactory sessionFactory = ContextWrapper.getContext().getBean(SessionFactory.class);
OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);

String qs = "from NodeDocumentVersion ndv where ndv.size=0";
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();
Query q = session.createQuery(qs);
List docVersionList = q.list();
tx.commit();

print("Number of nodes: " + docVersionList.size() + "<br/>");

for (Object obj : docVersionList) {
    NodeDocumentVersion ndv = (NodeDocumentVersion) obj;
    okmDocument.purgeVersionHistory(null, ndv.getParent());
}

session.close();
print("done");
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Convert TIF to PDF

Transform all the TIF files in the repository to PDF.

- *There's an output log file named "ConvertTIFToPDFXXX.log" in the logs folders.*
- *Because large repositories the SQL query is executed with a database cursor.*

```
import org.springframework.web.context.*;
import com.openkm.module.db.base.*;
import com.openkm.util.*;
import com.openkm.servlet.frontend.util.ConversionUtils;
import org.hibernate.*;

WebApplicationContext cc = (WebApplicationContext) ContextWrapper.getContext();
SessionFactory sessionFactory = cc.getBean(SessionFactory.class);
ConversionUtils conversionUtils = cc.getBean(ConversionUtils.class);
StatelessSession session = sessionFactory.openStatelessSession();
String LOG_NAME = "ConvertTIFToPDF";
ScrollableResults results = null;

try {
  FileLogger.info(LOG_NAME, "### BEGIN conversion from TIF to PDF");
  String sql = "select nb.NBS_UUID from OKM_NODE_BASE nb where nb.NBS_NAME LIKE '%.tif'";
  Query query = session.createSQLQuery(sql);
  query.setReadOnly(true);
  query.setFetchSize(10);

  // Execute query
  results = query.scroll(ScrollMode.FORWARD_ONLY);

  while (results.next()) {
 String nodeUuid = results.get(0);
 try {
  String dstPath = conversionUtils.convertToPDF(nodeUuid);
  FileLogger.error(LOG_NAME, nodeUuid + " > " + dstPath);
 } catch (Exception e) {
  FileLogger.error(LOG_NAME, "Error in node " + nodeUuid + " caused by " + e.getMessage());
  FileLogger.error(LOG_NAME + "-Error", "Error in node " + nodeUuid + " caused by \n" + StackTraceUtils.toString(e
 }
  }

  FileLogger.info(LOG_NAME, "### END conversion from TIF to PDF");
} catch (Exception e) {
  print(StackTraceUtils.toString(e));
  FileLogger.error(LOG_NAME, StackTraceUtils.toString(e));
} finally {
  results.close();
  session.close();
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Count all children subfolder

Count all children folders - and subfolders into, recursion - into first level folder (okm:root children folders).

```
import java.util.ArrayList;
import java.util.List;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

class Util {

    public int childCount(String uuid) {
        String qsUUIDList = "select uuid from NodeFolder nf where nf.parent=:parent";
        Session session = null;
        Transaction tx = null;
        int value = 0;
        List uuidList = new ArrayList();

        try {
            SessionFactory sessionFactory = ContextWrapper.getContext().getBean(SessionFactory.class);

            session = sessionFactory.openSession();
            tx = session.beginTransaction();
            Query q = session.createQuery(qsUUIDList).setCacheable(true);
            q.setString("parent", uuid);
            uuidList = q.list();
            tx.commit();
            value = uuidList.size();
            print(value);

        } catch (HibernateException e) {
            tx.rollback();
        } finally {
            session.close();
        }

        for (Object childUuid : uuidList) {
            value = value + childCount((String) childUuid);
        }
        return value;
    }
}


OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
Util util = new Util();
// /okm:root = 151f3a54-f370-47d6-801a-d20faecec180
for (Folder fld : okmFolder.getChildren(null, "151f3a54-f370-47d6-801a-d20faecec180")) {
    print(fld.getPath() + "->" + util.childCount(fld.getUuid()) + "<br/>");
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Exporter

Repository export execution.

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

import com.openkm.api.OKMFolder;
import com.openkm.bean.ContentInfo;
import com.openkm.core.Config;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FileLogger;
import com.openkm.util.impexp.HTMLInfoDecorator;
import com.openkm.util.impexp.RepositoryExporter;

OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
RepositoryExporter repositoryExporter = ContextWrapper.getContext().getBean(RepositoryExporter.class);

// Configuration parameters
String token = DbSessionManager.getInstance().getSystemToken();
String repoPath = "/okm:root";
String fsPath = "/home/openkm/export";
boolean history = false;

// Start
FileWriter fw = null;
try {
    FileLogger.info("LOG_FILE_NAME", "Started");
    // Open outputstream
    String fileDate = new SimpleDateFormat("yyyyMMdd").format(new Date());
    String fileName = Config.LOG_DIR + File.separator + "LOG_FILE_NAME" + "_" + fileDate + ".log";
    File file = new File(fileName);
    fw = new FileWriter(file);

    // Starting the process
    ContentInfo cInfo = okmFolder.getContentInfo(token, repoPath);
    FileLogger.info("LOG_FILE_NAME",
            "Files & directories to export:" + (cInfo.getDocuments() + cInfo.getFolders()));
    File dir = new File(fsPath);
    repositoryExporter.exportFolder(token, repoPath, dir, history, fw,
            new HTMLInfoDecorator((int) cInfo.getDocuments() + (int) cInfo.getFolders()));
    FileLogger.info("CrontabRepositoryExporter", "Finished");
} catch (Exception e) {
    try {
        if (fw != null) {
            fw.close();
        }
    } catch (IOException e1) { // Ignore
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Export repository folder to server as a ZIP file

This script exports all contents of an OpenKM folder to application server as a ZIP file.

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.StringWriter;

import org.apache.commons.io.IOUtils;

import com.openkm.util.ArchiveUtils;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.impexp.RepositoryExporter;
import com.openkm.util.impexp.TextInfoDecorator;

FileUtils fileUtils = ContextWrapper.getContext().getBean(FileUtils.class);
RepositoryExporter repositoryExporter = ContextWrapper.getContext().getBean(RepositoryExporter.class);
ArchiveUtils archiveUtils = ContextWrapper.getContext().getBean(ArchiveUtils.class);

// Destination
File file = new File("/home/openkm/import/folderExport.zip");
// OpenKM folder contents to be exported
String fldPath = "/okm:root/invoices";

FileOutputStream os = new FileOutputStream(file);
StringWriter out = new StringWriter();
File tmp = fileUtils.createTempDir();
repositoryExporter.exportFolder(null, fldPath, tmp, false, out, new TextInfoDecorator(fldPath));
archiveUtils.createZip(tmp, "import", os);

org.apache.commons.io.FileUtils.deleteDirectory(tmp);
IOUtils.closeQuietly(out);
os.flush();
os.close();
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Folders deep

The script finds all folders with some deep.



*The variable **MAX_DEPTH=5** indicates the folder deep.*

```
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.openkm.api.OKMFolder;
import com.openkm.api.OKMRepository;
import com.openkm.core.Config;
import com.openkm.core.DatabaseException;
import com.openkm.db.bean.NodeBase;
import com.openkm.util.ContextWrapper;

Logger log = LoggerFactory.getLogger("com.openkm.folder.deep");
int MAX_DEPTH = 5;
int count = 0;

void findNodePathHelper(Session session, NodeBase parentNode, int depth) {
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);

    String qs = "from NodeBase nb where nb.parent=:parent";
    Query q = session.createQuery(qs).setCacheable(true);
    q.setString("parent", parentNode.getUuid());

    for (Object obj : q.list()) {
        NodeBase nb = (NodeBase) obj;
        if (depth == MAX_DEPTH && okmFolder.isValid(null, nb.getUuid())) {
            print("uuid:" + nb.getUuid() + " path:" + okmRepository.getNodePath(null, nb.getUuid()) + "<br/>");
        } else if (depth < MAX_DEPTH) {
            findNodePathHelper(session, nb, depth + 1);
        }
    }
}

log.info("***** Process BEGIN *****");
String qs = "from NodeBase nb where nb.parent=:parent";
Session session = null;

log.info("***** Process BEGIN *****");
String qs = "from NodeBase nb where nb.parent=:parent";
Session session = null;
Transaction tx = null;

try {
    SessionFactory sessionFactory = ContextWrapper.getContext().getBean(SessionFactory.class);

    session = sessionFactory.openSession();
    tx = session.beginTransaction();

    // First level nodes
    Query q = session.createQuery(qs);
```

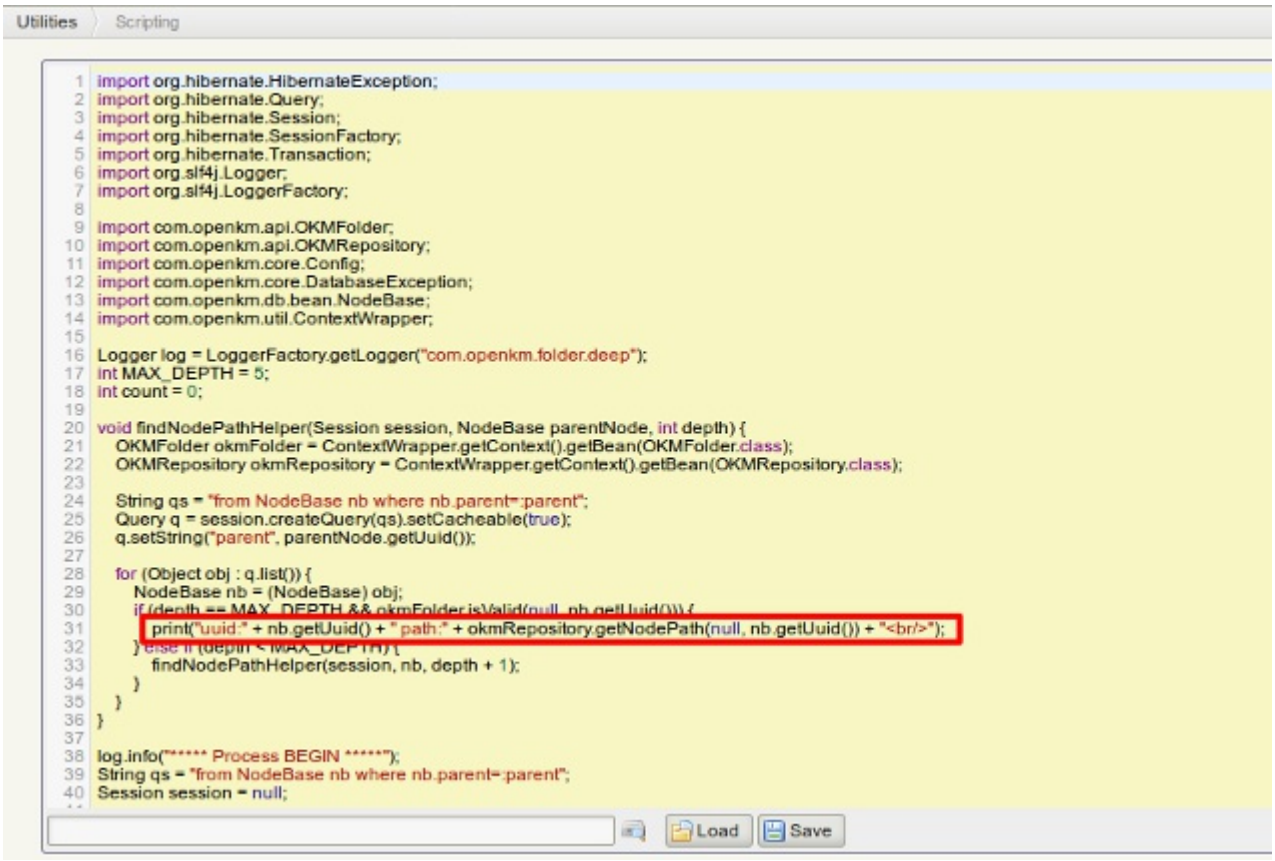
```
    q.setString("parent", Config.ROOT_NODE_UUID);

    for (Object obj : q.list()) {
        NodeBase nb = (NodeBase) obj;
        // Process in depth
        findNodePathHelper(session, nb, 0);
    }
    tx.commit();
} catch (HibernateException e) {
    tx.rollback();
    throw new DatabaseException(e.getMessage(), e);
} finally {
    session.close();
}

log.info("***** Process END *****");
```

Images

Execution



Results

Script error
Script result
Script output
uuid:e18393ad-912f-4b0a-8095-596456d52ae8 path:/okm:root/test 1.2/test2/test3/test4/test5 uuid:c0b84029-da90-472a-80e5-44819b2a47f3 path:/okm:root/test 1.1/test2/test3/test4/test5 uuid:5dbc701b-658e-4e4b-ba8b-710a7e831fac path:/okm:root/test 1.3/test2/test3/test4/test5 uuid:70606548-586e-4a10-94c1-7b32d7734f4d path:/okm:trash/okmAdmin/test 1.4/test2/test3/test4

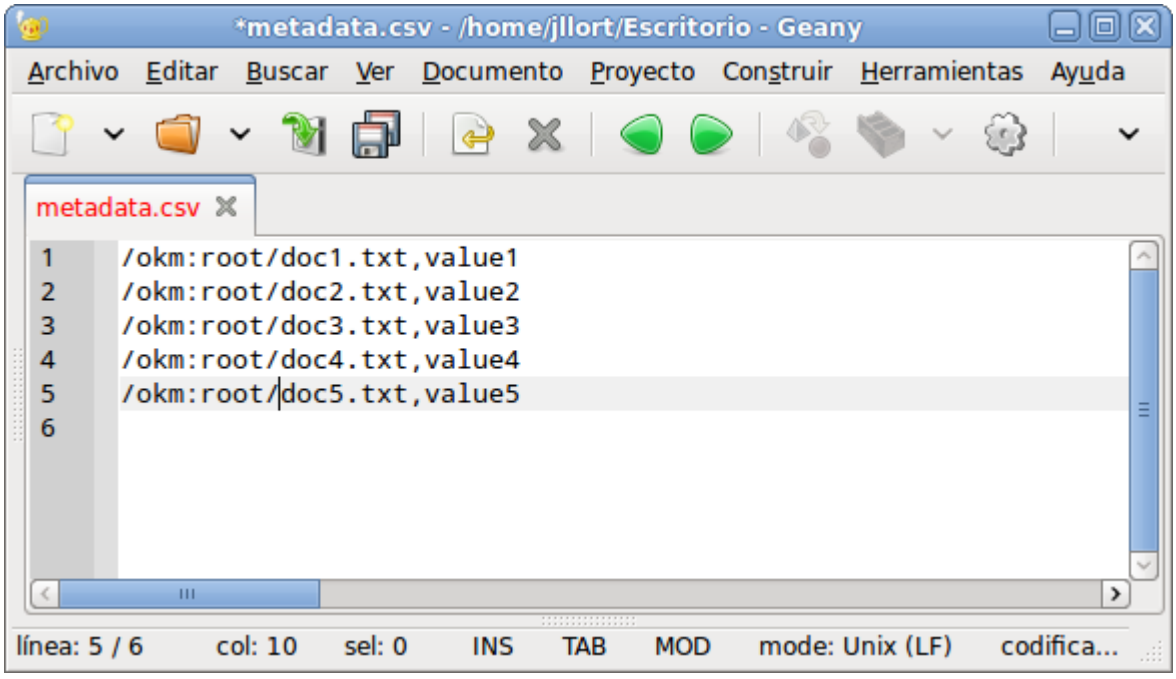
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Import data from CSV file

The script can be executed from the administration scripting (or used as automation script too) to import metadata values.

- *There's a metadata CSV file with two columns, one for the document path and other for the metadatada value. The script can be easily changed to use more than two columns.*
- *Script searches documents by document name. In theory there's only one document with the same name in the system, otherwise script shows error. Here could be used document path from first csv column, but to get more complete sample it has been used a search engine to find document.*
- *For each document found, script add metadata group with CSV values.*

The CSV file has two colum, first column contains document path and second column has metadata value.



Download CSV sample file csv_metadata.zip.

The metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
"http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
    <property-group label="Metadata" name="okg:metadata">
        <input label="Document ID" name="okp:metadata.value" type="text"/>
    </property-group>
</property-groups>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

The script:

```
import java.io.FileReader;
import java.io.Reader;
import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

import com.googlecode.jcsv.CSVStrategy;
import com.googlecode.jcsv.reader.CSVReader;
import com.googlecode.jcsv.reader.internal.CSVReaderBuilder;
import com.googlecode.jcsv.reader.internal.DefaultCSVEntryParser;
import com.openkm.api.OKMDocument;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;
import com.openkm.util.PathUtils;

String grpName = "okg:metadata";
String FILE_LOG_NAME = "CSVLOG";
String META_PATH = "/home/openkm/csv/";
String META_FILE_NAME = "metadata.csv";
int uniqueFileName = 0;
int valueColumn = 1;

// Format defintion
char delimiter = ',';
char quoteCharacter = '"';
char commentIndicator = '#';
boolean skipHeader = false;
boolean ignoreEmptyLines = true;
CSVStrategy strategy = new CSVStrategy(delimiter, quoteCharacter, commentIndicator, skipHeader, ignoreEmptyL
// File reader
Reader reader = new FileReader(META_PATH + META_FILE_NAME);
// CSV reader
CSVReader csvParser = new CSVReaderBuilder(reader).strategy(strategy).entryParser(new DefaultCSVEntryParse
List data = csvParser.readAll();
int count = 0;
int countFound = 0;
int countNotDocument = 0;
int moreThanOneDocumentFound = 0;
int notFound = 0;
int noName = 0;

OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class);
PathUtils pathUtils = ContextWrapper.getContext().getBean(PathUtils.class);

for (int i = 0; i < data.size(); i++) {
    String[] row = (String[]) data.get(i);
    String docPath = row[uniqueFileName];
    print(count + ">>>> " + docPath);

    if (docPath != null && !docPath.equals("")) {
        QueryParams queryParams = new QueryParams();
        queryParams.setDomain(QueryParams.DOCUMENT);
        queryParams.setName(pathUtils.getName(docPath));
        Collection results = okmSearch.find(null, queryParams);

        if (results.size() == 1) {
```

```
            QueryResult queryResult = (QueryResult) results.iterator().next();
            if (queryResult.getNode() != null && queryResult.getNode() instanceof Document) {
                print("found");
                countFound++;

                Map properties = new HashMap();
                properties.put("okp:metadata.value", row[valueColumn]);

                // Add Group and metadata
                okmPropertyGroup.addGroup(null, docPath, grpName, properties);
            } else {
                print("error is not document");
                countNotDocument++;
            }
        } else if (results.size() > 1) {
            print("error more than one document found can not decide");
            moreThanOneDocumentFound++;
        } else {
            print("not found");
            notFound++;
        }
    } else {
        print("error document has no name");
        noName++;
    }

    print("</br>");

    count++;
}

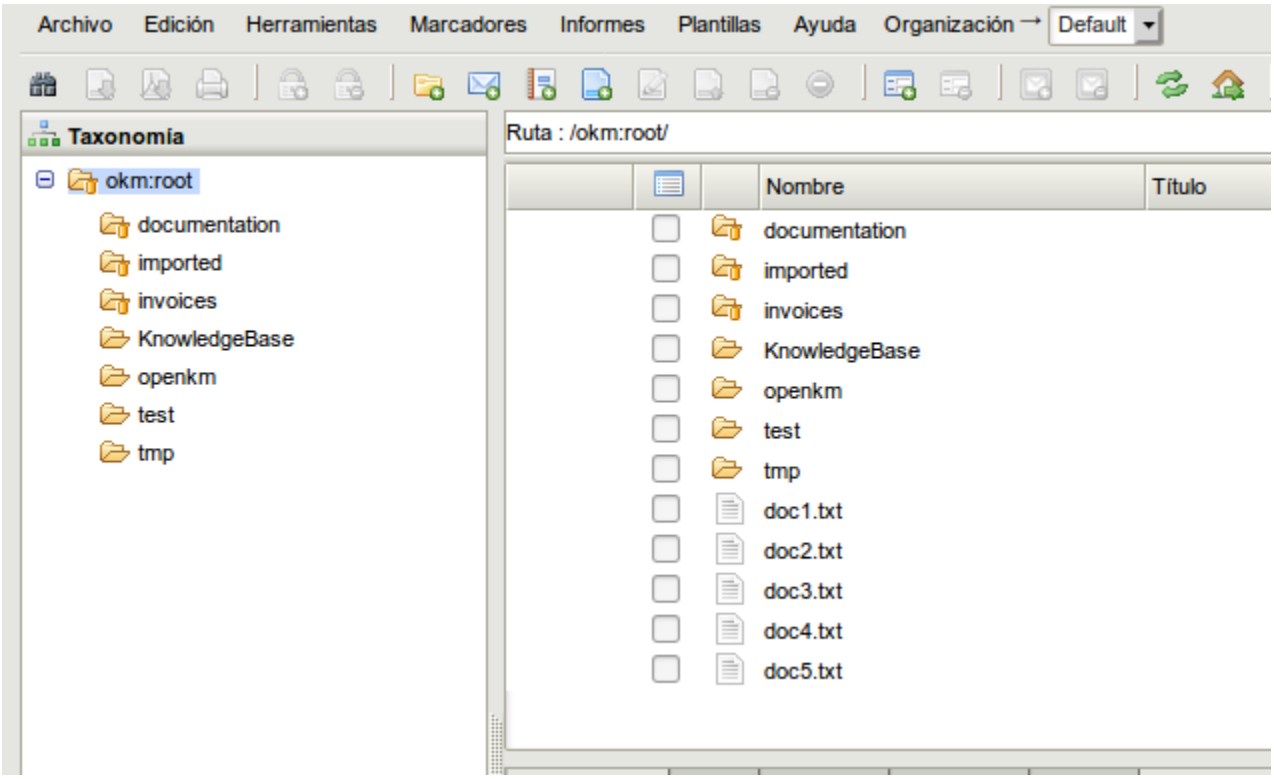
print("Total:" + count + "</br>");
print("Found:" + countFound + "</br>");
print("Error not document:" + countNotDocument + "</br>");
print("Error more then one document found:" + moreThanOneDocumentFound + "</br>");
print("Error not found:" + notFound + "</br>");
print("Error name empty:" + notFound + "</br>");
```

Image

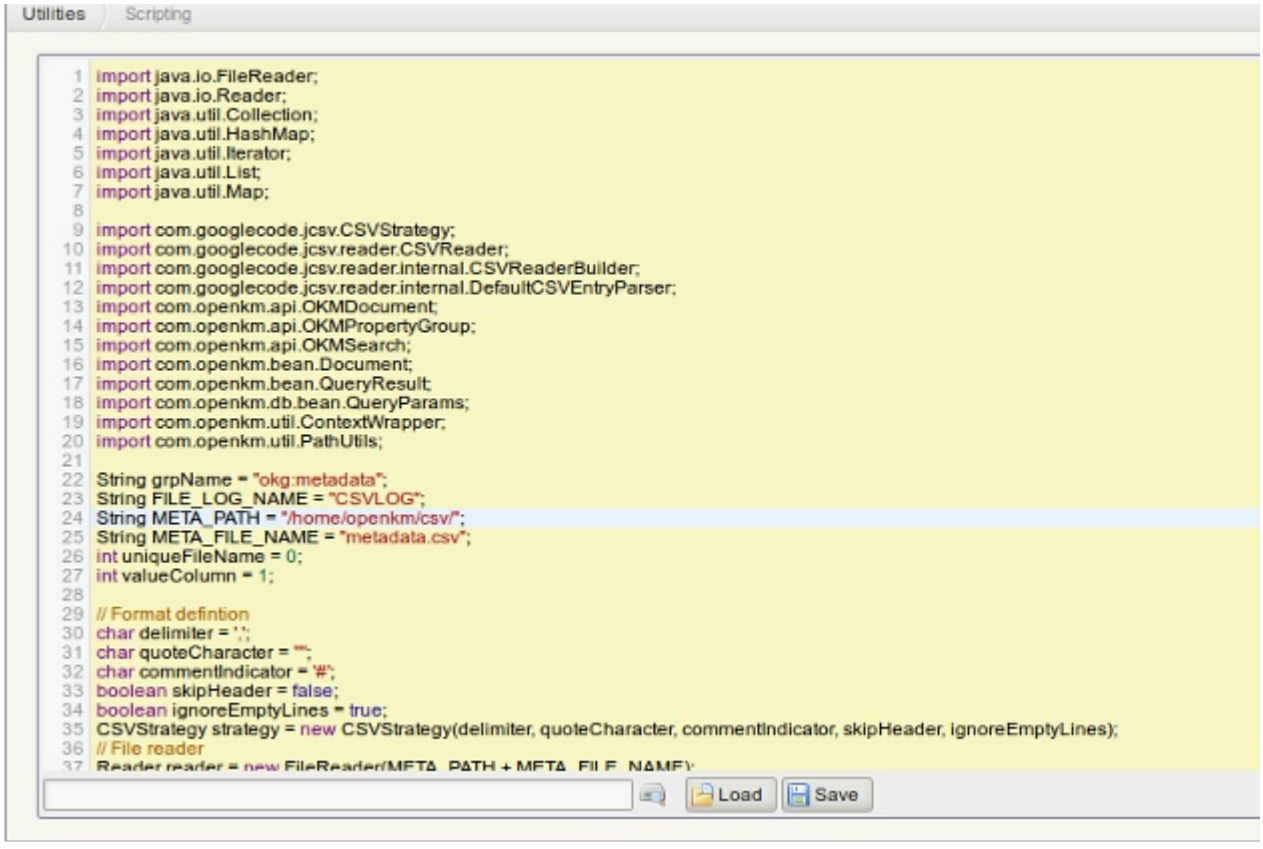
Register the metadata group



Repository view


Archivo
Edición
Herramientas
Marcadores
Informes
Plantillas
Ayuda
Organización
Default
Taxonomía
okm:root
documentation
imported
invoices
KnowledgeBase
openkm
test
tmp
Ruta : /okm:root/
Nombre
Título
documentation
imported
invoices
KnowledgeBase
openkm
test
tmp
doc1.txt
doc2.txt
doc3.txt
doc4.txt
doc5.txt

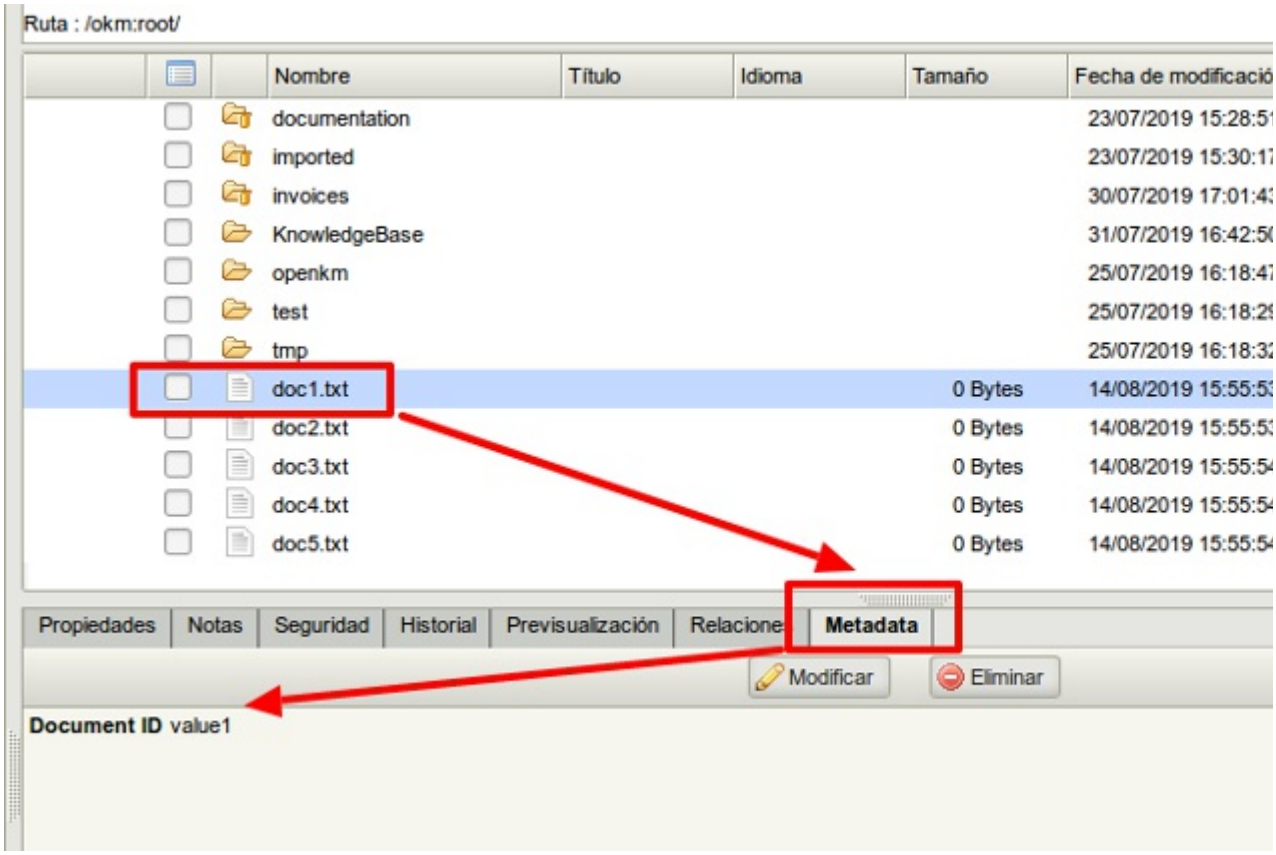
Execution


Utilities
Scripting
import java.io.FileReader;
import java.io.Reader;
import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import com.googlecode.jcsv.CSVStrategy;
import com.googlecode.jcsv.reader.CSVReader;
import com.googlecode.jcsv.reader.internal.CSVReaderBuilder;
import com.googlecode.jcsv.reader.internal.DefaultCSVEntryParser;
import com.openkm.api.OKMDocument;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;
import com.openkm.util.PathUtils;
String grpName = "okg:metadata";
String FILE_LOG_NAME = "CSVLOG";
String META_PATH = "/home/openkm/csv/";
String META_FILE_NAME = "metadata.csv";
int uniqueFileName = 0;
int valueColumn = 1;
// Format defintion
char delimiter = ',';
char quoteCharacter = '"';
char commentIndicator = '#';
boolean skipHeader = false;
boolean ignoreEmptyLines = true;
CSVStrategy strategy = new CSVStrategy(delimiter, quoteCharacter, commentIndicator, skipHeader, ignoreEmptyLines);
// File reader
Load
Save

Script results

Script error
Script result
Script output
0>>>> /okm:root/doc1.txt found 1>>>> /okm:root/doc2.txt found 2>>>> /okm:root/doc3.txt found 3>>>> /okm:root/doc4.txt found 4>>>> /okm:root/doc5.txt found Total:5 Found:5 Error not document:0 Error more then one document found:0 Error not found:0 Error name empty:0

Imported metadata



Script - Purge all folders into some parent

That script purges all folders into a parent, without previously moving files to trash.



The purge action is a non recovery action that removes definitely the objects from OpenKM repository. Use it with care.

```
import java.util.ArrayList;
import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

import com.openkm.api.OKMFolder;
import com.openkm.db.bean.NodeFolder;
import com.openkm.util.ContextWrapper;

String qs = "from NodeFolder nf where nf.parent='2ed9586e-570c-4d27-a107-eeb0a5f181be'";
Session session = null;
Transaction tx = null;
boolean found = true;

try {
    SessionFactory sessionFactory = ContextWrapper.getContext().getBean(SessionFactory.class);
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);

    while (found) {

        session = sessionFactory.openSession();
        Query q = session.createQuery(qs);
        q.setMaxResults(100);
        found = false;
        List fldList = new ArrayList();

        for (NodeFolder fld : q.list()) {
            found = true;
            fldList.add(fld.getUuid());
        }

        for (String uuid : fldList) {
            try {
                okmFolder.purge(null, uuid);
            } catch (Exception e) {
                print(e.getMessage() + "<br/>");
            }
        }
    }
} catch (Exception e) {
    print(e.getMessage());
} finally {
    session.close();
}
```

Script - Purge all users trashes

```
import com.openkm.api.OKMDocument;
import com.openkm.api.OKMFolder;
import com.openkm.api.OKMMail;
import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.bean.Mail;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.util.ContextWrapper;

try {
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
    OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);

    String token = DbSessionManager.getInstance().getSystemToken();

    for (Folder trash : okmFolder.getChildren(token, "/okm:trash")) {
        print("Trash: " + trash.getPath() + "<br/>");

        for (Folder fld : okmFolder.getChildren(token, trash.getPath())) {
            print("About to delete folder: " + fld.getPath() + "<br/>");
            okmFolder.purge(token, fld.getUuid());
        }

        for (Document doc : okmDocument.getChildren(token, trash.getPath())) {
            print("About to delete document: " + doc.getPath() + "<br/>");

            if (okmDocument.isLocked(token, doc.getUuid())) {
                okmDocument.forceUnlock(token, doc.getUuid());
            }

            okmDocument.purge(token, doc.getUuid());
        }

        for (Mail mail : okmMail.getChildren(token, trash.getUuid())) {
            print("About to delete mail: " + mail.getPath() + "<br/>");
            okmMail.purge(token, mail.getUuid());
        }
    }
} catch (Exception e) {
    print(e.getMessage());
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Put documents from folder and descendants into pending text extractio

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.openkm.api.OKMFolder;
import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.db.service.LegacySrv;
import com.openkm.util.ContextWrapper;

Logger log = LoggerFactory.getLogger("com.openkm.scripting");
int MAX_DEPTH = Integer.MAX_VALUE;

void nodeTask(String uuid, int depth) throws Exception {
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
    LegacySrv legacySrv = ContextWrapper.getContext().getBean(LegacySrv.class);

    String sql = "UPDATE OKM_NODE_DOCUMENT SET NDC_TEXT_EXTRACTED='F' WHERE NBS_UUID IN (S
            + uuid + "')";
    legacySrv.executeSQL(sql);

    for (Folder fld : okmFolder.getChildren(null, uuid)) {
        log.info("Folder: {}", fld.getPath());

        if (depth < MAX_DEPTH) {
            nodeTask(fld.getUuid(), depth + 1);
        }
    }
}

try {
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
    log.info("***** Process BEGIN *****");
    String fldPath = "/okm:root/KnowledgeBase";
    String fldUuid = okmRepository.getNodeUuid(null, fldPath);
    nodeTask(fldUuid, 0);
    log.info("***** Process END *****");
} catch (Exception e) {
    print(e.getMessage());
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Read PDF form fields

This is a script what demonstrates how to read PDF form fields. Sample document formtempl.pdf.

```
import com.lowagie.text.pdf.*;
import com.openkm.util.*;
import com.openkm.api.*;
import org.apache.commons.io.IOUtils;
import java.io.InputStream;

OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
InputStream is = null;

try {
 // Replace UUID of the document by yours
 is = okmDocument.getContent(null, "e5a84c26-f8ed-4e7a-8a05-257ea4db3569", false);
 PdfReader reader = new PdfReader(is);
 PRAcroForm form = reader.getAcroForm();
 AcroFields fields = reader.getAcroFields();

 if (form != null) {
  for (PRAcroForm.FieldInformation field : form.getFields()) {
   String fieldName = field.getName();
   String fieldValue = fields.getField(fieldName);
   print("Field: " + fieldName + ", Value: " + fieldValue);
  }
 }
} finally {
 IOUtils.closeQuietly(is);
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Recursive character renaming

Renames a character in all folders or documents from some initial path.

```
// Recursive renaming & to -
import com.openkm.api.OKMDocument;
import com.openkm.api.OKMFolder;
import com.openkm.api.OKMRepository;
import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;
import com.openkm.util.PathUtils;

void findNodePathHelper(String uuid) throws Exception {
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
    PathUtils pathUtils = ContextWrapper.getContext().getBean(PathUtils.class);

    for (Folder fld : okmFolder.getChildren(null, uuid)) {
        String fldName = pathUtils.getName(fld.getPath());

        if (fldName.contains("&")) {
            print(fld.getPath() + "<br>");
            String newName = fldName.replaceAll("&", "-");
            okmFolder.rename(null, fld.getUuid(), newName);
        }

        for (Document doc : okmDocument.getChildren(null, fld.getPath())) {
            String docName = pathUtils.getName(doc.getPath());

            if (docName.contains("&")) {
                print(doc.getPath() + "<br>");
                String newName = docName.replaceAll("&", "-");
                okmDocument.rename(null, doc.getUuid(), newName);
            }
        }

        findNodePathHelper(fld.getUuid());
    }
}

try {
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
    String fldPath = "/okm:root";
    String fldUuid = okmRepository.getNodeUuid(null, fldPath);
    findNodePathHelper(fldUuid);
} catch (Exception e) {
    print(e.getMessage());
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Recursive repository traversal

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.openkm.api.OKMDocument;
import com.openkm.api.OKMFolder;
import com.openkm.api.OKMRepository;
import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

Logger log = LoggerFactory.getLogger("com.openkm.scripting");
int MAX_DEPTH = Integer.MAX_VALUE;

void nodeTask(String uuid, int depth) throws Exception {
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);

    for (Document doc : okmDocument.getChildren(null, uuid)) {
        log.info("Document: {}", doc.getPath());
        print("Document: " + doc.getPath() + "<br>");
    }

    for (Folder fld : okmFolder.getChildren(null, uuid)) {
        log.info("Folder: {}", fld.getPath());
        print("Document: " + fld.getPath() + "<br>");

        if (depth < MAX_DEPTH) {
            nodeTask(fld.getUuid(), depth + 1);
        }
    }
}

log.info("***** Process BEGIN *****");
try {
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
    String fldPath = "/okm:root";
    String fldUuid = okmRepository.getNodeUuid(null, fldPath);
    nodeTask(fldUuid, 0);
} catch (Exception e) {
    print(e.getMessage());
}
log.info("***** Process END *****");
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Recursive change security

```
import java.util.HashMap;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.openkm.api.OKMAuth;
import com.openkm.api.OKMDocument;
import com.openkm.api.OKMFolder;
import com.openkm.api.OKMMail;
import com.openkm.api.OKMRecord;
import com.openkm.bean.CommonUser;
import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.bean.Mail;
import com.openkm.bean.Permission;
import com.openkm.bean.Record;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.util.ContextWrapper;

Logger log = LoggerFactory.getLogger("com.openkm.scripting");
int MAX_DEPTH = Integer.MAX_VALUE;
Map grantUsers = new HashMap();
Map grantRoles = new HashMap();
Map revokeUsers = new HashMap();
Map revokeRoles = new HashMap();

OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

void nodeTask(String uuid, int depth) throws Exception {
    setSecurity(uuid);
    for (Document doc : okmDocument.getChildren(null, uuid)) {
        setSecurity(doc.getUuid());
        log.info("Document: {}", doc.getPath());
    }

    for (Folder fld : okmFolder.getChildren(null, uuid)) {
        if (depth < MAX_DEPTH) {
            nodeTask(fld.getUuid(), depth + 1);
        }
    }

    for (Mail mail : okmMail.getChildren(null, uuid)) {
        setSecurity(mail.getUuid());
        log.info("Mail: {}", mail.getPath());
    }

    for (Record rec : okmRecord.getChildren(null, uuid)) {
        log.info("Record: {}", rec.getPath());

        if (depth < MAX_DEPTH) {
            nodeTask(rec.getUuid(), depth + 1);
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
void setSecurity(String uuid) throws PathNotFoundException, AccessDeniedException, DatabaseException {
    // remove all
    okmAuth.changeSecurity(null, uuid, new HashMap(), revokeUsers, new HashMap(), revokeRoles, false);
    // set specific
    okmAuth.changeSecurity(null, uuid, grantUsers, new HashMap(), grantRoles, new HashMap(), false);
}

try {
    log.info("***** Process BEGIN *****");
    String parentUuid = "162735a1-661c-4555-bf53-4e9115f38c62"; // Choose your folder or record UUID here
    // Loading users and roles to be removed
    for (CommonUser user : okmAuth.getUsers(null)) {
        revokeUsers.put(user.getId(), Permission.ALL);
    }
    for (String role : okmAuth.getRoles(null)) {
        revokeRoles.put(role, Permission.ALL);
    }

    // Loading users and roles to be added ( allowed permissions are ALL = READ + WRITE + DELETE + SECURIT
    grantRoles.put("ROLE_USER", Permission.READ);
    grantUsers.put("okmAdmin", Permission.ALL);
    grantUsers.put("sochoa", Permission.READ + Permission.WRITE);

    nodeTask(parentUuid, 0);
    log.info("***** Process END *****");
} catch (Exception e) {
    print(e.getMessage());
}
```

Script - Security change at low level

This is an administrator script that should be used only when to propagate security changes under a folder which have a significant number of nodes is needed. These type of changes made from UI can take a lot of time to be propagated across the repository and this script does it much faster.



*Our recommendation is to use it when changes affects more than **100.000 nodes** otherwise we recommend do it from UI.*



The security change starts with some folder UUID node and propagates across all children. The logic is to remove actual grants for each node and replace by newer based only on roles. Note that you can build other type of logic. Take the example as a basis.

- *The variable **UUID** is the folder Unique Identifier.*
- *The variable **MAX_DEPTH** sets how much deep should advance the script in subfolders children (note than 0 indicates the UUID folder).*
- *The variables **roleName** and **roleGrant** arrays are the pairs of role name and roles grants to be added.*
- *The variable **LOG_FILE_NAME** is the name of the log file saved under $TOMCAT_HOME folder.*

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

import com.openkm.bean.Permission;
import com.openkm.core.Config;

// PARAMETERS
String LOG_FILE_NAME = "Security";
int MAX_DEPTH = Integer.MAX_VALUE;
String folder_UUID = "162735a1-661c-4555-bf53-4e9115f38c62";
String[] roleName = {"ROLE_X", "ROLE_Y"};
int[] roleGrant = {Permission.READ, Permission.READ | Permission.WRITE};

public static class FileLogger {

    public static final String DATE_FORMAT = "yyyy-MM-dd HH:mm:ss,SSS";
    public static final String LEVEL_INFO = "INFO ";
    public static final String LEVEL_WARN = "WARN ";
    public static final String LEVEL_ERROR = "ERROR";

    public static void info(String baseName, String message) throws IOException {
        logWrite(baseName, LEVEL_INFO, message);
    }
```

```
    public static void logWrite(String baseName, String level, String message) throws IOException {
        Writer sLogger = new FileWriter(getLogFile(baseName), true);
        sLogger.write(getLogEntry(level, message));
        sLogger.flush();
        sLogger.close();
    }

    public static String getLogFile(String baseName) {
        String fileDate = new SimpleDateFormat("yyyyMMdd").format(new Date());
        return Config.HOME_DIR + File.separator + baseName + "_" + fileDate + ".log";
    }

    public static void info(String baseName, String message, String param1, String param2) throws IOException {
        logWrite(baseName, LEVEL_INFO, message, param1, param2);
    }

    public static void warm(String baseName, String message, String param1, String param2) throws IOException {
        logWrite(baseName, LEVEL_WARN, message, param1, param2);
    }

    public static void error(String baseName, String message, String param1, String param2) throws IOException {
        logWrite(baseName, LEVEL_ERROR, message, param1, param2);
    }

    public static void logWrite(String baseName, String level, String message, String param1, String param2)
            throws IOException {
        Writer sLogger = new FileWriter(getLogFile(baseName), true);
        sLogger.write(getLogEntry(level, message, param1, param2));
        sLogger.flush();
        sLogger.close();
    }

    public static String getLogEntry(String level, String message) {
        StringBuilder sb = new StringBuilder();
        sb.append(new SimpleDateFormat(DATE_FORMAT).format(new Date()));
        sb.append(" ");
        sb.append(level);
        sb.append(" ");
        sb.append(message);
        sb.append("\n");
        return sb.toString();
    }

    public static String getLogEntry(String level, String message, String param1, String param2) {
        Object[] params = {param1, param2};
        StringBuilder sb = new StringBuilder();
        sb.append(new SimpleDateFormat(DATE_FORMAT).format(new Date()));
        sb.append(" ");
        sb.append(level);
        sb.append(" ");
        sb.append(MessageFormat.format(message, params));
        sb.append("\n");
        return sb.toString();
    }
}

public void updateSecurity(Connection con, String UUID, int level) throws SQLException, IOException {
    FileLogger.info(LOG_FILE_NAME, "updateSecurity level:''{0}'' with UUID:''{1}''", String.valueOf(level), UUID);
    PreparedStatement ps = null;
    ResultSet rs = null;

    if (level == 0) {
        setParentNodeSecurity(con, UUID);
        removeAllChildrenNodeSecurity(con, UUID);
        setAllChildrenNodeSecurity(con, UUID);
    }
```

```
    if (level < MAX_DEPTH) {
        String sqlfindFolderChildren = "SELECT NBS_UUID FROM OKM_NODE_BASE WHERE NBS_PARENT=? an

        try {
            ps = con.prepareStatement(sqlfindFolderChildren);
            ps.setString(1, UUID);
            rs = ps.executeQuery();

            while (rs.next()) {
                String childUUID = rs.getString("NBS_UUID");
                removeAllChildrenNodeSecurity(con, childUUID);
                setAllChildrenNodeSecurity(con, childUUID);
                updateSecurity(con, childUUID, level + 1);
            }
        } catch (SQLException e) {
            FileLogger.error(LOG_FILE_NAME, "updateSecurity with UUID:"{0}" con error "{1}"", UUID, e.getMessage()
        } finally {
            ps.close();
            rs.close();
        }
    }
}

public void removeAllChildrenNodeSecurity(Connection con, String UUID) throws SQLException, IOException {
    PreparedStatement ps = null;
    String deleteChildrenNodeRoles = "DELETE FROM OKM_NODE_ROLE_PERMISSION where NRP_NODE in (S
    String deleteChildrenNodeUsers = "DELETE FROM OKM_NODE_USER_PERMISSION where NUP_NODE in (S

    try {
        ps = con.prepareStatement(deleteChildrenNodeRoles);
        ps.setString(1, UUID);
        ps.executeUpdate();
        ps = con.prepareStatement(deleteChildrenNodeUsers);
        ps.setString(1, UUID);
        ps.executeUpdate();
        con.commit();
    } catch (SQLException e) {
        FileLogger.error(LOG_FILE_NAME, "removeAllChildrenNodeSecurity with UUID:"{0}" con error "{1}"", UUID,
                e.getMessage());
    } finally {
        ps.close();
    }
}

public void setAllChildrenNodeSecurity(Connection con, String UUID) throws SQLException, IOException {
    PreparedStatement ps = null;
    ResultSet rs = null;
    String sqlfindAllChildren = "SELECT NBS_UUID FROM OKM_NODE_BASE WHERE NBS_PARENT=?";
    String insertRoleSecurityChildren = "INSERT INTO OKM_NODE_ROLE_PERMISSION (NRP_NODE,NRP_PER

    try {
        ps = con.prepareStatement(sqlfindAllChildren);
        ps.setString(1, UUID);
        rs = ps.executeQuery();
        ps = con.prepareStatement(insertRoleSecurityChildren);

        while (rs.next()) {
            String childUUID = rs.getString("NBS_UUID");

            for (int i = 0; i < roleName.length; i++) {
                ps.setString(1, childUUID);
                ps.setInt(2, roleGrant[i]);
                ps.setString(3, roleName[i]);
                ps.addBatch();
            }
        }
```

```
            ps.executeBatch();
            con.commit();
        } catch (SQLException e) {
            FileLogger.error(LOG_FILE_NAME, "removeAllChildrenNodeSecurity with UUID:"{0}" con error "{1}"", UUID,
                    e.getMessage());
        } finally {
            ps.close();
            rs.close();
        }
    }

    public void setParentNodeSecurity(Connection con, String UUID) throws SQLException, IOException {
        PreparedStatement ps = null;
        String deleteChildrenNodeRoles = "DELETE FROM OKM_NODE_ROLE_PERMISSION where NRP_NODE =?";
        String deleteChildrenNodeUsers = "DELETE FROM OKM_NODE_USER_PERMISSION where NUP_NODE =?";
        String insertRoleSecurityChildren = "INSERT INTO OKM_NODE_ROLE_PERMISSION (NRP_NODE,NRP_PERI

        try {
            ps = con.prepareStatement(deleteChildrenNodeRoles);
            ps.setString(1, UUID);
            ps.executeUpdate();
            ps = con.prepareStatement(deleteChildrenNodeUsers);
            ps.setString(1, UUID);
            ps.executeUpdate();
            ps = con.prepareStatement(insertRoleSecurityChildren);

            for (int i = 0; i < roleName.length; i++) {
                ps.setString(1, UUID);
                ps.setInt(2, roleGrant[i]);
                ps.setString(3, roleName[i]);
                ps.addBatch();
            }

            ps.executeBatch();
            con.commit();
        } catch (SQLException e) {
            FileLogger.error(LOG_FILE_NAME, "setParentNodeSecurity with UUID:"{0}" con error "{1}"", UUID, e.getMess
        } finally {
            ps.close();
        }
    }

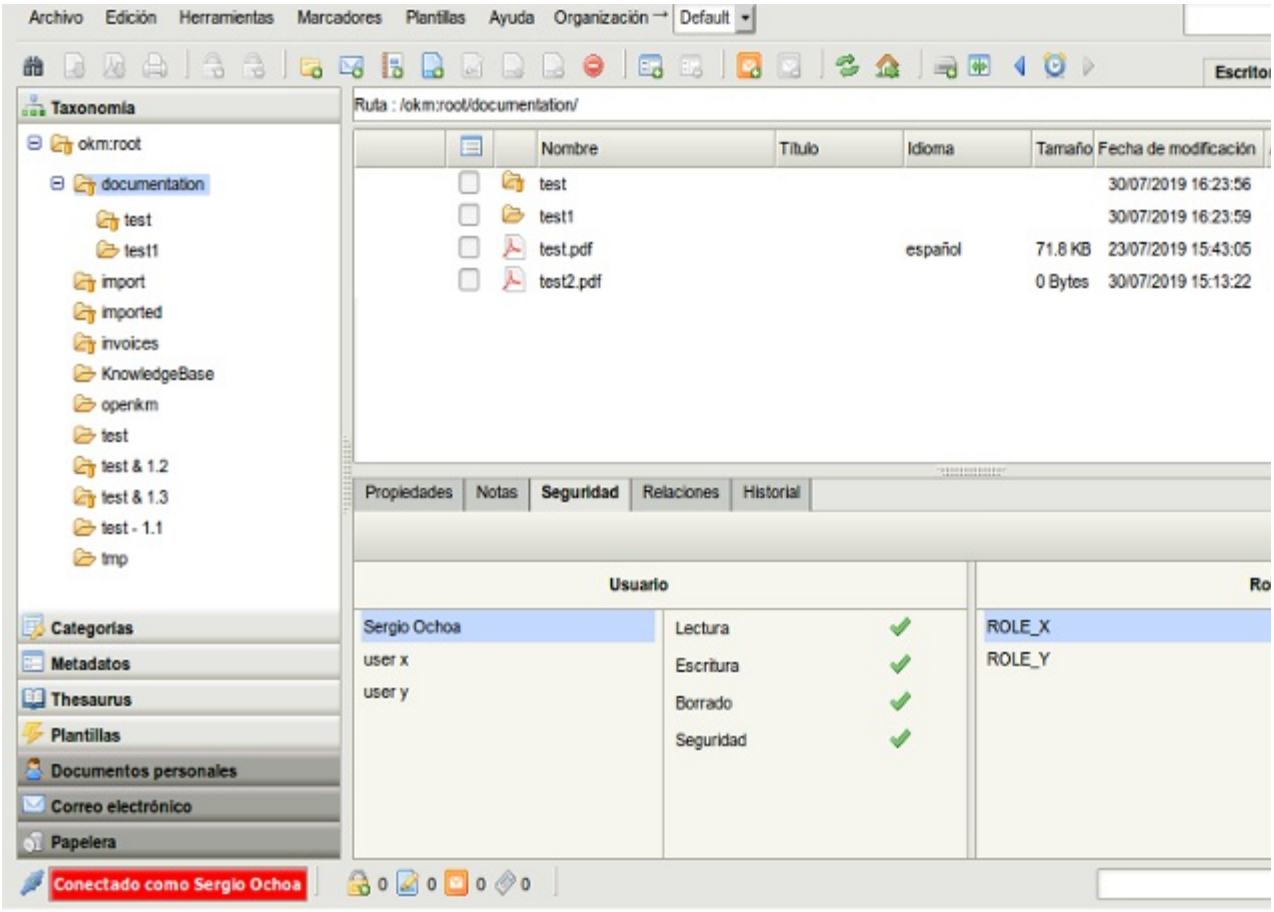

    try {
        FileLogger.info(LOG_FILE_NAME, "**** Security change started ****");
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = null;
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/okmdb", "openkm", "*secret*");
        con.setAutoCommit(false);
        updateSecurity(con, folder_UUID, 0);
        con.close();
        FileLogger.info(LOG_FILE_NAME, "**** Security change ended ****");
    } catch (Exception e) {
        print(e.getMessage());
    }
```



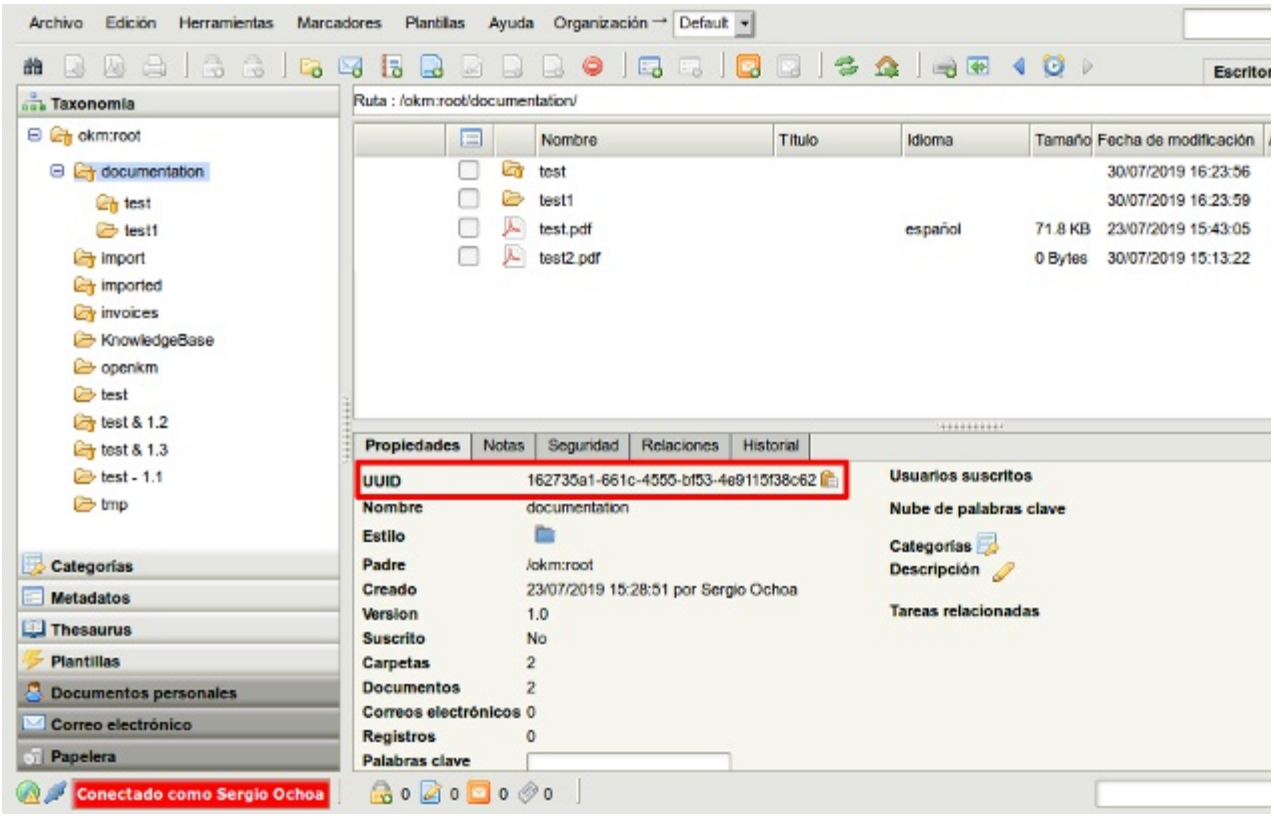
After the script execution we must reindex Lucene indexes because the security changes has been done at the database level and it is necessary for maintaining integrity with search engine.

Images

Initial security status

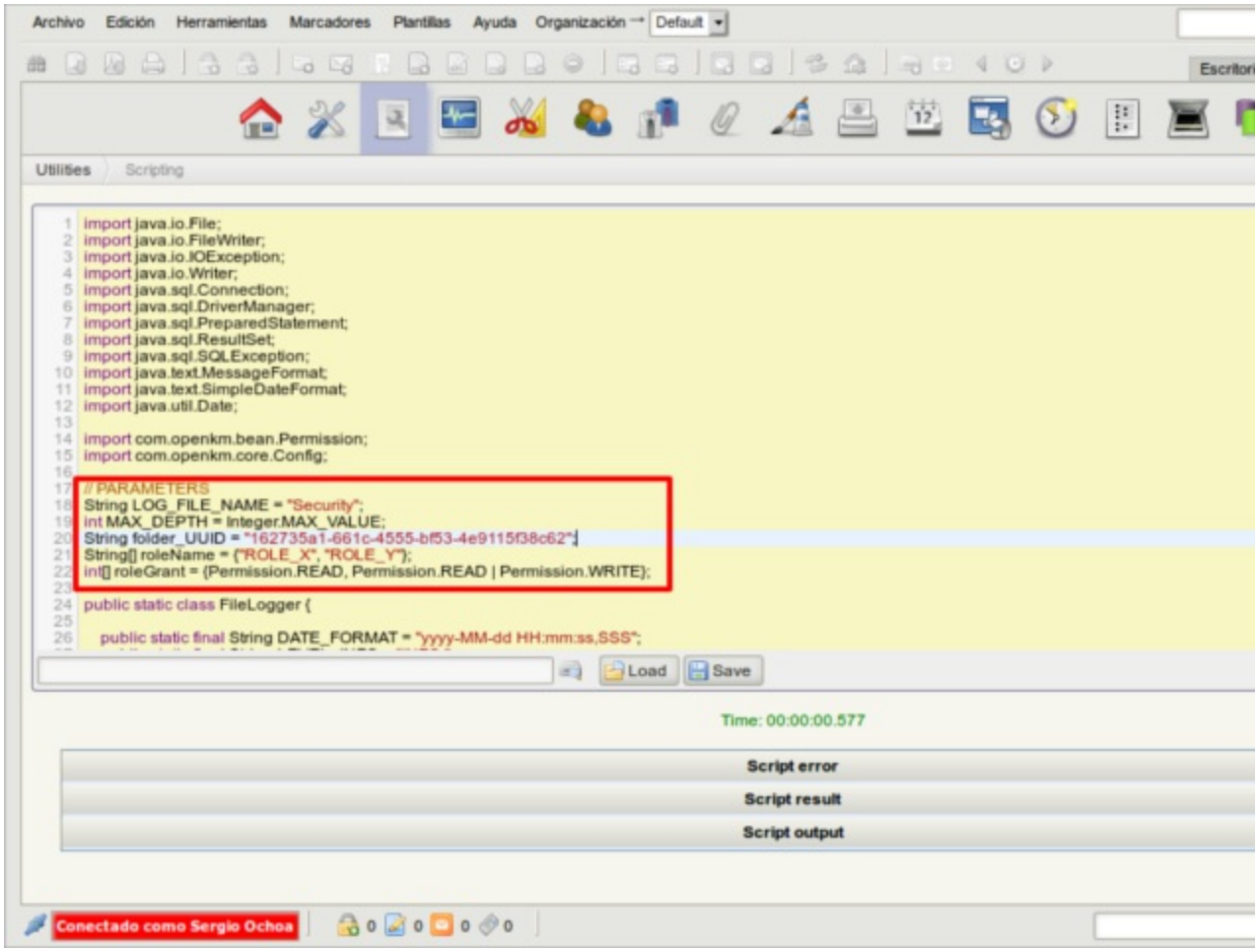


UUID of the folder



Execution

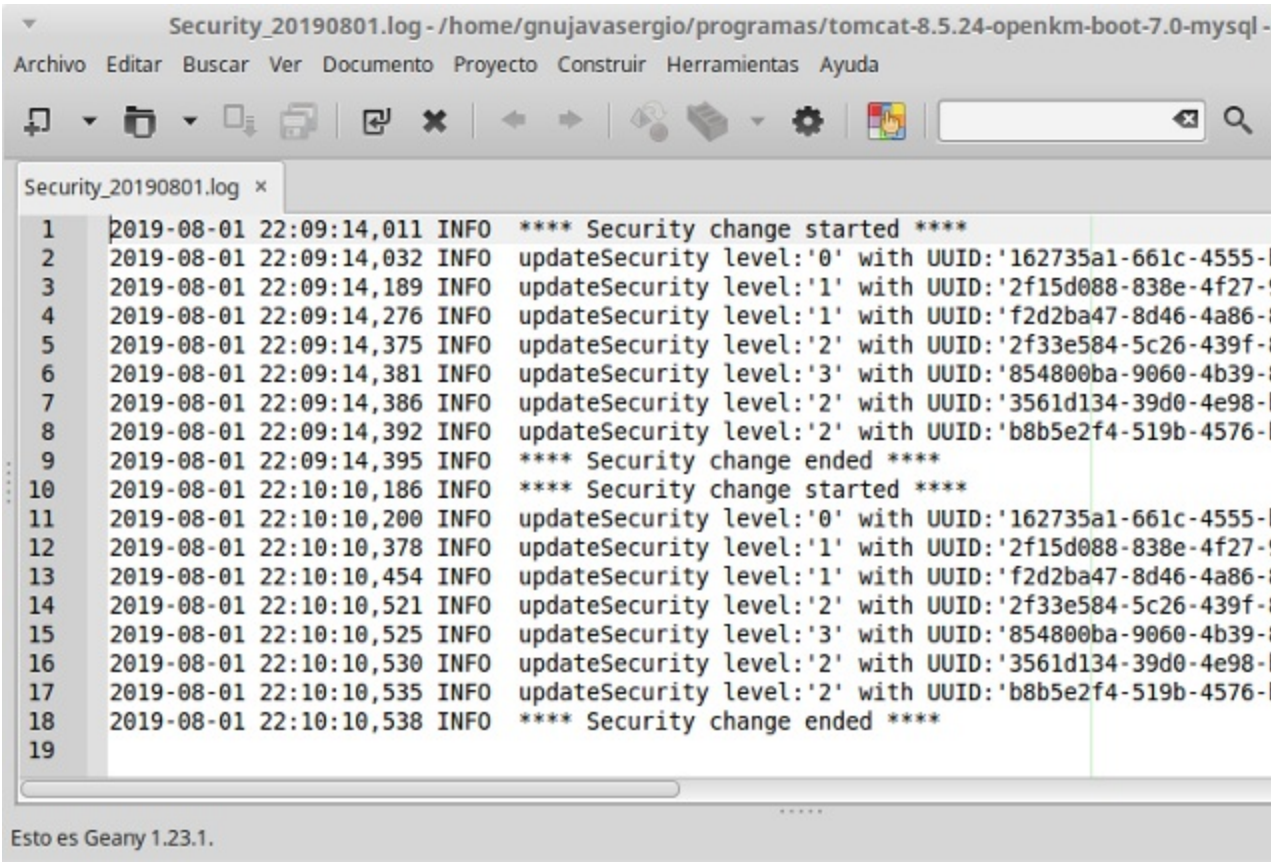
Change script parameters by your owns.



Log

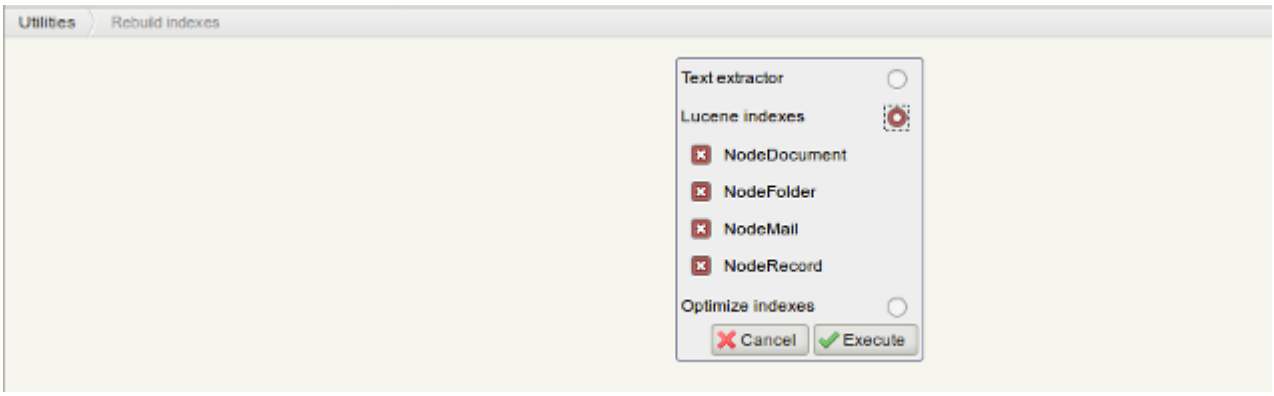
Consider take a look at file log to see progress of the process and check for errors.

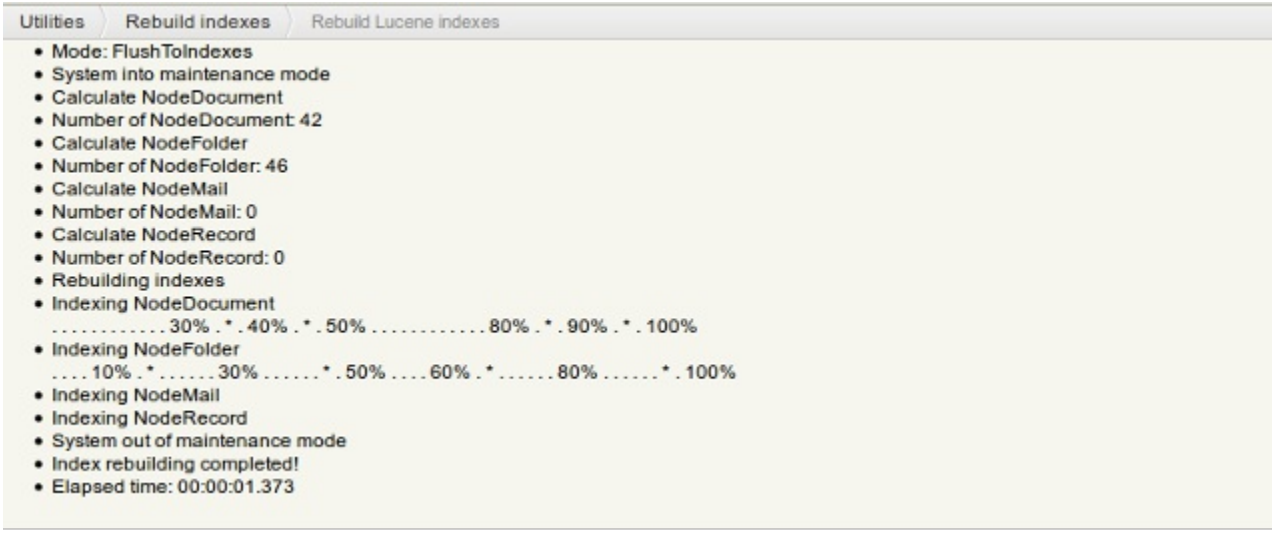
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



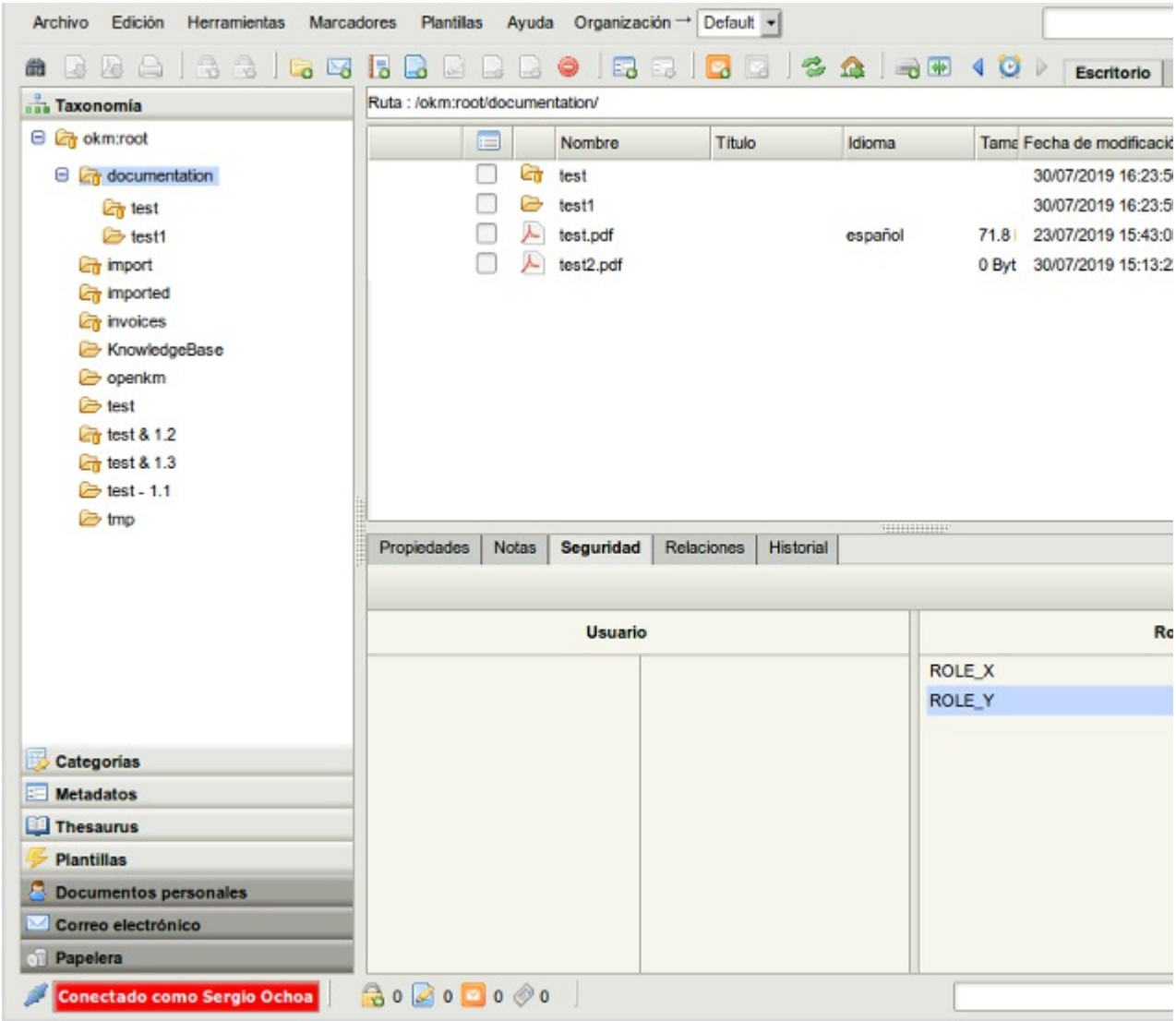
Rebuild Lucene indexes

Caused by changes of the security at a database level, it is necessary for maintaining integrity with search engine, reindex Lucene indexes.





Security changed applied



Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script samples for automation

Script	*Description*
Script for automation - Autonumeric	*Sets an unique number for each uploaded document.*
Script for automation - Set unique document name based on metadata	*Each time a document is uploaded the name of the document is changed based on a metadata group values.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script for automation - Autonumeric

Creates an unique number for each new document.

- *There's a metadata with tree fields which stores data.*
- *Each time a new document is uploaded, automation is executed. Automation script creates a new unique number identifier using metadata database sequence feature.*
- *The **document id** is a 6 digits incremental numeric*
- *The **revision** number starts with 1*
- *The **code** is a unique id + "-" + revision*

To get it running it's needed to register a metadata group and then create and automation task based on scripting.

This is only a sample implementation and not intended to be used in production.

Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
"http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Autonumeric" name="okg:autonumber">
    <input label="Document ID" name="okp:autonumber.id" type="text" readonly="true"/>
    <input label="Revision" name="okp:autonumber.revision" type="text" readonly="true"/>
    <input label="Code" name="okp:autonumber.code" type="text" readonly="true"/>
  </property-group>
</property-groups>
```

The automation script:

```
import com.openkm.db.service.*;
import com.openkm.db.bean.*;
import com.openkm.api.*;
import com.openkm.util.*;

getNextValue() {
  ConfigSrv cfgSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
  Config cfg = cfgSrv.findByPk("autonumber.data.id");

  if (cfg == null) {
    cfg = new Config();
    cfg.setType(Config.INTEGER);
    cfg.setKey("autonumber.data.id");
    cfg.setValue("0");
  }

  int value = Integer.parseInt(cfg.getValue()) + 1;
```

```
  cfg.setValue(String.valueOf(value));
  cfgSrv.save(cfg);

  return String.format("%04d", value);
}

try {
  // Set metadata
  String id = getNextValue();
  String rev = "1";
  String code = id + "-" + rev;

  // Add metadata
  Map props = new HashMap();
  props.put("okp:autonumber.id",id);
  props.put("okp:autonumber.revision",rev);
  props.put("okp:autonumber.code",code);

  OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class);
  okmPropertyGroup.addGroup(null, uuid, "okg:autonumber", props);
} catch (Exception e) {
  e.printStackTrace();
}
```

Register metadata group

Create automation task based on metadata group creation event:

- ***Name**: Autonumeric*
- ***Event**: Document create*
- ***At**: Post*

*Use the previous scripting in **ExecuteScripting** action.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script for automation - Set unique document name based on metadata

Each time a new document is uploaded, the script sets a unique document name based in metadata values.

- *Property **okp:data.id** stores a unique auto incremental value.*
- *Property **okp:data.project.code** stores a project code value.*
- *Property **okp:data.customer** stores a customer code.*
- *Property **okp:data.description** stores a document description.*
- *When a new document is uploaded the user must fill all the fields except **okp:data.id** which is automatically set by OpenKM.*
- *When Metadata group is changed - event triggered- is executed automation code which generates **okp:data.id** and renames the document based on mask **projectCode-autonumericId-clientCode-description.documentExtension**.*

To get it running it's needed to register a metadata group and then create and automation task based on scripting.

This is only a sample implementation and not intended to be used in production.

Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Data" name="okg:data">
    <input label="Id" type="text" name="okp:data.id" readonly="true"/>
    <input label="Project code" type="text" name="okp:data.project_code">
      <validator type="req"/>
      <validator type="num"/>
      <validator type="maxlen" parameter="6"/>
      <validator type="minlen" parameter="6"/>
    </input>
    <select label="Customer" name="okp:data.customer" type="simple"
          optionsQuery="select CUS_ID, CUS_NAME from MY_CUSTUMER">
      <validator type="req"/>
    </select>
    <input label="Description" type="text" name="okp:data.description">
      <validator type="req"/>
      <validator type="maxlen" parameter="150"/>
    </input>
  </property-group>
</property-groups>
```

Database table definition:

```
create table MY_CUSTUMER (
  CUS_ID int not null auto_increment,
  CUS_NAME varchar(128),
  primary key(CUS_ID)
);

insert into MY_CUSTUMER(CUS_ID, CUS_NAME) values (1, 'Customer 1');
insert into MY_CUSTUMER(CUS_ID, CUS_NAME) values (2, 'Customer 2');
```

The script:

```
import com.openkm.db.service.*;
import com.openkm.db.bean.*;
import com.google.gson.*;
import com.openkm.api.*;
import com.openkm.util.*;

getNextValue() {
  ConfigSrv cfgSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
  Config cfg = cfgSrv.findByPk("autoincrement.data.id");

  if (cfg == null) {
    cfg = new Config();
    cfg.setType(Config.INTEGER);
    cfg.setKey("autoincrement.data.id");
    cfg.setValue("0");
  }

  int value = Integer.parseInt(cfg.getValue()) + 1;
  cfg.setValue(String.valueOf(value));
  cfgSrv.save(cfg);

  return String.format("%04d", value);
}

try {
  if ("okg:data".equals(env.get("propGroupName"))) {
    // Get metadata
    Map props = env.get("propGroupProperties");
    String prjCode = props.get("okp:data.project_code");
    String desc = props.get("okp:data.description");
    String cust = props.get("okp:data.customer");
    String id = props.get("okp:data.id");

    String[] opts = new Gson().fromJson(cust, String[].class);
    String clientCode = opts[0];

    // Set metadata
    if (id == null || id.isEmpty()) {
      id = getNextValue();
      props.put("okp:data.id", id);
      env.put("propGroupProperties", props);
    }

    // Rename document
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
    String path = okmRepository.getNodePath(null, uuid);
    String docExt = FileUtils.getFileExtension(PathUtils.getName(path));
    String newName = prjCode + "-" + id + "-" + clientCode + "-" + desc + "." + docExt;
    print("New doc name: " + newName);

    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
    okmDocument.rename(null, uuid, newName);
  }
```

```
} catch (Exception e) {
  e.printStackTrace();
}
```

Register automation rule

Create automation task based on metadata group creation event:

- ***Name**: AutoRename*
- ***Event**: Add metadata group*
- ***At**: Pre*

*Use the previous scripting in **ExecuteScripting** action.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Crontab samples

Script	*Description*
Crontab sample - Basic file system document importer	*The script imports files from the server file system.*
Crontab sample - Document importer with metadata values	*Imports files with metadata values from the server file system.*
Contab sample - Generate HTML report	*Generates a HTML report from OpenKM activity log.*
Crontab task - XML importer	*Import files with metadata values from XML files.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Crontab sample - Basic file system document importer

The script import files from file system of the server.

- *Files are imported from the server file system.*
- *Files are periodically imported by a crontab task to a OpenKM folder (/okm:root/Scans).*

```
package com.openkm.plugin.cron;

import java.io.File;
import java.io.FileInputStream;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.openkm.api.OKMDocument;
import com.openkm.api.OKMFolder;
import com.openkm.core.ItemExistsException;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.util.ContextWrapper;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class SystemDocumentImporter extends BaseCronPlugin implements CronAdapter {

    private Logger log = LoggerFactory.getLogger(SystemDocumentImporter.class);

    private OKMDocument document = ContextWrapper.getContext().getBean(OKMDocument.class);
    private OKMFolder folder = ContextWrapper.getContext().getBean(OKMFolder.class);
    private String token = DbSessionManager.getInstance().getSystemToken();

    @Override
    public String getName() {
        return "System Document Importer";
    }

    @Override
    public String getCronExpression() {
        return "0 5 * * * *";
    }

    @Override
    public void execute() {
        String okmPath = "/okm:root/Scans/";
        File file = new File("/home/opemkm/import");
        autoImport(okmPath, file);
    }

    private void autoImport(String okmPath, File fldpath) {
        try {
            log.info("Scanning " + fldpath.getName());
            for (File file : fldpath.listFiles()) {
                log.info("Importing " + file.getName());
                try {
                    if (file.isDirectory()) {
                        try {
                            folder.createSimple(token, okmPath + file.getName());
                        } catch (ItemExistsException ie) {
```

```
                    log.error("folder already exists<br>");
                    // Folder already exists - just ignore exception
                }
                autoImport(okmPath + file.getName() + "/", file);
            } else {
                // Check if file is still being written to
                long length = file.length();
                Thread.sleep(1000);
                if (file.length() > length) {
                    continue; // Skip file this time
                }
                FileInputStream fis = new FileInputStream(file);
                document.createSimple(token, okmPath + file.getName(), fis);
                fis.close();
            }
            log.info("Created " + okmPath + file.getName());
        } catch (Exception e) {
            log.info("Exception:" + e);
            // Something bad happened to prevent import. Skip to next file.
            continue;
        }
        file.delete();
    }
  } catch (Exception e) {
    log.error(e.getMessage());
  }
 }

}
```

Register a new plugin

- *To install new plugin **create a jar file** and **copy into your $TOMCAT/plugins** folder. More information at [Creating a JAR file with Eclipse](#).*
- *Go to **Administration** > **Utilities** > **Plugins** and click at top right the*  *Reload plugins button.*
- *In **plugins table will be shown** the new plugin.*

Images

Crontab task

Crontab

Show 15 entries

Name	Expression	Class Name	Last Begin	
Antivirus Checker Worker	0 */5 * * * *	com.openkm.plugin.cron.AntivirusChecker	18-08-2019 11:45:00	18-
Check Disk Space	0 0 0 * * *	com.openkm.plugin.cron.CheckDiskSpace	18-08-2019 00:00:00	18-
Check Repository Integrity	0 0 0 * * SAT	com.openkm.plugin.cron.CheckRepositoryIntegrity		
Dashboard Activity Purge	0 0 0 * * *	com.openkm.plugin.cron.DashboardActivity	18-08-2019 00:00:00	18-
Document Importer With Metadata	0 5 * * * *	com.openkm.plugin.cron.DocumentImporterWithMetadata	18-08-2019 11:27:28	18-
Process Pending Task	0 */5 * * * *	com.openkm.plugin.cron.ProcessPendingTask	18-08-2019 11:45:00	18-
Purge Empty Datastore Directories	0 0 0 * * *	com.openkm.plugin.cron.PurgeEmptyDatastoreDirectories	18-08-2019 00:00:00	18-
Purge Empty Extraction Directories	0 0 0 * * *	com.openkm.plugin.cron.PurgeEmptyExtractionDirectories	18-08-2019 00:00:00	18-
Purge Old Trash Nodes	0 0 0 * * *	com.openkm.plugin.cron.PurgeOldTrashesNodes	18-08-2019 00:00:00	18-
Purge Temporal Printing Files	0 0 0 * * *	com.openkm.plugin.cron.PurgeTemporalPrintingFiles	18-08-2019 00:00:01	18-
Purge Tomcat Logs	0 0 0 * * *	com.openkm.plugin.cron.PurgeTomcatLogs	18-08-2019 00:00:00	18-
Repository Info	0 0 0 * * *	com.openkm.plugin.cron.RepositoryInfo	18-08-2019 00:00:00	18-
Send Grouped Notifications	0 0 0 * * *	com.openkm.plugin.cron.SendGroupedNotifications	18-08-2019 00:00:00	18-
System Document Importer	0 5 * * * *	com.openkm.plugin.cron.SystemDocumentImporter	18-08-2019 02:05:02	18-
Text Extractor Issues	0 0 0 * * *	com.openkm.plugin.cron.TextExtractorIssues	18-08-2019 00:00:00	18-

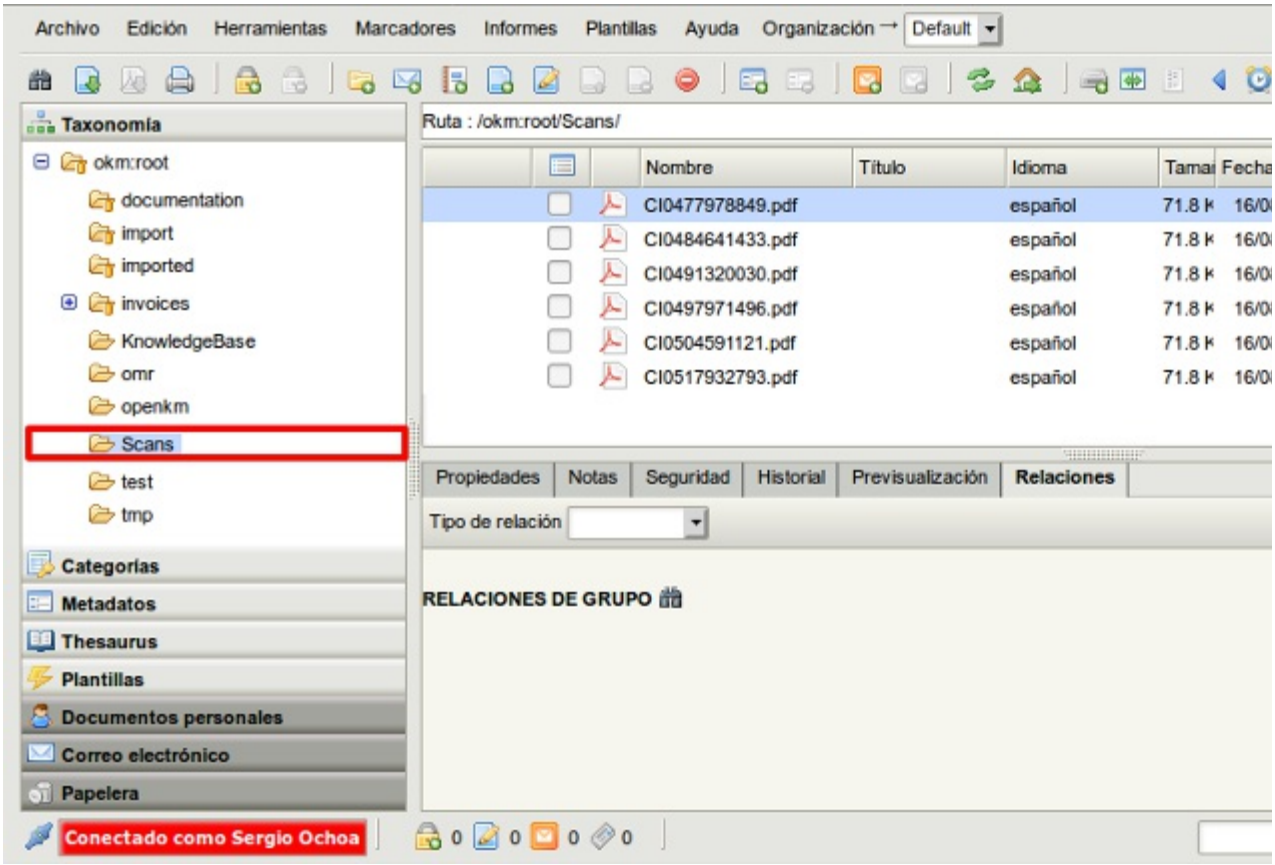
Showing 1 to 15 of 17 entries

Files in the application server

CI0477978849.pdf	244,2 KB Adobe Acrobat Document	29/09/11
CI0484641433.pdf	270,3 KB Adobe Acrobat Document	29/09/11
CI0491320030.pdf	276,4 KB Adobe Acrobat Document	29/09/11
CI0497971496.pdf	279,0 KB Adobe Acrobat Document	29/09/11
CI0504591121.pdf	278,9 KB Adobe Acrobat Document	29/09/11
CI0517932793.pdf	305,1 KB Adobe Acrobat Document	29/09/11

Result

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


Archivo
Edición
Herramientas
Marcadores
Informes
Plantillas
Ayuda
Organización
Default
Taxonomía
okm:root
documentation
import
imported
invoices
KnowledgeBase
omr
openkm
Scans
test
tmp
Categorías
Metadatos
Thesaurus
Plantillas
Documentos personales
Correo electrónico
Papelera
Conectado como Sergio Ochoa
Ruta : /okm:root/Scans/
Nombre
Título
Idioma
Tamaı
Fecha
CI0477978849.pdf
CI0484641433.pdf
CI0491320030.pdf
CI0497971496.pdf
CI0504591121.pdf
CI0517932793.pdf
español
71.8 K
Propiedades
Notas
Seguridad
Historial
Previsualización
Relaciones
Tipo de relación
RELACIONES DE GRUPO

Crontab sample - Document importer with metadata values

The script imports files with metadata values from the file system of the server.

- *Files stored in the OpenKM server have a file name format like some_text - (CUPS,REN).pdf*
- *The variable **grpName** is the metadata group to be inserted.*
- *The variable **contractUUID** is the UUID of the OpenKM where the files will be imported.*
- *The variable **systemFolder** is the file system path.*
- *Verifies paths and extracts metadata from text between the characters "**(**" "**)**" present in the file name.*
- *Files are stored at basePath/year/month (when year or month folders are not present are automatically created).*
- *At the end, it send a mail with import results.*

Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
"http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
   <property-group label="Contract" name="okg:contract">
   <input label="Cups" type="text" name="okp:contract.cups" width="200px"/>
   <input label="Año" type="text" name="okp:contract.year" width="200px"/>
   <select label="Mes" name="okp:contract.month" type="simple">
      <option label="Enero" value="enero"/>
    <option label="Febrero" value="febrero"/>
    <option label="Marzo" value="marzo"/>
      <option label="Abril" value="abril"/>
      <option label="Mayo" value="mayo"/>
      <option label="Junio" value="junio"/>
      <option label="Julio" value="julio"/>
      <option label="Agosto" value="agosto"/>
      <option label="Septiembre" value="septiembre"/>
      <option label="Octubre" value="octubre"/>
      <option label="Noviembre" value="noviembre"/>
      <option label="Diciembre" value="diciembre"/>
   </select>
  </property-group>
</property-groups>
```

The script:

```
package com.openkm.plugin.cron;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.Calendar;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.cxf.bus.extension.ExtensionException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import com.google.gson.Gson;
import com.openkm.api.OKMDocument;
import com.openkm.api.OKMFolder;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.api.OKMRepository;
import com.openkm.bean.Document;
import com.openkm.core.AccessDeniedException;
import com.openkm.core.Config;
import com.openkm.core.DatabaseException;
import com.openkm.core.ItemExistsException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.PromotedAsRecordLockException;
import com.openkm.core.RepositoryException;
import com.openkm.core.UnsupportedMimeTypeException;
import com.openkm.core.VirusDetectedException;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.plugin.automation.AutomationException;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.MailUtils;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class DocumentImporterWithMetadata extends BaseCronPlugin implements CronAdapter {

    private Logger log = LoggerFactory.getLogger(DocumentImporterWithMetadata.class);

    @Autowired
    private OKMDocument okmDocument;

    @Autowired
    private OKMFolder okmFolder;

    @Autowired
    private OKMRepository okmRepository;

    @Autowired
    private OKMPropertyGroup okmPropertyGroup;

    @Autowired
    private MailUtils mailUtils;

    @Autowired
    private FileUtils fileUtils;

    private String token = DbSessionManager.getInstance().getSystemToken();
    private List<Status> bad = new ArrayList<>();
    private List<Status> good = new ArrayList<>();
    private List<String> toAddress = new ArrayList<>();

    @Override
    public String getName() {
        return "Document Importer With Metadata";
    }

    @Override
    public String getCronExpression() {
```

```
    return "0 5 * * * *";
}

@Override
public void execute() {
    try {
        String msg = documentImporterWithMetadata();

        if (msg != null && !msg.isEmpty()) {
            // Sending mails
            toAddress.add("gdgsucre@mail.com");
            mailUtils.sendMessage(toAddress, "Importing report", msg);
            // Or send mail admin user
            mailUtils.sendCronTaskResult(getName(), msg);
        }
    } catch (Exception e) {
        log.error(e.getMessage());
    }
}

private String documentImporterWithMetadata()
        throws AccessDeniedException, PathNotFoundException, RepositoryException, DatabaseException,
        UnsupportedEncodingException, ItemExistsException, AutomationException, PromotedAsRecordLockExcep
    // Months
    List<String> month = new ArrayList<>();
    month.add("ENERO");
    month.add("FEBRERO");
    month.add("MARZO");
    month.add("ABRIL");
    month.add("MAYO");
    month.add("JUNIO");
    month.add("JULIO");
    month.add("AGOSTO");
    month.add("SEPTIEMBRE");
    month.add("OCTUBRE");
    month.add("NOVIEMBRE");
    month.add("DICIEMBRE");

    String grpName = "okg:contract";
    String contractUUID = "99bd0a50-fc98-409d-9b92-62528c42707f";
    String systemFolder = "/home/openkm/pending_to_import_folder";
    String basePath = okmRepository.getNodePath(token, contractUUID);
    Gson gson = new Gson();

    // Loading files
    File files = new File(systemFolder);
    File[] listOfFiles = files.listFiles();

    for (int i = 0; i < listOfFiles.length; i++) {
        File file = listOfFiles[i];
        String name = file.getName();
        String extension = fileUtils.getFileExtension(name);
        if (file.isFile() && extension.equals("pdf")) {
            Status status = new Status();
            status.fileName = name;
            if (status.fileName.indexOf("(") > 0 && status.fileName.indexOf("(") < status.fileName.indexOf(")")) {
                // Get metadata
                String metadata = status.fileName.substring((status.fileName.indexOf("(") + 1),
                        status.fileName.lastIndexOf(")"));

                // get data
                String[] data = metadata.split(",");
                if (data.length != 2) {
                    status.error = "Incorrect format -> name (cups,REN)<br>";
                } else {
                    status.cups = data[0].toUpperCase().replaceAll(" ", "");
                    Calendar calendar = Calendar.getInstance();
```

```
calendar.setTimeInMillis(file.lastModified());
status.year = String.valueOf(calendar.get(Calendar.YEAR));
status.month = month.get(calendar.get(Calendar.MONTH));

if (!data[1].toUpperCase().equals("REN")) {
    status.error = "Error type REN not found<br>";
}

if (status.error.equals("")) {
    // test if folder year exists otherside create it
    String path = basePath + "/" + status.year;

    if (!okmRepository.hasNode(token, path)) {
        okmFolder.createSimple(token, path);
    }

    // test if folder month exists otherside create it
    path = path + "/" + status.month;

    if (!okmRepository.hasNode(token, path)) {
        okmFolder.createSimple(token, path);
    }

    // Create document and adding metadata
    try {
        // Removing extra ( contents )
        String fileName = status.fileName.substring(0, status.fileName.indexOf("("));

        // Removing - and spaces at ends
        while (fileName.substring(fileName.length() - 1).equals(" ")
                || fileName.substring(fileName.length() - 1).equals("-")) {
            fileName = fileName.substring(0, fileName.length() - 1);
        }

        fileName = fileName + ".pdf";
        path = path + "/" + fileName;
        long length = file.length();
        Thread.sleep(1000);
        if (file.length() > length) {
            continue; // Skip file this time
        }
        FileInputStream fis = new FileInputStream(file);
        Document doc = okmDocument.createSimple(token, path, fis);
        status.dstPath = path;
        fis.close();

        // Create new metadata
        Map<String, String> properties = new HashMap<>();
        properties.put("okp:contract.cups", status.cups);
        properties.put("okp:contract.year", status.year);
        properties.put("okp:contract.month", gson.toJson(new String[]{status.month.toLowerCase()}));

        okmPropertyGroup.addGroup(token, doc.getPath(), grpName, properties);

        // Delete file
        boolean success = file.delete();
        if (!success) {
            status.error += "File can not been deleted";
        }
    } catch (PathNotFoundException e) {
        status.error += "PathNotFoundException";
    } catch (ItemExistsException e) {
        status.error += "ItemExistsException";
    } catch (UnsupportedMimeTypeException e) {
        status.error += "UnsupportedMimeTypeException";
    } catch (VirusDetectedException e) {
        status.error += "VirusDetectedException";
```

```
                    } catch (RepositoryException e) {
                        status.error += "RepositoryException";
                    } catch (DatabaseException e) {
                        status.error += "DatabaseException";
                    } catch (ExtensionException e) {
                        status.error += "ExtensionException";
                    } catch (IOException e) {
                        status.error += "IOException";
                    } catch (Exception e) {
                        status.error += e.getMessage();
                    }
                }
            }
        } else {
            status.error = "Document format incorrect -> nombre - (cups,REN)<br>";
        }

        if (!status.error.equals("")) {
            bad.add(status);
        } else {
            good.add(status);
        }
    }
}

StringBuffer result = new StringBuffer();
result.append("<h1>Import report</h1>");
result.append("</br></br>");
result.append("<table boder=\"0\" cellpadding=\"2\" cellspacing=\"0\" width=\"100%\">");
result.append("<tr>");
result.append("<td bgcolor=\"silver\"><b>Name</b></td>");
result.append("<td bgcolor=\"silver\"><b>CUPS</b></td>");
result.append("<td bgcolor=\"silver\"><b>Año</b></td>");
result.append("<td bgcolor=\"silver\"><b>Mes</b></td>");
result.append("<td bgcolor=\"silver\"><b>Error</b></td>");
result.append("</tr>");
result.append("<tr>");
result.append("<td colspan=\"6\" bgcolor=\"silver\"><b>Errors:" + bad.size() + "</b></td>");
result.append("</tr>");

for (Status status : bad) {
    result.append("<tr>");
    result.append("<td>" + status.fileName + "</td>");
    result.append("<td>" + status.cups + "</td>");
    result.append("<td>" + status.year + "</td>");
    result.append("<td>" + status.month + "</td>");
    result.append("<td><font color=\"red\">" + status.error + "</font></td>");
    result.append("</tr>");
}

result.append("</table>");
result.append("</br></br>");
result.append("<table boder=\"0\" cellpadding=\"2\" cellspacing=\"0\" width=\"100%\">");
result.append("<tr>");
result.append("<td bgcolor=\"silver\"><b>Name</b></td>");
result.append("<td bgcolor=\"silver\"><b>CUPS</b></td>");
result.append("<td bgcolor=\"silver\"><b>Year</b></td>");
result.append("<td bgcolor=\"silver\"><b>Month</b></td>");
result.append("<td bgcolor=\"silver\"><b>Destination</b></td>");
result.append("</tr>");
result.append("<tr>");
result.append("<td colspan=\"6\" bgcolor=\"silver\"><b>Imported correctly:" + good.size() + "</b></td>");
result.append("</tr>");

for (Status status : good) {
    result.append("<tr>");
    result.append("<td>" + status.fileName + "</td>");
```

```
            result.append("<td>" + status.cups + "</td>");
            result.append("<td>" + status.year + "</td>");
            result.append("<td>" + status.month + "</td>");
            result.append("<td><a href=\"" + Config.APPLICATION_URL + "?docPath=" + URLEncoder.encode(status.d
                    + "\">" + status.dstPath + "</font></td>");
            result.append("</tr>");
        }

        result.append("</table>");

        return result.toString();
    }

    private class Status {

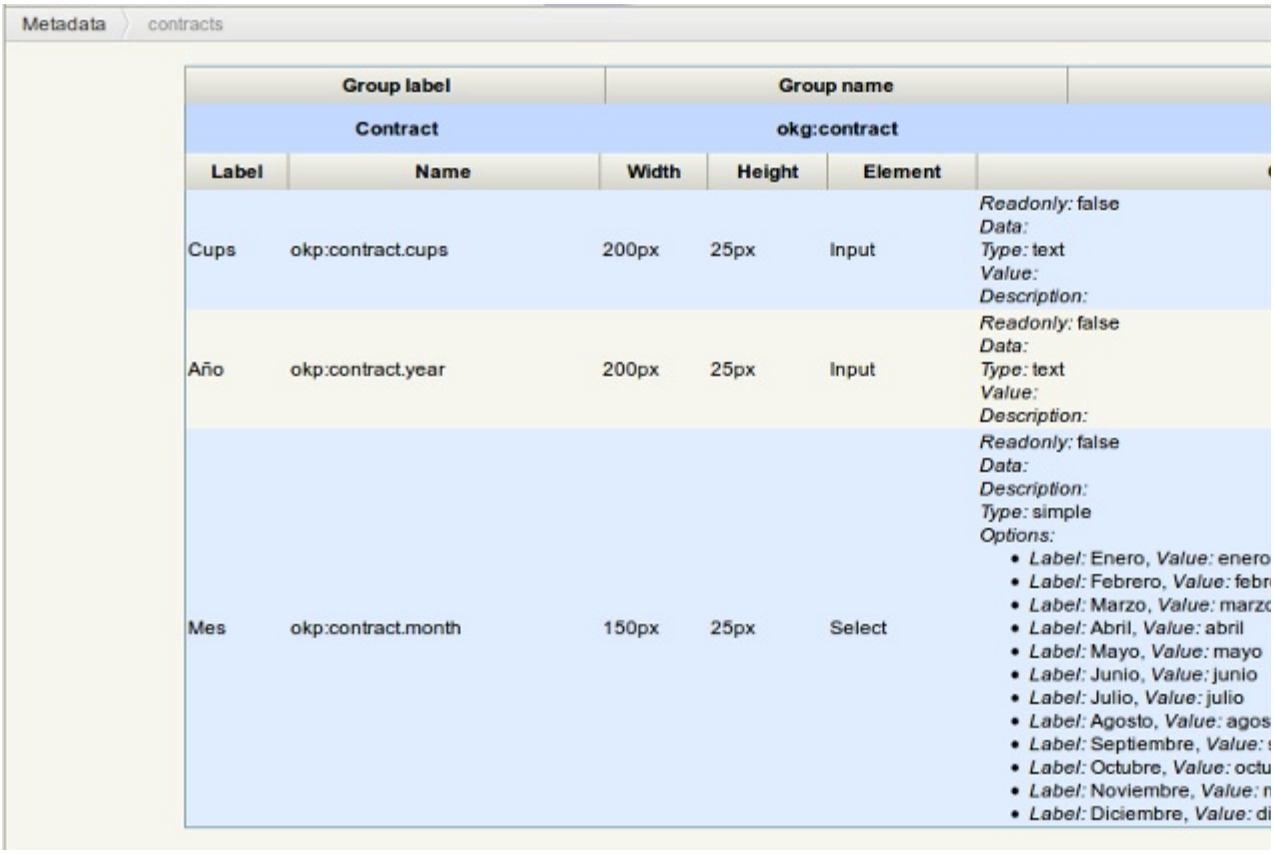
        public String fileName = "";
        public String dstPath = "";
        public String cups = "";
        public String month = "";
        public String year = "";
        public String error = "";
    }
}
```

Register a new plugin

- *To install new plugin **create a jar file** and **copy into your $TOMCAT/plugins** folder. More information at [Creating a JAR file with Eclipse](#).*
- *Go to **Administration** > **Utilities** > **Plugins** and click at top right the*  *Reload plugins button.*
- *In **plugins table will be shown** the new plugin.*

Images

Register metadata definition

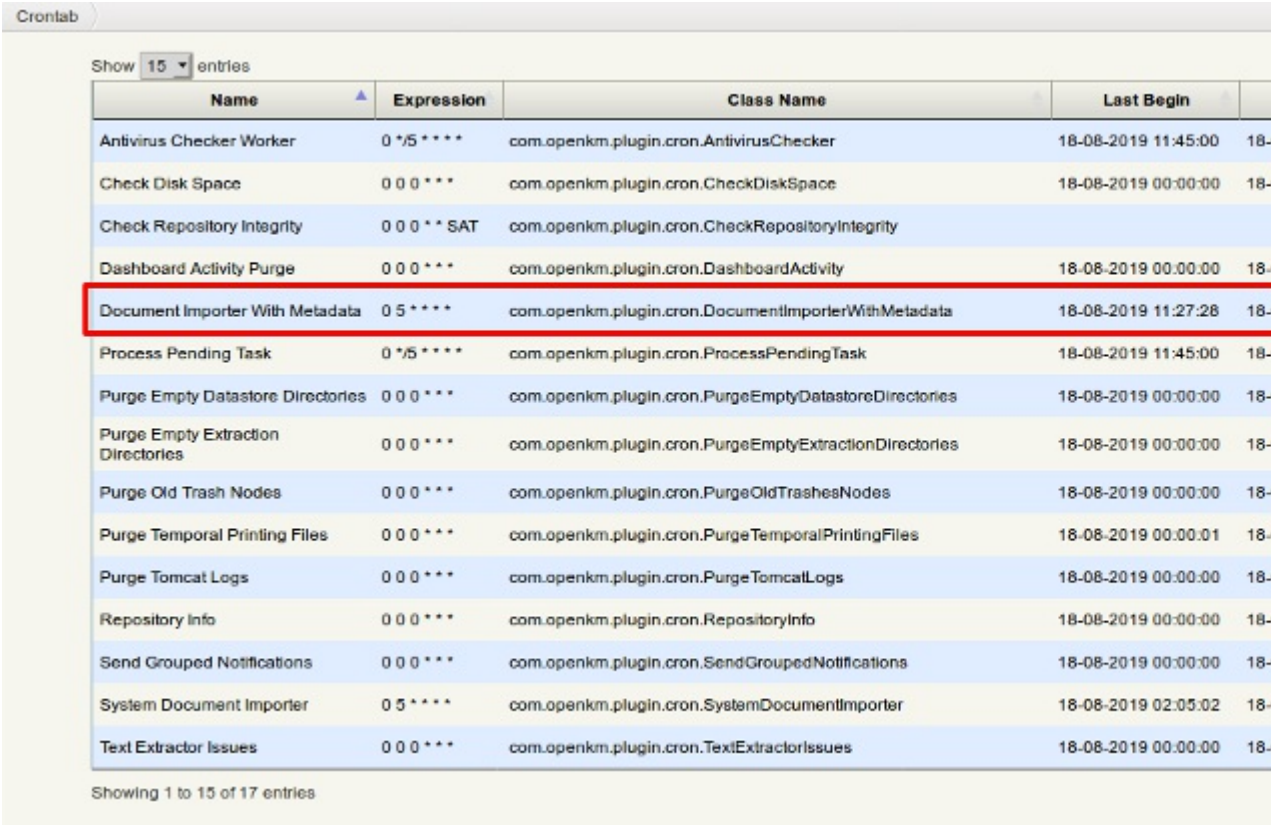


Metadata contracts

Group label		Group name			
Contract		okg:contract			
Label	**Name**	**Width**	**Height**	**Element**	
Cups	okp:contract.cups	200px	25px	Input	*Readonly:* false *Data:* *Type:* text *Value:* *Description:*
Año	okp:contract.year	200px	25px	Input	*Readonly:* false *Data:* *Type:* text *Value:* *Description:*
Mes	okp:contract.month	150px	25px	Select	*Readonly:* false *Data:* *Description:* *Type:* simple *Options:* • *Label:* Enero, *Value:* enero • *Label:* Febrero, *Value:* febr • *Label:* Marzo, *Value:* marzo • *Label:* Abril, *Value:* abril • *Label:* Mayo, *Value:* mayo • *Label:* Junio, *Value:* junio • *Label:* Julio, *Value:* julio • *Label:* Agosto, *Value:* agos • *Label:* Septiembre, *Value:* • *Label:* Octubre, *Value:* octu • *Label:* Noviembre, *Value:* • *Label:* Diciembre, *Value:* di

Crontab task

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

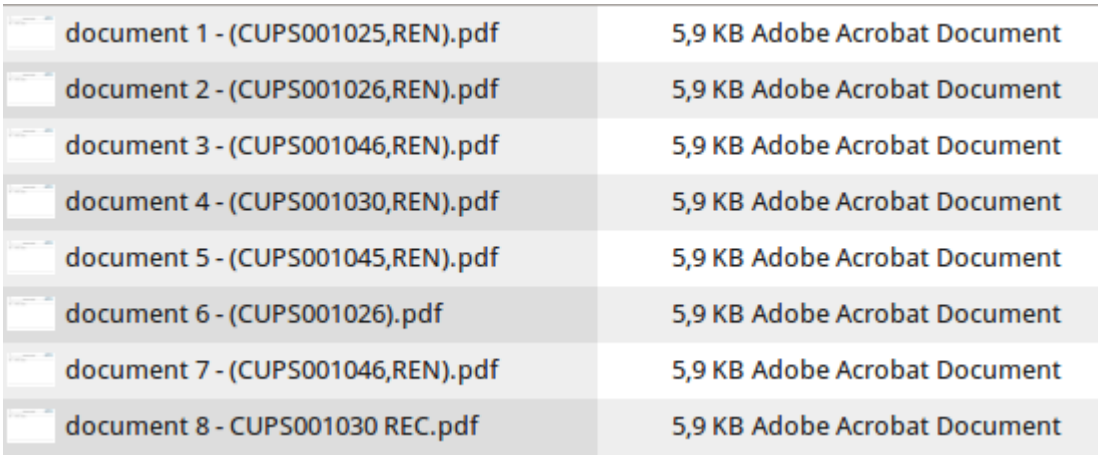

Crontab

Show 15 entries

Name	Expression	Class Name	Last Begin	
Antivirus Checker Worker	0 */5 * * * *	com.openkm.plugin.cron.AntivirusChecker	18-08-2019 11:45:00	18-
Check Disk Space	0 0 0 * * *	com.openkm.plugin.cron.CheckDiskSpace	18-08-2019 00:00:00	18-
Check Repository Integrity	0 0 0 * * SAT	com.openkm.plugin.cron.CheckRepositoryIntegrity		
Dashboard Activity Purge	0 0 0 * * *	com.openkm.plugin.cron.DashboardActivity	18-08-2019 00:00:00	18-
Document Importer With Metadata	0 5 * * * *	com.openkm.plugin.cron.DocumentImporterWithMetadata	18-08-2019 11:27:28	18-
Process Pending Task	0 */5 * * * *	com.openkm.plugin.cron.ProcessPendingTask	18-08-2019 11:45:00	18-
Purge Empty Datastore Directories	0 0 0 * * *	com.openkm.plugin.cron.PurgeEmptyDatastoreDirectories	18-08-2019 00:00:00	18-
Purge Empty Extraction Directories	0 0 0 * * *	com.openkm.plugin.cron.PurgeEmptyExtractionDirectories	18-08-2019 00:00:00	18-
Purge Old Trash Nodes	0 0 0 * * *	com.openkm.plugin.cron.PurgeOldTrashesNodes	18-08-2019 00:00:00	18-
Purge Temporal Printing Files	0 0 0 * * *	com.openkm.plugin.cron.PurgeTemporalPrintingFiles	18-08-2019 00:00:01	18-
Purge Tomcat Logs	0 0 0 * * *	com.openkm.plugin.cron.PurgeTomcatLogs	18-08-2019 00:00:00	18-
Repository Info	0 0 0 * * *	com.openkm.plugin.cron.RepositoryInfo	18-08-2019 00:00:00	18-
Send Grouped Notifications	0 0 0 * * *	com.openkm.plugin.cron.SendGroupedNotifications	18-08-2019 00:00:00	18-
System Document Importer	0 5 * * * *	com.openkm.plugin.cron.SystemDocumentImporter	18-08-2019 02:05:02	18-
Text Extractor Issues	0 0 0 * * *	com.openkm.plugin.cron.TextExtractorIssues	18-08-2019 00:00:00	18-

Showing 1 to 15 of 17 entries

Files in application server



Result

Mail notification

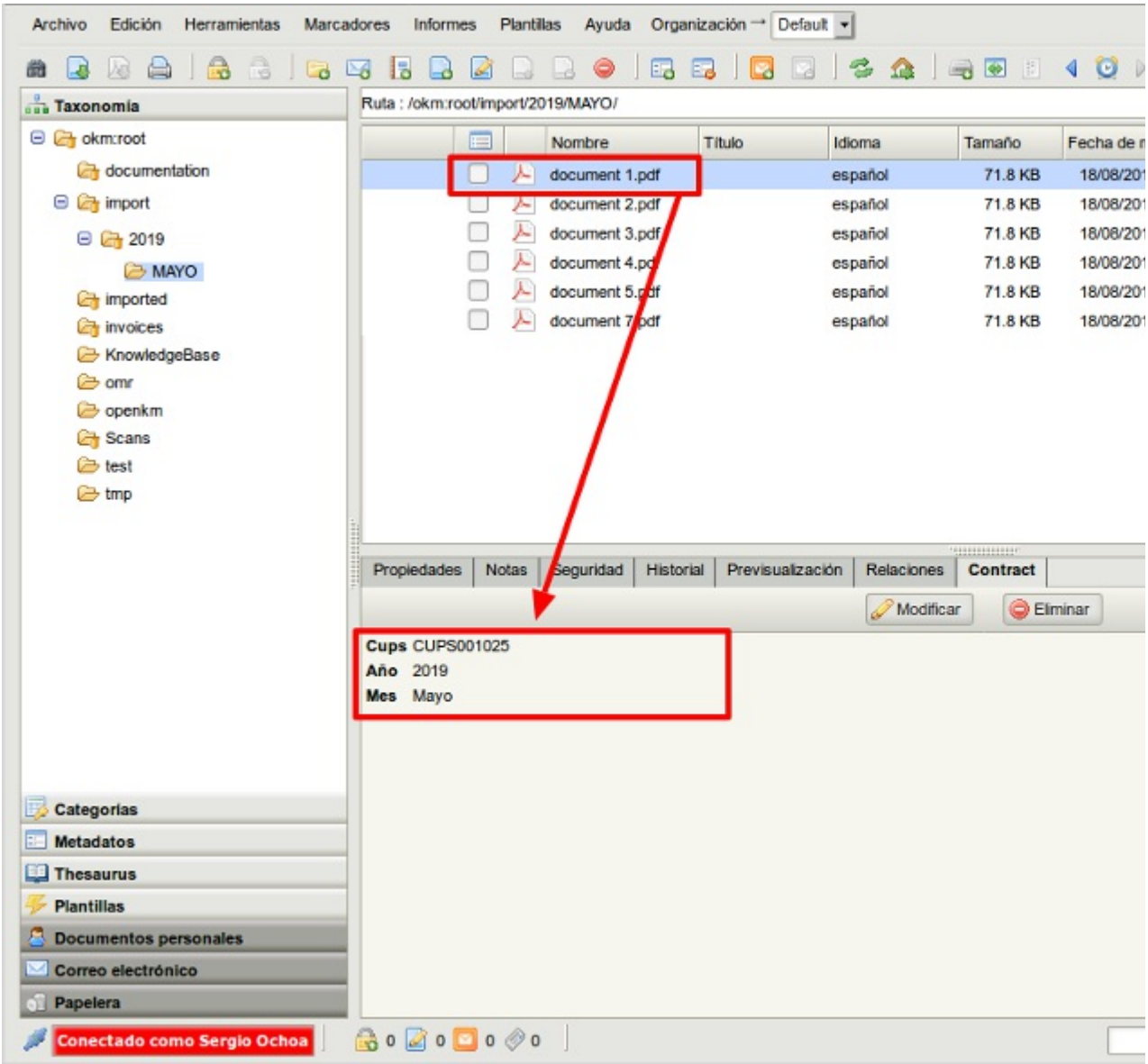
Import report

Name	CUPS	Año	Mes	Error
Errors:2				
document 8 - (CUPS001030,REC).pdf	CUPS001030	2019	MAYO	Error type RE
document 6 - (CUPS001026).pdf				Incorrect for

Name	CUPS	Year	Month	Destination
Imported correctly:6				
document 3 - (CUPS001046,REN).pdf	CUPS001046	2019	MAYO	/okm:root/import/201
document 2 - (CUPS001026,REN).pdf	CUPS001026	2019	MAYO	/okm:root/import/201
document 7 - (CUPS001046,REN).pdf	CUPS001046	2019	MAYO	/okm:root/import/201
document 4 - (CUPS001030,REN).pdf	CUPS001030	2019	MAYO	/okm:root/import/201
document 5 - (CUPS001045,REN).pdf	CUPS001045	2019	MAYO	/okm:root/import/201
document 1 - (CUPS001025,REN).pdf	CUPS001025	2019	MAYO	/okm:root/import/201

Imported files into the repository

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


Archivo
Edición
Herramientas
Marcadores
Informes
Plantillas
Ayuda
Organización
Default
Taxonomía
okm:root
documentation
import
2019
MAYO
imported
invoices
KnowledgeBase
omr
openkm
Scans
test
tmp
Ruta : /okm:root/import/2019/MAYO/
Nombre
Título
Idioma
Tamaño
document 1.pdf
document 2.pdf
document 3.pdf
document 4.pdf
document 5.pdf
document 7.pdf
español
71.8 KB
Propiedades
Notas
Seguridad
Historial
Previsualización
Relaciones
Contract
Modificar
Eliminar
Cups CUPS001025
Año 2019
Mes Mayo
Categorías
Metadatos
Thesaurus
Plantillas
Documentos personales
Correo electrónico
Papelera
Conectado como Sergio Ochoa

Contab sample - Generate HTML report

This examples generates an HTML report from the OpenKM activity log:

```
package com.openkm.plugin.cron;

import com.openkm.db.service.LegacySrv;
import com.openkm.util.MailUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.io.FileUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.io.File;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.List;

@PluginImplementation
public class SampleHtmlReport extends BaseCronPlugin implements CronAdapter {
 private static final Logger log = LoggerFactory.getLogger(SampleHtmlReport.class);

 @Autowired
 private LegacySrv legacySrv;

 @Override
 public String getName() {
  return "Sample HTML report";
 }

 @Override
 public String getCronExpression() {
  return "0 0 0 * * *";
 }

 @Override
 public void execute() {
  try {
   Calendar to = Calendar.getInstance();
   Calendar from = (Calendar) to.clone();
   from.add(Calendar.DAY_OF_YEAR, -7);
   DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
   String query = String.format("select ACT_DATE, ACT_USER, ACT_ACTION, ACT_ITEM, ACT_PATH from OKM_
     + "where ACT_DATE between '%s' and '%s'", df.format(from.getTime()), df.format(to.getTime()));

   StringBuilder result = new StringBuilder();
   result.append("<h1>Sample report</h1>");
   result.append("</br></br>");
   result.append("<table boder=\"0\" cellpadding=\"2\" cellspacing=\"0\" width=\"100%\">");
   result.append("<tr>");
   result.append("<th bgcolor=\"silver\"><b>Date</b></td>");
   result.append("<th bgcolor=\"silver\"><b>User</b></td>");
   result.append("<th bgcolor=\"silver\"><b>Action</b></td>");
   result.append("<th bgcolor=\"silver\"><b>Item</b></td>");
   result.append("<th bgcolor=\"silver\"><b>Path</b></td>");
   result.append("</tr>");

   for (List<String> row : legacySrv.executeSQL(query)) {
```

```
    result.append("<tr>");
    result.append("<td>").append(row.get(0)).append("</td>");
    result.append("<td>").append(row.get(1)).append("</td>");
    result.append("<td>").append(row.get(2)).append("</td>");
    result.append("<td>").append(row.get(3)).append("</td>");
    result.append("<td>").append(row.get(4)).append("</td>");
    result.append("</tr>");
   }

   result.append("</table>");

   DateFormat dfHour = new SimpleDateFormat("yyyy-MM-dd_HHmmss");
   File output = new File("report-" + dfHour.format(to.getTime()) + ".html");
   FileUtils.writeStringToFile(output, result.toString(), StandardCharsets.UTF_8);
   log.info("Report written to {}", output.getAbsolutePath());
  } catch (Exception e) {
   log.error(e.getMessage(), e);
  }
 }
}
```

Register a new plugin

- *To install new plugin **create a jar file** and **copy into your $TOMCAT/plugins** folder. More information at [Creating a JAR file with Eclipse](#).*
- *Go to **Administration** > **Utilities** > **Plugins** and click at top right the Reload plugins button.*
- *In **plugins table will be shown** the new plugin.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Crontab task - XML importer

Import files with metadata values from XML files.

This is a crontab task base in Java class packaged in a JAR.

- *The files to be imported are stored at **/home/openkm/import/Output** OpenKM server. (Variable **SYSTEM_FOLDER**+Output).*
- *The XML files with metadata values are stored at **/home/openkm/import/logfile** OpenKM server. (Variable **SYSTEM_FOLDER**+logfile).*
- *The files are imported to the OpenKM folder "**/okm:root/import**". (Variable **OPENKM_FOLDER**).*

The crontab does three actions:

- *First, import all files from "**/home/openkm/import/Output**" and delete them from the server after has being imported.*
- *Then load and parse XML files located at "**/home/openkm/import/logfile**" and set the metadata group values of the documents.*
- *Finally, the XML files are deleted.*

Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
   <property-group label="Data" name="okg:xmldata">
      <input label="Folio" name="okp:xmldata.folio" type="text"/>
      <input label="Rut" name="okp:data.rut" type="text"/>
      <input label="Placa patente" name="okp:xmldata.placa" type="text"/>
      <input label="Nombre" name="okp:xmldata.nombres" type="text"/>
      <input label="Apellido parterno" name="okp:xmldata.apellido_padre" type="text"/>
      <input label="Apellido materno" name="okp:xmldata.apellido_madre" type="text"/>
      <input label="Año contable" name="okp:xmldata.year" type="text"/>
      <input label="Tipo documento" name="okp:xmldata.documento" type="text"/>
   </property-group>
</property-groups>
```

XML files structure:

```
<?xml version="1.0" ?>
<INDEX_LOG>
   <Batch BatchID="2013-04-16">
      <File>
         <Filename>10201200365212 - 01.pdf</Filename>
         <Fields>
            <Field>
               <Name>Folio</Name>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
            <Value>10201200365245</Value>
         </Field>
         <Field>
            <Name>Rut</Name>
            <Value>9419475-JK</Value>
         </Field>
         <Field>
            <Name>Placa Patente</Name>
            <Value>XG412190</Value>
         </Field>
         <Field>
            <Name>Nombres</Name>
            <Value>JOSEP</Value>
         </Field>
         <Field>
            <Name>Apellido Paterno</Name>
            <Value>LLORT</Value>
         </Field>
         <Field>
            <Name>Apellido Materno</Name>
            <Value>TELLA</Value>
         </Field>
         <Field>
            <Name>Año</Name>
            <Value>2012</Value>
         </Field>
         <Field>
            <Name>Tipo de Documento</Name>
            <Value>Comprobante de Permiso</Value>
         </Field>
      </Fields>
   </File>
   <!-- etc. -->   </Batch>
</INDEX_LOG>
```

JAVA class:

```
package com.openkm.plugin.cron;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.openkm.api.OKMDocument;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.bean.PropertyGroup;
import com.openkm.bean.QueryResult;
```

```
import com.openkm.core.AccessDeniedException;
import com.openkm.core.DatabaseException;
import com.openkm.core.ItemExistsException;
import com.openkm.core.ParseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.PromotedAsRecordLockException;
import com.openkm.core.RepositoryException;
import com.openkm.core.UnsupportedMimeTypeException;
import com.openkm.core.ValidationFormException;
import com.openkm.core.VirusDetectedException;
import com.openkm.db.bean.QueryParams;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.plugin.PluginNotFoundException;
import com.openkm.plugin.automation.AutomationException;
import com.openkm.principal.PrincipalAdapterException;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class XmlImporter extends BaseCronPlugin implements CronAdapter {
    private Logger log = LoggerFactory.getLogger(XmlImporter.class);
    private static final String SYSTEM_FOLDER = "/home/openkm/import";
    private static final String OPENKM_FOLDER = "/okm:root/import";
    private static final String ATTRIBUTE_FILENAME = "Filename";
    private static final String ATTRIBUTE_FIELDS = "Fields";
    private static final String ATTRIBUTE_FIELD = "Field";
    private static final String ATTRIBUTE_NAME = "Name";
    private static final String ATTRIBUTE_VALUE = "Value";
    private static final String FIELD_NAME_FOLIO = "folio";
    private static final String FIELD_NAME_RUT = "rut";
    private static final String FIELD_NAME_PLACA_PATENTE = "placa patente";
    private static final String FIELD_NAME_NOMBRES = "nombres";
    private static final String FIELD_NAME_APELLIDO_PATERNO = "apellido paterno";
    private static final String FIELD_NAME_APELLIDO_MATERNO = "apellido materno";
    private static final String FIELD_NAME_ANO = "año";
    private static final String FIELD_NAME_TIPO_DOCUMENTO = "tipo de documento";

    @Autowired
    private OKMDocument okmDocument;

    @Autowired
    private OKMSearch okmSearch;

    @Autowired
    private OKMPropertyGroup okmPropertyGroup;

    private String token = DbSessionManager.getInstance().getSystemToken();

    @Override
    public String getName() {
        return "Xml Importer";
    }

    @Override
    public String getCronExpression() {
        return "0 5 * * * *";
    }

    @Override
    public void execute() {
        try {
            importFiles();
            importMetadata();
        } catch (Exception e) {
            log.error(e.getMessage());
            e.getStackTrace();
        }
```

```
}

/**
 * importMetadata
 */
public void importMetadata() throws PathNotFoundException, AccessDeniedException, RepositoryException, Da
        AutomationException, PluginNotFoundException, ValidationFormException, ParseException, IOException,
        ParserConfigurationException, SAXException {
    String fileName = "";
    String folio = "";
    String rut = "";
    String placaPatente = "";
    String nombres = "";
    String apellidoPaterno = "";
    String apellidoMaterno = "";
    String ano = "";
    String tipoDocumento = "";

    File folder = new File(SYSTEM_FOLDER + "/logfile");
    File[] listOfFiles = folder.listFiles();

    for (int i = 0; i < listOfFiles.length; i++) {
        File xmlFile = listOfFiles[i];
        if (xmlFile.isFile() && xmlFile.getName().toLowerCase().endsWith("indexlog.xml")) {
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            dbf.setNamespaceAware(true);
            dbf.setAttribute("http://xml.org/sax/features/namespaces", Boolean.TRUE);

            DocumentBuilder db = dbf.newDocumentBuilder();
            org.w3c.dom.Document xmlDoc = db.parse(xmlFile);
            xmlDoc.getDocumentElement().normalize();
            NodeList fileNodesList = xmlDoc.getElementsByTagName("File");

            for (int x = 0; x < fileNodesList.getLength(); x++) {
                fileName = "";
                folio = "";
                rut = "";
                placaPatente = "";
                nombres = "";
                apellidoPaterno = "";
                apellidoMaterno = "";
                ano = "";
                tipoDocumento = "";
                Node childFildNode = fileNodesList.item(x);

                if (childFildNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element fileElement = (Element) childFildNode; // attibute node
                    fileName = fileElement.getElementsByTagName(ATTRIBUTE_FILENAME).item(0).getTextContent()
                    Node fieldsNode = fileElement.getElementsByTagName(ATTRIBUTE_FIELDS).item(0); // Only one

                    // Fields
                    if (fieldsNode.getNodeType() == Node.ELEMENT_NODE) {
                        Element fieldsElement = (Element) fieldsNode; // fields

                        // node
                        NodeList fieldList = fieldsElement.getElementsByTagName(ATTRIBUTE_FIELD);
                        for (int y = 0; y < fieldList.getLength(); y++) {
                            Node fieldNode = fieldList.item(y);
                            if (fieldNode.getNodeType() == Node.ELEMENT_NODE) {
                                Element fieldElement = (Element) fieldNode; // attibute

                                // node
                                String name = fieldElement.getElementsByTagName(ATTRIBUTE_NAME).item(0).getTextCo
                                        .trim();
                                String value = fieldElement.getElementsByTagName(ATTRIBUTE_VALUE).item(0).getTextC
                                        .trim();
```

```
                    if (name.toLowerCase().equals(FIELD_NAME_FOLIO)) {
                        folio = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_RUT)) {
                        rut = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_PLACA_PATENTE)) {
                        placaPatente = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_NOMBRES)) {
                        nombres = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_APELLIDO_PATERNO)) {
                        apellidoPaterno = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_APELLIDO_MATERNO)) {
                        apellidoMaterno = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_ANO)) {
                        ano = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_TIPO_DOCUMENTO)) {
                        tipoDocumento = value;
                    }
                }
            }
        }

        if (fileName != null && !fileName.equals("")) {
            QueryParams queryParams = new QueryParams();
            queryParams.setDomain(QueryParams.DOCUMENT);
            queryParams.setName(fileName);
            List<QueryResult> results = okmSearch.find(token, queryParams);

            if (results.size() == 1) {
                for (QueryResult queryResult : results) {
                    if (queryResult.getNode() != null && queryResult.getNode() instanceof Document) {
                        log.info(fileName + " -ok");
                        boolean found = false;

                        for (PropertyGroup group : okmPropertyGroup.getGroups(token,
                                queryResult.getNode().getPath())) {
                            if (group.getName().equals("okg:xmldata")) {
                                found = true;
                            }
                        }

                        Map<String, String> properties = new HashMap<String, String>();
                        properties.put("okp:xmldata.folio", folio);
                        properties.put("okp:xmldata.rut", rut);
                        properties.put("okp:xmldata.placa", placaPatente);
                        properties.put("okp:xmldata.nombres", nombres);
                        properties.put("okp:xmldata.apellido_padre", apellidoPaterno);
                        properties.put("okp:xmldata.apellido_madre", apellidoMaterno);
                        properties.put("okp:xmldata.year", ano);
                        properties.put("okp:xmldata.documento", tipoDocumento);
                        if (!found) {
                            okmPropertyGroup.addGroup(token, queryResult.getNode().getPath(), "okg:xmldata",
                                    properties);
                        } else {
                            okmPropertyGroup.setProperties(token, queryResult.getNode().getPath(), "okg:xmldata
                                    properties);
                        }
                    }
                }
            } else if (results.size() > 1) {
                log.error(fileName + " - error");
            } else {
                log.error(fileName + " - not found");
            }
        }
    }
}
```

```
                xmlFile.delete();
            }
        }
    }

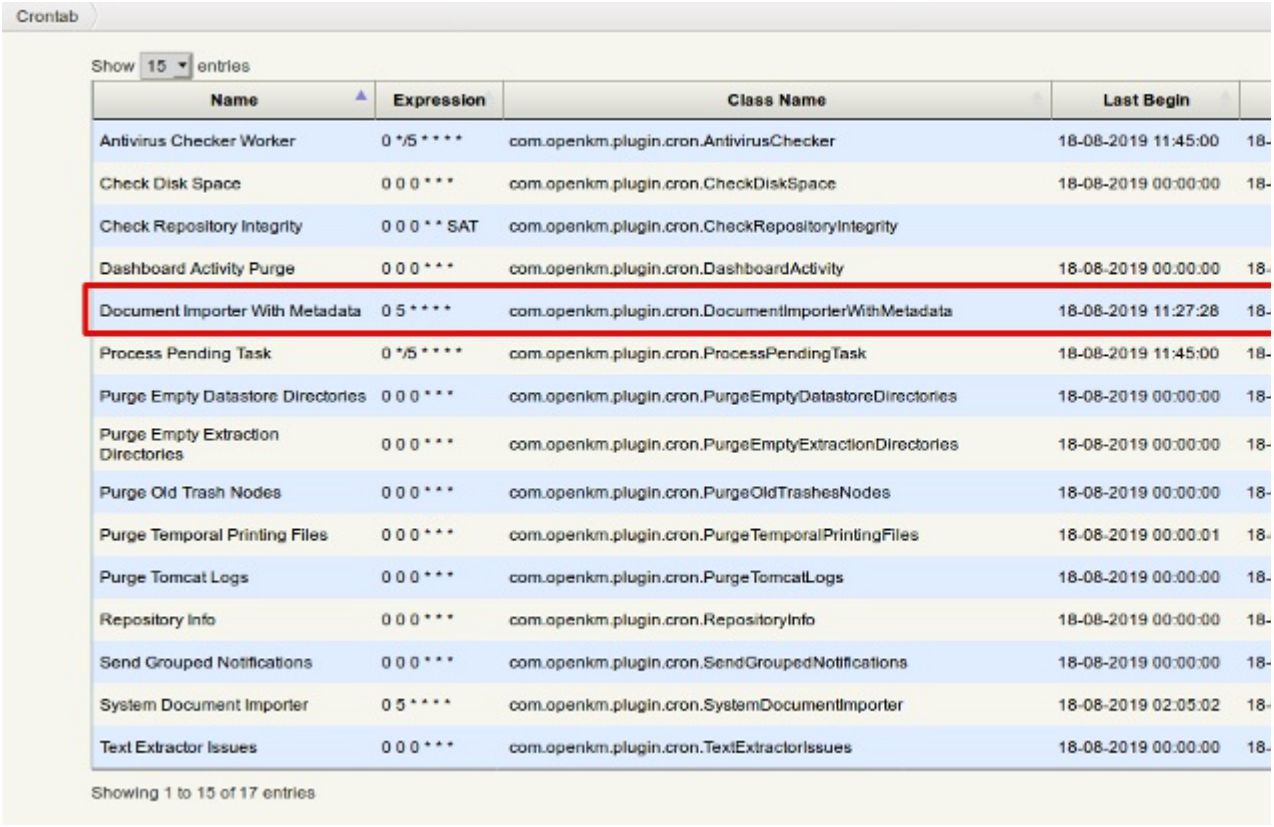
    /**
     * importFiles
     */
    public void importFiles() throws UnsupportedMimeTypeException, VirusDetectedException, ItemExistsException,
            PathNotFoundException, AccessDeniedException, RepositoryException, DatabaseException, AutomationEx
            PromotedAsRecordLockException, PrincipalAdapterException, IOException {
        // Loading files
        File folder = new File(SYSTEM_FOLDER + "/Output");
        File[] listOfFiles = folder.listFiles();
        for (int i = 0; i < listOfFiles.length; i++) {
            File file = listOfFiles[i];
            if (file.isFile() && file.getName().toLowerCase().endsWith(".pdf")) {
                Document doc = new Document();
                doc.setPath(OPENKM_FOLDER + "/" + file.getName());
                FileInputStream fis = new FileInputStream(file);
                doc = okmDocument.create(token, doc, fis);
                file.delete();
            }
        }
    }
}
```

Register a new plugin

- *To install the new plugin,* ***create a jar file*** *and* ***copy it into your $TOMCAT/plugins*** *folder—more information at [Creating a JAR file with Eclipse](#).*
- *Go to* ***Administration*** *>* ***Utilities*** *>* ***Plugins*** *and click at the top right the*  *Reload plugins button.*
- *In* ***plugins, the table will be shown*** *the new plugin.*

Images

Crontab task

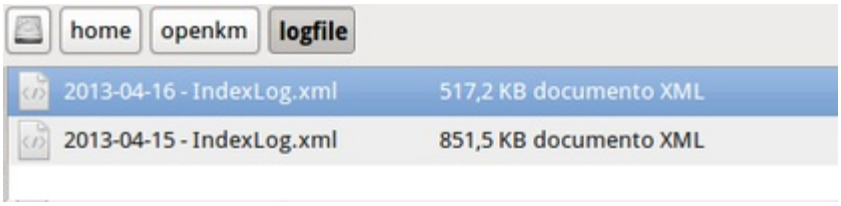


Register metadata definition

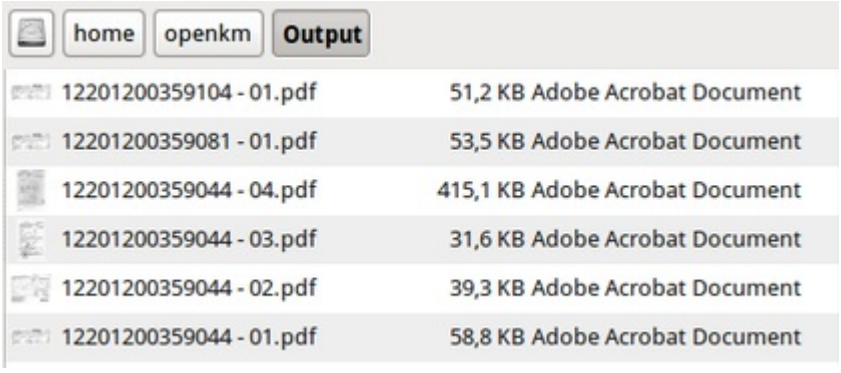
Metadata contracts

Group label		Group name			
Contract		okg:contract			
Label	**Name**	**Width**	**Height**	**Element**	
Cups	okp:contract.cups	200px	25px	Input	*Readonly:* false *Data:* *Type:* text *Value:* *Description:*
Año	okp:contract.year	200px	25px	Input	*Readonly:* false *Data:* *Type:* text *Value:* *Description:*
Mes	okp:contract.month	150px	25px	Select	*Readonly:* false *Data:* *Description:* *Type:* simple *Options:* • *Label:* Enero, *Value:* enero • *Label:* Febrero, *Value:* febr • *Label:* Marzo, *Value:* marzo • *Label:* Abril, *Value:* abril • *Label:* Mayo, *Value:* mayo • *Label:* Junio, *Value:* junio • *Label:* Julio, *Value:* julio • *Label:* Agosto, *Value:* agos • *Label:* Septiembre, *Value:* • *Label:* Octubre, *Value:* octu • *Label:* Noviembre, *Value:* • *Label:* Diciembre, *Value:* di

Logfile folder

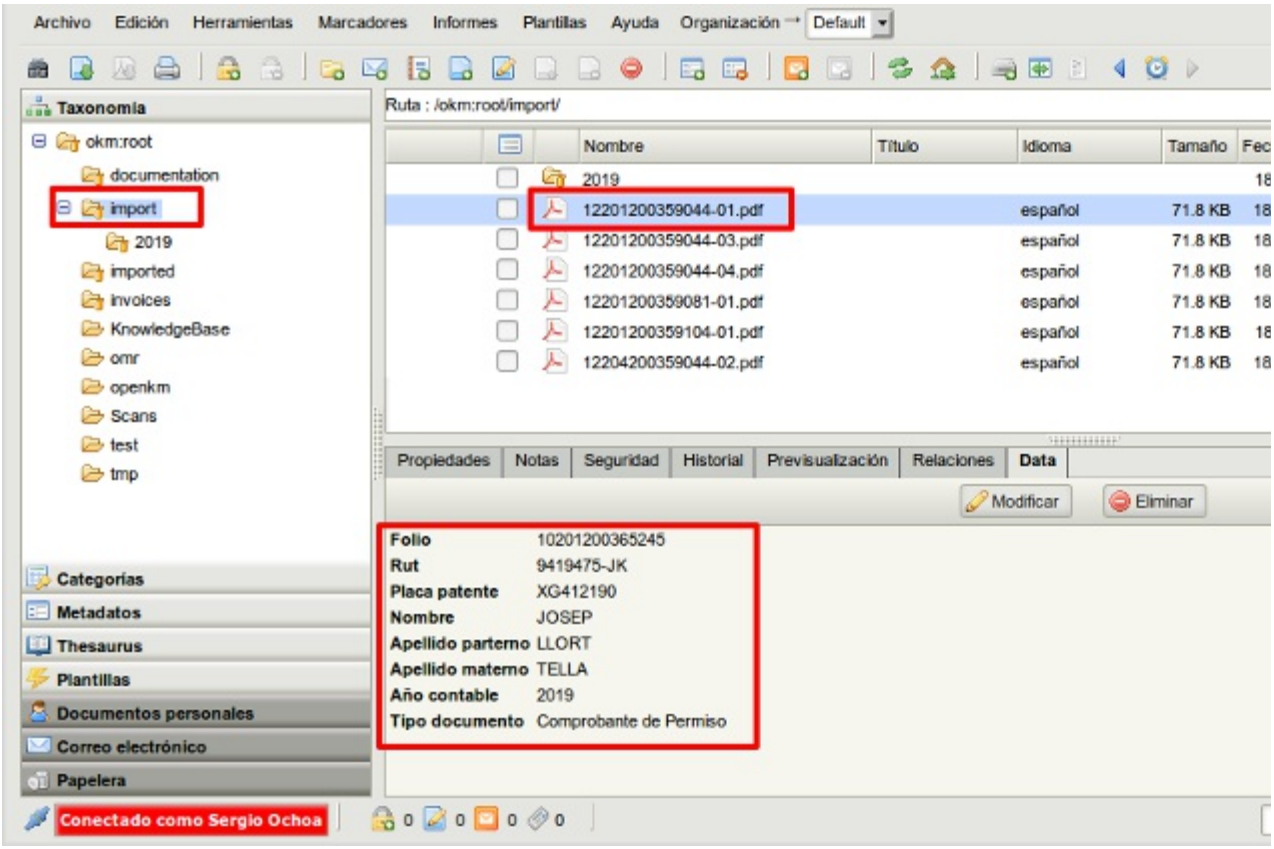


Output folder



Result

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


Archivo
Edición
Herramientas
Marcadores
Informes
Plantillas
Ayuda
Organización
Default
Taxonomia
okm:root
documentation
import
2019
imported
invoices
KnowledgeBase
omr
openkm
Scans
test
tmp
Categorías
Metadatos
Thesaurus
Plantillas
Documentos personales
Correo electrónico
Papelera
Conectado como Sergio Ochoa
Ruta : /okm:root/import/
Nombre
Título
Idioma
Tamaño
2019
12201200359044-01.pdf
12201200359044-03.pdf
12201200359044-04.pdf
12201200359081-01.pdf
12201200359104-01.pdf
12204200359044-02.pdf
español
71.8 KB
Propiedades
Notas
Seguridad
Historial
Previsualización
Relaciones
Data
Modificar
Eliminar
Folio 10201200365245
Rut 9419475-JK
Placa patente XG412190
Nombre JOSEP
Apellido parterno LLORT
Apellido materno TELLA
Año contable 2019
Tipo documento Comprobante de Permiso

Variables by automation event

Events can be pre (previously) and post (posterior). For example ***pre*** *document creation event, will be executed before document creation and then can be changed several event vars to change parent folder, document name among others. In case* ***post*** *document creation event, will be executed after document creation. In this case there're no associated vars to the event you'll get NodeDocument object to apply directly any change across OpenKM API.*

Available environment variables

Variable	*Object class*	*Description*
AutomationUtils.TENANT	*long*	*Tenant*
AutomationUtils.CATEGORIES	*Set<String>*	*List of categories*
AutomationUtils.CREATION_DATE	*Calendar*	*Creation date*
AutomationUtils.DESTINATION_NODE	*NodeFolder*	*Destination folder*
AutomationUtils.DOCUMENT_NODE	*NodeDocument*	*Document node*
AutomationUtils.DOCUMENT_FILE	*File*	*Document File*
AutomationUtils.DOCUMENT_NAME	*String*	*Document name*
AutomationUtils.DOCUMENT_VERSION	*String*	*Document version*
AutomationUtils.EVENT	*AutomationRule.EnumEvents*	*Event*
AutomationUtils.EVENT_AT	*String*	*Event at*
AutomationUtils.FOLDER_NODE	*NodeFolder*	*Folder node*
AutomationUtils.KEYWORDS	*Set<String>*	*List of keywords*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AutomationUtils.NODE_BASE	*NodeBase* • *NodeDocument* • *NodeFolder* • *NodeMail* • *NodeRecord* *These nodes extends from NodeBase.*	*Node base*
AutomationUtils.NODE_BASE_LIST	*List<NodeBase>* • *NodeDocument* • *NodeFolder* • *NodeMail* • *NodeRecord* *These nodes extends from NodeBase.*	*Node base list*
AutomationUtils.NOTES	*List<NodeNote>*	*List of notes*
AutomationUtils.MAIL_BCC	*String[]*	*Mail bcc*
AutomationUtils.MAIL_CC	*String[]*	*Mail cc*
AutomationUtils.MAIL_CONTENT	*String*	*Mail content*
AutomationUtils.MAIL_FROM	*String*	*Mail from*
AutomationUtils.MAIL_NODE	*NodeMail*	*Mail node*
AutomationUtils.MAIL_RECEIVED_DATE	*Calendar*	*Mail sent date*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AutomationUtils.MAIL_REPLY	*String[]*	*Mail reply*
AutomationUtils.MAIL_SENT_DATE	*Calendar*	*Mail sent date*
AutomationUtils.MAIL_SUBJECT	*String*	*Mail subject*
AutomationUtils.MAIL_TO	*String[]*	*Mail to*
AutomationUtils.MIME_TYPE	*String*	*Mime Type*
AutomationUtils.NAME	*String*	*Name*
AutomationUtils.PROPERTY_GROUP_NAME	*String*	*Property group name*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*Map<String, String>*	*Property group properties*
AutomationUtils.PROPERTY_GROUPS	*Set<NodeProperty>*	*List of property groups (metadata groups)*
AutomationUtils.RECORD_NODE	*NodeRecord*	*Record node*
AutomationUtils.TEXT_EXTRACTED	*String*	*Document text extracted*
AutomationUtils.LANGUAGE_DETECTED	*String*	*Language detected*
AutomationUtils.UPLOAD_RESPONSE	*String*	*Upload response*
AutomationUtils.TITLE	*String*	*Title*
AutomationUtils.WIKIPAGE	*wikiPage*	*The wiki page*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AutomationUtils.NODE_CLASS	*Long*	*The NodeClass id*
AutomationUtils.NEW_NODE_CLASS	*Long*	*The new NodeClass*
AutomationUtils.OLD_NODE_CLASS	*Long*	*The old NodeClass*
AutomationUtils.TASK_ID	*Long*	*The Task id*
AutomationUtils.TASK_OBJECT	*TaskManagerTask*	*Task object*
AutomationUtils.USER	*com.openkm.bean.CommonUser*	*User*
AutomationUtils.DOCUMENT_SIZE	*Long*	*Document size*

Document

Create Document

At Pre

Variable	***For changing***
AutomationUtils.CATEGORIES	*Yes*
AutomationUtils.CREATION_DATE	*No*
AutomationUtils.NAME	*Yes*
AutomationUtils.MIME_TYPE	*Yes*
AutomationUtils.KEYWORDS	*Yes*
AutomationUtils.NOTES	*Yes*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre	
AutomationUtils.NODE_CLASS	*Yes*
AutomationUtils.DESTINATION_NODE	*Yes*
AutomationUtils.PROPERTY_GROUPS	*Yes*
AutomationUtils.TITLE	*Yes*
AutomationUtils.WIKIPAGE	*Yes*
AutomationUtils.DOCUMENT_SIZE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_FILE	*No*

Update Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***

AutomationUtils.DOCUMENT_NODE ***At Pre***	*No*

Delete Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Rename Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Purge Document

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.TENANT	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.TENANT	*No*

Move Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Restore Version Document

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*

Download Document from UI

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*
AutomationUtils.DOCUMENT_NAME	*Yes*
AutomationUtils.DOCUMENT_NODE	*No*
At post	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre ***Variable***	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*
AutomationUtils.DOCUMENT_NAME	*No*
AutomationUtils.DOCUMENT_NODE	*No*

Download Document from UI for preview

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_NAME	*Yes*
AutomationUtils.DOCUMENT_VERSION	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_NAME	*No*
AutomationUtils.DOCUMENT_VERSION	*No*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Print Document

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.DOCUMENT_NODE	
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.DOCUMENT_NODE	*No*

Stamp Document

At pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Virus detected Document

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre	
Variable	***For changing***
AutomationUtils.USER	*No*
AutomationUtils.DOCUMENT_NAME	*No*
AutomationUtils.VIRUS_INFO	*No*

Set Node Class Document

At Pre	
Variable	***For changing***
AutomationUtils.NODE_CLASS	*No*
AutomationUtils.OLD_NODE_CLASS	*No*
AutomationUtils.NEW_NODE_CLASS	*Yes*
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.NODE_CLASS	*No*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Folder

Create Folder

At Pre	
Variable	***For changing***
AutomationUtils.CATEGORIES	*Yes*
AutomationUtils.CREATION_DATE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
AutomationUtils.NAME	*Yes*
AutomationUtils.NOTES	*Yes*
AutomationUtils.KEYWORDS	*Yes*
AutomationUtils.PROPERTY_GROUPS	*Yes*
AutomationUtils.WIKIPAGE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Delete Folder



The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Purge Folder

The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Move Folder

The event is only triggered on the root node and the destination node. If the node has subnodes, an event will not be generated from them.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Rename Folder

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Export Folder as ZIP

At Pre	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Variable	***For changing***
At Pre	
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Export Folder as JAR

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Mail

Create mail

At Pre	
Variable	***For changing***
AutomationUtils.CATEGORIES	*Yes*

AutomationUtils.CREATION_DATE ***At Pre***	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
AutomationUtils.MAIL_FROM	*No*
AutomationUtils.MAIL_REPLY	*No*
AutomationUtils.MAIL_TO	*No*
AutomationUtils.MAIL_CC	*No*
AutomationUtils.MAIL_BCC	*No*
AutomationUtils.MAIL_SENT_DATE	*No*
AutomationUtils.MAIL_RECEIVED_DATE	*No*
AutomationUtils.MAIL_SUBJECT	*No*
AutomationUtils.MAIL_CONTENT	*No*
AutomationUtils.NAME	*Yes*
AutomationUtils.KEYWORDS	*Yes*
AutomationUtils.MIME_TYPE	*No*
AutomationUtils.TITLE	*Yes*
At post	

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Variable	*For changing*
At Pre	
AutomationUtils.MAIL_NODE	*No*

Move Mail

> *The event is only triggered on the root node and the destination node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Download mail from UI

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***

At Pre	
AutomationUtils.MAIL_NODE	*No*

Set Node Class Mail

At Pre	
Variable	***For changing***
AutomationUtils.NODE_CLASS	*No*
AutomationUtils.OLD_NODE_CLASS	*No*
AutomationUtils.NEW_NODE_CLASS	*Yes*
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
AutomationUtils.NODE_CLASS	*No*

Mail import attachments Mail

At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Delete Mail

> *The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Purge Mail

> *The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

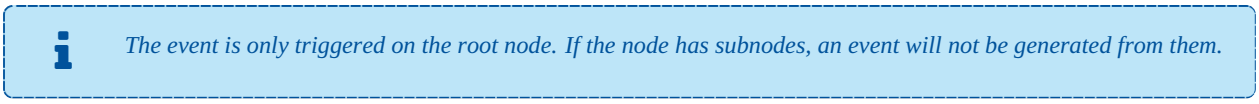
Record

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Create record

At Pre	
Variable	***For changing***
AutomationUtils.DESTINATION_NODE	*Yes*
AutomationUtils.NAME	*Yes*
AutomationUtils.NODE_CLASS	*Yes*
AutomationUtils.TITLE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*

Delete record



At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
At post	
Variable	***For changing***

At Pre	
AutomationUtils.RECORD_NODE	*No*

Purge record

> *The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*

Move Record

> *The event is only triggered on the root node and the destination node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	

Variable *At Pre*	*For changing*
AutomationUtils.RECORD_NODE	*No*

Set Node Class Record

At Pre	
Variable	***For changing***
AutomationUtils.NODE_CLASS	*No*
AutomationUtils.OLD_NODE_CLASS	*No*
AutomationUtils.NEW_NODE_CLASS	*Yes*
AutomationUtils.RECORD_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
AutomationUtils.NODE_CLASS	*No*

Property Group (metadata)

Add property group

At Pre	
Variable	***For changing***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

AutomationUtils.NODE_BASE	*No*
At Pre	
AutomationUtils.PROPERTY_GROUP_NAME	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*Yes*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.PROPERTY_GROUP_NAME	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*No*

Set property group

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.PROPERTY_GROUP_NAME	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*Yes*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre *AutomationUtils.PROPERTY_GROUP_NAME*	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*No*

Remove property group

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.PROPERTY_GROUP_NAME	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.PROPERTY_GROUP_NAME	*No*

Auth

Create user

At Pre	
Variable	***For changing***
AutomationUtils.USER	*No*

At post *At Pre*	
Variable	***For changing***
AutomationUtils.USER	*No*

User login

At post	
Variable	***For changing***
AutomationUtils.USER	*No*

User logout

At Pre	
Variable	***For changing***
AutomationUtils.USER	*No*

Task

Create Task

At Pre	
Variable	***For changing***
AutomationUtils.TASK_OBJECT	*Yes*
At post	

Variable	***For changing***
At Pre	
AutomationUtils.TASK_OBJECT	*No*

Update Task

At Pre	
Variable	***For changing***
AutomationUtils.TASK_OBJECT	*Yes*
At post	
Variable	***For changing***
AutomationUtils.TASK_OBJECT	*No*

Delete Task

At Pre	
Variable	***For changing***
AutomationUtils.TASK_ID	*No*
At post	
Variable	***For changing***
AutomationUtils.TASK_ID	*No*

Note

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Create Note

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_TEXT	*No*
At post	
Variable	***For changing***
AutomationUtils.NOTE_UUID	*No*

Update Note

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_TEXT	*No*
AutomationUtils.NOTE_UUID	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

At Pre	
AutomationUtils.NOTE_TEXT	*No*
AutomationUtils.NOTE_UUID	*No*

Delete Note

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_UUID	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_UUID	*No*

General

Text extraction

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

AutomationUtils.TENANT	*No*
At Pre	
At post	
Variable	***For changing***
AutomationUtils.TEXT_EXTRACTED	*Yes*
AutomationUtils.LANGUAGE_DETECTED	*Yes*

Export nodes to zip

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE_LIST	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE_LIST	*No*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Vault of stuff

Field / Property	*Description*
Convert Exchange Legacy DN	*Changes all the Legacy DN strings into proper mail addresses.*
Download document with direct link	*Downloads a document by url.*
Import document with bash script using REST	*Imports a document with a bash script that uses REST Webservices API.*
Creating users from scripts	*Creation of users from generated scripts with file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Convert Exchange Legacy DN

The script changes all the Legacy DN strings into proper mail addresses. Therefore there's an export file of the global address list from our IT as .csv containing all the Exchange Legacy DNs and their corresponding mail addresses. With this data the database can be altered the Exchange Legacy DN can be changed to normal mail addresses.

CSV sample:

```
/o=EMail/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=somefancyname;f.ancy@sa
/o=EMail/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=anotherfancyname;an.fancy
/o=EMail/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=m.mueller;m.mueller@mydo
```

```
<?php
# Change ExchangeDN to Mailadresses
# Malte Küppers
#
# CSV Legacy File Format:       Exchange Legacy DN ; Mailadresse
#
#


# variables
$row = 0;
$legacy_file = "/root/Hilfsdateien/LegacyExchangeDN.txt" # Set to your filepath


if (($handle = fopen($legacy_file, "r")) !== FALSE) {
    while (($data = fgetcsv($handle, 10000, ";")) !== FALSE) {
                $exchdn[$row]=strtoupper($data[0]);     #Exchange Legacy DN
                $mail[$row]=$data[1];                   #Mailadresse
                $row++;

    }
    fclose($handle);
        #echo $row ." lines read";
}

#Open Database Connection
$db = mysqli_connect("127.0.0.1", "<USERNAME>", "<PASSWORD>", "okmdb");
if(!$db)
{
        exit("Verbindungsfehler: ".mysqli_connect_error());
}

# Mail FROM ändern
$ergebnis = mysqli_query($db, "SELECT DISTINCT NML_FROM FROM OKM_NODE_MAIL");
while($row = mysqli_fetch_object($ergebnis)){
                #only adresses that needs to be changed (Starting with /o= )
                if(strpos($row->NML_FROM,"/o")!==false) {

                        $key = array_search(strtoupper($row->NML_FROM), $exchdn);
                        if(is_numeric($key)){
                                echo $row->NML_FROM ." found at Pos #".$key ."<br>";
                                $qry = "UPDATE OKM_NODE_MAIL SET NML_FROM = '".$mail[$key]."' WHERE NML_FROM
                                #echo $qry ."<br><br>";
                                $update = mysqli_query($db,$qry);
                                }
                }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
}

# Mail TO ändern
$ergebnis = mysqli_query($db, "SELECT DISTINCT NML_TO FROM OKM_NODE_MAIL_TO");
while($row = mysqli_fetch_object($ergebnis)){
        #Nur Adressen die nicht geändert wurden
        if(strpos($row->NML_TO,"/o")!==false) {

                $key = array_search(strtoupper($row->NML_TO), $exchdn);
                if(is_numeric($key)){
                        echo $row->NML_TO ." gefunden an Pos #".$key ."<br>";
                        $qry = "UPDATE OKM_NODE_MAIL_TO SET NML_TO = '".$mail[$key]."' WHERE NML_TO ='
                        #echo $qry ."<br><br>";
                        $update = mysqli_query($db,$qry);
                        }
        }
}
# Mail CC ändern
$ergebnis = mysqli_query($db, "SELECT DISTINCT NML_CC FROM OKM_NODE_MAIL_CC");
while($row = mysqli_fetch_object($ergebnis)){
        #Nur Adressen die nicht geändert wurden
        if(strpos($row->NML_CC,"/o")!==false){

                $key = array_search(strtoupper($row->NML_CC), $exchdn);
                if(is_numeric($key)){
                        echo $row->NML_CC ." gefunden an Pos #".$key ."<br>";
                        $qry = "UPDATE OKM_NODE_MAIL_CC SET NML_CC = '".$mail[$key]."' WHERE NML_CC =
                        #echo $qry ."<br><br>";
                        $update = mysqli_query($db,$qry);
                        }
        }
}


# Mail BCC ändern
$ergebnis = mysqli_query($db, "SELECT DISTINCT NML_BCC FROM OKM_NODE_MAIL_BCC");
while($row = mysqli_fetch_object($ergebnis)){
        #Nur Adressen die nicht geändert wurden
        if(strpos($row->NML_BCC,"/o")!==false){

                $key = array_search(strtoupper($row->NML_BCC), $exchdn);
                if(is_numeric($key)){
                        echo $row->NML_BCC ." gefunden an Pos #".$key ."<br>";
                        $qry = "UPDATE OKM_NODE_MAIL_BCC SET NML_BCC = '".$mail[$key]."' WHERE NML_BC
                        #echo $qry ."<br><br>";
                        $update = mysqli_query($db,$qry);
                        }
        }
}

?>
```



Additional information at our public forum.

Download document with direct link

You can download a document from OpenKM using a command line utility like ***wget****. There is a service with BASIC authentication for this task. You can reference a document by UUID using the* ***uuid*** *parameter. In the same URL you need to specify the user and password.*

> *When you share the user and password for downloading files is a good practice configure the user profile disallowing all the features. That will lock UI access to the user.*

Available parameters:

- *The parameter* ***uuid*** *is the document UUID*
- *The parameter* ***inline*** *set when document must be downloaded as attachment or shown inline. It's a boolean parameter, allowed values are true or false.*

> *When you use parameter path, it is a good practice to encode url to prevent problems with special characters.*

Sample:

```
http://user:password@@localhost:8080/openkm/download?uuid=def34bcc-3291-4e11-9aa3-e86ae32f5a6c
```

This ease some integration tasks:

```
$ wget http://okmAdmin:admin@localhost:8080/openkm/download?uuid=xxx-yyy-zzz
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Import document with bash script using REST

Simplified version of the script which make use of RESTful API.

```
OKM_SERVICE="http://localhost:8080/openkm"
OKM_USER="okmAdmin"
OKM_PASSWORD="admin"
OKM_UUID="754aefd5-c11c-4954-a94e-b062cf3fed97"
OKM_PATH="/okm:root/upload"
LOCAL_PATH="."
TMP=$(mktemp)
RESET='\e[0m'
ERROR='\e[1;31m'
OK='\e[1;34m'

for DIR in $(find $LOCAL_PATH -type d); do
  DIR=${DIR:1:${#DIR}}
  if [ -n "$DIR" ]; then
    RETCODE=$(curl -u $OKM_USER:$OKM_PASSWORD --write-out %{http_code} -X POST --silent --output $TMP
      -H "Content-Type: application/json" -d "$DIR" "$OKM_SERVICE/rest/folders/parents/$OKM_UUID")
    if [ $RETCODE != "200" ]; then
      echo -e "${ERROR}ERROR -> $(cat $TMP)${RESET}";
    else
      echo -e "${OK}Folder created: ${OKM_PATH}${DIR}${RESET}"
    fi
  fi
done

for FILE in $(find $LOCAL_PATH -type f); do
  FILE=${FILE:2:${#FILE}}
  RETCODE=$(curl -u $OKM_USER:$OKM_PASSWORD --write-out %{http_code} -X POST --silent --output $TMP
    -F "docPath=$OKM_PATH/$FILE" -F content=@$FILE "$OKM_SERVICE/rest/documents/parents/$OKM_UUID",
  if [ $RETCODE != "200" ]; then
    echo -e "${ERROR}ERROR -> $(cat $TMP)${RESET}";
  else
    echo -e "${OK}Document created: ${OKM_PATH}/${FILE}${RESET}"
  fi
done

# Cleanup
rm -f $TMP
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Creating users from scripts

The script allows you to create users and assign them the user role.

Download file create-users_okm.xlsx

*The string "**newId**" is the **user Id** what will be set.*

*The string "**newPassword**" is the **user password** what will be set.*

*The string "**newMail@mail.com**" is the **user mail** what will be set.*

*The string "**newRealName**" is the **user real name** what will be set.*

```
import com.openkm.util.ContextWrapper;
import com.openkm.api.OKMAuth;
import com.openkm.bean.User;

OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
User user = new User("newId", "newPassword", "newMail@mail.com", "New Real Name", true);
okmAuth.createUser(null, user);
okmAuth.assignRole(null, user.getId(), "ROLE_USER");
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

API description

OpenKM API classes:

OpenKM API class	*Description*
OKMActivity	*Manage the activites and actions*
OKMAuth	*Manage security and users. For example add or remove grants on a node, create or modify users or getting the profiles.*
OKMBookmark	*Manage the user bookmarks.*
OKMDashboard	*Manage all data shown at dashboard.*
OKMDocument	*Manage documents. For example create, move or delete a document.*
OKMFolder	*Manage folders. For example create, move or delete a folder.*
OKMMail	*Manage mails. For example create, move or delete a mails.*
OKMNote	*Manage notes on any node type. For example create, edit or delete a note on a document, folder, mail or record.*
OKMNotification	*Manage notifications. For example add or remove subscriptions on a document or a folder.*
OKMProperty	*Manage categories and keywords. For example add or remove keywords on a document, folder, mail or record.*
OKMPropertyGroup	*Manage metadata. For example add metadata group, set metadata fields.*
OKMRecord	*Manage records. For example create, move or delete a record.*
OKMRelation	*Manage relations between nodes. For example create a relation (parent-child) between two*

	documents.
OKMReport	*Manage reports.*
OKMRepository	*A lot of stuff related with repository. For example get the properties of main root node (/okm:root).*
OKMSearch	*Manage search feature. For example manage saved queries or perform a new query to the repository.*
OKMStats	*General stats of the repository.*
OKMTask	*Manage task. For example create a new task.*
OKMUserConfig	*Manage user home configuration.*
OKMWorkflow	*Manage workflows. For example execute a new workflow.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMActivity

With OKMActivity you can explore the OpenKM activity log data.

Methods

findByFilter

Description:

Method	Return values	Description
findByFilter(String token, ActivityFilter filter)	**List<Activity>**	*Returns a list of all the activity log.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMActivity;
import com.openkm.db.bean.Activity;
import com.openkm.db.bean.ActivityFilter;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMActivity okmActivity = ContextWrapper.getContext().getBean(OKMActivity.class);

            Calendar beginDate = Calendar.getInstance();
            beginDate.add(Calendar.MONTH, -1);
            Calendar endDate = Calendar.getInstance();
            String item = "f84a2e1f-a858-4e53-9c09-36519d903782";
            String user = "okmAdmin";

            ActivityFilter filter = new ActivityFilter();
            filter.setBegin(beginDate);
            filter.setEnd(endDate);
            filter.setItem(item);
            filter.setUser(user);
            filter.setPage(0);
            filter.setLength(20);
            for (Activity activity : okmActivity.findByFilter(null, filter)) {
                System.out.println(activity);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

countByFilter

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
countByFilter(String token, ActivityFilter filter)	***int***	*Returns the number of activity log rows.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMActivity;
import com.openkm.db.bean.ActivityFilter;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMActivity okmActivity = ContextWrapper.getContext().getBean(OKMActivity.class);

            Calendar beginDate = Calendar.getInstance();
            beginDate.add(Calendar.MONTH, -1);
            Calendar endDate = Calendar.getInstance();
            String item = "f84a2e1f-a858-4e53-9c09-36519d903782";
            String user = "okmAdmin";

            ActivityFilter filter = new ActivityFilter();
            filter.setBegin(beginDate);
            filter.setEnd(endDate);
            filter.setItem(item);
            filter.setUser(user);
            filter.setPage(0);
            filter.setLength(20);
            System.out.println(okmActivity.countByFilter(null, filter));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActions

Description:

Method	*Return values*	*Description*
getActions(String token)	***List<String>***	*Returns a list of all the activity log actions.*

Example

```
package com.openkm;

import com.openkm.api.OKMActivity;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMActivity okmActivity = ContextWrapper.getContext().getBean(OKMActivity.class);

            for (String action : okmActivity.getActions(null)) {
                System.out.println(action);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMAuth

Used for managing security and users. For example, add or remove grants on a node, create or modify users, or get the profiles.



Starting from version 8.1.14, some methods return the DbUser object instead of the old CommonUser, which has been deprecated.

Basics

*The class **com.openkm.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.*



To set READ and WRITE access, you should do:

```
int permission = Permission.READ + Permission.WRITE;
```

To check if you have permission access,s you should do:

```
// permission is a valid integer value
if ((permission | Permission.WRITE) = Permission.WRITE) {
  // Has WRITE grants.
}
```

*In almost all methods, you'll see a parameter named "**nodeId**". The value of this parameter can be a valid node **UUID** (folder, document, mail, record).*



Example of nodeId:

- *Using UUID -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**";*

*Also, in all methods, you'll see a parameter named "**token**". Because the Cron tasks are executed in the background without authentication, the methods used in this scenario might use the token parameter. From the default application execution context, you must use "**null**" value, which indicates that the application must use the "**user session**".*



*In special cases, you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

login

Description:

Method	*Return values*	*Description*
login()	***void***	*Simulates user UI login process.*

When a user logs in from UI, some background process creates main user nodes, like /okm:trash. Unfortunately, if the user has never logged in from UI and login from the API, these nodes are still not created and will raise an error, because it is necessary at the beginning execute the login method.

The user must be logged in before executing the method.

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.login();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

login

Description:

Method	*Return values*	*Description*
login(String user, String pass)	***String***	*The login process returns an authentication token.*

Login token by default have an expiration time of 24h.

After executing the login method, all the next calls will automatically use the authentication token.

Each time the login method is executed, the login token is replaced with a newer.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



When a user login in to the application for the first time, is called internally calls a method that creates user specific folders, like /okm:trash/userId etc.

From API point of view, this method should be only executed first time user accessing from API, for creating specific user folder structure.

Otherwise, you can get errors, for example PathNotExistsException /okm:trash/userId when deleting objects, because the folders have not been created.

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.login("userId","password");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

grantUser

Description:

Method	*Return values*	*Description*
grantUser(String token, String nodeId, String user, int permissions, boolean recursive)	***void***	*Add a user grant on a node.*
The parameter recursive only has sense when the nodeId is a folder or record node. *When the parameter recursive is true, the change will be applied to the node and its descendants.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            // Add sochoa write grants at the node but not descendants
            okmAuth.grantUser(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "sochoa", Permission.ALL_GRANTS, f

            // Add all okmAdmin grants at the node and descendants
            okmAuth.grantUser(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "okmAdmin", Permission.ALL_GRANT
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

revokeUser

Description:

Method	*Return values*	*Description*
revokeUser(String token, String nodeId, String user, int permissions, boolean recursive)	***void***	*Remove user grant on a node.*

The parameter recursive only has sense when the nodeId is a folder or a record node.

When the parameter recursive is true, the change will be applied to the node and descendants.

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            // Add sochoa write grants at the node but not descendants
            okmAuth.revokeUser(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "sochoa", Permission.ALL_GRANTS

            // Remove all okmAdmin grants at the node and descendants
            okmAuth.revokeUser(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "okmAdmin", Permission.ALL_GRAN
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

getGrantedUsers

Description:

Method	*Return values*	*Description*
getGrantedUsers(String token, String nodeId)	***Map<String, Integer>***	*Return the granted users of a node.*

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            Map<String, Integer> grants = okmAuth.getGrantedUsers(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338");
            for (String role : grants.keySet()) {
                System.out.println(role + "->" + grants.get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

grantRole

Description:

Method	*Return values*	*Description*
grantRole(String token, String nodeId, String role, int permissions, boolean recursive)	***void***	*Add a role grant on a node.*
The parameter recursive only has sense when the nodeId is a folder or a record node. *When the parameter recursive is true, the change will be applied to the node and descendants.*		

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            // Add ROLE_USER write grants at the node but not descendants
            okmAuth.grantRole(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "ROLE_USER", Permission.ALL_GRA

            // Add all ROLE_ADMIN grants to the node and descendants
            okmAuth.grantRole(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "ROLE_ADMIN", Permission.ALL_GR
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

revokeRole

Description:

Method	*Return values*	*Description*
revokeRole(String token, String nodeId, String role, int permissions, boolean recursive)	***void***	*Remove role grant on a node.*
The parameter recursive only has sense when the nodeId is a folder or a record node. *When the parameter recursive is true, the change will be applied to the node and its descendants.*		

Example:

```
package com.openkm;
import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            // Remove ROLE_USER write grants at the node but not descendants
            okmAuth.revokeRole(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "ROLE_USER", Permission.ALL_GR

            // Remove all ROLE_ADMIN grants to the node and descendants
            okmAuth.revokeRole(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "ROLE_ADMIN", Permission.ALL_G
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

getGrantedRoles

Description:

Method	*Return values*	*Description*
getGrantedRoles(String token, String nodeId)	***Map<String, Integer>***	*Return the granted roles of a node.*

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            Map<String, Integer> grants = okmAuth.getGrantedRoles(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338");
            for (String role : grants.keySet()) {
                System.out.println(role + "->" + grants.get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUsers

Description:

Method	*Return values*	*Description*
getUsers(String token)	***List<DbUser>***	*Return the list of all the users.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.DbUser;
import com.openkm.util.ContextWrapper;

public class Test {
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            for (DbUser user : okmAuth.getUsers(null)) {
                System.out.println(user.getId());
                System.out.println(user.getName());
                System.out.println(user.getEmail());
                System.out.println(user.getActive());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRoles

Description:

Method	*Return values*	*Description*
getRoles(String token)	***List<String>***	*Return the list of all the roles.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            for (String role : okmAuth.getRoles(null)) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUsersByRole

Description:

Method	*Return values*	*Description*
getUsersByRole(String token, String role)	***List<DbUser>***	*Return the list of all the users who have been assigned a role.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.DbUser;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            for (DbUser user : okmAuth.getUsersByRole(null, "ROLE_ADMIN")) {
                System.out.println(user.getId());
                System.out.println(user.getName());
                System.out.println(user.getEmail());
                System.out.println(user.getActive());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRolesByUser

Description:

Method	*Return values*	*Description*
getRolesByUser(String token, String user)	***List<String>***	*Return the list of all the roles assigned to a user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            for (String role : okmAuth.getRolesByUser(null, "okmAdmin")) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

changeSecurity

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
public void changeSecurity(String token, String nodeId, Map<String, Integer> grantUsers, Map<String, Integer> revokeUsers, Map<String, Integer> grantRoles, Map<String, Integer> revokeRoles, boolean recursive)	***void***	*Change the security of a node.*

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            String nodeId = "b153c4b7-3d1c-4589-bd42-0ed0f34fd338";
            Map<String, Integer> grantUsers = new HashMap<>();
            Map<String, Integer> revokeUsers = new HashMap<>();
            Map<String, Integer> grantRoles = new HashMap<>();
            grantRoles.put("ROLE_TEST", Permission.READ | Permission.WRITE);
            Map<String, Integer> revokeRoles = new HashMap<>();
            okmAuth.changeSecurity(null, nodeId, grantUsers, revokeUsers, grantRoles, revokeRoles, false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

overwriteSecurity

Description:

Method	*Return values*	*Description*
public void overwriteSecurity(String token, String nodeId, Map<String, Integer> grantUsers, Map<String, Integer> grantRoles, boolean recursive)	***void***	*Overwrite the security of a node.*

Example:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            String nodeId = "b153c4b7-3d1c-4589-bd42-0ed0f34fd338";
            Map<String, Integer> grantUsers = new HashMap<>();
            Map<String, Integer> grantRoles = new HashMap<>();
            grantRoles.put("ROLE_TEST", Permission.READ | Permission.WRITE);
            okmAuth.overwriteSecurity(null, nodeId, grantUsers, grantRoles, false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createUser

Description:

Method	*Return values*	*Description*
createUser(String token, User user)	***void***	*Create a new user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.User;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            User user = new User();
            user.setId("test");
            user.setPassword("password.2019");
            user.setEmail("some@mail.com");
            user.setName("User Name");
            user.setActive(true);
            okmAuth.createUser(null, user);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

deleteUser

Description:

Method	*Return values*	*Description*
deleteUser(String token, String user)	***void***	*Delete a user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.deleteUser(null, "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateUser

Description:

Method	*Return values*	*Description*
updateUser(String token, User user)	***void***	*Update a user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.User;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            User user = new User();
            user.setId("test");
            user.setPassword("newpassword");
            user.setEmail("some@mail.com");
            user.setName("Test");
            user.setActive(false);
```

```
            okmAuth.updateUser(null, user);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createRole

Description:

Method	*Return values*	*Description*
createRole(String token, Role role)	***void***	*Create a new role.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Role;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            Role role = new Role();
            role.setId("ROLE_TEST");
            role.setActive(true);
            okmAuth.createRole(null, role);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteRole

Description:

Method	*Return values*	*Description*
deleteRole(String token, String role)	***void***	*Delete a role.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.deleteRole(null, "ROLE_TEST");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateRole

Description:

Method	*Return values*	*Description*
updateRole(String token, String role, boolean active)	***void***	*Update a role.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;

public class Test {
    public static void main(String[] args) {
        try {
            OKMAuth.getInstance().updateRole(null, "ROLE_TEST",true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

assignRole

Description:

Method	*Return values*	*Description*
assignRole(String token, String user, String role)	***void***	*Assign a role to a user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.assignRole(null, "test", "ROLE_USER");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeRole

Description:

Method	*Return values*	*Description*
removeRole(String token, String user, String role)	***void***	*Remove a role from a user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.removeRole(null, "test", "ROLE_USER");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProfiles

Description:

Method	*Return values*	*Description*
getProfiles(String token, boolean filterByActive)	***List<Profile>***	*Return the list of all profiles.*

The parameter filterByActive, when enabled, the method will return only the active profiles; otherwise will return all available profiles.



Each user has assigned one profile that enables more or fewer of the OpenKM UI features.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.db.bean.Profile;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            for (Profile profile : okmAuth.getProfiles(null, true)) {
                System.out.println(profile);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserProfile

Description:

Method	*Return values*	*Description*
getUserProfile(String token, String userId)	***Profile***	*Return the profile assigned to a user.*

> *Each user has assigned one profile that enables more or fewer of the OpenKM UI features.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            System.out.println(okmAuth.getUserProfile(null, "okmAdmin"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setUserProfile

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
setUserProfile(String token, String userId, long profileId)	***void***	*Change the assigned profile to a user.*

Each user has assigned one profile that enables more or less of the OpenKM UI features.

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.db.bean.Profile;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            // Set the profile named "default" to the user
            for (Profile profile : okmAuth.getProfiles(null, true)) {
                if (profile.getName().equals("default")) {
                    okmAuth.setUserProfile(null, "okmAdmin", profile.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

logout

Description:

Method	*Return values*	*Description*
logout()	*void*	*Execute the logout method on the OpenKM side.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.logout(null);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUser

Description:

Method	*Return values*	*Description*
getUser(String token, String userId)	***DbUser***	*Return user data.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.DbUser;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            DbUser commonUser = okmAuth.getUser(null, "okmAdmin");
            System.out.print(commonUser);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setUserPermissions

Description:

Method	*Return values*	*Description*
public void setUserPermissions(String token, String nodeId, String user, int permissions, boolean recursive)	***void***	*Update user permissions on a node.*
The parameter recursive only has sense when the uuid is a folder or record node. *When the parameter recursive is true, the changes will be applied to the node and descendants.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            // Update permissions of sochoa at the node but not descendants
            okmAuth.setUserPermissions(null, "3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "sochoa", Permission.REAL

            // Update permissions of okmAdmin at the node and descendants
            okmAuth.setUserPermissions(null, "3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "okmAdmin", Permission.AL
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRolePermissions

Description:

Method	*Return values*	*Description*
setRolePermissions(String token, String nodeId, String role, int permissions, boolean recursive)	***void***	*Update role permissions on a node.*
The parameter recursive only has sense when the uuid is a folder or record node. *When the parameter recursive is true, the changes will be applied to the node and descendants.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Permission;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            // Update permissions of ROLE_USER at the node but not descendants
            okmAuth.setRolePermissions(null, "3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_USER", Permissior
```

```
            // Update permissions of ROLE_ADMIN at the node and descendants
            okmAuth.setRolePermissions(null, "3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_ADMIN", Permissio
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserTenants

Description:

Method	*Return values*	*Description*
getUserTenants(String token)	***List<Tenant>***	*Return the list of all tenants to which the user has access.*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.api.OKMAuth;
import com.openkm.db.bean.Tenant;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            List<Tenant> tenants = okmAuth.getUserTenants(null);
            for (Tenant tenant : tenants) {
                System.out.println(tenant);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setUserTenant

Description:

Method	*Return values*	*Description*
setUserTenant(String token, long tenantId)	***void***	*Change the assigned tenant to a user.*

> *A user can have access to several tenants but only have one assigned.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            long tenantId = 1;
            okmAuth.setUserTenant(null, tenantId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isPasswordExpired

Description:

Method	*Return values*	*Description*
isPasswordExpired(String token)	*Boolean*	*Check if the user password has expired.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            Boolean isExpired = okmAuth.isPasswordExpired(null);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

resetPassword

Description:

Method	*Return*	*Description*

values		
resetPassword(String userId)	***void***	*Sends an email with a link that, if the user presses it, will reset the password.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            String userId = "okmAdmin";
            okmAuth.resetPassword(userId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMBookmark

Used for managing bookmarks of users. For example add or remove bookmark of a node for a user.

Basics

*On almost methods you'll see parameter named "**nodeId**". The value of this parameter can be some valid node **UUID** (folder, document, mail, record).*

Example of nodeId:

- *Using UUID -> "**3887ae75-dd67-4019-b4d1-2f6962815403**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

add

Description:

Method	*Return values*	*Description*
add(String token, String nodeId, String name)	***Bookmark***	*Add a new bookmark of a node.*
The parameter name indicates the name which will be used to store the bookmark. Usually is used the same name of the nodeId.		

Example:

```
package com.openkm;

import com.openkm.api.OKMBookmark;
import com.openkm.db.bean.Bookmark;
import com.openkm.util.ContextWrapper;

public class Test {
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    public static void main(String[] args) {
        try {
            OKMBookmark okmBookmark = ContextWrapper.getContext().getBean(OKMBookmark.class);
            Bookmark bookmark = okmBookmark.add(null, "3887ae75-dd67-4019-b4d1-2f6962815403", "document.doc
            System.out.println(bookmark);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

get

Description:

Method	*Return values*	*Description*
get(String token, int bmId)	***Bookmark***	*Gets the bookmark data.*
The parameter bmId is the value of the unique bookmark identifier.		

Example:

```
package com.openkm;

import com.openkm.api.OKMBookmark;
import com.openkm.db.bean.Bookmark;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMBookmark okmBookmark = ContextWrapper.getContext().getBean(OKMBookmark.class);
            int bmId = 2;
            Bookmark bookmark = okmBookmark.get(null, bmId);
            System.out.println(bookmark);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

remove

Description:

Method	*Return values*	*Description*
remove(String token, int bmId)	***void***	*Delete a bookmark.*
The parameter bmId is the value of the unique bookmark identifier.		

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMBookmark;

public class Test {
    public static void main(String[] args) {
        try {
            int bmId = 23;
            OKMBookmark.getInstance().remove(null, bmId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	*Return values*	*Description*
rename(String token, int bmId, String newName	***Bookmark***	*Rename a bookmark.*
The parameter bmId is the value of the unique bookmark identifier.		

Example:

```
package com.openkm;

import com.openkm.api.OKMBookmark;
import com.openkm.db.bean.Bookmark;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMBookmark okmBookmark = ContextWrapper.getContext().getBean(OKMBookmark.class);
            int bmId = 2;
            Bookmark bookmark = okmBookmark.rename(null, bmId, "newname.doc");
            System.out.println(bookmark);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAll

Description:

Method	*Return values*	*Description*

getAll(String token)	***List<Bookmark>***	*Gets all the bookmarks of the logged user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMBookmark;
import com.openkm.db.bean.Bookmark;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMBookmark okmBookmark = ContextWrapper.getContext().getBean(OKMBookmark.class);
            for (Bookmark bookmark : okmBookmark.getAll(null)) {
                System.out.println(bookmark);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMDashboard

Basics

*On all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

getUserCheckedOutDocuments

Description:

Method	*Return values*	*Description*
getUserCheckedOutDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of documents checkout by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserCheckedOutDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastModifiedDocuments

Description:

Method	*Return values*	*Description*
getUserLastModifiedDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of the last documents modified by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastModifiedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockedDocuments

Description:

Method	*Return values*	*Description*
getUserLockedDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of the locked documents by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*

- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLockedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockedRecords

Description:

Method	*Return values*	*Description*
getUserLockedRecords(String token, long offset, long limit)	***DashboardRecordResultSet***	*Get the list of the locked records by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserLockedRecords(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockekFolders

Description:

Method	*Return values*	*Description*
getUserLockedFolders(int offset, int limit)	***DashboardResultSet***	*Returns a list of the folders locked by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardResultSet resultSet = okmDashboard.getUserLockedFolders(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockedMails

Description:

Method	*Return values*	*Description*
getUserLockedMails(int offset, int limit)	***DashboardResultSet***	*Returns a list of the mails locked by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardResultSet resultSet = okmDashboard.getUserLockedMails(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedDocuments

Description:

Method	*Return values*	*Description*
getUserSubscribedDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of the subscribed documents by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserSubscribedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedFolders

Description:

Method	*Return values*	*Description*
getUserSubscribedFolders(String token, long offset, long limit)	***DashboardFolderResultSet***	*Get the list of the subscribed folders by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*

- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.bean.DashboardFolderResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardFolderResultSet resultSet = okmDashboard.getUserSubscribedFolders(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardFolderResult dashboardFolderResult : resultSet.getResults()) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedRecords

Description:

Method	*Return values*	*Description*
getUserSubscribedRecords(String token, long offset, long limit)	***DashboardRecordResultSet***	*Get the list of the subscribed records by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserSubscribedRecords(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedDocuments

Description:

Method	*Return values*	*Description*
getUserLastCreatedDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of the last created documents by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these

values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastCreatedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedDocumentsNotes

Description:

Method	*Return values*	*Description*
getUserLastCreatedDocumentsNotes(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of the last created documents/notes by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastCreatedDocumentsNotes(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedFolders

Description:

Method	*Return values*	*Description*
getUserLastCreatedFolders(String token, long offset, long limit)	***DashboardFolderResultSet***	*Get the list of the last created folders by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*

- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.bean.DashboardFolderResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardFolderResultSet resultSet = okmDashboard.getUserLastCreatedFolders(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardFolderResult dashboardFolderResult : resultSet.getResults()) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedFoldersNotes

Description:

Method	*Return values*	*Description*
getUserLastCreatedFoldersNotes(String token, long offset, long limit)	***DashboardFolderResultSet***	*Get the list of the last created folders/notes by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.bean.DashboardFolderResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardFolderResultSet resultSet = okmDashboard.getUserLastCreatedFoldersNotes(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardFolderResult dashboardFolderResult : resultSet.getResults()) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedRecords

Description:

Method	*Return values*	*Description*
getUserLastCreatedRecords(String token, long offset, long limit)	***List<DashboardRecordResult>***	*Get the list of the last created records by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserLastCreatedRecords(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedRecordsNotes

Description:

Method	*Return values*	*Description*
getUserLastCreatedRecordsNotes(String token, long offset, long limit)	***List<DashboardRecordResult>***	*Get the list of the last created records/notes by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserLastCreatedRecordsNotes(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastDownloadedDocuments

Description:

Method	*Return values*	*Description*
getUserLastDownloadedDocuments(String token,		*Get the list of the last*

long offset, long limit)	***DashboardDocumentResultSet***	*download documents by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastDownloadedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastImportedMails

Description:

Method	*Return values*	*Description*

getUserLastImportedMails(String token, long offset, long limit)	***DashboardMailResultSet***	*Get the list of the last imported mails by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardMailResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardMailResult dashboardMailResult : okmDashboard.getUserLastImportedMails(null, "okmAdmi
                System.out.println(dashboardMailResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedMailsNotes

Description:

Method	*Return values*	*Description*

getUserLastCreatedMailsNotes(String token, long offset, long limit)	***DashboardMailResultSet***	*Get the list of the last imported mails/notes by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardMailResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardMailResult dashboardMailResult : okmDashboard.getUserLastCreatedMailsNotes(null, "okm
                System.out.println(dashboardMailResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastImportedMailAttachments

Description:

Method	*Return values*	*Description*

getUserLastImportedMailAttachments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of the last imported mail attachments by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastImportedMailAttachments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserDocumentsSize

Description:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Method	Return values	Description
getUserDocumentsSize(String token)	**long**	*Return the total size in bytes of the user documents.*

> *When user upload or modify a document, the entire size of the contents - and history versions in case of update - are assigned to the users.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FormatUtil;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            long sizeInBytes = okmDashboard.getUserDocumentsSize(null);
            System.out.println(FormatUtil.formatSize(sizeInBytes));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSearches

Description:

Method	Return values	Description
getUserSearches(String token)	**List<QueryParams>**	*Get the list of all "user news" saved by the user.*

> *The "user saved" searches are executed periodically for getting news based on queries from the repository.*
>
> *More information at Working with the dashboard and Working with search views.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (QueryParams qp : okmDashboard.getUserSearches(null)) {
                System.out.println(qp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

find

Description:

Method	*Return values*	*Description*
find(String token, int qpId)	***List<DashboardNodeResult>***	*Execute a "user news".*
The param qpId is the unique identifier of the "users news" search.		

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardNodeResult;
import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.bean.Mail;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            int qpId = 12;
            for (DashboardNodeResult dashboardNodeResult : okmDashboard.find(null, qpId)) {
                if (dashboardNodeResult.getNode() instanceof Document) {
                    Document doc = (Document) dashboardNodeResult.getNode();
                    System.out.println(doc);
                } else if (dashboardNodeResult.getNode() instanceof Folder) {
                    Folder fld = (Folder) dashboardNodeResult.getNode();
                    System.out.println(fld);
                } else if (dashboardNodeResult.getNode() instanceof Record) {
                    Record rec = (Record) dashboardNodeResult.getNode();
                    System.out.println(rec);
                } else if (dashboardNodeResult.getNode() instanceof Mail) {
                    Mail mail = (Mail) dashboardNodeResult.getNode();
                    System.out.println(mail);
                }
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

getLastWeekTopDownloadedDocuments

Description:

Method	*Return values*	*Description*
getLastWeekTopDownloadedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the top downloaded documents during the last week by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastWeekTopDownloadedD
                System.out.println(dashboardDocumentResult);
            }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastMonthTopDownloadedDocuments

Description:

Method	*Return values*	*Description*
getLastMonthTopDownloadedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the top downloaded documents during the last month by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
```

```
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastMonthTopDownloaded
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastWeekTopModifiedDocuments

Description:

Method	*Return values*	*Description*
getLastWeekTopModifiedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the top modified documents during the last week by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastWeekTopModifiedDocu
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastMonthTopModifiedDocuments

Description:

Method	*Return values*	*Description*
getLastMonthTopModifiedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the top modified documents during the last month by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
```

```
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastMonthTopModifiedDoc
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastModifiedDocuments

Description:

Method	*Return values*	*Description*
getLastModifiedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the last modified documents by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
```

```
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastModifiedDocuments(nu
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastCreatedDocuments

Description:

Method	*Return values*	*Description*
getLastCreatedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the last created documents by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
```

```
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastCreatedDocuments(nu
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastCheckedOutDocuments

Description:

Method	*Return values*	*Description*
getLastCheckedOutDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the last documents on edition by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastCheckedOutDocument
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastCreatedFolders

Description:

Method	*Return values*	*Description*
getLastCreatedFolders(String token, long offset, long limit)	***List<DashboardFolderResult>***	*Get the list of the last created folders by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardFolderResult dashboardFolderResult : okmDashboard.getLastCreatedFolders(null, 0, 5)) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastCreatedRecords

Description:

Method	*Return values*	*Description*
getLastCreatedRecords(String token, long offset, long limit)	***List<DashboardRecordResult>***	*Get the list of the last created records by all the users.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardRecordResult dashboardRecordResult : okmDashboard.getLastCreatedRecords(null, 0, 5)) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

visitNode

Description:

Method	*Return values*	*Description*
visitNode(String token, String source, String node, Calendar date)	***void***	*Mark a node as a visited.*

> *From UI, the nodes are shown strong until are visited by the user. This method mark the node as visited by the user in the database.*

Available sources:

- *LastWeekTopDownloadedDocuments*
- *LastMonthTopDownloadedDocuments*
- *LastWeekTopModifiedDocuments*
- *LastMonthTopModifiedDocuments*
- *LastModifiedDocuments*
- *LastCreatedDocuments*
- *LastCreatedFolders*
- *LastCreatedRecords*
- *LastCheckoutDocuments*
- *UserLockedDocuments*
- *UserLockedRecord*
- *UserCheckedOutDocuments*
- *UserLastModifiedDocuments*
- *UserLastDownloadedDocuments*
- *UserSubscribedDocuments*
- *UserSubscribedFolders*
- *UserSubscribedRecords*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

- *UserLastCreatedDocuments*
- *UserLastCreatedFolders*
- *UserLastCreatedRecords*
- *UserLastImportedMails*
- *UserLastImportedMailAttachments*
- *In case "user new", the source name is the name of the query.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMDashboard;
import com.openkm.util.ContextWrapper;

public class Test {

 public static void main(String[] args) {
  try {
   OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
   Calendar cal = Calendar.getInstance();
   okmDashboard.visitNode(null, "LastWeekTopDownloadedDocuments", "/okm:root/document.pdf", cal);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMDocument

Basics

*On almost methods you'll see parameter named "**docId**". The value of this parameter can be some valid document **UUID** node.*

Example of docId:

- *Using UUID -> "**c41f9ea0-0d6c-45da-bae4-d72b66f42d0f**";*

*About the parameter named "**fldId**", the value of this parameter can be some valid folder **UUID or** record node.*

Example of fldId:

- *Using UUID -> "**c6785440-0d6c-45da-1234-d9874563d0f**";*

*About the parameter named "**dstId**", the value of this parameter can be some valid folder **UUID or** record node.*

Example of dstId:

- *Using UUID -> "**c4455667-0d6c-45da-1234-d98564323e0**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

create

Description:

Method	*Return values*	*Description*
create(String token, Document doc, InputStream is)	***Document***	*Create a new document.*
The parameter doc must be initialized with the OpenKM path value into the repository. Optionally you can also set		

keywords and title values.



The other parameteres are not taken in consideration for document creation.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMDocument;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        InputStream is = null;
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            is = new FileInputStream("/home/openkm/sample.pdf");
            Document doc = new Document();
            doc.setPath("/okm:root/sample.pdf");
            // Optional
            doc.setTitle("sample document");
            doc.getKeywords().add("key1");
            doc.getKeywords().add("key2");
            // Create document
            doc = okmDocument.create(null, doc, is);
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
        }
    }
}
```

createSimple

Description:

Method	*Return values*	*Description*
createSimple(String token, String docPath, InputStream is)	***Document***	*Create a new document.*
The parameter docPath must be initialized with the OpenKM path value into the repository.		

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMDocument;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        InputStream is = null;
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);

            is = new FileInputStream("/home/openkm/sample.pdf");
            String docPath = "/okm:root/sample.pdf";
            Document doc = okmDocument.createSimple(null, docPath, is);
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
        }
    }
}
```

delete

Description:

Method	*Return values*	*Description*
delete(String token, String docId)	***void***	*Delete a document.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.delete(null, "d50133e3-dbfa-4d01-a109-28785cd48f40");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	*Return values*	*Description*
getProperties(String token, String docId)	***Document***	*Return a document properties.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            Document doc = okmDocument.getProperties(null, "3887ae75-dd67-4019-b4d1-2f6962815403");
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContent

Description:

Method	*Return values*	*Description*
getContent(String token, String docId, boolean checkout)	***InputStream***	*Return the document content.*
When paramer checkout is true, the document is marked for edition.		

Example:

```
package com.openkm;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    public static void main(String[] args) {
        InputStream is = null;
        OutputStream fos = null;
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            is = okmDocument.getContent(null, "3887ae75-dd67-4019-b4d1-2f6962815403", false);
            File tmp = new File("/home/openkm/sample.pdf");
            fos = new FileOutputStream(tmp);
            IOUtils.copy(is, fos);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        }
    }
}
```

getContentByVersion

Description:

Method	*Return values*	*Description*
getContentByVersion(String token, String docId, String versionId)	***InputStream***	*Return the document content by version.*
The parameter versionIs is the name of the document version.		

Example:

```
package com.openkm;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        InputStream is = null;
        OutputStream fos = null;
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            is = okmDocument.getContentByVersion(null, "f3a977f7-abfc-4f58-a033-988903b31473", "1.1");
            File tmp = new File("/home/openkm/sample.pdf");
            fos = new FileOutputStream(tmp);
            IOUtils.copy(is, fos);
        } catch (Exception e) {
```

```
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        }
    }
}
```

getExtractedText

Description:

Method	*Return values*	*Description*
getExtractedText(String token, String docId)	***String***	*Return the extracted text of the document.*



When document is upload to OpenKM goes into search engine queue to be processed. Until the process be finished this method will return an empty string.

From Database query view you can execute a query to verify is a document has yet been processed:

```
SELECT NDC_TEXT_EXTRACTED FROM OKM_NODE_DOCUMENT where NBS_UUID='1048eb46-2f32-4011-90e4-e084d838a592'
```

NBS_UUID value is the document UUID value. You can get it from properties tab in desktop view.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            String indexedText = okmDocument.getExtractedText(null, "f3a977f7-abfc-4f58-a033-988903b31473");
            System.out.println(indexedText);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Descripcion:

Method	*Return values*	*Description*

getChildren(String token, String fldId)	***List<Document>***	*Returns a list of all documents which their parent is fldId.*
The parameter fldId can be a folder or a record node.		

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            for (Document doc : okmDocument.getChildren(null, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	*Return values*	*Description*
rename(String token, String docId, String newName)	***Document***	*Changes the name of a document.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            Document doc = okmDocument.rename(null, "f3a977f7-abfc-4f58-a033-988903b31473", "samples_renamed
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

setProperties

Description:

Method	*Return values*	*Description*
setProperties(String token, Document doc)	***void***	*Changes some document properties.*

Variables allowed to be changed:

- *Title*
- *Description*
- *Language*
- *Associated categories*
- *Associated keywords*



Only not null and not empty variables will be taken in consideration.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            Document doc = okmDocument.getProperties(null, "f3a977f7-abfc-4f58-a033-988903b31473");
            doc.getKeywords().add("key1");
            doc.setDescription("some description");
            okmDocument.setProperties(null, doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setLanguage

Description:

Method	Return values	Description
setLanguage(String token, String docId, String lang)	**void**	*Sets the document language.*

The parameter lang must be ISO 691-1 compliant.

> *More information at: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.setLanguage(null, "f3a977f7-abfc-4f58-a033-988903b31473", "en");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTitle

Description:

Method	Return values	Description
setTitle(String token, String docId, String title)	**void**	*Sets the document title.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.setTitle(null, "f3a977f7-abfc-4f58-a033-988903b31473", "Document title");
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

checkout

Description:

Method	*Return values*	*Description*
checkout(String token, String docId)	***void***	*Marks the document for edition.*

Only one user can modify a document at a time.

Before starting edition you must do a checkout action that locks the edition process for other users and allows edition only to the user who has executed the action.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.checkout(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

cancelCheckout

Description:

Method	*Return values*	*Description*
cancelCheckout(String token, String docId)	***void***	*Cancels a document edition.*



This action can only be done by the user who previously executed the checkout action.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.cancelCheckout(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceCancelCheckout

Description:

Method	*Return values*	*Description*
forceCancelCheckout(String token, String docId)	***void***	*Cancels a document edition.*

This method allows to cancel edition on any document.

It is not mandatory to execute this action by the same user who previously executed the checkout action.



This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.forceCancelCheckout(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isCheckedOut

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
isCheckedOut(String token, String docId)	***void***	*Returns a boolean that indicates if the document is on edition or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            System.out.println(okmDocument.isCheckedOut(null, "f3a977f7-abfc-4f58-a033-988903b31473"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkin

Description:

Method	*Return values*	*Description*
checkin(String token, String docId, InputStream is, String comment)	***Version***	*Updates a document with a new version and returns an object with new Version values.*

> *Only the user who started the edition - checkout - is allowed to update the document.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;
```

```
import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        InputStream is = null;
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            is = new FileInputStream("/home/openkm/sample.pdf");
            okmDocument.checkin(null, "f3a977f7-abfc-4f58-a033-988903b31473", is, "optional comment");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
        }
    }
}
```

checkin

Description:

Method	*Return values*	*Description*
checkin(String token, String docId, InputStream is, String comment, int increment)	***Version***	*Updates a document with a new version and returns an object with new Version values.*

The increment value depends on VersionNumeration implementation. Common values are:

- *VersionNumerationAdapter.INCREASE_DEFAULT for default.*
- *VersionNumerationAdapter.INCREASE_MINOR for increasing a minor version.*
- *VersionNumerationAdapter.INCREASE_MAJOR for increasing a major version.*
- *VersionNumerationAdapter.INCREASE_MAJOR_MINOR for increasing a major-minor version.*



More information about VersionNumeration at Creating your own Version Number Adapter.



Only the user who started the edition - checkout - is allowed to update the document.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;
```

```
import com.openkm.api.OKMDocument;
import com.openkm.plugin.vernum.VersionNumerationAdapter;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        InputStream is = null;
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            is = new FileInputStream("/home/openkm/sample.pdf");
            okmDocument.checkin(null, "f3a977f7-abfc-4f58-a033-988903b31473", is, "optional comment",
                    VersionNumerationAdapter.INCREASE_MINOR);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
        }
    }
}
```

lock

Description:

Method	*Return values*	*Description*
lock(String token, String docId)	***LockInfo***	*Locks a document and returns an object with the Lock information.*



Only the user who locked the document is allowed to unlock.

A locked document cannot be modified by other users.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.LockInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            LockInfo lockInfo = okmDocument.lock(null, "f3a977f7-abfc-4f58-a033-988903b31473");
            System.out.println(lockInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

unlock

Description:

Method	*Return values*	*Description*
unlock(String token, String docId)	***LockInfo***	*Unlocks a locked document.*



Only the user who locked the document is allowed to unlock.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.unlock(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceUnlock

Description:

Method	*Return values*	*Description*
forceUnlock(String token, String docId)	***LockInfo***	*Unlocks a locked document.*

This method allows to unlock any locked document.

It is not mandatory execute this action by the same user who previously executed the checkout lock action.



This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.forceUnlock(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isLocked

Description:

Method	*Return values*	*Description*
isLocked(String token, String docId)	***void***	*Returns a boolean that indicates if the document is locked or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            System.out.println(okmDocument.isLocked(null, "f3a977f7-abfc-4f58-a033-988903b31473"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLockInfo

Description:

Method	*Return values*	*Description*
getLockInfo(String token, String docId)	***LockInfo***	*Returns an object with the Lock information.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.LockInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            LockInfo lockInfo = okmDocument.getLockInfo(null, "f3a977f7-abfc-4f58-a033-988903b31473");
            System.out.println(lockInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

Method	*Return values*	*Description*
purge(String token, String docId)	***void***	*The document is definitely removed from repository.*

Usually you will purge documents into /okm:trash/userId - the personal trash user locations - but is possible to directly purge any document from the whole repository.

> *When a document is purged only will be able to be restored from a previously repository backup. The purge action removes the document definitely from the repository.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.purge(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

move

Description:

Method	*Return values*	*Description*
move(String token, String docId, String dstId)	***void***	*Move document into some folder or record.*
The values of the dstId parameter should be a folder or record UUID.		

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.move(null, "f3a977f7-abfc-4f58-a033-988903b31473", "b67fe3ae-bcca-4947-b8ea-0cc7aedd
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String docId, String dstId)	***void***	*Copy a document into some folder or record.*

The values of the dstId parameter should be a folder or record UUID.

> *Only the binary data and the security grants are copied to destination, the metadata, keywords, etc. of the document are not copied.*
>
> *See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.copy(null, "f3a977f7-abfc-4f58-a033-988903b31473", "b67fe3ae-bcca-4947-b8ea-0cc7aedd4
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String docId, String dstId, String newName)	***void***	*Copy a document into some folder or record.*

The values of the dstId parameter should be a folder or record UUID.

When the parameter newName value is null,the document will preservate the same name.



Only the binary data and the security grants are copied to destination, the metadata, keywords, etc. of the document are not copied.

*See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.copy(null, "f3a977f7-abfc-4f58-a033-988903b31473", "b67fe3ae-bcca-4947-b8ea-0cc7aedd4
                    "samples_renamed.pdf");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedCopy

Description:

Method	*Return values*	*Description*
extendedCopy(String token, String docId, String dstId, String docName, ExtendedAttributes extAttr)	***void***	*Copies a document with associated data into some folder or record.*

The values of the dstId parameter should be a folder or a record UUID.

When the parameter docName value is null, the document will preserve the same name.



By default only the binary data and the security grants, the metadata, keywords, etc. of the document are not copied.

Additional:

- *When the category parameter is true the original values of the categories will be copied.*
- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original value of the notes will be copied.*
- *When wiki parameter is true the original value of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            ExtendedAttributes extAttr = new ExtendedAttributes();
            // copy keywords
            extAttr.setKeywords(true);
            extAttr.setCategories(true);
            extAttr.setNotes(true);
            extAttr.setPropertyGroups(true);

            okmDocument.extendedCopy(null, "f3a977f7-abfc-4f58-a033-988903b31473", "b67fe3ae-bcca-4947-b8ea-0
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getVersionHistorySize

Description:

Method	*Return values*	*Description*
getVersionHistorySize(String token, String docId)	***long***	*Returns the sum in bytes of all documents into documents history.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FormatUtil;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            System.out.println(FormatUtil.formatSize(okmDocument.getVersionHistorySize(null, "f3a977f7-abfc-4f58-a0
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	*Return values*	*Description*
isValid(String token, String docId)	***boolean***	*Returns a boolean that indicates if the node is a document or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            boolean valid = okmDocument.isValid(null, "f3a977f7-abfc-4f58-a033-988903b31473");
            System.out.println(valid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPath

Description:

Method	*Return values*	*Description*
getPath(String token, String uuid)	***String***	*Converts a document UUID to document path.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            String docPath = okmDocument.getPath(null, "f3a977f7-abfc-4f58-a033-988903b31473");
            System.out.println(docPath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDetectedLanguages

Description:

Method	*Return values*	*Description*
getDetectedLanguages(String token)	***List<String>***	*Return a list of available document languages what OpenKM can identify.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            for (String languages : okmDocument.getDetectedLanguages(null)) {
                System.out.println(languages);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFromTemplate

Description:

Method	*Return values*	*Description*
createFromTemplate(String token, String docId, String dstPath, Map<String, String> properties, ExtendedAttributes attributes)	***Document***	*Creates a new document from template and returns an object Document.*

The dstPath can be a folder or a record UUID.

The parameter language is optional.

When the template uses metadata groups to fill fields into, then these values are mandatory and must be set into properties parameter.

For more information about Templates and metadata check: Creating templates.

Additional attribute parameters:

- *When category parameter is true the original values of the categories will be copied.*
- *When keywords parameter is true the original values of the keywords will be copied.*
- *When propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When notes parameter is true the original values of the notes will be copied.*

- *When wiki parameter is true the original values of the wiki will be copied.*

Example:

The example below is based on Creating PDF template sample.

```
package com.openkm;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;
import java.util.List;

import com.google.gson.Gson;
import com.openkm.api.OKMDocument;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;
import com.openkm.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:tpl.name", "Some name");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));

            List<String> list = new ArrayList<>();
            list.add("java");
            properties.put("okp:tpl.language", new Gson().toJson(list).toString());

            okmDocument.createFromTemplate(null, "86633fe8-1b0d-48ac-a716-7685022adad8", "/okm:root/templates
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateFromTemplate

Description:

Method	*Return values*	*Description*
updateFromTemplate(String token, String docId, String dstId, Map<String, String> properties)	***void***	*Updates a document previously created from the template and returns an object Document.*

This method only has sense when the template uses metadata groups to fill fields in.

For more information about Templates and metadata check: Creating templates .

Example:

The example below is based on Creating PDF template sample.

```
package com.openkm;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.google.gson.Gson;
import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;
import com.openkm.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:tpl.name", "Some name");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));

            List<String> list = new ArrayList<>();
            list.add("java");
            properties.put("okp:tpl.language", new Gson().toJson(list).toString());

            okmDocument.updateFromTemplate(null, "86633fe8-1b0d-48ac-a716-7685022adad8", "b67fe3ae-bcca-494
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setAnnotations

Description:

Method	*Return values*	*Description*
setAnnotations(String token, String docId, String verName, String annotation)	***void***	*Set document annotations of some document version.*

Example:

```
package com.openkm;
import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.setAnnotations(null, "3887ae75-dd67-4019-b4d1-2f6962815403", "1.1", "The annotation text.
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAnnotations

Description:

Method	*Return values*	*Description*
getAnnotations(String token, String docId, String verName)	***String***	*Return the document annotations of some document version.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            String annotation = okmDocument.getAnnotations(null, "3887ae75-dd67-4019-b4d1-2f6962815403", "1.1");
            System.out.println(annotation);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCheckedOut

Description:

Method	*Return values*	*Description*

getCheckedOut(String token)	***List<Document>***	*Return the list of all documents checkout in the repository*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            for (Document doc : okmDocument.getCheckedOut(null)) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stamp

Description:

Method	*Return values*	*Description*
stamp(String token, String docId, int type, long stId)	***void***	*Stamp a document.*

The parameter type indicates stamp type:

- *"text" with value 0;*
- *"image" with value 1;*

The parameter stId is the unique identifier value of the stamp definition. You can get it from Administration > Stamp view.

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            int type = 0; // indicate text stamp definition
```

```
            long stId = 1; // unique identifier of the stamp definition
            okmDocument.stamp(null, "3887ae75-dd67-4019-b4d1-2f6962815403", type, stId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampText

Description:

Method	*Return values*	*Description*
stampText(String token, String docId, long stId, String range, String exprX, String exprY, String text)	***void***	*Stamp in a document with a custom text.*

> *The **id** is the id of an image stamp which is created in the administrator.*
>
> *The **exprX** is the expression to set the position in X coordinates.*
>
> *The **exprY** is the expression to set the position in Y coordinates.*

> *In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:*
>
> - *IMAGE_WIDTH*
> - *IMAGE_HEIGHT*
> - *PAGE_WIDTH*
> - *PAGE_HEIGHT*
> - *PAGE_CENTER*
> - *PAGE_MIDDLE*
>
> *So to center a stamp in the page you can use:*
>
> | | |
> |---|---|
> | ***Expr. X*** | *PAGE_CENTER* |
> | ***Expr. Y*** | *PAGE_MIDDLE* |

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;
```

```
import java.io.FileInputStream;
import java.io.InputStream;

public class Test {

    public static void main(String[] args) {
        try {
            long stampTextId = 1;
            String text = "OpenKM";
            String exprX = "PAGE_CENTER";
            String exprY = "PAGE_MIDDLE";
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.stampText(null, "05b14eee5-2e57-4d1d-925a-c9bc3f526e24", stampTextId, "", exprX, exprY,
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampImage

Description:

Method	*Return values*	*Description*
stampImage(String token, String docId, long stId, String range, String exprX, String exprY, InputStream is)	***void***	*Stamp in a document with a custom image.*

*The **id** is the id of an image stamp which is created in the administrator.*

*The **exprX** is the expression to set the position in X coordinates.*

*The **exprY** is the expression to set the position in Y coordinates.*

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- *IMAGE_WIDTH*
- *IMAGE_HEIGHT*
- *PAGE_WIDTH*
- *PAGE_HEIGHT*
- *PAGE_CENTER*
- *PAGE_MIDDLE*

So to center a stamp in the page you can use:

Expr. X	*PAGE_CENTER*

	Expr. Y	PAGE_MIDDLE	

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

import java.io.FileInputStream;
import java.io.InputStream;

public class Test {

    public static void main(String[] args) {
        try {
            long id = 1;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.stampImage(null, "055b5206-35d0-4351-ba92-c304bbba7ddf", id, "", exprX, exprY, is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	Return values	Description
setDescription(String token, String docId, String description)	**void**	*Set a description.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.setDescription(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

setNodeClass

Description:

Method	*Return values*	*Description*
setNodeClass(String token, String docId, long ncId)	***void***	*Set the NodeClass.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            long ncId = 2;
            okmDocument.setNodeClass(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDispositionStage

Description:

Method	*Return values*	*Description*
setDispositionStage(String token, String docId, long stage)	***void***	*Set the disposition stage*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            long stage = 1;
            okmDocument.setDispositionStage(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

createWizard

Description:

Method	*Return values*	*Description*
createWizard(String token, String docPath, long nodeClass, InputStream is)	***WizardNode***	*Create a new document with wizard.*

*The parameters **docPath** should be any valid folder or record **PATH**.*

> *The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:*
>
> - *Add keyword*
> - *Add Categories*
> - *Add Metadata*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMDocument;
import com.openkm.bean.WizardNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            WizardNode wn = okmDocument.createWizard(null, "/okm:root/test/logo.png", 0, is);
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNumberOfPages

Description:

Method	*Return values*	*Description*
getNumberOfPages(String token, String uuid)	***int***	*Get the number of pages of a document.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            System.out.println("The number of pages is : " + okmDocument.getNumberOfPages(null, "b2f88679-e3fd-4f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPageAsImage

Description:

Method	*Return values*	*Description*
getPageAsImage(String token, String uuid, int pageNumber, int maxWidth, int maxHeight)	***String***	*Return specific page as an image encoded as a String in B64.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            System.out.println(okmDocument.getPageAsImage(null, "3fe350e2-69e8-4681-a97c-6ac4ba6c1524", 1, 150
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

isConvertibleToPDF

Description:

Method	*Return values*	*Description*
isConvertibleToPDF(String docId)	***Boolean***	*Returns a boolean that indicates if the node is convertible to PDF or not.*



*In case of a node of type **PDF file** will be returned a **false** value.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            System.out.println(okmDocument.isConvertibleToPDF("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setIndexable

Description:

Method	*Return values*	*Description*
setIndexable(String uuid, boolean enabled)	***void***	*Sets the document indexable.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
```

```
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.setIndexable("19938640-fd34-421c-90c8-6b435e4b7d79", true);
            Document document = okmDocument.getDocumentProperties("19938640-fd34-421c-90c8-6b435e4b7d79")
            System.out.println("Document indexable:" + document.isIndexable());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

mergePdf

Description:

Method	*Return values*	*Description*
mergePdf(String token, String destinationPath, String docName, List<String> paths)	***void***	*Merge two or more PDF files.*
The parameters ***destinationPath*** *should be any valid folder or record PATH.*		

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            List<String> uuids = new ArrayList<>();
            uuids.add("/okm:root/test/document.pdf");
            uuids.add("/okm:root/test/document1.pdf");
            String fldUuid = "/okm:root/mergue";
            okmDocument.mergePdf(null, fldUuid, "MergePdf.pdf", uuids);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMFolder

Basics

*On almost methods you'll see parameter named "**fldId**". The value of this parameter can be some valid folder **UUID**.*

Example of fldId:

- *Using UUID -> "**b67fe3ae-bcca-4947-b8ea-0cc7aedd4474**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

create

Description:

Method	*Return values*	*Description*
create(String token, Folder fld)	***Folder***	*Creates a new folder and returns as a result an object Folder.*

*The variable **path** into the parameter **fld**, must be initializated. It indicates the folder path into OpenKM.*

```
Folder fld = new Folder();
fld.setPath("/okm:root/test");
```

The other variables of the Folder (fld) will not take any effect on the folder creation.

We suggest using the method below to create the Folder Simply rather this one.

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            Folder fld = new Folder();
            fld.setPath("/okm:root/test");
            okmFolder.create(null, fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createSimple

Description:

Method	*Return values*	*Description*
createSimple(String token, String fldPath)	***Folder***	*Creates a new folder and returns as a result an object Folder.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            String fldPath = "/okm:root/test";
            okmFolder.createSimple(null, fldPath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	*Return values*	*Description*
getProperties(String token, String fldId)	***Folder***	*Return the folder properties.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            Folder fld = okmFolder.getProperties(null, "9f64248c-e049-4706-b89e-6d725842c7f7");
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	*Return values*	*Description*
delete(String token, String fldId)	***void***	*Delete a folder.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.delete(null, "9f64248c-e049-4706-b89e-6d725842c7f7");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

Method	*Return values*	*Description*

purge(String token, String fldId)	***void***	*The folder is definitely removed from the repository.*

Usually you will purge folders into /okm:trash/userId - the personal trash user locations - but it is possible to directly purge any folder from the whole repository.



When a folder is purged, it will only be able to be restored from a previously repository backup. The purge action removes the folder definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.delete(null, "9f64248c-e049-4706-b89e-6d725842c7f7");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	*Return values*	*Description*
rename(String token, String fldId, String newName)	***Folder***	*Rename a folder.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.rename(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263", "newname");
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

move

Description:

Method	*Return values*	*Description*
move(String token, String fldId, String dstId)	***void***	*Moves folder into some folder or record*
The values of the dstId parameter should be a folder or record UUID.		

Example:

```
package com.openkm;
import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.move(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263", "b67fe3ae-bcca-4947-b8ea-0cc7aedd44
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String fldId, String dstId)	***void***	*Copies a folder into a folder or record.*

The values of the dstId parameter should be a folder or record UUID.



Only the security grants are copied to destination, the metadata, keywords, etc. of the folder are not copied.

*See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.copy(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263", "b67fe3ae-bcca-4947-b8ea-0cc7aedd44
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String fldId, String dstId, String newName)	***void***	*Copies a folder into a folder or record.*

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, folder will preservate the same name.



Only the security grants are copied to destination, the metadata, keywords, etc. of the folder are not copied.

*See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.copy(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263", "b67fe3ae-bcca-4947-b8ea-0cc7aedd44
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedCopy

Description:

Method	*Return values*	*Description*
extendedCopy(String token, String fldId, String dstId, ExtendedAttributes extAttr)	***void***	*Copies a folder with the associated data into some folder or record.*

The values of the dstId parameter should be a folder or record UUID.

By default only the binary data and the security grants, the metadata, keywords, etc. of the folder are not copyed.

Additional extended attributes parameters:

- *When the category parameter is true the original values of the categories will be copied.*
- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original values of the notes will be copied.*
- *When the wiki parameter is true the original values of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);

            ExtendedAttributes extAttr = new ExtendedAttributes();
            extAttr.setKeywords(true);
            extAttr.setNotes(true);
            extAttr.setCategories(true);
            extAttr.setPropertyGroups(true);

            okmFolder.extendedCopy(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263", "b67fe3ae-bcca-4947-b8ea-0c
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

extendedCopy

Description:

Method	*Return values*	*Description*
extendedCopy(String token, String fldId, String dstId, ExtendedAttributes extAttr, String newName)	***void***	*Copies a folder with the associated data into some folder or record.*

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, folder will preservate the same name.

By default only the binary data and the security grants, the metadata, keywords, etc. of the folder are not copyed.

Additional extended attributes parameters:

- *When the category parameter is true the original values of the categories will be copied.*
- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original values of the notes will be copied.*
- *When the wiki parameter is true the original values of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);

            ExtendedAttributes extAttr = new ExtendedAttributes();
            extAttr.setKeywords(true);
            extAttr.setNotes(true);
            extAttr.setCategories(true);
            extAttr.setPropertyGroups(true);
```

```
            okmFolder.extendedCopy(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263", "b67fe3ae-bcca-4947-b8ea-0c
                    extAttr, "newname");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

getChildren

Description:

Method	*Return values*	*Description*
getChildren(String token, String fldId)	***List<Folder>***	*Returns a list of all folders which their parent is fldId.*
The parameter fldId can be a folder or a record node.		

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            for (Folder fld : okmFolder.getChildren(null, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContentInfo

Description:

Method	*Return values*	*Description*
getContentInfo(String token, String fldId)	***ContentInfo***	*Return and object ContentInfo with information about folder.*

The ContentInfo object retrives information about:

- *The number of folders into.*
- *The number of documents into.*
- *The number of records into.*
- *The number of mails into.*
- *The size in bytes of all objects into the folder.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.bean.ContentInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            ContentInfo ci = okmFolder.getContentInfo(null, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474");
            System.out.println(ci);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	*Return values*	*Description*
isValid(String token, String fldId)	***boolean***	*Returns a boolean that indicates if the node is a folder or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            boolean valid = okmFolder.isValid(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263");
            System.out.println(valid);
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPath

Description:

Method	*Return values*	*Description*
getPath(String token, String uuid)	***String***	*Convert folder UUID to folder path.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            String path = okmFolder.getPath(null, "7aceb86c-1dc8-4551-aebd-b7bd73d01263");
            System.out.println(path);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setStyle

Description:

Method	*Return values*	*Description*
setStyle(String token, String fldId, long styleId)	***void***	*Set the folder style.*

More information at Folder style.

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.setStyle(null, "/okm:root/test", 1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createMissingFolders

Description:

Method	*Return values*	*Description*
createMissingFolders(String token, String fldPath)	***void***	*Create missing folders based on all the path nodes.*

> *The method checks all subfolders and when detects some one is missing, it creates all missing subfolders from there.*

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.createMissingFolders(null, "/okm:root/test/missing/from/here");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addNodeClassChildren

Description:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Method	Return values	Description
addNodeClassChildren(String token, String fldId, long ncId)	**void**	Add the NodeClass.

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            long ncId = 2;
            okmFolder.addNodeClassChildren(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	Return values	Description
setDescription(String token, String uuid, String description)	**void**	Set a description.

Example:

```
package com.openkm;

import com.openkm.api.OKMFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            okmFolder.setDescription(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMImport

Basics

*On almost methods you'll see parameter named "**parentUuid**". The value of this parameter can be some valid document **UUID** node.*

Example of docId:

- *Using UUID -> "**c41f9ea0-0d6c-45da-bae4-d72b66f42d0f***

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

createDocument

Description:

Method	*Return values*	*Description*
createDocument(String token, String parentUuid, String docName, File file)	***NodeDocument***	*Create a new document.*
*The parameter **parentUuid** must be a **valid folder or record UUID**.* *The parameter **file** is the file of the document.*		

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
import com.openkm.api.OKMImport;
import com.openkm.db.bean.NodeDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMImport okmImport = ContextWrapper.getContext().getBean(OKMImport.class);
            File doc = new File("/home/openkm/sample.pdf");
            NodeDocument nodeDoc = okmImport.createDocument(null, "0a19e616-1d11-4ef7-a7c7-f1e109f8ea3c", "do
            System.out.println(nodeDoc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFolder

Description:

Method	*Return values*	*Description*
createDocument(String token, String parentUuid, String fldName)	***NodeFolder***	*Create a new folder.*
*The parameter **parentUuid** must be a **valid folder or record UUID**.* *The parameter **fldName** is the name of the folder.*		

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMImport;
import com.openkm.db.bean.NodeFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMImport okmImport = ContextWrapper.getContext().getBean(OKMImport.class);
            NodeFolder nodeFld = okmImport.createFolder(null, "0a19e616-1d11-4ef7-a7c7-f1e109f8ea3c", "newFolder
            System.out.println(nodeFld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

createRecord

Description:

Method	*Return values*	*Description*
createRecord(String token, String parentUuid, String recName)	*NodeRecord*	*Create a new record.*
*The parameter **parentUuid** must be a **valid folder or record UUID**.* *The parameter **recName** is the name of the record.*		

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMImport;
import com.openkm.db.bean.NodeRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMImport okmImport = ContextWrapper.getContext().getBean(OKMImport.class);
            NodeRecord nodeRec = okmImport.createRecord(null, "0a19e616-1d11-4ef7-a7c7-f1e109f8ea3c", "newRec
            System.out.println(nodeRec);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createMail

Description:

Method	*Return values*	*Description*
createMail(String token, String parentUuid, File file)	*NodeMail*	*Create a new mail.*
*The parameter **parentUuid** must be a **valid folder or record UUID**.* *The parameter **file** is the file of the mail.*		

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMImport;
import com.openkm.db.bean.NodeMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMImport okmImport = ContextWrapper.getContext().getBean(OKMImport.class);
            NodeMail nodeMail = okmImport.createMail(null, "0a19e616-1d11-4ef7-a7c7-f1e109f8ea3c", "newMailName
            System.out.println(nodeMail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importFolder

Description:

Method	*Return values*	*Description*
importFolder(String token, String parentUuid, File folder, int threads)	***void***	*Import local folder hierarchy.*

*The parameter **parentUuid** must be a **valid folder or record UUID**.*

*The value **threads** set the **number of concurrent threads** used by import feature. This value should **never be upper than the number of the server cores**. If you have 20 cores a good starting point might be use 10 threads, take in consideration other software like database, OS etc... are also using the hardware.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMImport;
```

```
import com.openkm.db.bean.NodeFolder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMImport okmImport = ContextWrapper.getContext().getBean(OKMImport.class);
            File fld = new File("/home/openkm/import/");
            okmImport.importFolder(null, "0a19e616-1d11-4ef7-a7c7-f1e109f8ea3c", fld, 4);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMMail

Basics

*On almost methods you'll see parameter named "**mailId**". The value of this parameter can be some valid mail **UUID**.*

Example of mailId:

- *Using UUID -> "**0747c0cb-5e4a-4088-92a9-ab97b401af75**";*

*About the parameter named "**fldId**", the value of this parameter can be some valid folder **UUID** or record node.*

Example of fldId:

- *Using UUID -> "**b67fe3ae-bcca-4947-b8ea-0cc7aedd4474**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

getProperties

Description:

Method	*Return values*	*Description*
getProperties(String token, String mailId)	***Mail***	*Returns the mail properties.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            Mail mail = okmMail.getProperties(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75");
            System.out.println(mail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createAttachment

Description:

Method	*Return values*	*Description*
createAttachment(String token, String mailId, String docName, InputStream is)	***Document***	*Add a attachment into the mail.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.poi.util.IOUtils;

import com.openkm.api.OKMMail;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        InputStream is = null;
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            is = new FileInputStream("/home/openkm/sample.pdf");
            Document doc = okmMail.createAttachment(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "sample.pdf",
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
        }
    }
}
```

deleteAttachment

Description:

Method	*Return values*	*Description*

deleteAttachment(String token, String mailId, String docId)	***void***	*Delete a attachment into the mail.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.deleteAttachment(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "c98d704f-c9f2-4f75-9923-878
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAttachments

Description:

Method	*Return values*	*Description*
getAttachments(String token, String mailId)	***List<Document>***	*Returns a list of all attachments of a mail.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            for (Document doc : okmMail.getAttachments(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Method	Return values	Description
delete(String token, String mailId)	**void**	Delete a mail.

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.delete(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

Method	Return values	Description
purge(String token, String mailId)	**void**	The mail is definitely removed from the repository.

Usually you will purge mails into /okm:trash/userId - the personal trash user locations - but it is possible to directly purge any mail from the whole repository.



When a mail is purged, it will only be able to be restored from a previously repository backup. The purge action removes the mail definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
```

```
            okmMail.purge(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	*Return values*	*Description*
rename(String token, String mailId, String newName)	***Mail***	*Rename a mail.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.rename(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "newname");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

move

Description:

Method	*Return values*	*Description*
move(String token, String mailId, String dstId)	***void***	*Moves mail into some folder or record*
The values of the dstId parameter should be a folder or record UUID.		

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.move(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "3636f3bb-779c-4851-b145-0d53e0702b0f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String mailId, String dstId)	***void***	*Copies a mail into a folder or record.*

The values of the dstId parameter should be a folder or record UUID.

> *Only the security grants are copied to destination, the metadata, keywords, etc. of the mail are not copied.*
>
> *See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.copy(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "3636f3bb-779c-4851-b145-0d53e0702b0f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*

copy(String token, String mailId, String dstId, String newName)	***void***	*Copies a mail into a folder or record.*

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, mail will preservate the same name.



Only the security grants are copied to destination, the metadata, keywords, etc. of the mail are not copied.

*See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.copy(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "3636f3bb-779c-4851-b145-0d53e0702b0f"
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedCopy

Description:

Method	*Return values*	*Description*
extendedCopy(String token, String mailId, String dstId, ExtendedAttributes extAttr)	***void***	*Copies a mail with the associated data into some folder or record.*

The values of the dstId parameter should be a folder or record UUID.



By default only the binary data and the security grants, the metadata, keywords, etc. of the mail are not copyed.

Additional extended attributes parameters:

- *When the category parameter is true the original values of the categories will be copied.*

- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original values of the notes will be copied.*
- *When the wiki parameter is true the original values of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            ExtendedAttributes extAttr = new ExtendedAttributes();
            extAttr.setNotes(true);
            extAttr.setKeywords(true);
            extAttr.setCategories(true);
            extAttr.setPropertyGroups(true);
            okmMail.extendedCopy(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "3636f3bb-779c-4851-b145-0d53
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedCopy

Description:

Method	*Return values*	*Description*
extendedCopy(String token, String mailId, String dstId, ExtendedAttributes extAttr, String newName)	***void***	*Copies a mail with the associated data into some folder or record.*

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, mail will preservate the same name.

By default only the binary data and the security grants, the metadata, keywords, etc. of the mail are not copyed.

Additional extended attributes parameters:

- *When the category parameter is true the original values of the categories will be copied.*
- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original values of the notes will be copied.*
- *When the wiki parameter is true the original values of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            ExtendedAttributes extAttr = new ExtendedAttributes();
            extAttr.setNotes(true);
            extAttr.setKeywords(true);
            extAttr.setCategories(true);
            extAttr.setPropertyGroups(true);
            okmMail.extendedCopy(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "3636f3bb-779c-4851-b145-0d53
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Description:

Method	*Return values*	*Description*
getChildren(String token, String fldId)	***List<Mail>***	*Returns a list of all mails which their parent is mailId.*
The parameter mailId can be a folder or a record node.		

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            for (Mail mail : okmMail.getChildren(null, "3636f3bb-779c-4851-b145-0d53e0702b0f")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	*Return values*	*Description*
isValid(String token, String mailId)	***boolean***	*Returns a boolean that indicates if the node is a mail or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            boolean valid = okmMail.isValid(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75");
            System.out.println(valid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPath

Description:

Method	*Return values*	*Description*
getPath(String token, String uuid)	***String***	*Convert folder UUID to mail path.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            String path = okmMail.getPath(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75");
            System.out.println(path);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMail

Description:

Method	*Return values*	*Description*
sendMail(String token, List<String> recipients, String subject, String body)	***void***	*Sends a mail.*

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            StringBuffer body = new StringBuffer();
            body.append("Test message body.");
            List<String> recipients = new ArrayList<>();
            recipients.add("gnu.java.sergio@gmail.com");
            okmMail.sendMail(null, recipients, "Testing sending mail from OpenKM", body.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTitle

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
setTitle(String token, String mailId, String title)	***void***	*Sets the mail title.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.setTitle(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "new title");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMailWithAttachments

Description:

Method	*Return values*	*Description*
sendMailWithAttachments(String token, List<String> toRecipients, List<String> ccRecipients, List<String> bccRecipients, List<String> replyToMails, String subject, String body, List<String> docsId, String dstId)	***Mail***	*Send mail message with attachement.*

The values of the dstId parameter should be a folder or record ***UUID****. The dstId parameter indicate where the mail will be stored in the repository after is sent.*

Other parameters:

- *toRecipients are a list of mail accounts destination.*
- *subject is the mail subject.*
- *ccRecipients, bccRecipients are a list of mail accounts destination (optional)*
- *replyToMails are a list of mails accounts for replying (optional)*
- *docsId are a list of valid document UUID already into OpenKM that will be send as attachment into mail (*

optional).

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            List<String> toRecipients = Arrays.asList("gnujavasergio@mail.com");
            List<String> ccRecipients = new ArrayList<>();
            List<String> bccRecipients = new ArrayList<>();
            List<String> replyToMails = new ArrayList<>();
            List<String> docsId = Arrays.asList("b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "d168924d-e801-426d-a222

            Mail mail = okmMail.sendMailWithAttachments(null, toRecipients, ccRecipients, bccRecipients, replyToMails
                    "some subject", "some body", docsId, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474");
            System.out.println(mail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importEml

Description:

Method	*Return values*	*Description*
importEml(String fldId, String title, InputStream is)	***Mail***	*Import a mail in EML format.*

The values of the fldId parameter should be a folder or record ***UUID****. The path parameter indicate where the mail will be stored in the repository after is imported.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;
```

```
import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

 public static void main(String[] args) {
  InputStream is = null;
       try {
        OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
          is = new FileInputStream("/home/openkm/sample1.eml");
          Mail mail = okmMail.importEml(null, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474", "some title", is);
          System.out.println(mail);
       } catch (Exception e) {
          e.printStackTrace();
       } finally {
          IOUtils.closeQuietly(is);
       }
 }
}
```

importMsg

Description:

Method	*Return values*	*Description*
importMsg(String fldId, String title, InputStream is)	***Mail***	*Import a mail in MSG format.*

The values of the fldId parameter should be a folder or record ***UUID****. The path parameter indicate where the mail will be stored in the repository after is imported.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
       InputStream is = null;
       try {
          OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
          is = new FileInputStream("/home/openkm/test subject.msg");
          Mail mail = okmMail.importMsg(null, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474", "some title", is);
          System.out.println(mail);
       } catch (Exception e) {
          e.printStackTrace();
```

```
        } finally {
            IOUtils.closeQuietly(is);
        }
    }
}
```

setNodeClass

Description:

Method	*Return values*	*Description*
setNodeClass(String token, String uuid, long ncId)	***void***	*Set the NodeClass.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            long ncId = 2;
            okmMail.setNodeClass(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDispositionStage

Description:

Method	*Return values*	*Description*
setDispositionStage(String token, String uuid, long stage)	***void***	*Set the disposition stage*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            long stage = 1;
            okmMail.setDispositionStage(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	*Return values*	*Description*
setDescription(String token, String uuid, String description)	***void***	*Set the description.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            okmMail.setDescription(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "some description");
            Mail mail = okmMail.getProperties(null, "b0747c0cb-5e4a-4088-92a9-ab97b401af75");
            System.out.println(mail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContent

Description:

Method	*Return values*	*Description*
geContent(String token, String mailId)	***InputStream***	*Retrieve mail content (binary data).*

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
```

```
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            OutputStream fos = new FileOutputStream("/home/openkm/test.eml");
            InputStream is = okmMail.getContent(null, "b405a504-d8cb-4166-ac51-22f68acee8c5");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizard

Description:

Method	*Return values*	*Description*
createWizard(String token, String path, String title, InputStream is, String type)	***WizardNode***	*Create a new email with a wizard.*

*The parameters **path** should be any valid folder or record **PATH**.*

Available types:

- *Mail.ORIGIN_MSG*
- *Mail.ORIGIN_EML*

> *The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:*
>
> - *Add keyword*
> - *Add Categories*
> - *Add Metadata*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.bean.WizardNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            InputStream is = new FileInputStream("/home/openkm/sample2.eml");
            WizardNode wn = okmMail.createWizard(null, "/okm:root/test", "test title", is, Mail.ORIGIN_EML);
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getExtractedText

Description:

Method	*Return values*	*Description*
getExtractedText(String token, String uuid)	***String***	*Returns an String with the extracted text by text extractor process.*

When a document is uploaded to OpenKM, it goes into text extraction queue. There's a crontab tab that periodically process this queue and extract document contents.



Unfortunately there is not a direct way to know if a document has been processed or not from the API, because this information is only stored at database level.

Althoughthis restriction, from API - only administrators users - can do database queries to retrieve this information. Check Repository samples for accessing database level.



More information at Statistics.

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            String text = okmMail.getExtractedText(null, "46762f90-82c6-4886-8d21-ad3017dd78a7");
            System.out.println(text);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forwardEmail

Description:

Method	*Return values*	*Description*
forwardEmail(String token, String uuid, List<String> users, List<String> roles, List<String> mails, String message)	***void***	*Forward an email to a list of users, roles or external emails.*

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            String mailUuid = "f123a950-0329-4d62-8328-0ff500fd42db";
            List<String> users = new ArrayList<>();
            users.add("userTest");
            List<String> roles = new ArrayList<>();
            roles.add("ROLE_USER");
            List<String> mails = new ArrayList<>();
            mails.add("test@none.com");

            okmMail.forwardEmail(null, mailUuid, users, roles, mails, "Any message");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMNode

Basics

*On almost methods you'll see parameter named "**nodeId**". The value of this parameter can be some valid **UUID** node.*

Example of docId:

- *Using UUID -> "**c41f9ea0-0d6c-45da-bae4-d72b66f42d0f**";*

*About the parameter named "**fldId**", the value of this parameter can be some valid folder **UUID or** record node.*

Example of fldId:

- *Using UUID -> "**c6785440-0d6c-45da-1234-d9874563d0f**";*

*About the parameter named "**dstId**", the value of this parameter can be some valid folder **UUID or** record node.*

Example of dstId:

- *Using UUID -> "**c4455667-0d6c-45da-1234-d98564323e0**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

getNodeByUuid

Description:

Method	*Return values*	*Description*
getNodeByUuid(String token, String uuid)	***Node***	*Get a node by uuid.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.bean.Node;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            Node node = okmNode.getNodeByUuid(null, "78f9c377-e557-48c7-804e-35c0ca284739");
            System.out.println(node);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getVersionHistory

Description:

Method	*Return values*	*Description*
getVersionHistory(String token, String nodeId)	***List<Version>***	*Returns the version history list of a node.*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.Version;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<Version> versions = okmNode.getVersionHistory(null, "78f9c377-e557-48c7-804e-35c0ca284739");
            for (Version version : versions) {
                System.out.println(version);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

restoreVersion

Description:

Method	Return values	Description
restoreVersion(String token, String nodeId, String versionId)	**void**	Restore the version of a node.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.restoreVersion(null, "78f9c377-e557-48c7-804e-35c0ca284739", "1.2");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameVersion

Description:

Method	Return values	Description
renameVersion(String token, String nodeId, String versionId, String newName)	**void**	Rename the version of a node.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.renameVersion(null, "78f9c377-e557-48c7-804e-35c0ca284739", "1.2", "newName");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeVersionHistory

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
purgeVersionHistory(String token, String nodeId)	***void***	*Purge the version history of a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.purgeVersionHistory(null, "3767deb4-21e7-4272-82be-fece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

mayBePromotedAsRecord

Description:

Method	*Return values*	*Description*
mayBePromotedAsRecord(String token, String nodeId, boolean fullEvaluation)	***PromoteAsRecordEvaluation***	*Returns the result of the evaluation to promote a node to the "record" status. When the node can not be promoted to the "record" status, the result of the evaluation contains information about the reason why can not be promoted.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.plugin.fileplan.record.bean.PromoteAsRecordEvaluation;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            PromoteAsRecordEvaluation pre = okmNode.mayBePromotedAsRecord(null, "3767deb4-21e7-4272-82be-f
```

```
            System.out.println(pre);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

promoteAsRecord

Description:

Method	*Return values*	*Description*
promoteAsRecord(String token, String nodeId)	***void***	*Promote the node to the "record" status.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.promoteAsRecord(null, "3767deb4-21e7-4272-82be-fece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

degradeRecord

Description:

Method	*Return values*	*Description*
degradeRecord(String token, String nodeId)	***void***	*Degrade a node with "record" status.*



This action only can be done by administrators.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.degradeRecord(null, "3767deb4-21e7-4272-82be-fece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isElectronicRecordPath

Description:

Method	*Return values*	*Description*
isElectronicRecordPath(String token, String nodeId)	***boolean***	*Returns true when the node is into an electronic record.*

Return true when one of the parents of the node is an electronic record.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            System.out.println(okmNode.isElectronicRecordPath(null, "3767deb4-21e7-4272-82be-fece5384fbab"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getElectronicRecordInPath

Description:

Method	*Return values*	*Description*

getElectronicRecordInPath(String token, String nodeId)	***Record***	*Get the electronic record in the path.*

Returns the first electronic record in the path of the node.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            Record record = okmNode.getElectronicRecordInPath(null, "3767deb4-21e7-4272-82be-fece5384fbab");
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenPaginated

Description:

Method	*Return values*	*Description*
getChildrenPaginated(String token, String nodeId, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList)	***PageInfo***	*Get children nodes paginated.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

*The parameter "**orderByFilter**" can have the following values name or created*

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.PageInfo;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.bean.NodeFolder;
import com.openkm.db.bean.NodeMail;
import com.openkm.db.bean.NodeRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<Class<? extends NodeBase>> filteredByNodeTypeList = new ArrayList<>();
            filteredByNodeTypeList.add(NodeDocument.class);
            filteredByNodeTypeList.add(NodeFolder.class);
            filteredByNodeTypeList.add(NodeMail.class);
            filteredByNodeTypeList.add(NodeRecord.class);

            // nodeId Folder and Record UUID or Path
            String nodeId = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            String orderByField = "created";
            boolean orderAsc = true;
            PageInfo pageInfo = okmNode.getChildrenPaginated(null, nodeId, 0, 10, "", orderByField, orderAsc,
                    filteredByNodeTypeList);
            System.out.println("Filtered Elements: " + pageInfo.getFilteredElements());
            System.out.println("Total Elements: " + pageInfo.getTotalElements());
            for (Object object : pageInfo.getObjects()) {
                NodeBase nodeBase = (NodeBase) object;
                System.out.println(nodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenByCategoryPaginated

Description:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Method	*Return values*	*Description*
getChildrenByCategoryPaginated(String token, String nodeId, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList)	***PageInfo***	*Get children nodes by category paginated.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

*The parameter "**orderByFilter**" can have the following values name or created*

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.PageInfo;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.bean.NodeFolder;
import com.openkm.db.bean.NodeMail;
import com.openkm.db.bean.NodeRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<Class<? extends NodeBase>> filteredByNodeTypeList = new ArrayList<>();
```

```
            filteredByNodeTypeList.add(NodeDocument.class);
            filteredByNodeTypeList.add(NodeFolder.class);
            filteredByNodeTypeList.add(NodeMail.class);
            filteredByNodeTypeList.add(NodeRecord.class);

            // nodeId Folder and Record UUID or Path
            String nodeId = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            String orderByField = "created";
            boolean orderAsc = true;
            PageInfo pageInfo = okmNode.getChildrenByCategoryPaginated(null, nodeId, 0, 10, "", orderByField, orderA
                    filteredByNodeTypeList);
            System.out.println("Filtered Elements: " + pageInfo.getFilteredElements());
            System.out.println("Total Elements: " + pageInfo.getTotalElements());
            for (Object object : pageInfo.getObjects()) {
                NodeBase nodeBase = (NodeBase) object;
                System.out.println(nodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getBreadcrumb

Description:

Method	*Return values*	*Description*
getBreadcrumb(String token, String fldId)	***List<BreadCrumbItem>***	*Get breadcrumb.*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.BreadCrumbItem;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<BreadCrumbItem> list = okmNode.getBreadcrumb(null, "39479efe-de5e-468e-91a7-24d2aa3f8837");
            for (BreadCrumbItem item : list) {
                System.out.println(item.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

subscribe

Description:

Method	*Return values*	*Description*
subscribe(String token, String nodeId)	***void***	*Adds a subscription to a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.subscribe(null, "39479efe-de5e-468e-91a7-24d2aa3f8837");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unsubscribe

Description:

Method	*Return values*	*Description*
unsubscribe(String token, String nodeId)	***void***	*Delete a subscription to a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.unsubscribe(null, "39479efe-de5e-468e-91a7-24d2aa3f8837");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importZip

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
importZip(String token, String fldUuid, InputStream inputStream)	***void***	*Import a zip file.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/import.zip");
            okmNode.importZip(null, "212e7c1f-443d-4aac-a12c-0b818ca03419", is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unzip

Description:

Method	*Return values*	*Description*
unzip(String token, String uuid, String destinationPath)	***void***	*Unzip file.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            String uuid = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801"; // zip File
            String destinationPath = "/okm:root/test"; // destination path
            okmNode.unzip(null, uuid, destinationPath);
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

exportZip

Description:

Method	*Return values*	*Description*
exportZip(List<String> paths, boolean withPath, boolean background, boolean excludeProperties)	***ByteArrayInputStream***	*Export as a zip file.*

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<String> uuids = new ArrayList<>();
            uuids.add("/okm:root/test/logo.png");
            uuids.add("/okm:root/test/invoice.pdf");

            OutputStream fos = new FileOutputStream("/home/openkm/import.zip");

            InputStream is = okmNode.exportZip(uuids, true, true, true);
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodesFromUuids

Description:

Method	*Return values*	*Description*

getNodesFromUuids(String token, String[] uuids)	***List<Node>***	*Return a list of nodes.*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.Node;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            String[] uuids = {"0f6463f3-4d36-4091-b518-4fe7c353ee70", "d386cff8-1d4d-472f-9c6d-f21955ec499a"};
            List<Node> nodes = okmNode.getNodesFromUuids(null, uuids);
            for (Node node : nodes) {
                System.out.println(node);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

evaluateZipDownload

Description:

Method	*Return values*	*Description*
evaluateZipDownload(String token, List<String> uuids)	***ZipDownloadEvaluationResult***	*Evaluate when a zip file can be downloaded in real-time or at background.*

Related configuration parameters:

- ***download.zip.live.max.file.size***
- ***download.zip.live.max.files***

More information at Performance configuration parameters

Example:

```
package com.openkm;
```

```
import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.ZipDownloadEvaluationResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<String> uuids = new ArrayList<>();
            uuids.add("0f6463f3-4d36-4091-b518-4fe7c353ee70");
            uuids.add("d386cff8-1d4d-472f-9c6d-f21955ec499a");

            ZipDownloadEvaluationResult result = okmNode.evaluateZipDownload(null, uuids);
            System.out.println(result);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

restore

Description:

Method	*Return values*	*Description*
restore(String token, String nodeId)	***Node***	*Restore a node from the trash.*

> *When a node is moved to trash, the application keeps the origin. When possible the application will try to restore in the same origin (sometimes is not possible because the origin no longer exist or security has been changed in the origin, etc.).*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.bean.Node;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            Node node = okmNode.restore(null, "0f6463f3-4d36-4091-b518-4fe7c353ee70");
            System.out.println(node);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

hasNodesLockedByOtherUser

Description:

Method	*Return values*	*Description*
hasNodesLockedByOtherUser(String token, String nodeId)	***booelan***	*Check if the node and descendants have been locked by other users*

This method is used to evaluate if it is possible to lock a hierarchy. For example, in case of a record, the lock affects all the nodes into what also will be locked.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            if (okmNode.hasNodesLockedByOtherUser(null, "0f6463f3-4d36-4091-b518-4fe7c353ee70")) {
                System.out.println("Is blocked");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

lock

Description:

Method	*Return values*	*Description*
lock(String token, String nodeId)	***LockInfo***	*Locks a node and returns an object with the Lock information.*

Only the user who locked the document is allowed to unlock.

A locked document cannot be modified by other users.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.lock(null, "1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceLock

Description:

Method	*Return values*	*Description*
forceLock(String token, String nodeId)	***void***	*Locks a locked node.*

This method allows to lock any locked node.



This action can only be done by users with administrator grants.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.forceLock(null, "1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unlock

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
unlock(String token, String nodeId)	***void***	*Unlocks a locked node.*



Only the user who locked the document is allowed to unlock.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.unlock(null, "1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceUnlock

Description:

Method	*Return values*	*Description*
forceUnlock(String token, String nodeId)	***void***	*Unlocks a locked node.*

This method allows to unlock any locked node.

It is not mandatory execute this action by the same user who previously executed the checkout lock action.



This action can only be done by a user with administrator grants.

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.forceUnlock(null, "1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isLocked

Description:

Method	*Return values*	*Description*
isLocked(String token, String nodeId)	***Boolean***	*Returns a boolean that indicates if the node is locked or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            System.out.println(okmNode.getLockInfo(null, "1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLockInfo

Description:

Method	*Return values*	*Description*
getLockInfo(String token, String nodeId)	***LockInfo***	*Returns an object with the Lock information*

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            System.out.println(okmNode.getLockInfo(null, "1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setComment

Description:

Method	*Return values*	*Description*
setComment(String token, String nodeId, String versionName, String comment)	***void***	*Sets the comment for a specific node version.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.setComment(null, "1ec49da9-1746-4875-ae32-9281d7303a62", "1.14", "Update comment");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMNote

Basics

*On most methods you'll see parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID**.*

Example of nodeId:

- *Using UUID -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

add

Description:

Method	*Return values*	*Description*
add(String token, String nodeId, String text)	***Note***	*Adds a note to a node and returns an object Note.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNote;
import com.openkm.bean.Note;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNote okmNote = ContextWrapper.getContext().getBean(OKMNote.class);
            Note note = okmNote.add(null, "f123a950-0329-4d62-8328-0ff500fd42db", "the note text");
            System.out.println(note);
        } catch (Exception e) {
            e.printStackTrace();
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
        }
    }
}
```

get

Description:

Method	*Return values*	*Description*
get(String token, String noteId)	***Note***	*Retrieves the note.*

> *The noteId is an UUID.*
>
> *The object Node has a variable named path, in that case the path contains an UUID.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNote;
import com.openkm.bean.Note;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNote okmNote = ContextWrapper.getContext().getBean(OKMNote.class);
            for (Note note : okmNote.list(null, "f123a950-0329-4d62-8328-0ff500fd42db")) {
                Note data = okmNote.get(null, note.getPath());
                System.out.println(data);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	*Return values*	*Description*
delete(String token, String noteId)	***void***	*Delete a note.*

> *The noteId is an UUID.*
>
> *The object Node has a variable named path, in that case the path contains an UUID.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNote;
import com.openkm.bean.Note;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNote okmNote = ContextWrapper.getContext().getBean(OKMNote.class);
            for (Note note : okmNote.list(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338")) {
                okmNote.delete(null, note.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

set

Description:

Method	*Return values*	*Description*
set(String token, String noteId, String text)	***void***	*Changes the note text.*

> *The noteId is an UUID.*
>
> *The object Node has a variable named path, in that case the path contains an UUID.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNote;
import com.openkm.bean.Note;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNote okmNote = ContextWrapper.getContext().getBean(OKMNote.class);
            for (Note note : okmNote.list(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338")) {
                okmNote.set(null, note.getPath(), "new note");
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

list

Description:

Method	*Return values*	*Description*
list(String token, String nodeId)	***List<Note>***	*Retrieves a list of all the notes of a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNote;
import com.openkm.bean.Note;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNote okmNote = ContextWrapper.getContext().getBean(OKMNote.class);
            for (Note note : okmNote.list(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338")) {
                System.out.println(note);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNoteHistories

Description:

Method	*Return values*	*Description*
getNoteHistories(String token, String nodeId)	***List<NoteHistory>***	*Return the historical notes of a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNote;
import com.openkm.bean.NoteHistory;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
```

```
        try {
            OKMNote okmNote = ContextWrapper.getContext().getBean(OKMNote.class);
            for (NoteHistory noteHistory : okmNote.getNoteHistories(null, "3c68b3a1-c65c-4b1e-84b5-9ce2712ca573"))
                System.out.println(noteHistory);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMNotification

Basics

*On most methods you'll see parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID**.*

Example of nodeId:

- *Using uuid -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**"*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

subscribe

Description:

Method	*Return values*	*Description*
subscribe(String token, String nodeId)	***void***	*Adds a subscription to a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNotification;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNotification okmNotification = ContextWrapper.getContext().getBean(OKMNotification.class);

            okmNotification.subscribe(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338");
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    }
}
```

unsubscribe

Description:

Method	*Return values*	*Description*
unsubscribe(String token, String nodeId)	***void***	*Delete a subscription to a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNotification;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNotification okmNotification = ContextWrapper.getContext().getBean(OKMNotification.class);
            okmNotification.unsubscribe(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSubscriptors

Description:

Method	*Return values*	*Description*
getSubscriptors(String token, String nodeId)	***Set<String>***	*Retrieves a list of all the subscriptors of a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNotification;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNotification okmNotification = ContextWrapper.getContext().getBean(OKMNotification.class);
            for (String subscriptor : okmNotification.getSubscriptors(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338")) {
                System.out.println(subscriptor);
            }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

notify

Description:

Method	*Return values*	*Description*
notify(String token, String nodeId, List<String> users, List<String> mails, String message, boolean attachment)	***void***	*Send a mail notification.*

The parameter users are a set of OpenKM users to be notified.

The parameter mails are a set of mails - usually external mails - to be notified.

The parameter message is the content body of the mail.

When attachment value is true the node is attached into the mail.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMNotification;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNotification okmNotification = ContextWrapper.getContext().getBean(OKMNotification.class);
            List<String> users = new ArrayList<>();
            users.add("sochoa");
            List<String> mails = new ArrayList<>();
            String body = "Body of the message";
            okmNotification.notify(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", users, mails, body, false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMProperty

Basics

*On most methods you'll see parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID**.*

Example of nodeId:

- *Using UUID -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**";*

*About the parameter named "**catId**", the value of this parameter can be a valid categories folder **UUID** node.*

Example of nodeId:

- *Using UUID -> "**5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

addCategory

Description:

Method	*Return values*	*Description*
addCategory(String token, String nodeId, String catId)	***void***	*Set a relation between a category and a node.*
The value of the catId parameter should be a category folder UUID.		

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            okmProperty.addCategory(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "5cbc35fc-23c0-48f5-a7bc-3cfa
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeCategory

Description:

Method	*Return values*	*Description*
removeCategory(String token, String nodeId, String catId)	***void***	*Removes a relation between a category and a node.*
The value of the catId parameter should be a category folder UUID.		

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            okmProperty.removeCategory(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "5cbc35fc-23c0-48f5-a7bc-3
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addKeyword

Description:

Method	*Return values*	*Description*

addKeyword(String token, String nodeId, String keyword)	***String***	*Add a keyword in a node.*

The keyword should be a single word without spaces, formats allowed:

- *"test"*
- *"two_words" (the character "_" is used for the junction).*



We also we suggest you to add keyword in lowercase format, because OpenKM is case sensitive.

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            String keyword = okmProperty.addKeyword(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "test");
            System.out.println(keyword);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeKeyword

Description:

Method	*Return values*	*Description*
removeKeyword(String token, String nodeId, String keyword)	***void***	*Removes a keyword from a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            okmProperty.removeKeyword(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "test");
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setEncryption

Description:

Method	*Return values*	*Description*
setEncryption(String token, String nodeId, String cipherName)	***void***	*Marks a document as an en cripted binary data in the repository.*

The parameter nodeId should be a document node.

The parameter cipherName saves information about the encription mechanism.



This method does not perform any kind of encryption, simply mark in the database that a document is encrypted.

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            okmProperty.setEncryption(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "phrase");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unsetEncryption

Description:

Method	*Return values*	*Description*
unsetEncryption(String token, String	***void***	*Marks a document is a normal binary data into*

nodeId)		*repository.*

The parameter nodeId should be a document node.



This method does not perform any kind of encryption, simply mark into the database that a document has been uncrypted.

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            okmProperty.unsetEncryption(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setSigned

Description:

Method	*Return values*	*Description*
setSigned(String token, String nodeId, boolean signed)	***void***	*Marks a document as signed or unsigned binary data into the repository.*

The parameter nodeId should be a document node.



This method does not perform any kind of digital signature process, simply mark into the database that a document is signed.

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            okmProperty.setSigned(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isSigned

Description:

Method	*Return values*	*Description*
isSigned(String token, String nodeId)	***boolean***	*Returns a boolean that indicates if the document is signed or not.*
The parameter nodeId should be a document node.		

Example:

```
package com.openkm;

import com.openkm.api.OKMProperty;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);
            System.out.println("Is the document signed: " + okmProperty.isSigned(null, "6330d2a0-529f-4c14-baa1-ce6a
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMPropertyGroup

Basics

*From the older OpenKM version we named "**Metadata Groups**" as "**Property Groups**".*

Although we understand this name does not help a lot to identify these methods with metadata ones, for historical reason, we continue maintaining the nomenclature.

For more information about Metadata.

*On most methods you'll see a parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID**.*

Example of nodeId:

- *Using UUID -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**";*

The class com.openkm.util.ISO8601 should be used to set and parse metadata date fields. The metadata field of type date values is stored in the application in ISO-8601 basic format.

To convert the retrieved metada field of type date to a valid date use:

```
Calendar cal = ISO8601.parseBasic(metadataFieldValue);
```

To save the date value in the metadata field of type date use:

```
Calendar cal = Calendar.getInstance(); // Present date
String metadataFieldValue = ISO8601.formatBasic(cal);
// metadataFieldValue can be saved into repository metadata field of type date
```

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

addGroup

Description:

Method	*Return values*	*Description*
addGroup(String token, String nodeId, String grpName, Map<String, String> properties)	***void***	*Adds an metadata group and its values to a node.*

The grpName should be a valid Metadata group name.

It is not mandatory to set in the properties parameter all fields values, is enough with the fields you wish to change its values.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Date" name="okp:technology.date" type="date" />
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <checkbox label="Important" name="okp:technology.important"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

To add several values on a metadata field of type multiple like this:

```
<select label="Type" name="okp:technology.type" type="multiple">
  <option label="Alfa" value="t1"/>
  <option label="Beto" value="t2"/>
  <option label="Omega" value="t3" />
</select>
```

You should do:

```
properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*Where \"**t1**\" and \"**t2**\" are valid values and character "**,**" is used as separator.*

Another option:

```
Gson gson = new Gson();
List<String> values = new ArrayList<>();
values.add("t1");
values.add("t2");
properties.put("okp:technology.type", gson.toJson(values));
```

Example:

```
package com.openkm;

import com.google.gson.Gson;
import com.openkm.Config;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.util.ISO8601;

import java.util.*;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)

            Map<String, String> properties = new HashMap<>();
            Gson gson = new Gson();
            List<String> values = new ArrayList<>();
            values.add("t1");
            values.add("t2");
            properties.put("okp:technology.type", gson.toJson(values));
            properties.put("okp:technology.language", "[ \"java\" ]");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:technology.date", ISO8601.formatBasic(cal));
            properties.put("okp:technology.comment", "comment sample");
            properties.put("okp:technology.description", "description sample");
            properties.put("okp:technology.important", String.valueOf(true));
            okmPropertyGroup.addGroup(null, "4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5", "okg:technology", properties)
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeGroup

Description:

Method	*Return values*	*Description*
removeGroup(String token, String nodeId, String grpName)	***void***	*Removes a metadata group of a node.*

The grpName should be a valid Metadata group name.

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            okmPropertyGroup.removeGroup(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "okg:consulting");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroups

Description:

Method	*Return values*	*Description*
getGroups(String token, String nodeId)	***List<PropertyGroup>***	*Retrieves a list of metadata groups assigned to a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.bean.PropertyGroup;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            for (PropertyGroup pg : okmPropertyGroup.getGroups(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338")) {
                System.out.println(pg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllGroups

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
getAllGroups(String token)	***List<PropertyGroup>***	*Retrieves a list of all metadata groups set into the application.*

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.bean.PropertyGroup;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            for (PropertyGroup pg : okmPropertyGroup.getAllGroups(null)) {
                System.out.println(pg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserGroups

Description:

Method	*Return values*	*Description*
getUserGroups(String token)	***List<PropertyGroup>***	*Retrieves a list of all metadata groups for the user set into the application.*

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.bean.PropertyGroup;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            for (PropertyGroup pg : okmPropertyGroup.getUserGroups(null)) {
                System.out.println(pg);
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	*Return values*	*Description*
getProperties(String token, String nodeId, String grpName)	***Map<String, String>***	*Retrieves a list of all metadata group elements and its node values.*

The grpName should be a valid Metadata group name.

> *The method is usually used to display form elements with its values to be shown or changed by used.*

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            Map<String, String> properties = okmPropertyGroup.getProperties(null, "b153c4b7-3d1c-4589-bd42-0ed0f3
            for (String key : properties.keySet()) {
                System.out.println(key + ": " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setProperties

Description:

Method	*Return values*	*Description*
setProperties(String token, String nodeId, String grpName,	***void***	*Changes the metadata group*

Map<String, String> properties)		*valuea of a node.*

The grpName should be a valid Metadata group name.

It is not mandatory to set in the properties parameter all fields values, is enough with the fields you wish to change its values.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Date" name="okp:technology.date" type="date" />
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <checkbox label="Important" name="okp:technology.important"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

To add several values on a metadata field of type multiple like this:

```
<select label="Type" name="okp:technology.type" type="multiple">
  <option label="Alfa" value="t1"/>
  <option label="Beto" value="t2"/>
  <option label="Omega" value="t3" />
</select>
```

You should do:

```
properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
```

*Where \"**t1**\" and \"**t2**\" are valid values and character "**,**" is used as separator.*

Another option:

```
Gson gson = new Gson();
List<String> values = new ArrayList<>();
values.add("t1");
values.add("t2");
```

```
properties.put("okp:technology.type", gson.toJson(values));
```

Example:

```
package com.openkm;

import com.google.gson.Gson;
import com.openkm.Config;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.util.ISO8601;

import java.util.*;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)

            Map<String, String> properties = new HashMap<>();
            Gson gson = new Gson();
            List<String> values = new ArrayList<>();
            values.add("t1");
            values.add("t2");
            properties.put("okp:technology.type", gson.toJson(values));
            properties.put("okp:technology.language", "[ \"java\" ]");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:technology.date", ISO8601.formatBasic(cal));
            properties.put("okp:technology.comment", "comment sample");
            properties.put("okp:technology.description", "description sample");
            properties.put("okp:technology.important", String.valueOf(true));
            okmPropertyGroup.setProperties(null, "4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5", "okg:technology", properti
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupForm

Description:

Method	*Return values*	*Description*
getPropertyGroupForm(String token, String grpName)	***List<FormElement>***	*Retrieves a list of all metadata group elements definition.*

The grpName should be a valid Metadata group name.

> *The method is usually used to display empty form elements for creating new metadata values.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.form.FormElement;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            for (FormElement fe : okmPropertyGroup.getPropertyGroupForm(null, "okg:consulting")) {
                System.out.println(fe);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasGroup

Description:

Method	*Return values*	*Description*
hasGroup(String token, String nodeId, String grpName)	***boolean***	*Returns a boolean that indicate if the node has or not a metadata group.*

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            boolean found = okmPropertyGroup.hasGroup(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "okg:consu
            System.out.println(found);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestions

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
getSuggestions(String token, String nodeId, String grpName, String propName)	***List<String>***	*Retrieves a list of a suggested metadata field values.*

The propName parameter should be a Metadata Select field type.

More information at Creating your own Suggestion Analyzer and Metadata Select field .

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            for (String value : okmPropertyGroup.getSuggestions(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "okg
                    "okp:technology.language")) {
                System.out.println(value);
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

registerDefinition

Description:

Method	*Return values*	*Description*
registerDefinition(String token, String pgDef, String pgName)	***void***	*Set the XML Metada groups definition into the repository.*



The method only can be executed by super user granted (ROLE_ADMIN member user).

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        InputStream is = null;
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            is = new FileInputStream("/home/files/PropertyGroups.xml");
            String pgDef = IOUtils.toString(is);
            okmPropertyGroup.registerDefinition(null, pgDef, "test");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(is);
        }
    }
}
```

getRegisteredDefinition

Description:

Method	Return values	Description
getRegisteredDefinition(String token)	**String**	*Returns the XML Metada groups definition.*



The method only can be executed by super user granted (ROLE_ADMIN member user).

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            String pgDef = okmPropertyGroup.getRegisteredDefinition(null);
            System.out.println(pgDef);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroups

Description:

Method	Return values	Description
getGroups(String token, String nodeId, String versionName)	**List<PropertyGroup>**	*Retrieves a list of metadata groups assigned to a specific version of a node*

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.PropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            for (PropertyGroup pg : okmPropertyGroup.getGroups(null, "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801", "1.2
                System.out.println(pg);
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroupProperties

Description:

Method	*Return values*	*Description*
getGroupProperties(String token, Long pgdId)	***List<PropertyGroup>***	*Retrieves a list of metadata groups properties.*

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.PropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            long pgdId = 1;
            for (PropertyGroup pg : okmPropertyGroup.getGroupProperties(null, pgdId)) {
                System.out.println(pg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	*Return values*	*Description*
getProperties(String token, String nodeId, String grpName, Sring versionName)	***Map<String, String>***	*Retrieves a map - (key, value) pairs - with Metadata group values assigned to a specific version of a node.*

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)

            Map<String, String> properties = okmPropertyGroup.getProperties(null, "82555dd1-bcc2-4e64-81cb-5f7c2b
            for (String key : properties.keySet()) {
                System.out.println(key + ": " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestBoxKeyValue

Description:

Method	*Return values*	*Description*
getSuggestBoxKeyValue(String token, String grpName, String propertyName, Sring key)	***KeyValue***	*Retrieves an object that contains the key and value of the suggestbox.*

Example:

```
package com.openkm;


import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.KeyValue;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)


            KeyValue kv = okmPropertyGroup.getSuggestBoxKeyValue(null, "okg:groupname", "okp:groupname.proper
            System.out.println(kv.getKey() + ": " + kv.getValue());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestBoxKeyValuesFiltered

Description:

Method	*Return values*	*Description*
getSuggestBoxKeyValuesFiltered(String token, String grpName, String propertyName, String filter)	***List<KeyValue>***	*Retrieves a list of objects that contains the key and value of the suggestbox filtered by value.*

Example:

```
package com.openkm;

import java.util.List;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.KeyValue;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)


            List<KeyValue> kvList = okmPropertyGroup.getSuggestBoxKeyValuesFiltered(null, "okg:groupname", "okp:g

            for (KeyValue kv : kvList) {
               System.out.println(kv.getKey() + ": " + kv.getValue());
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMRecord

Basics

*On almost methods you'll see parameter named "**recId**". The value of this parameter can be some valid record **UUID**.*

Example of recId:

- *Using UUID -> "**7f867772-7476-4176-b86f-bdeb8bce1c5f**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

create

Description:

Method	*Return values*	*Description*
create(String token, Record rec)	***Record***	*Creates a new record and returns as a result an object Record.*

*The variable **path** into the parameter **rec**, must be initializated. It indicates the folder path into OpenKM.*

```
Record rec = new Record();
rec.setPath("/okm:root/PKI-100200");
```

The other variables of the Record (rec) will not take any effect on the record creation.

We suggest using the method below to create the Record Simply rather this one.

Example:

```
package com.openkm;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
import com.openkm.api.OKMRecord;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            Record rec = new Record();
            rec.setPath("/okm:root/test");
            okmRecord.create(null, rec);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createSimple

Description:

Method	*Return values*	*Description*
createSimple(String token, String recPath)	***Record***	*Creates a new record and returns as a result an object Record.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            String recPath = "/okm:root/PKI-100200";
            okmRecord.createSimple(null, recPath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	*Return values*	*Description*
getProperties(String token, String recId)	***Record***	*Return the record properties.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            Record rec = okmRecord.getProperties(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
            System.out.println(rec);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	*Return values*	*Description*
delete(String token, String recId)	***void***	*Delete a record.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.delete(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

Method	*Return values*	*Description*

purge(String token, String recId)	***void***	*The record is definitely removed from the repository.*

Usually you will purge folders into /okm:trash/userId - the personal trash user locations - but it is possible to directly purge any record from the whole repository.



When a record is purged, it will only be able to be restored from a previously repository backup. The purge action removes the record definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.purge(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	*Return values*	*Description*
rename(String token, String recId, String newName)	***Record***	*Rename a record.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.rename(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f", "newname");
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

move

Description:

Method	*Return values*	*Description*
move(String token, String recId, String dstId)	***void***	*Moves record into some folder or record*
The values of the dstId parameter should be a folder or record UUID.		

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.move(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f", "b6848ac2-345f-454e-964a-eba4dcc020
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String recId, String dstId)	***void***	*Copies a record into a folder or record.*

The values of the dstId parameter should be a folder or record UUID.



Only the binary data and the security grants are copied to destination, the metadata, keywords, etc. of the document are not copied.

*See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.copy(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f", "b6848ac2-345f-454e-964a-eba4dcc020e
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String recId, String dstId, String newName)	***void***	*Copies a record into a folder or record.*

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, folder will preservate the same name.

> *Only the binary data and the security grants are copied to destination, the metadata, keywords, etc. of the document are not copied.*
>
> *See "**extendedCopy**" method for this feature.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.copy(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f", "b6848ac2-345f-454e-964a-eba4dcc020e
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

extendedCopy

Description:

Method	*Return values*	*Description*
extendedCopy(String token, String recId, String dstId, String newName, ExtendedAttributes extAttr)	***void***	*Copies a record with associated data into some folder or record.*

The values of the dstId parameter should be a folder or a record UUID.

When the parameter newName value is null, the record will preserve the same name.

By default only the binary data and the security grants, the metadata, keywords, etc. of the document are not copied.

Additional:

- *When the category parameter is true the original values of the categories will be copied.*
- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original value of the notes will be copied.*
- *When wiki parameter is true the original value of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            ExtendedAttributes extAttr = new ExtendedAttributes();
            extAttr.setKeywords(true);
            extAttr.setCategories(true);
            extAttr.setNotes(true);

            okmRecord.extendedCopy(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f", "b67fe3ae-bcca-4947-b8ea-0cc
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	*Return values*	*Description*
isValid(String token, String recId)	***boolean***	*Returns a boolean that indicates if the node is a record or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            boolean valid = okmRecord.isValid(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
            System.out.println(valid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPath

Description:

Method	*Return values*	*Description*
getPath(String token, String recId)	***String***	*Convert folder UUID to record path.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            String path = okmRecord.getPath(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
            System.out.println(path);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Description:

Method	*Return values*	*Description*
getChildren(String token, String fldId)	***List<Record>***	*Returns a list of all records which their parent is fldId.*
The parameter fldId can be a folder or a record node.		

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            for (Record rec : okmRecord.getChildren(null, "b6848ac2-345f-454e-964a-eba4dcc020e6")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

lock

Description:

Method	*Return values*	*Description*
lock(String token, String recId)	***LockInfo***	*Locks a record and return an object with the Lock information.*



Only the user who locked the record is allowed to unlock.

A locked record can not be modified by other users.

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.lock(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unlock

Description:

Method	*Return values*	*Description*
unlock(String token, String recId)	***LockInfo***	*Unlock a locked record.*

Only the user who locked the record is allowed to unlock.

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.unlock(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceUnlock

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
forceUnlock(String token, String recId)	***void***	*Unlock a locked record.*

This method allows to unlock any locked record.

It is not mandatory to execute this action by the same user who previously executed the checkout lock action.



This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.forceUnlock(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTitle

Description:

Method	*Return values*	*Description*
setTitle(String token, String recId, String title)	***void***	*Sets a record title.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
```

```
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.setTitle(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f", "new title");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setNodeClass

Description:

Method	*Return values*	*Description*
setNodeClass(String token, String recId, long ncId)	***void***	*Set the NodeClass.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);

            long ncId = 2;
            okmRecord.setNodeClass(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDispositionStage

Description:

Method	*Return values*	*Description*
setDispositionStage(String token, String recId, long stage)	***void***	*Set the disposition stage*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);

            long stage = 1;
            okmRecord.setDispositionStage(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addNodeClassChildren

Description:

Method	*Return values*	*Description*
addNodeClassChildren(String token, String recId, long ncId)	***void***	*Add NodeClass of the record in the folder parents.*
Used for navigation purposes to evaluating when a folder contains a node of type node class. *The method propagate the node class value in the parent containers (folders) to indicate there's a node of this type.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);

            long ncId = 2;
            okmRecord.addNodeClassChildren(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	*Return values*	*Description*

setDescription(String token, String uuid, String description)	***void***	*Set a description.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);

            okmRecord.setDescription(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizard

Description:

Method	*Return values*	*Description*
createWizard(String token, String recPath, String title, long nodeClass)	***WizardNode***	*Create a new record with wizard.*

The parameters ***recPath*** *should be any valid folder or record* ***UUID***

> *The WizardNode contains a list of pending actions what should be done to complete the process of record creation. These might be:*
>
> - *Add keyword*
> - *Add Categories*
> - *Add Metadata*

Example:

```
package com.openkm;

import com.openkm.api.OKMRecord;
```

```
import com.openkm.bean.WizardNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);

            WizardNode wn = okmRecord.createWizard(null, "1f323e88-64ee-4f57-91e2-9276f8c603f9", "PKI-100200",
            System.out.print(wn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMRelation

Basics

*On most methods you'll see parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID**.*

Example of nodeId:

- *Using UUID -> "**d168924d-e801-426d-a222-00124a1461bd**";*

*Also on all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken()
```

Methods

getRelationTypes

Description:

Method	*Return values*	*Description*
getRelationTypes(String token, String type)	***List<RelationType>***	*Retrieves a list of all relations defined of a type.*

Available types values:

- *NodeRelationType.BIDIRECTIONAL*
- *NodeRelationType.PARENT_CHILD*
- *NodeRelationType.MANY_TO_MANY*

More information at Relation types .

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationType;
import com.openkm.db.bean.NodeRelationType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationType rt : okmRelation.getRelationTypes(null, NodeRelationType.BIDIRECTIONAL)) {
                System.out.println(rt);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

add

Description:

Method	*Return values*	*Description*
add(String token, String nodeAId, String nodeBId, long relationType)	***void***	*Sets a relation between two nodes.*
The parameters ***nodeAId*** *and* ***nodeBId*** *should be any valid document, folder, mail or record* ***UUID***.		

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationType;
import com.openkm.db.bean.NodeRelationType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationType rt : okmRelation.getRelationTypes(null, NodeRelationType.BIDIRECTIONAL)) {
                // looking for a relation named invoice
                if (rt.getTitle().equals("invoice")) {
                    okmRelation.add(null, "d168924d-e801-426d-a222-00124a1461bd", "1f8ee9b8-0357-447b-95d0-c7b0a
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

delete

Description:

Method	*Return values*	*Description*
delete(String token, long relationId)	***void***	*Deletes a relation.*



Only when the relation will not be used by any node is able to be deleted, otherwise you'll get an error.

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.Relation;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (Relation relation : okmRelation.getRelations(null, "d168924d-e801-426d-a222-00124a1461bd")) {
                if (relation.getRelationTitle().equals("invoice")) {
                    okmRelation.delete(null, relation.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRelations

Description:

Method	*Return values*	*Description*
getRelations(String token, String nodeId)	***List<Relation>***	*Retrieves a list of all relations of a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
```

```
import com.openkm.bean.Relation;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (Relation relation : okmRelation.getRelations(null, "d168924d-e801-426d-a222-00124a1461bd")) {
                System.out.println(relation);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRelationGroups

Description:

Method	*Return values*	*Description*
getRelationGroups(String token, String nodeId)	***List<RelationGroup>***	*Retrieves a list of all relation groups of a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationGroup rg : okmRelation.getRelationGroups(null, "d168924d-e801-426d-a222-00124a1461bd")) {
                System.out.println(rg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addGroup

Description:

Method	*Return values*	*Description*

addGroup(String token, String nodeId, String groupName, long type)	***void***	*Adds a relation group at a node.*

On a relation group only has sense to apply a relation type of NodeRelationType.MANY_TO_MANY.



More information at Relation types .

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationType;
import com.openkm.db.bean.NodeRelationType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationType rt : okmRelation.getRelationTypes(null, NodeRelationType.MANY_TO_MANY)) {
                if (rt.getTitle().equals("staple")) {
                    okmRelation.addGroup(null, "d168924d-e801-426d-a222-00124a1461bd", "staple group", rt.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addNodeToGroup

Description:

Method	*Return values*	*Description*
addNodeToGroup(String token, String nodeId, long groupId)	***void***	*Adds a node to an existing relation group.*

On a relation group only has sense to apply a relation type of NodeRelationType.MANY_TO_MANY.



More information at Relation types .

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationGroup rg : okmRelation.getRelationGroups(null, "d168924d-e801-426d-a222-00124a1461bd")) {
                if (rg.getRelationTitle().equals("staple")) {
                    okmRelation.deleteGroup(null, "d168924d-e801-426d-a222-00124a1461bd", rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteGroup

Description:

Method	*Return values*	*Description*
deleteGroup(String token, String nodeId, long groupId)	***void***	*Remove a node from a relation group.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationGroup rg : okmRelation.getRelationGroups(null, "8f101a85-88e7-4abe-8175-c5fea3e17d8b")) {
                if (rg.getName().equals("staple group")) {
                    okmRelation.deleteGroup(null, "46762f90-82c6-4886-8d21-ad3017dd78a7", rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroup

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
getGroup(String token, long groupId)	***RelationGroup***	*Finds a relation group by id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            long groupId = 1;
            RelationGroup rg = okmRelation.getGroup(null, groupId);
            System.out.println(rg);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setGroupName

Description:

Method	*Return values*	*Description*
setGroupName(String token, long groupId, String groupName)	***void***	*Changes the relation group name.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            long groupId = 1;
            okmRelation.setGroupName(null, groupId, "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMReport

Basics

You can set several reports formats based in document mime type:

Contants defined into Report class:

- *Report.FORMAT_TEXT = "text/plain"*
- *Report.FORMAT_HTML = "text/html"*
- *Report.FORMAT_CSV = "text/csv"*
- *Report.FORMAT_PDF = "application/pdf"*
- *Report.FORMAT_RTF = "application/rtf"*
- *Report.FORMAT_ODT = "application/vnd.oasis.opendocument.text"*
- *Report.FORMAT_XLSX = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"*
- *Report.FORMAT_DOCX = "application/vnd.openxmlformats-officedocument.wordprocessingml.document"*

Methods

list

Description:

Method	*Return values*	*Description*
list(String token, boolean active)	***List<Report>***	*Returns a list of reports.*
*When variable **active** when is set to **true** will be returned only the active reports otherwise all the reports.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMReport;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMReport okmReport = ContextWrapper.getContext().getBean(OKMReport.class);
            System.out.println(okmReport.list(null, true));
        } catch (Exception e) {
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
            e.printStackTrace();
        }
    }
}
```

get

Description:

Method	*Return values*	*Description*
get(String token, long rpId)	***Report***	*Returns an object Report.*
*The variable **rpId** is the long value used to identify the report.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMReport;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMReport okmReport = ContextWrapper.getContext().getBean(OKMReport.class);
            System.out.println(okmReport.get(null, 1));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

execute

Description:

Method	*Return values*	*Description*
execute(String token, String pluginName, Map<String, Object> params)	***PageInfo***	*Returns a PageInfo object.*
*The variable **pluginName** is the name of a plugin defined in OpenKM.*		

Example:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
package com.openkm;


import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;


import com.openkm.api.OKMReport;
import com.openkm.bean.Report;
import com.openkm.util.ContextWrapper;
import com.openkm.bean.PageInfo;

public class Test {

    public static void main(String[] args) {
        try {
            OKMReport okmReport = ContextWrapper.getContext().getBean(OKMReport.class);

            Calendar yesterday = Calendar.getInstance();
            yesterday.add(Calendar.DATE, -1);
            Calendar today = Calendar.getInstance();

            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

            PageInfo pageInfo = new HashMap<>(); // report parameters
            params.put("from_date", sdf.format(yesterday.getTime()));
            params.put("to_date", sdf.format(today.getTime()));
            System.out.println(okmReport.execute(null, "pluginName", params));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

execute

Description:

Method	*Return values*	*Description*
execute(String token, long rpId, Map<String, Object> params, String format)	***InputStream***	*Returns the inputstream of the report.*
*The variable **rpId** is the long value used to identify the report.*		

Example:

```
package com.openkm;

import java.io.InputStream;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;
```

```
import java.util.Map;

import com.openkm.api.OKMReport;
import com.openkm.bean.Report;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMReport okmReport = ContextWrapper.getContext().getBean(OKMReport.class);

            Calendar yesterday = Calendar.getInstance();
            yesterday.add(Calendar.DATE, -1);
            Calendar today = Calendar.getInstance();

            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

            Map<String, Object> params = new HashMap<>(); // report parameters
            params.put("from_date", sdf.format(yesterday.getTime()));
            params.put("to_date", sdf.format(today.getTime()));
            InputStream is = okmReport.execute(null, 1, params, Report.FORMAT_PDF);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMRepository

*On all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*



*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

getRootFolder

Description:

Method	*Return values*	*Description*
getRootFolder(String token)	***Folder***	*Returns the object Folder of node "/okm:root".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getRootFolder(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTrashFolder

Description:

Method	*Return values*	*Description*
getTrashFolder(String token)	***Folder***	*Returns the object Folder of node "/okm:trash/{userId}".*

The returned folder will be the user trash folder.

> *For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:trash/okmAdmin".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getTrashFolder(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTrashFolderBase

Description:

Method	*Return values*	*Description*
getTrashFolderBase(String token)	***Folder***	*Returns the object Folder of node "/okm:trash".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getTrashFolderBase(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

getTemplatesFolder

Description:

Method	*Return values*	*Description*
getTemplatesFolder(String token)	***Folder***	*Returns the object Folder of node "/okm:templates".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getTemplatesFolder(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalFolder

Description:

Method	*Return values*	*Description*
getPersonalFolder(String token)	***Folder***	*Returns the object Folder of node "/okm:personal/{userId}".*

The returned folder will be the user personal folder.

> *For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:personal/okmAdmin".*

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getPersonalFolder(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalFolderBase

Description:

Method	*Return values*	*Description*
getPersonalFolderBase(String token)	***Folder***	*Returns the object Folder of node "/okm:personal".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getPersonalFolderBase(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailFolder

Description:

Method	*Return values*	*Description*
getMailFolder(String token)	***Folder***	*Returns the object Folder of node "/okm:mail/{userId}".*

The returned folder will be the user mail folder.



For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:mail/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getMailFolder(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailFolderBase

Description:

Method	*Return values*	*Description*
getMailFolderBase(String token)	***Folder***	*Returns the object Folder of node "/okm:mail".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getMailFolderBase(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

getThesaurusFolder

Description:

Method	*Return values*	*Description*
getThesaurusFolder(String token)	***Folder***	*Returns the object Folder of node "/okm:thesaurus".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getThesaurusFolder(null);
            System.out.println(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategoriesFolder

Description:

Method	*Return values*	*Description*
getCategoriesFolder(String token)	***Folder***	*Returns the object Folder of node "/okm:categories".*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            Folder fld = okmRepository.getCategoriesFolder(null);
            System.out.println(fld);
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

purgeTrash

Description:

Method	*Return values*	*Description*
purgeTrash(String token)	***Folder***	*Definitively removes from repository all nodes to "/okm:trash/{userId}".*

For example if the method is executed by the user "okmAdmin" then the purged trash will be "/okm:trash/okmAdmin".

When a node is purged it will only be able to be restored from a previously repository backup. The purge action removes the node definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            okmRepository.purgeTrash(null);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUpdateMessage

Description:

Method	*Return values*	*Description*
getUpdateMessage(String token)	***String***	*Retrieves a message when there is a new OpenKM release.*

There's an official OpenKM update message service available which is based on your local OpenKM version.

The most common message is that a new OpenKM version has been released.

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            String msg = okmRepository.getUpdateMessage(null);
            System.out.println(msg);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRepositoryUuid

Description:

Method	*Return values*	*Description*
getRepositoryUuid(String token)	***String***	*Retrieves the installation unique identifier.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            String uuid = okmRepository.getRepositoryUuid(null);
            System.out.println(uuid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasNode

Description:

Method	Return values	Description
hasNode(String token, String nodeId)	**boolean**	Returns a node that indicate if a node exists or not.
The value of the parameter nodeId can be a valid UUID.		

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            boolean found = okmRepository.hasNode(null, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474");
            System.out.println(found);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodePath

Description:

Method	Return values	Description
getNodePath(String token, String uuid)	**String**	Converts a node UUID to path.

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            String path = okmRepository.getNodePath(null, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474");
            System.out.println(path);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

getNodeUuid

Description:

Method	*Return values*	*Description*
getNodeUuid(String token, String path)	***String***	*Converts a node path to UUID.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            String uuid = okmRepository.getNodeUuid(null, "/okm:root/api");
            System.out.println(uuid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAppVersion

Description:

Method	*Return values*	*Description*
getAppVersion(String token)	***AppVersion***	*Returns information about OpenKM version.*

Example:

```
package com.openkm;
import com.openkm.api.OKMRepository;
import com.openkm.bean.AppVersion;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            AppVersion appVersion = okmRepository.getAppVersion(null);
            System.out.println(appVersion);
        } catch (Exception e) {
            e.printStackTrace();
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
        }
    }
}
```

copyAttributes

Description:

Method	*Return values*	*Description*
copyAttributes(String token, String srcId, String dstId, ExtendedAttributes extAttr)	***void***	*Copy attributes from a node to other..*

The values of the dstId parameter should be a node ***UUID***.

- *When the category parameter is true the original values of the categories will be copied.*
- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the property Groups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original values of the notes will be copied.*
- *When the wiki parameter is true the original values of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
            ExtendedAttributes extAttr = new ExtendedAttributes();
            extAttr.setKeywords(true);
            extAttr.setCategories(true);
            extAttr.setNotes(true);
            extAttr.setPropertyGroups(true);
            extAttr.setWiki(true);
            okmRepository.copyAttributes(null, "d168924d-e801-426d-a222-00124a1461bd", "9f64248c-e049-4706-b89
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMetadataFolder

Description:

Method	*Return values*	*Description*
getMetadataFolder(String token)	***Folder***	*Return the metadata folder.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);

            Folder folder = okmRepository.getMetadataFolder(null);
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChangeLog

Description:

Method	*Return values*	*Description*
getChangeLog(String token, String nodePath, Calendar modifyFrom)	***List<ChangeLogged>***	*Return the list of changes in some path and subfolders.*

- *The method is used by desktop synchronization application for retrieving the changes.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMRepository;
import com.openkm.bean.ChangeLogged;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);

            Calendar now = Calendar.getInstance();
            for (ChangeLogged cl : okmRepository.getChangeLog(null, "/okm:root/synchronized", now)) {
                System.out.println(cl);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAppLicense

Description:

Method	*Return values*	*Description*
getAppLicense(String token)	***LicenseInfo***	*Return license information.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRepository;
import com.openkm.util.ContextWrapper;
import com.openkm.util.Update.LicenseInfo;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);

            LicenseInfo licenseInfo = okmRepository.getAppLicense(null);
            System.out.println(licenseInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMSearch

Basics

Almosts all methods use QueryParams. Here there're some tips about how using it.

Variables	*Type*	*Allow wildcards*	*Restrictions*
domain	***long***	*No.*	*Available values:* • *QueryParams.DOCUMENT* • *QueryParams.FOLDER* • *QueryParams.MAIL* • *QueryParams.RECORD* *By default the value is set to QueryParams.DOCUMENT.* *For searching documents and folders use value:* `(QueryParams.DOCUMENT | QueryParams.FOLDER)`
author	***String***	*No.*	*The value must be a valid userId.*
name	***String***	*Yes.*	
title	***String***	*Yes.*	
keywords	***Set<String>***	*Yes.*	
categories	***Set<String>***	*No.*	*Values should be a category UUID, not use path value.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

content		*Yes.*	
mimeType		*No.*	*The value should be a valid and registered MIME type.* *Only can be applied to documents node.*
language		*No.*	*The value should be a valid language.* *Only can be applied to documents node.*
folder		*No.*	*When empty is used by default "/okm:root" node.* *The value should be a valid UUID, not use a path value.*
folderRecursive	***Boolean***	*No.*	*Only has sense to set this variable to true when the variable folder is not*
lastModifiedFrom	***Calendar***	*No.*	
lastModifiedTo	***Calendar***	*No.*	
mailSubject	***String***	*Yes.*	*Only apply to mail nodes.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

mailFrom	***String***	*Yes.*	*Only apply to mail nodes.*
mailTo		*Yes.*	*Only apply to mail nodes.*
notes		*Yes.*	

properties	***Map<String, String>***	*Yes on almost.*	*On metadata field values like "date" can not be applied wilcards.*
			
			The map of the properties is composed of pairs:
			
			('metadata_field_name','metada_field_value")
			
			For example:
			
			`Map<String, String> properties = new HashMap();`
			`properties.put("okp:consulting.name", "name value");`
			
			Filtering by range of dates:
			
			`Calendar to = Calendar.getInstance(); // today`
			`to.set(0, Calendar.HOUR);`
			`to.set(0, Calendar.MINUTE);`
			`to.set(0, Calendar.SECOND);`
			`to.set(0, Calendar.MILLISECOND);`
			`Calendar from = (Calendar) to.clone();`
			`from.add(-3, Calendar.DATE); // three days before`
			`Map<String,String> properties = new HashMap<>();`
			`properties.put("okp:consulting.date", ISO8601.formatBasic(from)`
			
			When filtering by a range of dates you must set both values (will be ignored from the OpenKM side.
			
			To filtering by a metadata field of type multiple like this:

```
<select label="Multiple" name="okp:consulting.multiple" type="m
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>
```

You should do:

```
properties.put("okp:consulting.multiple", "one;two");
```

*Where "**one**" and "**two**" are valid values and character "**;**" is used as sep*



The search operation is done only by AND logic.

Wildcard examples:

Variable	*Example*	*Description*
name	***test*.html***	*Any document that starts with characters "test" and ends with characters ".html"*
name	***test?.html***	*Any document that starts with the characters "test" followed by a single character and ends with characters ".html"*
name	***?test****	*Any of the documents where the first character doesn't matter but is followed by the characters, "test".*

Methods

find

Description:

Method	*Return values*	*Description*
find(String token, QueryParams params)	***List<QueryResult>***	*Returns a list of results filtered by the values of the queryParams parameter.*

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            for (QueryResult qr : okmSearch.find(null, qParams)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

find

Description:

Method	*Return values*	*Description*
find(String token, QueryParams params, String propertiesPlugin)	***List<QueryResult>***	*Returns a list of results filtered by the values of the queryParams parameter.*



The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            for (QueryResult qr : okmSearch.find(null, qParams, "com.openkm.plugin.properties.SimpleNodeProperties",
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findPaginated

Description:

Method	*Return values*	*Description*
findPaginated(String token, QueryParams params, int offset, int limit)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");
            ResultSet rs = okmSearch.findPaginated(null, qParams, 20, 10);
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadata

Description:

Method	*Return values*	*Description*
findWithMetadata(String token, QueryParams params, String propertiesPlugin, List<String> groups)	***List<QueryResult>***	*Returns a list of results filtered by the values of the queryParams parameter.*

*The parameter "**propertiesPlugin**" must be a canonical class name of the class which implements the NodeProperties interface.*

*The parameter "**groups**" is the name metadata group.*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import java.util.ArrayList;
```

```
import java.util.List;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:tpl");

            for (QueryResult qr : okmSearch.findWithMetadata(null, qParams, "com.openkm.plugin.properties.SimpleNo
                    groups)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadataPaginated

Description:

Method	*Return values*	*Description*
findWithMetadataPaginated(String token, QueryParams params, int offset, int limit, String propertiesPlugin, List<String> groups)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" used to skip that many results before the beginning to return results.*

*The parameter "**propertiesPlugin**" must be a canonical class name of the class which implements the NodeProperties interface.*

*The parameter "**groups**" is the name metadata group.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*

- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:tpl");

            ResultSet rs = okmSearch.findWithMetadataPaginated(null, qParams, 0, 10, "com.openkm.plugin.properties
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQuery

Description:

Method	*Return values*	*Description*
findByQuery(String token,	***List<QueryResult>***	*Returns a list of paginated results filtered by the value of the*

String query)		*query parameter.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (QueryResult qr : okmSearch.findByQuery(null, "text:grial AND name:o*.pdf")) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQuery

Description:

Method	*Return values*	*Description*
findByQuery(String token, String query, String propertiesPlugin)	***ResultSet***	*Returns a list of paginated results filtered by the value of the query parameter.*

*The parameter "**propertiesPlugin**" must be the canonical class name of the class which implements the NodeProperties interface.*

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (QueryResult qr : okmSearch.findByQuery(null, "text:grial AND name:o*.pdf",
                    "com.openkm.plugin.properties.SimpleNodeProperties")) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQuery

Description:

Method	*Return values*	*Description*
findByQuery(String token, String query, String sortField, boolean sortReverse, String propertiesPlugin)	***ResultSet***	*Returns a list of paginated results filtered by the value of the query parameter.*

*The parameter "**propertiesPlugin**" must be the canonical class name of the class which implements the NodeProperties interface.*

*Available **sortField** values:*

```
name
author
lastModified
```

*The **syntax** to use in the statement parameter is the pair '**field:value**'. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (QueryResult qr : okmSearch.findByQuery(null, "text:grial AND name:o*.pdf", "name", true, "com.openkm
                System.out.println(qr);
            }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQueryPaginated

Description:

Method	*Return values*	*Description*
findByQueryPaginated(String token, String query, int offset, int limit)	***ResultSet***	*Returns a list of paginated results filtered by the value of the query parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            ResultSet rs = okmSearch.findByQueryPaginated(null, "text:grial AND name:o*.pdf", 10, 10);
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQueryPaginated

Description:

Method	*Return values*	*Description*
findByQueryPaginated(String token, String query, int offset, int limit, String propertiesPlugin)	***ResultSet***	*Returns a list of paginated results filtered by the value of the query parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

*The parameter "**propertiesPlugin**" must be a canonical class name of the class which implements the NodeProperties interface.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

case you can use NodeProperties classes for retrieving the Object variables what you really need.

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            ResultSet rs = okmSearch.findByQueryPaginated(null, "text:grial AND name:o*.pdf", 0, 10,
                    "com.openkm.plugin.properties.SimpleNodeProperties");
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQueryPaginated

Description:

Method	*Return values*	*Description*
findByQueryPaginated(String token, String query, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)	***ResultSet***	*Returns a list of paginated results filtered by the value of the query parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

*The parameter "**propertiesPlugin**" must be a canonical class name of the class which implements the NodeProperties interface.*

*Available **sortField** values:*

```
name
author
lastModified
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            ResultSet rs = okmSearch.findByQueryPaginated(null, "text:grial AND name:o*.pdf", "name", true, 0, 10,
                    "com.openkm.plugin.properties.SimpleNodeProperties");
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleNodeBaseByQueryPaginated

Description:

Method	*Return values*	*Description*
findSimpleNodeBaseByQueryPaginated(String token, String query, String sortField, boolean sortReverse, int offset, int limit)	***SimpleNodeBaseResultSet***	*Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

*Available **sortField** values:*

```
name
author
lastModified
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.SimpleNodeBase;
import com.openkm.ws.rest.util.SimpleNodeBaseResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");
            SimpleNodeBaseResultSet resultSet = okmSearch.findSimpleNodeBasePaginated(null, qParams, "name", t
            System.out.println("Total results:" + resultSet.getTotal());

            for (SimpleNodeBase simpleNodeBase : resultSet.getResults()) {
                System.out.println(simpleNodeBase);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveSearch

Description:

Method	*Return values*	*Description*
saveSearch(String token, QueryParams params)	***Long***	*Saves search parameters.*
The variable queryName of the parameter params should have to be initialized.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            // Save the search to be used later
            qParams.setQueryName("sample search");
            okmSearch.saveSearch(null, qParams);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateSearch

Description:

Method	*Return values*	*Description*
updateSearch(String token, QueryParams params)	***void***	*Saves a search parameters.*



Only can be updated as a saved search created by the same user user who's executing the method.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (QueryParams qParams : okmSearch.getAllSearches(null)) {
                if (qParams.getQueryName().equals("sample search")) {
                    // Change some value.
                    qParams.setName("admin*.html");
                    okmSearch.updateSearch(null, qParams);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSearch

Description:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Method	Return values	Description
getSearch(String token, int qpId)	**QueryParams**	Gets saved search.



Only can be updated as a saved search created by the same user user who's executing the method.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            int qpId = 1; // Some valid search id
            QueryParams qParams = okmSearch.getSearch(null, qpId);
            System.out.println(qParams);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllSearches

Description:

Method	Return values	Description
getAllSearches(String token)	**List<QueryParams>**	Retrieves a list of all saved search.



Only can be updated as a saved search created by the same user who's executing the method.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (QueryParams qParams : okmSearch.getAllSearches(null)) {
```

```
            System.out.println(qParams);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

deleteSearch

Description:

Method	*Return values*	*Description*
deleteSearch(String token, long qpId)	***void***	*Deletes a saved search.*

Only can be updated as a saved search created by the same user user who's executing the method.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            int qpId = 1; // Some valid search id
            okmSearch.deleteSearch(null, qpId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getKeywordMap

Description:

Method	*Return values*	*Description*
getKeywordMap(String token, List<String> filter)	***Map<String, Integer>***	*Returns a map of keywords with its count value filtered by other keywords.*

Example:

- *Doc1.txt has keywords "test", "one", "two".*

- *Doc2.txt has keywords "test", "one"*
- *Doc3.txt has keywords "test", "three".*

The results filtering by "test" -> "one", "two", "three".

The results filtering by "one" -> "test", "two".

The results filtering by "two" -> "test", "one".

The results filtering by "three" -> "test".

The results filtering by "one" and "two" -> "test"

```
package com.openkm;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Map;

import com.openkm.api.OKMSearch;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            // All keywords without filtering
            System.out.println("Without filtering");
            Map<String, Integer> keywords = okmSearch.getKeywordMap(null, new ArrayList<String>());

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used :" + keywords.get(key));
            }

            // Keywords filtered
            System.out.println("Filtering");
            keywords = okmSearch.getKeywordMap(null, Arrays.asList("test"));

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used :" + keywords.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedDocuments

Description:

Method	*Return values*	*Description*
getCategorizedDocuments(String token, String categoryId)	***List<Document>***	*Retrieves a list of all documents related with a category.*

The values of the categoryId parameter should be a category folder UUID or path.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Document doc : okmSearch.getCategorizedDocuments(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f"))
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedFolders

Description:

Method	*Return values*	*Description*
getCategorizedFolders(String token, String categoryId)	***List<Folder>***	*Retrieves a list of all folders related to a category.*
The values of the categoryId parameter should be a category folder UUID or path.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Folder fld : okmSearch.getCategorizedFolders(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedMails

Description:

Method	*Return values*	*Description*
getCategorizedMails(String token, String categoryId)	***List<Mail>***	*Retrieves a list of all mails related to a category.*
The values of the categoryId parameter should be a category folder UUID or path.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Mail mail : okmSearch.getCategorizedMails(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedRecords

Description:

Method	*Return values*	*Description*
getCategorizedRecords(String token, String categoryId)	***List<Record>***	*Retrieves a list of all records related to a category.*
The values of the categoryId parameter should be a category folder UUID or path.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Record rec : okmSearch.getCategorizedRecords(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentsByKeyword

Description:

Method	*Return values*	*Description*
getDocumentsByKeyword(String token, String keyword)	***List<Document>***	*Retrieves a list of all documents related to a keyword.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Document doc : okmSearch.getDocumentsByKeyword(null, "test")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFoldersByKeyword

Description:

Method	*Return values*	*Description*
getFoldersByKeyword(String token, String keyword)	***List<Folder>***	*Retrieves a list of all folders related to a keyword.*

```
package com.openkm;
```

```
import com.openkm.api.OKMSearch;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Folder fld : okmSearch.getFoldersByKeyword(null, "test")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailsByKeyword

Description:

Method	*Return values*	*Description*
getMailsByKeyword(String token, String keyword)	***List<Mail>***	*Retrieves a list of all mails related to a keyword.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Mail mail : okmSearch.getMailsByKeyword(null, "test")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRecordsByKeyword

Description:

Method	*Return values*	*Description*
getRecordsByKeyword(String token, String keyword)	***List<Record>***	*Retrieves a list of all records related to a keyword.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Record rec : okmSearch.getRecordsByKeyword(null, "test")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentsByPropertyValue

Description:

Method	*Return values*	*Description*
getDocumentsByPropertyValue(String token, String group, String property, String value	***List<Document>***	*Retrieves a list of all documents related with a metadata field value.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Document doc : okmSearch.getDocumentsByPropertyValue(null, "okg:consulting", "okp:consulting.name
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFoldersByPropertyValue

Description:

Return

Method	values	Description
getFoldersByPropertyValue(String token, String group, String property, String value	**List<Folder>**	Retrieves a list of all folders related with a metadata field value.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Folder fld : okmSearch.getFoldersByPropertyValue(null, "okg:consulting", "okp:consulting.name", "value
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailsByPropertyValue

Description:

Method	Return values	Description
getMailsByPropertyValue(String token, String group, String property, String value	**List<Mail>**	Retrieves a list of all mails related with a metadata field value.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Mail mail : okmSearch.getMailsByPropertyValue(null, "okg:consulting", "okp:consulting.name", "value"))
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

getRecordsByPropertyValue

Description:

Method	*Return values*	*Description*
getRecordsByPropertyValue(String token, String group, String property, String value)	***List<Record>***	*Retrieves a list of all records related with a metadata field value.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Record rec : okmSearch.getRecordsByPropertyValue(null, "okg:consulting", "okp:consulting.name", "val
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findMailPaginated

Description:

Method	*Return values*	*Description*
findMailPaginated(String token, Map<String, String> filter, String sortField, boolean sortReverse, int type, int offset, int limit)	***ResultSet***	*Returns a list of paginated mail results filtered by the values of the parameters.*

Allowed filter key values are:

- *Mail.SENT_DATE*
- *Mail.SUBJECT*
- *Mail.TO*

Send data value format must be:

```
ISO8601.formatBasic(from)+","+ISO8601.formatBasic(to)
```

Allowed sortField values are:

- *"okm:" for a non sorted.*
- *Mail.TYPE*
- *Mail.SUBJECT*
- *Mail.FROM*
- *Mail.SENT_DATE*
- *Mail.SIZE*
- *Mail.REPLY*
- *Mail.TO*
- *Mail.CC*
- *Mail.BCC*
- *Mail.CONTENT*

Allowed type values are:

- *OKMSearch.TYPE_MAIL_ALL*
- *OKMSearch.TYPE_MAIL_WITHOUT_ATTACHMENT*
- *OKMSearch.TYPE_MAIL_WITH_ATTACHMENT*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

```
package com.openkm;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
import java.util.HashMap;
import java.util.Map;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            Map<String, String> filter = new HashMap<>();
            String sortField = "okm:";
            int type = OKMSearch.TYPE_MAIL_ALL;
            ResultSet rs = okmSearch.findMailPaginated(null, filter, sortField, false, type, 0, 10);
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

csvExport

Description:

Method	*Return values*	*Description*
csvExport(String lang, QueryParams queryParams, boolean compact, String filename)	***InputStream***	*Export as a csv a list of results filtered by the values of the queryParams parameter.*

*The parameter **lang** must be ISO 691-1 compliant.*

More information at: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;
import org.apache.commons.io.IOUtils;
```

```
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            InputStream is = ws.search.csvExport("es-ES", params, false, "export.csv");
            OutputStream fos = new FileOutputStream("/home/openkm/okm/export.csv");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMStamp

Methods

calculateStampCoordinates

Description:

Method	*Return values*	*Description*
calculateStampCoordinates(String token, long imgToStampWidth, long imgToStampHeight, long floatingDivWidth,long floatingDivHeight, String exprX, String exprY, String stampType, String stampAlign)	***StampCoordinates***	*Return the calculated stamp coordinates.*

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- *IMAGE_WIDTH*
- *IMAGE_HEIGHT*
- *PAGE_WIDTH*
- *PAGE_HEIGHT*
- *PAGE_CENTER*
- *PAGE_MIDDLE*

So to center a stamp in the page you can use:

Expr. X	*PAGE_CENTER*
Expr. Y	*PAGE_MIDDLE*

*The parameter **stampAlign.***

Available values:

- *text*
- *image*

*The parameter **stampAlign** is the Text Align*

Available values:

- *left*
- *center*
- *right*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.StampCoordinates;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long floatingDivWidth = 250;
            long floatingDivHeight = 300;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            String stampType = "text";
            String stampAlign = "center";
            StampCoordinates stampCoordinates = okmStamp.calculateStampCoordinates(null, imgToStampWidth, img
                    floatingDivWidth, floatingDivHeight, exprX, exprY, stampType, stampAlign);
            System.out.println("X = " + stampCoordinates.getX() + ", Y = " + stampCoordinates.getY());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

calculateStampExpressions

Description:

Method	*Return values*	*Description*
calculateStampExpressions(String token, long imgToStampWidth, long imgToStampHeight, long posX, long posY)	***StampExpressions***	*Return the calculated stamp expressions for X and Y.*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;
```

```
import com.openkm.ws.rest.util.StampExpressions;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long posX = 10;
            long posY = 50;
            StampExpressions stampExpressions = okmStamp.calculateStampExpressions(null, imgToStampWidth, img
            System.out.println("X = " + stampExpressions.getExprX() + ", Y = " + stampExpressions.getExprY());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampDocument

Description:

Method	*Return values*	*Description*
stampDocument(String token, long id, int type, String uuid)	***void***	*Stamp the document*

The parameter ***type.***

- *Value 0 for type text*
- *Value 1 for type image*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            long stampId = 1;
            okmStamp.stampDocument(null, stampId, 1, "babe24df-c443-49a5-9453-39dfc9b27924");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

stampTextCustom

Description:

Method	*Return values*	*Description*
stampTextCustom(String token, String docId, long stId, String exprX, String exprY, String range, String text)	***void***	*Stamp in a document with a custom text.*

*The **id** is the id of an image stamp which is created in the administrator.*

*The **exprX** is the expression to set the position in X coordinates.*

*The **exprY** is the expression to set the position in Y coordinates.*

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- *IMAGE_WIDTH*
- *IMAGE_HEIGHT*
- *PAGE_WIDTH*
- *PAGE_HEIGHT*
- *PAGE_CENTER*
- *PAGE_MIDDLE*

So to center a stamp in the page you can use:

Expr. X	*PAGE_CENTER*
Expr. Y	*PAGE_MIDDLE*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

import java.io.FileInputStream;
import java.io.InputStream;

public class Test {

    public static void main(String[] args) {
        try {
            long stampTextId = 1;
```

```
            String text = "OpenKM";
            String exprX = "PAGE_CENTER";
            String exprY = "PAGE_MIDDLE";
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.stampTextCustom(null, "05b14eee5-2e57-4d1d-925a-c9bc3f526e24", stampTextId, exprX, e
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampImageCustom

Description:

Method	*Return values*	*Description*
void stampImageCustom(String token, String uuid, String id, String exprX, String exprY, String range, String personalStampUuid)	***void***	*Stamp a document with a custom image.*

> *The **personalStampUuid** must be a valid document **UUID**. The document must be an image (jpg or png).*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            long stampId = 3;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            okmStamp.stampImageCustom(null, "babe24df-c443-49a5-9453-39dfc9b27924", stampId, exprX, exprY, "",
                    "928de61e-6ceb-4f94-bf20-9ba2405c5cee");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampImageCustom

Description:

Method	Return values	Description
void stampImageCustom(String token, String uuid, String id, String exprX, String exprY, String range, byte[] image)	**void**	*Stamp a document with a custom image.*

> *The **image** must be an image (jpg or png).*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;
import org.apache.commons.io.*;
import java.io.*;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            InputStream is = FileInputStream("/home/test/image.png");
            byte[] image = IOUtils.toByteArray(is);
            long stampId = 3;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            okmStamp.stampImageCustom(null, "babe24df-c443-49a5-9453-39dfc9b27924", stampId, exprX, exprY, "",
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllStamps

Description:

Method	Return values	Description
getAllStamps(String token)	**List<StampItem>**	*Returns a list of stamps*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;
```

```
import com.openkm.ws.rest.util.StampItem;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            List<StampItem> result = okmStamp.getAllStamps();
            for (StampItem stampItem : result) {
                System.out.println(stampItem.getName());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStampTextByPk

Description:

Method	*Return values*	*Description*
getStampTextByPk(String token, long itemId, String docId)	***StampText***	*Return the stamp Object by Id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.extension.db.bean.StampText;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            int stampId = 1;
            StampText stampText = okmStamp.getStampTextByPk(null, stampId, "928de61e-6ceb-4f94-bf20-9ba2405c
            System.out.println(stampText.getName());
            System.out.println(stampText.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStampImageByPk

Description:

Method	*Return values*	*Description*
getStampImageByPk(String token, long id, String docId)	***StampImage***	*Return the stamp Object by Id.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.extension.db.bean.StampImage;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            int stampId = 3;
            StampImage stampImage = okmStamp.getStampImageByPk(null, stampId, "928de61e-6ceb-4f94-bf20-9ba2
            System.out.println(stampImage.getName());
            System.out.println(stampImage.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStampBarcodeByPk

Description:

Method	*Return values*	*Description*
getStampBarcodeByPk(String token, long id, String docId)	***StampBarcode***	*Return the stamp Object by Id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.extension.db.bean.StampBarcode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            int stampId = 3;
            StampBarcode stampBarcode = okmStamp.getStampBarcodeByPk(null, stampId, "928de61e-6ceb-4f94-bf2
            System.out.println(stampBarcode.getName());
            System.out.println(stampBarcode.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

calculateFlyImageDimensions

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
calculateFlyImageDimensions(String token, String uuid, long imgToStampWidth, long imgToStampHeight,long floatingImageWidth, long floatingImageHeight, int pageNumber)	***StampImageSize***	*Return the calculated image dimensions of the stamp.*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.StampImageSize;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long floatingImageWidth = 300;
            long floatingImageHeight = 300;
            int pageNumber = 2;

            StampImageSize stampImageSize = okmStamp.calculateFlyImageDimensions(null, "babe24df-c443-49a5-9
            System.out.println(stampImageSize);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalStamps

Description:

Method	*Return values*	*Description*
getPersonalStamps(String token)	***List<Document>***	*Returns a list of personal seals.*

Return a list of documents of type image (jpg or png) from /okm:templates/ userId/stamps

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.api.OKMStamp;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            List<Document> result = okmStamp.getPersonalStamps(null);
            for (Document doc : result) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalStampImage

Description:

Method	*Return values*	*Description*
getPersonalStampImage(String token, String uuid, long id)	***StampPersonalImage***	*Return the personal seal processed.*

> *When stamping a document with an image, the image also can have a text layer.*
>
> *This method processes the image and returns an image with the text layer.*
>
> *The **uuid** must be one of the values returned by the method getPersonalStamps.*

Example:

```
package com.openkm;

import com.openkm.api.OKMStamp;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.StampPersonalImage;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMStamp okmStamp = ContextWrapper.getContext().getBean(OKMStamp.class);
            long stampId = 1;
            StampPersonalImage stampPersonalImage = okmStamp.getPersonalStampImage(null, "928de61e-6ceb-4f
            System.out.println(stampPersonalImage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMStats

Basics

*On all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

getDocumentsByContext

Description:

Method	Return values	Description
getDocumentsByContext(String token)	**StatsInfo**	Return stats information about documents by context.

Example:

```
package com.openkm;

import com.openkm.api.OKMStats;
import com.openkm.bean.StatsInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStats okmStats = ContextWrapper.getContext().getBean(OKMStats.class);
            StatsInfo sInfo = okmStats.getDocumentsByContext(null);
            System.out.println(sInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentsSizeByContext

Description:

Method	Return values	Description

getDocumentsByContext(String token)	***StatsInfo***	*Return stats information about documents size by context.*

Example:

```
package com.openkm;

import com.openkm.api.OKMStats;
import com.openkm.bean.StatsInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStats okmStats = ContextWrapper.getContext().getBean(OKMStats.class);
            StatsInfo sInfo = okmStats.getDocumentsSizeByContext(null);
            System.out.println(sInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFoldersByContext

Description:

Method	*Return values*	*Description*
getFoldersByContext(String token)	***StatsInfo***	*Return stats information about folders by context.*

Example:

```
package com.openkm;

import com.openkm.api.OKMStats;
import com.openkm.bean.StatsInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStats okmStats = ContextWrapper.getContext().getBean(OKMStats.class);
            StatsInfo sInfo = okmStats.getFoldersByContext(null);
            System.out.println(sInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailsByContext

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	Return values	Description
getMailsByContext(String token)	**StatsInfo**	Return stats information about mails by context.

Example:

```
package com.openkm;

import com.openkm.api.OKMStats;
import com.openkm.bean.StatsInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStats okmStats = ContextWrapper.getContext().getBean(OKMStats.class);
            StatsInfo sInfo = okmStats.getMailsByContext(null);
            System.out.println(sInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRecordsByContext

Description:

Method	Return values	Description
getRecordsByContext(String token)	**StatsInfo**	Return stats information about records by context.

Example:

```
package com.openkm;

import com.openkm.api.OKMStats;
import com.openkm.bean.StatsInfo;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMStats okmStats = ContextWrapper.getContext().getBean(OKMStats.class);
            StatsInfo sInfo = okmStats.getRecordsByContext(null);
            System.out.println(sInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMTask

Basics

Task fields description:

Field	*Type*	*Description*	*Mandatory*
Id	***Long***	*Internal task id.* *It is automatically set by the application.*	*Not applicable.*
Owner	***String***	*It contains the OpenKM userId.* *It is automatically set by the application.*	*Not applicable.*
Subject	***String***	*The topic of the task.*	*Yes.*
Description	***String***	*The description of the task.*	*No.*
Start	***Calendar***	*When the task might start.*	*Yes.*
End	***Calendar***	*When the task might ending.*	*No.*
Status	***TaskStatus***	*The task status.* *You administering the list of status values.*	*Yes.*
Project	***TaskProject***	*The project related to the task.* *You administering the list of project values.*	*Yes.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Type	***TaskType***	*The task type.* *You administering the list of task type values.*	*Yes.*
Progress	***Integer***	*The numeric progress status.* *The range of allowed values are from 0 to 100. Where 100% indicates a completed task.*	*Yes.*
RepeatGroup	***Long***	*When task is repeated along time is a member of a group. The group is identified by an unique repeat group id.*	*No.*
ReminderStartValue	***Integer***	*How many days before the task starting, the system might send a mail notification to the user.*	*No.*
ReminderEndValue	***Integer***	*How many days before the task ending, the system might send a mail notification to the user.*	*No.*
ReminderStartUnit	***String***	*Reminder start units.* *Allowed values:* • *"m" for minutes.* • *"h" for hours.* • *"d" for days.*	*Mandatory when **ReminderStartValue** greater than 0.*
ReminderEndUnit	***String***	*Reminder end units.* *Allowed values:* • *"m" for minutes.* • *"h" for hours.* • *"d" for days.*	*Mandatory when **ReminderEndValue** greater than 0.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Users	***Set<String>***	*Collection of assigned users to the task.*	*No. But At least should be a user or role assigned.*
Roles	***Set<String>***	*Collection of assigned roles to the task.*	*No. But At least should be a user or role assigned.*
Documents	***Set<String>***	*List of UUID's of the related documents.*	*No.*
Folders	***Set<String>***	*List of UUID's of the related folders.*	*No.*
Mails	***Set<String>***	*List of UUID's of the related mails.*	*No.*
Records	***Set<String>***	*List of UUID's of the related records.*	*No.*

*On all methods, there's a parameter named "**token**". When accessing application across REST, the login process returns a token. The token is used to identify the user on all the exposed methods. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

create

Description:

Method	*Return values*	*Description*
create(String token, Task task)	***Task***	*Create a new task.*

Example:

```
package com.openkm;
import java.util.Calendar;
```

```
import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.bean.TaskProject;
import com.openkm.bean.TaskStatus;
import com.openkm.bean.TaskType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            Task task = new Task();
            task.setSubject("task subject");
            task.setDescription("description");
            Calendar start = Calendar.getInstance();
            start.add(Calendar.DATE, 15);
            task.setStart(start);
            long taskStatusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            TaskStatus status = okmTask.findStatus(null, taskStatusId);
            TaskProject project = okmTask.findProject(null, projectId);
            TaskType type = okmTask.findType(null, typeId);
            task.setStatus(status);
            task.setProject(project);
            task.setType(type);
            task.setProgress(0);
            okmTask.create(null, task);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

update

Description:

Method	*Return values*	*Description*
update(String token, Task task)	***Task***	*Update a task.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long taskId = 1; // A valid task id
            Task task = okmTask.find(null, taskId);
```

```
            task.setSubject("subject modified");
            okmTask.update(null, task);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

find

Description:

Method	*Return values*	*Description*
find(String token, Task task)	***Task***	*Find a task.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long taskId = 1; // A valid task id
            Task task = okmTask.find(null, taskId);
            System.out.println(task);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	*Return values*	*Description*
delete(String token, Task task)	***void***	*Delete a task.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long taskId = 1; // A valid task id
            okmTask.delete(null, taskId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getByProject

Description:

Method	*Return values*	*Description*
getByProject(String token, long projectId)	***List<Task>***	*Retrieve a list of task related with a project.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            for (Task task : okmTask.getByProject(null, projectId)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getByType

Description:

Method	*Return values*	*Description*
getByType(String token, long typeId)	***List<Task>***	*Retrieve a list of task related with a task type.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long typeId = 1; // A valid type id
            for (Task task : okmTask.getByType(null, typeId)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAssigned

Description:

Method	*Return values*	*Description*
getAssigned(String token, long projectId, long typeId, long statusId, String orderColumn)	***List<Task>***	*Retrieve a list of tasks assigned to a user and filtered.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
            for (Task task : okmTask.getAssigned(null, projectId, typeId, statusId, orderColumn)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAssigned

Description:

Method	*Return values*	*Description*
getAssigned(String token, long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit)	***List<Task>***	*Retrieve a list of tasks assigned to a user, filtered and paginated.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query
            for (Task task : okmTask.getAssigned(null, projectId, typeId, statusId, orderColumn, true, 0, 10)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

countAssigned

Description:

Method	*Return values*	*Description*
countAssigned(String token, long projectId, long typeId, long statusId)	***Long***	*Return the number of tasks assigned to a user.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = okmTask.countAssigned(null, projectId, typeId, statusId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActive

Description:

Method	*Return values*	*Description*
getActive(String token, long projectId, long typeId, long statusId, String orderColumn)	***List<Task>***	*Retrieve a list of active tasks assigned to a user and filtered.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query
            for (Task task : okmTask.getActive(null, projectId, typeId, statusId, orderColumn)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActive

Description:

Method	*Return values*	*Description*
getActive(String token, long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit)	***List<Task>***	*Retrieve a list of active tasks assigned to a user, filtered and paginated.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query
            for (Task task : okmTask.getActive(null, projectId, typeId, statusId, orderColumn, true, 0, 10)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

countActive

Description:

Method	*Return values*	*Description*
countActive(String token, long projectId, long typeId, long statusId)	***Long***	*Return the number of active tasks of a user.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = okmTask.countAssigned(null, projectId, typeId, statusId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinished

Description:

Method	*Return values*	*Description*
getFinished(String token, long projectId, long typeId, long statusId, String orderColumn)	***List<Task>***	*Retrieve a list of finished tasks assigned to a user and filtered.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query
            for (Task task : okmTask.getFinished(null, projectId, typeId, statusId, orderColumn)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinished

Description:

Method	*Return values*	*Description*
getFinished(String token, long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit)	***List<Task>***	*Retrieve a list of finished tasks assigned to a user, filtered and paginated.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query
            for (Task task : okmTask.getFinished(null, projectId, typeId, statusId, orderColumn, true, 0, 10)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

countFinished

Description:

Method	*Return values*	*Description*
countFinished(String token, long projectId, long typeId, long statusId)	***Long***	*Return the number of finished tasks of a user.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = okmTask.countFinished(null, projectId, typeId, statusId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findStatus

Description:

Method	*Return values*	*Description*
findStatus(String token, long statusId)	***TaskStatus***	*Return a task status object by id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskStatus;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long statusId = 1; // A valid status id
```

```
            TaskStatus ts = okmTask.findStatus(null, statusId);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findProject

Description:

Method	*Return values*	*Description*
findProject(String token, long projectId)	***TaskProject***	*Return a task project object by id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskProject;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            TaskProject tp = okmTask.findProject(null, projectId);
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findType

Description:

Method	*Return values*	*Description*
findType(String token, long typeId)	***TaskType***	*Return a task type object by id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskType;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long typeId = 1; // A valid type id
            TaskType tt = okmTask.findType(null, typeId);
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findNote

Description:

Method	*Return values*	*Description*
findNote(String token, long noteId)	***TaskNote***	*Return a task note object by id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskNote;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long noteId = 2; // A valid note id
            TaskNote tn = okmTask.findNote(null, noteId);
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStatuses

Description:

Method	*Return values*	*Description*
getStatuses(String token)	***List<TaskStatus>***	*Retrieve a list of all the task status.*

Example:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskStatus;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            for (TaskStatus ts : okmTask.getStatuses(null)) {
                System.out.println(ts);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProjects

Description:

Method	*Return values*	*Description*
getProjects(String token, boolean filterByActive)	***List<TaskProject>***	*Retrieve a list of all the task projects.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskProject;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            for (TaskProject tp : okmTask.getProjects(null, true)) {
                System.out.println(tp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTypes

Description:

Method	*Return values*	*Description*

getTypes(String token, boolean filterByActive)	***List<TaskType>***	*Retrieve a list of all the task types.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            for (TaskType tt : okmTask.getTypes(null, true)) {
                System.out.println(tt);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotes

Description:

Method	*Return values*	*Description*
getNotes(String token, long taskId)	***List<TaskNote>***	*Retrieve a list of all the notes of a task.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskNote;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long taskId = 1; // A valid task id
            for (TaskNote tn : okmTask.getNotes(null, taskId)) {
                System.out.println(tn);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

createStatus

Description:

Method	*Return values*	*Description*
createStatus(String token, String name, boolean finish)	***TaskStatus***	*Create a new task status.*

> *Depending on your logic, several status can be ending status. For example, status named "closed" or "cancelled". The boolean parameter "finish" indicates if your status in an "ending status".*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskStatus;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            TaskStatus ts = okmTask.createStatus(null, "cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createProject

Description:

Method	*Return values*	*Description*
createProject(String token, String name, String description, boolean active)	***TaskProject***	*Create a new task project.*

> *The boolean parameter "active", indicates if your project is active or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskProject;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            TaskProject tp = okmTask.createProject(null, "Project one", "Description", true);
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createType

Description:

Method	*Return values*	*Description*
createType(String token, String name, String description, boolean active)	***TaskType***	*Create a new task type.*

> *The boolean parameter "active", indicates if your type is active or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            TaskType tt = okmTask.createType(null, "Type one", "Description", true);
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createNote

Description:

Method	Return values	Description
createNote(String token, long taskId, String text)	**TaskNote**	Create a new note to a task.

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskNote;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long taskId = 1; // A valid task id
            TaskNote tn = okmTask.createNote(null, taskId, "New note");
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateStatus

Description:

Method	Return values	Description
updateStatus(String token, long id, String name, boolean finish)	**TaskStatus**	Update a task status.

> *Depending on your logic, several status can be ending status. For example, status named "closed" or "cancelled". The boolean parameter "finish" indicates if your status in an "ending status".*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskStatus;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long statusId = 1; // A valid task id
```

```
            TaskStatus ts = okmTask.findStatus(null, statusId);
            System.out.println(ts);
            ts = okmTask.updateStatus(null, statusId, "cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateProject

Description:

Method	*Return values*	*Description*
updateProject(String token, long id, String name, String description, boolean active)	***TaskProject***	*Update a task project.*

The boolean parameter "active", indicates if your project is active or not.

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskProject;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 4; // A valid project id
            TaskProject tp = okmTask.findProject(null, projectId);
            System.out.println(tp);
            tp = okmTask.updateProject(null, projectId, "cancelled", "Description", false);
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateType

Description:

Method	*Return values*	*Description*

updateType(String token, long id, String name, String description, boolean active)	***TaskType***	*Update a task type.*

> *The boolean parameter "active", indicates if your type is active or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long typeId = 4; // A valid type id
            TaskType tp = okmTask.findType(null, typeId);
            System.out.println(tp);
            tp = okmTask.updateType(null, typeId, "Type one", "Description", false);
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateNote

Description:

Method	*Return values*	*Description*
updateNote(String token, long id, String text)	***TaskNote***	*Update note of a task.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskNote;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
```

```
            long noteId = 1; // A valid note id
            TaskNote tn = okmTask.updateNote(null, noteId, "Note updated");
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteStatus

Description:

Method	*Return values*	*Description*
deleteStatus(String token, long statusId)	***void***	*Delete a task status.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long statusId = 4; // A valid status id
            okmTask.deleteStatus(null, statusId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteProject

Description:

Method	*Return values*	*Description*
deleteProject(String token, long projectId)	***void***	*Delete a task project.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 5; // A valid project id
            okmTask.deleteProject(null, projectId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteType

Description:

Method	*Return values*	*Description*
deleteType(String token, long typeId)	***void***	*Delete a task type.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long typeId = 4; // A valid type id
            okmTask.deleteType(null, typeId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteNote

Description:

Method	*Return values*	*Description*
deleteNote(String token, long noteId)	***void***	*Delete note of a task.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long noteId = 1; // A valid note id
            okmTask.deleteNote(null, noteId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getByNode

Description:

Method	*Return values*	*Description*
getByNode(String token, String nodeUuid)	***List<Task>***	*Return the list of tasks assigned to a specific node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            String uuid = "d168924d-e801-426d-a222-00124a1461bd"; // A valid uuid
            for (Task task : okmTask.getByNode(null, uuid)) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getHistory

Description:

Method	*Return values*	*Description*
getHistory(String token, long taskId)	***List<TaskHistory>***	*Return the historic of a specific task.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskHistory;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long taskId = 1; // A valid task id
            for (TaskHistory taskHistory : okmTask.getHistory(null, taskId)) {
                System.out.println(taskHistory);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAssigned

Description:

Method	*Return values*	*Description*
getAssigned(String token, String subject, int progress, long projectId, long typeId, long statusId, Calendar lastModified, Calendar start, Calendar end, String orderColumn, boolean orderAsc, int offset, int limit)	***TaskList***	*Retrieve a list of tasks assigned to a user, filtered and paginated.*

Filter parameters description:

- *The subject parameter the topic of the task.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.TaskList;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            String subject = "test subject";
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
            boolean orderAsc = true;
            int offset = 0;
            int limit = 10;
            Calendar lastModified = Calendar.getInstance();
            Calendar start = Calendar.getInstance();
            start.add(Calendar.DATE, -30);
            Calendar end = Calendar.getInstance();

            TaskList taskList = okmTask.getAssigned(null, subject, 10, projectId, typeId, statusId, lastModified, start, end
                    orderColumn, orderAsc, offset, limit);
            for (Task task : taskList.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

countAssigned

Description:

Method	*Return values*	*Description*
countAssigned(String token, String subject, int progress, long projectId, long typeId, long statusId, Calendar lastModified, Calendar start, Calendar end)	***long***	*Return the number of tasks assigned to a user.*

Filter parameters description:

- *The subject parameter the topic of the task.*
- *The progress parameter is the progress of the task.*
- *The statusId parameters must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            String subject = "test subject";
            int progress = 10;
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            Calendar lastModified = Calendar.getInstance();

            Calendar start = Calendar.getInstance();
            start.add(Calendar.DATE, -30);

            Calendar end = Calendar.getInstance();

            long total = okmTask.countAssigned(null, subject, progress, projectId, typeId, statusId, lastModified, start, er
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
}
```

getActive

Description:

Method	*Return values*	*Description*
getActive(String token, String subject, int progress, long projectId, long typeId, long statusId, Calendar lastModified, Calendar start, Calendar end, String orderColumn, boolean orderAsc, int offset, int limit)	***TaskList***	*Retrieve a list of active tasks assigned to a user, filtered and paginated.*

Filter parameters description:

- *The subject parameter the topic of the task.*
- *The progress parameter is the progress of the task.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*

- *offset=10*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.TaskList;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            String subject = "test subject";
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "progress"; // A valid TaskManagerTask class parameter name used in HQL query
            boolean orderAsc = true;
            int offset = 0;
            int limit = 10;
            Calendar lastModified = Calendar.getInstance();
            Calendar start = Calendar.getInstance();
            start.add(Calendar.DATE, -30);
            Calendar end = Calendar.getInstance();

            TaskList taskList = okmTask.getActive(null, subject, 10, projectId, typeId, statusId, lastModified, start, end,
                    orderColumn, orderAsc, offset, limit);
            for (Task task : taskList.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

countActive

Description:

Method	*Return values*	*Description*
countActive(String token, String subject, int progress, long projectId, long typeId, long statusId, Calendar lastModified, Calendar start, Calendar end)	***long***	*Return the number of active tasks to a user.*

Filter parameters description:

- *The subject parameter the topic of the task.*
- *The progress parameter is progress of the task.*
- *The statusId parameters must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            String subject = "test subject";
            int progress = 10;
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            Calendar lastModified = Calendar.getInstance();

            Calendar start = Calendar.getInstance();
            start.add(Calendar.DATE, -30);

            Calendar end = Calendar.getInstance();

            long total = okmTask.countActive(null, subject, progress, projectId, typeId, statusId, lastModified, start, end);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinished

Description:

Method	*Return values*	*Description*
getFinished(String token, String subject, int progress, long projectId, long typeId, long statusId,	***TaskList***	*Retrieve a list of finished tasks assigned to a user, filtered and*

Calendar lastModified, Calendar start, Calendar end, String orderColumn, boolean orderAsc, int offset, int limit)		*paginated.*

Filter parameters description:

- *The subject parameter the topic of the task.*
- *The progress parameter is the progress of the task.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.TaskList;
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            String subject = "test subject";
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "end"; // A valid TaskManagerTask class parameter name used in HQL query
            boolean orderAsc = true;
            int offset = 0;
            int limit = 10;
            Calendar lastModified = Calendar.getInstance();
            Calendar start = Calendar.getInstance();
            start.add(Calendar.DATE, -30);
            Calendar end = Calendar.getInstance();

            TaskList taskList = okmTask.getFinished(null, subject, 10, projectId, typeId, statusId, lastModified, start, end,
                    orderColumn, orderAsc, offset, limit);
            for (Task task : taskList.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

countFinished

Description:

Method	*Return values*	*Description*
countFinished(String token, String subject, int progress, long projectId, long typeId, long statusId, Calendar lastModified, Calendar start, Calendar end)	***long***	*Return the number of finished tasks of a user.*

Filter parameters description:

- *The subject parameter the topic of the task.*
- *The progress parameter is the progress of the task.*
- *The statusId parameters must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            String subject = "test subject";
            int progress = 10;
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            Calendar lastModified = Calendar.getInstance();

            Calendar start = Calendar.getInstance();
            start.add(Calendar.DATE, -30);

            Calendar end = Calendar.getInstance();

            long total = okmTask.countFinished(null, subject, progress, projectId, typeId, statusId, lastModified, start, en
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAssignedCount

Description:

Method	*Return values*	*Description*
getAssignedCount(String token, long statusId, long projectId, long typeId)	***long***	*Return the number of tasks assigned to a user.*

Filter parameters description:

- *The statusId parameters must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id

            long total = okmTask.getAssignedCount(null, projectId, typeId, statusId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActiveCount

Description:

Method	*Return values*	*Description*
getActiveCount(String token, long statusId, long projectId, long typeId)	***Long***	*Return the number of active tasks of a user.*

Filter parameters description:

- *The statusId parameters must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
```

```
            long total = okmTask.getActiveCount(null, projectId, typeId, statusId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotified

Description:

Method	*Return values*	*Description*
getNotified(String token, long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit)	***List<Task>***	*Retrieve a list of notified tasks assigned to a user, paginated.*

Filter parameters description:

- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*

- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.Task;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.TaskList;

public class Test {

    public static void main(String[] args) {
        try {
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            String orderColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query

            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            TaskList taskList = okmTask.getNotified(null, projectId, typeId, statusId, orderColumn, true, 0, 10);
            for (Task task : taskList.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinishedCount

Description:

Method	*Return values*	*Description*
getFinishedCount(String token, long statusId, long projectId, long typeId)	***Long***	*Return the number of finished tasks of a user.*

Filter parameters description:

- *The statusId parameters must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameters must be a valid type id.*

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id

            long total = okmTask.getFinishedCount(null, projectId, typeId, statusId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMUserConfig

*On all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

setHome

Description:

Method	*Return values*	*Description*
setHome(String token, String nodeId)	***void***	*Sets the user home.*

By default user home value is "/okm:root".

Example:

```
package com.openkm;

import com.openkm.api.OKMUserConfig;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMUserConfig okmUserConfig = ContextWrapper.getContext().getBean(OKMUserConfig.class);
            okmUserConfig.setHome(null, "/okm:root/test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getConfig

Description:

Method	*Return values*	*Description*
getConfig(String token)	***UserConfig***	*Return the user configuration.*

Returned values are:

- *The userId.*
- *The home path.*
- *The home type (folder, document, mail or record).*
- *The home node (the UUID).*

Example:

```
package com.openkm;

import com.openkm.api.OKMUserConfig;
import com.openkm.db.bean.UserConfig;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMUserConfig okmUserConfig = ContextWrapper.getContext().getBean(OKMUserConfig.class);
            UserConfig uc = okmUserConfig.getConfig(null);
            System.out.println(uc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OKMWorkflow

*On all methods you'll see parameter named "**token**". Because the Cron Task are executed in background without authentication, the methods used in this scenario might use the token parameter. From default application execution context you must use "**null**" value what indicates to the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

registerProcessDefinition

Description:

Method	*Return values*	*Description*
registerProcessDefinition(String token, InputStream is)	***void***	*Registers a new workflow.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import com.openkm.util.ContextWrapper;
import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMWorkflow;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);
            InputStream is = new FileInputStream("/opt/files/Purchase.par")
            okmWorkflow.registerProcessDefinition(null, is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteProcessDefinition

Description:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Method	*Return values*	*Description*
deleteProcessDefinition(String token, long processDefinitionId)	***void***	*Deletes a workflow.*
The parameter processDefinitionId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);
            long pdId = 5; // Valid workflow process definition
            okmWorkflow.deleteProcessDefinition(null, pdId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProcessDefinitionImage

Description:

Method	*Return values*	*Description*
getProcessDefinitionImage(String token, long processDefinitionId, String node)	***byte[]***	*Returns a workflow image process definition.*
The parameter processDefinitionId value is a valid workflow process definition. *The parameter node is a node with a workflow in execution.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;
import org.apache.commons.io.IOUtils;

import java.io.ByteArrayInputStream;
```

```
import java.io.File;
import java.io.FileOutputStream;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pdId = 5; // Valid workflow process definition
            byte[] img = okmWorkflow.getProcessDefinitionImage(null, pdId, "/okm:root/sample.pdf");
            ByteArrayInputStream in = new ByteArrayInputStream(img);
            FileOutputStream out = new FileOutputStream(new File("/home/test/img.jpg"));
            IOUtils.copy(in, out);
            IOUtils.closeQuietly(in);
            IOUtils.closeQuietly(out);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProcessDefinitionForms

Description:

Method	*Return values*	*Description*
Map<String, List<FormElement>> getProcessDefinitionForms(String token, long processDefinitionId)	***Map<String, List<FormElement>>***	*Returns a map with all workflow form definitions.*
The parameter processDefinitionId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.form.FormElement;
import com.openkm.util.ContextWrapper;

import java.util.List;
import java.util.Map;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pdId = 5; // Valid workflow process definition
            Map<String, List<FormElement>> map = okmWorkflow.getProcessDefinitionForms(null, pdId);
            for (String key : map.keySet()) {
                System.out.println("group:" + key);
                for (FormElement fe : map.get(key)) {
```

```
                System.out.println(fe);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

getProcessDefinition

Description:

Method	*Return values*	*Description*
ProcessDefinition getProcessDefinition(String token, long processDefinitionId)	***ProcessDefinition***	*Returns a workflow process definition.*
The parameter processDefinitionId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.ProcessDefinition;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pdId = 5; // Valid workflow process definition
            ProcessDefinition pd = okmWorkflow.getProcessDefinition(null, pdId);
            System.out.println(pd);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

runProcessDefinition

Description:

Method	*Return values*	*Description*
runProcessDefinition(String token, long processDefinitionId, String uuid, List<FormElement> variables)	***ProcessInstance***	*Executes a workflow on a*

		node.

The parameter processDefinitionId value is a valid workflow process definition.

The parameter uuid can be any document, mail, folder or record UUID.

The parameter values are form element values needed for starting the workflow (not all workflows need form values for starting).

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.form.FormElement;
import com.openkm.bean.form.Input;
import com.openkm.bean.form.TextArea;
import com.openkm.util.ContextWrapper;

import java.util.ArrayList;
import java.util.List;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pdId = 8041; // Some valid workflow process definition id
            List<FormElement> feList = new ArrayList<>(); // Case as part of starting workflow are required form parame
            Input price = new Input();
            price.setName("price");
            price.setValue("1000");
            feList.add(price);
            TextArea textArea = new TextArea();
            textArea.setName("description");
            textArea.setValue("some description here");
            feList.add(textArea);
            okmWorkflow.runProcessDefinition(null, pdId, "f86cc22d-9b50-434f-a940-b04cea9c0048", feList);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendProcessInstanceSignal

Description:

Method	*Return values*	*Description*
sendProcessInstanceSignal(String token, long processInstanceId, String transitionName)	***ProcessInstance***	*Send a signal to a process intance id.*

The parameter processInstanceId value is a valid workflow process instance.

The parameter uuid can be any document, mail, folder or record UUID.

The parameter transactionName is the choosen transaction.

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pId = 1046; // Some valid process instance id
            okmWorkflow.sendProcessInstanceSignal(null, pId, "jump2");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

endProcessInstance

Description:

Method	*Return values*	*Description*
endProcessInstance(String token, long processInstanceId)	***void***	*Ends a process intance id.*

The parameter processInstanceId value is a valid workflow process instance.

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pId = 1046; // Some valid process instance id
            okmWorkflow.endProcessInstance(null, pId);
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

deleteProcessInstance

Description:

Method	*Return values*	*Description*
deleteProcessInstance(String token, long processInstanceId)	***void***	*Delete a process intance id.*
The parameter processInstanceId value is a valid workflow process instance.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pId = 1046; // Some valid process instance id
            okmWorkflow.deleteProcessInstance(null, pId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findProcessInstances

Description:

Method	*Return values*	*Description*
findProcessInstances(String token, long processDefinitionId)	***List<ProcessInstance>***	*Retrieves a list of all process instances of some registered workflows definition.*
The parameter processDefinitionId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.ProcessDefinition;
import com.openkm.bean.workflow.ProcessInstance;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            // Get all workflow definitions
            for (ProcessDefinition pd : okmWorkflow.findAllProcessDefinitions(null)) {
                System.out.println("WF definition: " + pd);

                // Get all process of some workflow definition
                for (ProcessInstance pi : okmWorkflow.findProcessInstances(null, pd.getId())) {
                    System.out.println("PI: " + pi);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findAllProcessDefinitions

Description:

Method	*Return values*	*Description*
findAllProcessDefinitions(String token, long processDefinitionId)	***List<ProcessDefinition>***	*Retrieves a list of all registered workflows definitions.*

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.ProcessDefinition;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            // Get all workflow definitions
            for (ProcessDefinition pd : okmWorkflow.findAllProcessDefinitions(null)) {
                System.out.println("WF definition: " + pd);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    }
}
```

findLatestProcessDefinitions

Description:

Method	*Return values*	*Description*
findLatestProcessDefinitions(String token)	***List<ProcessDefinition>***	*Retrieves a list of the last workflows definitions.*



Several versions of the same workflow can be registered.

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.ProcessDefinition;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            // Get all workflow definitions
            for (ProcessDefinition pd : okmWorkflow.findLatestProcessDefinitions(null)) {
                System.out.println("WF definition: " + pd);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findLastProcessDefinition

Description:

Method	*Return values*	*Description*
findLastProcessDefinition(String token, String name)	***ProcessDefinition***	*Retrieves last workflow definition of some specific workflow.*

The parameter name identifies an specific workflow definitions group.

Several workflow definition versions that have the same name.

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.ProcessDefinition;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            ProcessDefinition pd = okmWorkflow.findLastProcessDefinition(null, "purchase");
            System.out.println("WF definition: " + pd);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProcessInstance

Description:

Method	*Return values*	*Description*
getProcessInstance(String token, long processInstanceId)	***ProcessInstance***	*Returns the process instance.*
The parameter processInstanceId is a valid process instance id.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.ProcessInstance;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pId = 8108; // Some valid process instance id
            ProcessInstance pi = okmWorkflow.getProcessInstance(null, pId);
            System.out.println("PI: " + pi);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

suspendProcessInstance

Description:

Method	*Return values*	*Description*
suspendProcessInstance(String token, long processInstanceId)	***void***	*Suspends the process instance.*
The parameter processInstanceId is a valid process instance id.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pId = 8108; // Some valid process instance id
            okmWorkflow.suspendProcessInstance(null, pId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

resumeProcessInstance

Description:

Method	*Return values*	*Description*
resumeProcessInstance(String token, long processInstanceId)	***void***	*Resume the process instance.*
The parameter processInstanceId is a valid process instance id.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long pild = 8108; // Some valid process instance id
            okmWorkflow.resumeProcessInstance(null, pild);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findUserTaskInstances

Description:

Method	*Return values*	*Description*
findUserTaskInstances(String token, int offset, int limit)	***List<TaskInstance>***	*Retrieves a list of task instances assigned to the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" is used to skip that many results before the beginning to return results.*

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.TaskInstance;
import com.openkm.bean.workflow.TaskInstanceResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            TaskInstanceResultSet resultSet = okmWorkflow.findUserTaskInstances(null, 0, 10);
            System.out.println("Total: " + resultSet.getTotal());
            // Get all user task intances
            for (TaskInstance ti : resultSet.getResults()) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

findPooledTaskInstances

Description:

Method	*Return values*	*Description*
findPooledTaskInstances(String token, int offset, int limit)	***List<TaskInstance>***	*Retrieves a list of pooled task instances assigned to the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" is used to skip that many results before the beginning to return results.*

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.TaskInstance;
import com.openkm.bean.workflow.TaskInstanceResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            TaskInstanceResultSet resultSet = okmWorkflow.findPooledTaskInstances(null, 0, 10);
            System.out.println("Total: " + resultSet.getTotal());
            // Get all user task intances
            for (TaskInstance ti : resultSet.getResults()) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findTaskInstances

Description:

Method	*Return values*	*Description*

findTaskInstances(String token, long processInstanceId)	***List<TaskInstance>***	*Retrieves a list of task instances by process intance.*
The parameter processInstanceId is a valid process instance id.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.TaskInstance;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            // Get all user task intances
            long pild = 8108; // Some valid process instance id
            for (TaskInstance ti : okmWorkflow.findTaskInstances(null, pild)) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTaskInstanceValues

Description:

Method	*Return values*	*Description*
setTaskInstanceValues(String token, long taskInstanceId, String transitionName, List<FormElement> values)	***void***	*Set a task instance values.*
The parameter taskInstanceId is a valid task instance id. *The parameter transitionName is the choosen transaction.* *The parameter values are form element values needed for starting the workflow (not all workflow tasks need form values).*		

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMWorkflow;
import com.openkm.bean.form.FormElement;
import com.openkm.util.ContextWrapper;

import java.util.ArrayList;
import java.util.List;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long tiId = 8110; // Some valid task instance id
            List<FormElement> feList = new ArrayList<>();
            okmWorkflow.setTaskInstanceValues(null, tiId, "approve", feList);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskInstance

Description:

Method	*Return values*	*Description*
getTaskInstance(String token, long taskInstanceId)	***TaskInstance***	*Returns a task instance.*
The parameter taskInstanceId is a valid task instance id.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.TaskInstance;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long tiId = 8110; // Some valid task instance id
            TaskInstance ti = okmWorkflow.getTaskInstance(null, tiId);
            System.out.println(ti);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTaskInstanceActorId

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
setTaskInstanceActorId(String token, long taskInstanceId, String actorId)	***void***	*Set the actor to a task instance.*
The parameter taskInstanceId is a valid task instance id. *The parameter actorId must be some valid OpenKM userId.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long tiId = 8110; // Some valid task instance id
            okmWorkflow.setTaskInstanceActorId(null, tiId, "okmAdmin");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

startTaskInstance

Description:

Method	*Return values*	*Description*
startTaskInstance(String token, long taskInstanceId)	***void***	*Starts a task instance.*
The parameter taskInstanceId is a valid task instance id.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long tiId = 8110; // Some valid task instance id
            okmWorkflow.startTaskInstance(null, tiId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

endTaskInstance

Description:

Method	*Return values*	*Description*
endTaskInstance(String token, long taskInstanceId, String transitionName)	***void***	*Ends a task instance.*
The parameter taskInstanceId is a valid task instance id. *The parameter transitionName is a transaction name.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long tiId = 8110; // Some valid task instance id
            okmWorkflow.endTaskInstance(null, tiId, "end");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getToken

Description:

Method	*Return values*	*Description*

getToken(String token, long tokenId)	***Token***	*Return a token object.*
The parameter tokenId is a valid token id.		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.Token;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);

            long tkId = 8110; // Some valid token id
            Token tk = okmWorkflow.getToken(null, tkId);
            System.out.println(tk);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendTokenSignal

Description:

Method	*Return values*	*Description*
sendTokenSignal(String token, long tokenId, String transitionName)	***Token***	*Send a signal.*
The parameter tokenId is a valid token id. *The parameter transitionName is a transaction name.*		

Example:

```
package com.openkm;

import com.openkm.api.OKMWorkflow;
import com.openkm.bean.workflow.Token;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMWorkflow okmWorkflow = ContextWrapper.getContext().getBean(OKMWorkflow.class);
```

```
            long tkId = 8110; // Some valid token id
            Token tk = okmWorkflow.sendTokenSignal(null, tkId, "end");
            System.out.println(tk);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Commons classes

OpenKM commons classes:

OpenKM common class	*Description*
ArchiveUtils	*Utility methods for archive handling. For example, to create a ZIP or JAR.*
ExecutionUtils	*Execution related methods. For example, to run a script or JAR.*
FileLogger	*Utility methods to generate a log. For example, to show error, info or warning.*
FileUtils	*File related utility methods. For example, to get the filename or extension.*
FormatUtil	*Utility methods used by formatters. For example, to format a file size or format time.*
GeneralUtils	*Generic utility methods. For example, to notify something to an user.*
ISO8601	*Utility class what provides bidirectional format transformation between java time objects and ISO8601 format used for metadata date fields.*
MailUtils	*Utility methods to send a mail. For example to send message or send document.*
PathUtils	*Utility methods to handle paths. For example, get the name or get the parent path.*
PDFUtils	*Utility methods to convert PDF files. For example, to stamp image or text.*
PrincipalUtils	*Utility methods for retrieving information from the security context. For example, get the actual userId from the security context session or get the actual tenant of the logged user.*
StackTraceUtils	*Utility method for the stack.*
TemplateUtils	*Utility methods to handle templates. For example, replace a text in a template.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

WorkflowUtils	*Utility methods to handle workflows.*
ShardUtils	*Utility methods to have a repository distributed on several disks*
LegacySrv	*Utility methods to perform SQL query and HQL queries.*
DbSessionManager	*Utility methods for managing the sessions of the users.*
ConfigSrv	*Utility methods to perform config parameter operations.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ArchiveUtils

Utility methods for archive handling. For example, to create a ZIP or JAR.

Methods

createZip

Description:

Method	*Return values*	*Description*
createZip(File path, OutputStream os)	***void***	*Create ZIP archive from file.*
***path**: The file to compress.* ***os**: The output stream of the file.*		

Example:

```
package com.openkm;

import java.io.ByteArrayOutputStream;
import java.io.File;

import com.openkm.util.ArchiveUtils;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ArchiveUtils archiveUtils = ContextWrapper.getContext().getBean(ArchiveUtils.class);
            File file = new File("/home/openkm/test.pdf");
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            archiveUtils.createZip(file, baos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createJar

Description:

Method	*Return values*	*Description*
createJar(File path, String root, OutputStream os)	***void***	*Recursively create JAR archive from directory.*

***path**: The file to compress.*

***root**: A text with destiny of the file.*

***os**: The output stream of the file.*

Example:

```
package com.openkm;

import java.io.File;
import java.io.FileOutputStream;
import java.io.StringWriter;

import com.openkm.util.ArchiveUtils;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.PathUtils;
import com.openkm.util.impexp.RepositoryExporter;
import com.openkm.util.impexp.TextInfoDecorator;

public class Test {

    public static void main(String[] args) {
        try {
            ArchiveUtils archiveUtils = ContextWrapper.getContext().getBean(ArchiveUtils.class);
            PathUtils pathUtils = ContextWrapper.getContext().getBean(PathUtils.class);
            FileUtils fileUtils = ContextWrapper.getContext().getBean(FileUtils.class);
            RepositoryExporter repositoryExporter = ContextWrapper.getContext().getBean(RepositoryExporter.class);

            StringWriter out = new StringWriter();

            File dirFile = new File("/home/openkm");
            final File tmpFile = File.createTempFile("test", ".jar", dirFile);
            FileOutputStream os = new FileOutputStream(tmpFile);

            File tmp = fileUtils.createTempDir();
            repositoryExporter.exportFolder(null, "/okm:root/jar", tmp, false, out, new TextInfoDecorator("/okm:root/jar"));

            // Jar files
            archiveUtils.createJar(tmp, pathUtils.getName("/home/openkm"), os);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

ExecutionUtils

Execution related methods. For example, to run a script or JAR..

Methods

runScript

Description:

Method	*Return values*	*Description*
runScript(File script)	***Object[]***	*Execute script from file.*
***script**: The name of the file to run.*		

Example:

```
package com.openkm;

import java.io.File;

import com.openkm.core.Config;
import com.openkm.util.ExecutionUtils;

public class Test {

    public static void main(String[] args) {
        try {
            File script = new File("/home/openkm/test.bsh");
            ExecutionUtils.runScript(script);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

runJar

Description:

Method	*Return values*	*Description*
runJar(File jar)	***Object***	*Execute jar from file.*
***jar**: The name of the file to run.*		

Example:

```
package com.openkm;

import java.io.File;

import com.openkm.util.ExecutionUtils;

public class Test {

    public static void main(String[] args) {
        try {
            File jar = new File("/home/openkm/test.jar");
            ExecutionUtils.getInstance().runJar(jar);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

runCmd

Description:

Method	*Return values*	*Description*
runCmd(String cmd)	***CmdExecutionResult***	*Execute command line.*
***cmd**: The command line to execute.*		

Example:

```
package com.openkm;

import com.openkm.util.ExecutionUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ExecutionUtils.runCmd("mkdir /home/openkm/test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

FileLogger

Utility methods to generate a log. For example, to show error, info or warning.

Methods

info

Description:

Method	*Return values*	*Description*
info(String baseName, String message, Object... params)	***void***	*Static file logger with info.*
***baseName**: The name of file to write.* ***message**: A text with message for info.*		

Example:

```
package com.openkm;

import com.openkm.util.FileLogger;

public class Test {

    public static void main(String[] args) {
        try {
            FileLogger.info("logName", "This is info");
            FileLogger.info("logName", "Parameters: %s %s", "one", "two");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

warn

Description:

Method	*Return values*	*Description*
warn(String baseName, String message, Object... params)	***void***	*Static file logger with warn.*
***baseName**: The name of file to write.* ***message**: A text with message for warning.*		

Example:

```
package com.openkm;

import com.openkm.util.FileLogger;

public class Test {

    public static void main(String[] args) {
        try {
            FileLogger.warn("logName", "This is warning");
            FileLogger.warn("logName", "Parameters: %s %s", "one", "two");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

error

Description:

Method	*Return values*	*Description*
error(String baseName, String message, Object... params)	***void***	*Static file logger with error.*
***baseName**: The name of file to write.* ***message**: A text with message for error.*		

Example:

```
package com.openkm;

import com.openkm.util.FileLogger;

public class Test {

    public static void main(String[] args) {
        try {
            FileLogger.error("logName", "This is error");
            FileLogger.error("logName", "Parameters: %s %s", "one", "two");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

FileUtils

File related utility methods. For example, to get the filename or extension.

Methods

getFileName

Description:

Method	*Return values*	*Description*
getFileName(String file)	***String***	*Returns the name of the file without the extension.*
***file**: The full name of the file.*		

Example:

```
package com.openkm;

import java.io.File;

import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.PathUtils;

public class Test {

    public static void main(String[] args) {
        try {
            File file = new File("/home/openkm/test.png");
            String docName = PathUtils.getName(file.getPath());
            String baseName = FileUtils.getFileName(docName);
            System.out.println(baseName);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFileExtension

Description:

Method	*Return values*	*Description*
getFileExtension(String file)	***String***	*Returns the filename extension.*
***file**: The full name of the file.*		

Example:

```
package com.openkm;
import java.io.File;

import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.PathUtils;

public class Test {

    public static void main(String[] args) {
        try {
            File file = new File("/home/openkm/test.png");
            String docName = PathUtils.getName(file.getPath());
            String extension = FileUtils.getFileExtension(docName);
            System.out.println(extension);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

FormatUtil

Utility methods used by formatters. For example, to format a file size or format time.

Methods

formatSize

Description:

Method	*Return values*	*Description*
formatSize(long bytes)	***String***	*Format the document size for human readers.*
***bytes**: The size of the file content.*		

Example:

```
package com.openkm;

import java.io.File;
import com.openkm.util.FormatUtil;

public class Test {

    public static void main(String[] args) {
        try {
            File file = new File("/home/openkm/test.png");
            System.out.println(FormatUtil.formatSize(file.length()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

parseSize

Description:

Method	*Return values*	*Description*
parseSize(String text)	***long***	*Parse human-readable sizes.*
***text**: The size to parse.*		

Example:

```
package com.openkm;
```

```
import com.openkm.util.FormatUtil;

public class Test {
    public static void main(String[] args) {
        try {
         System.out.println(FormatUtil.parseSize("1024.7"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

formatDate

Description:

Method	*Return values*	*Description*
formatDate(Calendar cal)	***String***	*Format calendar date.*
***cal**: The date to format.*		

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.util.FormatUtil;

public class Test {
    public static void main(String[] args) {
        try {
         System.out.println(FormatUtil.formatDate(Calendar.getInstance()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ISO8601

Utility class what provides bidirectional format transformation between java time objects and ISO8601 format used for metadata date fields.

Methods

parseBasic

Description:

Method	*Return values*	*Description*
parseBasic(String value)	*String*	*Return date from an ISO8601 basic format string.*
***value**: A ISO86101 basic format string value.*		

Example:

```
package com.openkm;

import java.text.SimpleDateFormat;
import java.util.Calendar;

import com.openkm.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        try {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
            Calendar cal = ISO8601.parseBasic("20180222000000");
            System.out.println(sdf.format(cal.getTime()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

formatBasic

Description:

Method	*Return values*	*Description*
formatBasic(Calendar calendar)	*Calendar*	*Return ISO8601 basic format from date.*
***calendar**: The date to format.*		

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        try {
            Calendar cal = Calendar.getInstance();
            System.out.println(ISO8601.formatBasic(cal.getTime()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

GeneralUtils

Generic utility methods. For example, to notify something to an user.

Methods

uiNotify

Description:

Method	*Return values*	*Description*
uiNotify(String msg)	***void***	*Notify to current authenticated user.*
***msg**: A text with the message of notify.*		



The user must be logged before executing the methods.

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.GeneralUtils;

public class Test {

    public static void main(String[] args) {
        try {
            GeneralUtils generalUtils = ContextWrapper.getContext().getBean(GeneralUtils.class);
            generalUtils.uiNotify("Message test!");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

MailUtils

Utility methods to send a mail. For example to send message or send document.

Methods

sendMessage

Description:

Method	*Return values*	*Description*
sendMessage(String toAddress, String subject, String content)	***void***	*Send mail without FROM addresses.*
***toAddress**: The destination addresses for mail.* ***subject**: The mail subject.* ***content**: The mail body.*		

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.MailUtils;

public class Test {

    public static void main(String[] args) {
        try {
            MailUtils mailUtils = ContextWrapper.getContext().getBean(MailUtils.class);
            mailUtils.sendMessage("noreply@openkm.com", "this is the subject", "This is the text");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMessage

Description:

Method	*Return values*	*Description*
sendMessage(String fromAddress, String toAddress, String subject, String content)	***void***	*Send mail with FROM addresses.*

fromAddress: *The from addresses for mail.*

toAddress*: The destination addresses for mail.*

subject*: The mail subject.*

content*: The mail body.*

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.MailUtils;

public class Test {

    public static void main(String[] args) {
        try {
            MailUtils mailUtils = ContextWrapper.getContext().getBean(MailUtils.class);
            mailUtils.sendMessage("from_noreply@openkm.com", "to_noreply@openkm.com", "this is the subject", "Th
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendDocuments

Description:

Method	*Return values*	*Description*
sendDocuments(String fromAddress, List<String> toAddresses, String subject, String text, List<String> docsPath)	***MimeMessage***	*Send document to non-registered OpenKM users.*

fromAddress: *The from addresses for mail.*

toAddress*: The list of destination addresses for mail.*

subject*: The mail subject.*

text*: The mail body.*

docsPath*: The path of the documents to be send.*

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.util.ContextWrapper;
import com.openkm.util.MailUtils;

public class Test {

    public static void main(String[] args) {
        try {
            MailUtils mailUtils = ContextWrapper.getContext().getBean(MailUtils.class);
            List<String> docsPath = new ArrayList<>();
            List<String> listAddress = new ArrayList<>();
            listAddress.add("to_noreply@openkm.com");
            docsPath.add("/home/openkm/test.png");
            docsPath.add("/home/openkm/test2.png");
            docsPath.add("/home/openkm/test3.png");
            mailUtils.sendDocuments("from_noreply@openkm.com", listAddress, "this is the subject", "This is the text",
                    docsPath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

PathUtils

Utility methods to handle paths. For example, get the name or get the parent path.

Methods

getParent

Description:

Method	Return values	Description
getParent(String path)	***String***	*Get parent node.*
***path**: The path of the file.*		

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.PathUtils;

public class Test {

    public static void main(String[] args) {
        try {
            System.out.println(PathUtils.getParent("/okm:root/logo.png"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getName

Description:

Method	Return values	Description
getName(String path)	***String***	*Get name node.*
***path**: The path of the file.*		

Example:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.PathUtils;

public class Test {

    public static void main(String[] args) {
        try {
            System.out.println(PathUtils.getName("/okm:root/logo.png"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContext

Description:

Method	*Return values*	*Description*
getContext(String path)	***String***	*Get path context. For example "/okm:root/test.txt" -> "/okm:root".*
***path**: The path of the file.*		

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.PathUtils;

public class Test {

    public static void main(String[] args) {
        try {
            System.out.println(PathUtils.getContext("/okm:root/logo.png"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

PDFUtils

Utility methods to convert PDF files. For example, to stamp image or text.

Methods

merge

Description:

Method	*Return values*	*Description*
merge(List<InputStream> inputs, OutputStream output)	***void***	*Merge several PDFs into a new one.*
***inputs**: List of inputs stream for merge.* ***output**: Output stream of the file.*		

Example:

```
package com.openkm;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.PDFUtils;
import com.openkm.util.PathUtils;

public class Test {

    public static void main(String[] args) {
        FileOutputStream fos = null;
        FileInputStream fis = null;
        File tmp = null;

        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            PDFUtils pdfUtils = ContextWrapper.getContext().getBean(PDFUtils.class);

            tmp = FileUtils.createTempFile("pdf");
            fos = new FileOutputStream(tmp);

            List<InputStream> inputs = new ArrayList<>();
            inputs.add(okmDocument.getContent(null, "c0984ecf-8d9e-40f8-b5fb-2c83d202bd9d", false));
            inputs.add(okmDocument.getContent(null, "b3d0af95-076c-4cc4-9b85-bd3e7eb2baa3", false));
```

```
            // Merge document
            pdfUtils.merge(inputs, fos);

            // Create document in repository
            String fldPath = PathUtils.getParent("/okm:root/test3.pdf"); // all documents are in same path
            String docPath = fldPath + "/docMerge.pdf";
            fis = new FileInputStream(tmp);
            okmDocument.createSimple(null, docPath, fis);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(fis);
            IOUtils.closeQuietly(fos);
            fileUtils.deleteQuietly(tmp);
        }
    }
}
```

stampImage

Description:

Method	*Return values*	*Description*
stampImage(InputStream input, byte[] image, OutputStream output)	***void***	*Stamp PDF document with image watermark.*

***input**: The input stream of the file.*

***image**: The content of the file.*

***output**: Output stream of the file.*

Example:

```
package com.openkm;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.file.Files;
import java.nio.file.Paths;

import com.openkm.util.ContextWrapper;
import com.openkm.util.PDFUtils;

public class Test {

    public static void main(String[] args) {
        try {
            PDFUtils pdfUtils = ContextWrapper.getContext().getBean(PDFUtils.class);
```

```
            String path = "/home/openkm/test.pdf";
            File file = new File(path);
            InputStream is = new FileInputStream(file);
            OutputStream out = new FileOutputStream(file);
            byte[] data = Files.readAllBytes(Paths.get("/home/luis/test.png"));
            pdfUtils.stampImage(is, data, out);
            is.close();
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampText

Description:

Method	*Return values*	*Description*
stampText(InputStream input, String text, OutputStream output)	***void***	*Stamp PDF document with text watermark.*

***input**: The input stream of the file.*

***text:** A text for the stamp.*

***output**: Output stream of the file.*

Example:

```
package com.openkm;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import com.openkm.util.ContextWrapper;
import com.openkm.util.PDFUtils;

public class Test {

    public static void main(String[] args) {
        try {
            PDFUtils pdfUtils = ContextWrapper.getContext().getBean(PDFUtils.class);

            String path = "/home/openkm/test.pdf";
            File file = new File(path);
            InputStream is = new FileInputStream(file);
            OutputStream out = new FileOutputStream(file);
            pdfUtils.stampText(is, "This is the text", out);
```

```
            is.close();
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNumberOfPages

Description:

Method	*Return values*	*Description*
getNumberOfPages(String token, String docId)	***int***	*Return the number of pages for a PDF.*
***input**: The input stream of the file.* ***text:** A text for the stamp.* ***docId**: PDF OpenKM unique identifier.*		

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.PDFUtils;

public class Test {

    public static void main(String[] args) {
        try {
            PDFUtils pdfUtils = ContextWrapper.getContext().getBean(PDFUtils.class);
            String docId = "e6b61e8d-4bf1-4c37-828f-09ecc34261cc";
            System.out.println(pdfUtils.getNumberOfPages(null, docId));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

PrincipalUtils

Utility methods for retrieving information from the security context. For example, get the actual userId from the security context session or get the actual tenant of the logged user.

Methods

getUser

Description:

Method	Return values	Description
getUser()	**String**	*Get the userId of the the logged user.*



The user must be logged before executing the methods.

Example:

```
package com.openkm;

import com.openkm.principal.PrincipalUtils;

public class Test {

    public static void main(String[] args) {
        try {
            System.out.println(PrincipalUtils.getUser());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTenantId

Description:

Method	Return values	Description
getTenantId()	**long**	*Get the tenantId of the logged user.*



The user must be logged before executing the methods.

Example:

```
package com.openkm;

import com.openkm.principal.PrincipalUtils;

public class Test {

    public static void main(String[] args) {
        try {
            long tenantId = PrincipalUtils.getTenantId();
            System.out.println(tenantId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

StackTraceUtils

Utility methods for the stack.

Methods

isCallingMe

Description:

Method	Return values	Description
isCallingMe(String className)	**Boolean**	*Return true when a className is in the current stack*

Example:

```
package com.openkm;

package com.openkm.test;

import com.openkm.spring.PrincipalUtils;

public class Test {
 public static void main(String[] args) throws Exception {
  System.out.println(StackTraceUtilsUtils.isCallingMe("com.openkm.automation.action.UpdateByMetadata"));
 }
}
```

toString

Description:

Method	Return values	Description
toString(Throwable t)	**String**	*Return exception stack as standard text.*

Example:

```
package com.openkm;

package com.openkm.test;

import com.openkm.spring.PrincipalUtils;

public class Test {
 public static void main(String[] args) throws Exception {
        try {
            // Force exception
            List keys = null;
   for (String key : keys) {
```

```
    }
        } catch (Exception e) {
    System.out.println(StackTraceUtilsUtils.toString(e));
   }
  }
 }
```

TemplateUtils

Utility methods to handle templates. For example, replace a text in a template.

Methods

templateExists

Description:

Method	*Return values*	*Description*
templateExists(String name)	***boolean***	*Check for template existence.*
name: *The name of the template.*		

Example:

```
package com.openkm;

import com.openkm.util.TemplateUtils;

public class Test {

    public static void main(String[] args) {
        try {
            System.out.println("Exist template: " + TemplateUtils.templateExists("OpenKM - NOTIFICATION"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

replace

Description:

Method	*Return values*	*Description*
replace(String name, String template, Map<String, Object> model)	***String***	*Quick replace utility function.*
name: *The name of the template.* ***template****: A text whith the template definition.* ***model****: A map use to reaplace elements in the template.*		

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Example:

```
package com.openkm;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

import com.openkm.core.Config;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.TemplateUtils;

public class Test {

    public static void main(String[] args) {
        try {
            FileUtils fileUtils = ContextWrapper.getContext().getBean(FileUtils.class);

            Map<String, Object> hm = new HashMap<>();
            File tmpFileIn = fileUtils.createTempFileFromMime("text/plain");
            hm.put("fileIn", tmpFileIn.getPath());
            String cmd = TemplateUtils.replace("SYSTEM_XLS2CSV", Config.SYSTEM_CATDOC_XLS2CSV, hm);
            System.out.println("cmd template: " + cmd);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

WorkflowUtils

Utility methods to handle workflows.

Methods

findProcessInstancesByNode

Description:

Method	*Return values*	*Description*
findProcessInstancesByNode(String uuid)	***List<ProcessInstance>***	*Retrieves a list of all process instances of a node*
uuid *string type is the uuid of the document, folder, mail or record.*		

Example:

```
package com.openkm;

import com.openkm.bean.workflow.ProcessInstance;
import com.openkm.util.WorkflowUtils;

public class Test {

    public static void main(String[] args) {
        try {
            String uuid = "some node uuid";
            for (ProcessInstance processInstance : WorkflowUtils.findProcessInstancesByNode(uuid)) {
                System.out.println(processInstance);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ShardUtils

Utility methods to have a repository distributed on several disks

Methods

setCurrentShard

Description:

Method	*Return values*	*Description*
setCurrentShard(Long shardId)	***void***	*Set the current shard.*

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.ShardUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            Long shardId = new Long(1);
            shardUtils.setCurrentShard(shardId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createShardDirs

Description:

Method	*Return values*	*Description*
createShardDirs(Long shardId)	***void***	*Create new shard folder*

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.ShardUtils;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            Long shardId = new Long(1);
            shardUtils.createShardDirs(shardId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

changeNodeShard

Description:

Method	*Return values*	*Description*
changeNodeShard(String uuid, Long newShard)	***void***	*Change shard of a node.*

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.ShardUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            Long newShardId = new Long(2);
            shardUtils.changeNodeShard("e2391b01-ce10-42c7-94a9-b0d6b394048e", newShardId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getShardIds

Description:

Method	*Return values*	*Description*
getShardIds()	***List<Long>***	*Get the shard ids.*

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
```

```
import com.openkm.util.ShardUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            for (long id : shardUtils.getShardIds()) {
                System.out.println(id);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getShardSize

Description:

Method	*Return values*	*Description*
getShardSize(Long shardId)	***Long***	*Get the size of a shard.*

Example:

```
package com.openkm;

import com.openkm.util.ContextWrapper;
import com.openkm.util.ShardUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            Long shardId = new Long(1);
            System.out.println(shardUtils.getShardSize(shardId));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getShardPaths

Description:

Method	*Return values*	*Description*
getShardPaths(Long shardId)	***List<String>***	*Get the path of a shard.*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.util.ContextWrapper;
import com.openkm.util.ShardUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            Long shardId = new Long(1);
            List<String> list = shardUtils.getShardPaths(shardId);
            for (String path : list) {
                System.out.println(path);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getShards

Description:

Method	*Return values*	*Description*
getShards()	***List<Shard>***	*Get the shards*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.db.bean.Shard;
import com.openkm.util.ContextWrapper;
import com.openkm.util.ShardUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            List<Shard> list = shardUtils.getShards();
            for (Shard shard : list) {
                System.out.println(shard);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getShardFolders

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Description:

Method	*Return values*	*Description*
getShardFolders()	***List<File>***	*Get shard folders.*

Example:

```
package com.openkm;

import java.io.File;
import java.util.List;

import com.openkm.util.ContextWrapper;
import com.openkm.util.ShardUtils;

public class Test {

    public static void main(String[] args) {
        try {
            ShardUtils shardUtils = ContextWrapper.getContext().getBean(ShardUtils.class);
            List<File> list = shardUtils.getShardFolders();
            for (File file : list) {
                System.out.println(file.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

LegacySrv

executeSQL

Description:

Method	*Return values*	*Description*
executeSQL(String sql)	***List<List<String>>***	*Executes SQL script.*

```
SELECT NBS_UUID, NBS_AUTHOR, NBS_NAME, NBS_NOTES FROM OKM_NODE_BASE LIMIT 10;
```

> *The parameter named sql only allows a single SQL script.*
>
> *This action can only be done by users with administrator grants.*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.db.service.LegacySrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            LegacySrv legacySrv = ContextWrapper.getContext().getBean(LegacySrv.class);
            List<List<String>> results = legacySrv.executeSQL("SELECT NBS_UUID, NBS_AUTHOR, NBS_NAME, NE
            for (List<String> row : results) {
                System.out.println("uuid: " + row.get(0));
                System.out.println("Author: " + row.get(1));
                System.out.println("Name: " + row.get(2));
                System.out.println("Notes: " + row.get(3));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeHQL

Description:

Method	*Return values*	*Description*
executeHQL(String hql)	***List<Object>***	*Executes HQL script.*

```
SELECT uuid, name from NodeBase where name = 'okm:root';
```



The parameter named hql only allows a single HQL script.

This action can only be done by users with administrator grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.db.service.LegacySrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            LegacySrv legacySrv = ContextWrapper.getContext().getBean(LegacySrv.class);
            List<Object> results = legacySrv.executeHQL("SELECT * from NodeBase nb where nb.name = 'okm:root';");
            for (Object obj : results) {
                if (obj instanceof Object[]) {
                    Object[] ao = (Object[]) obj;

                    for (Object o : ao) {
                        System.out.println(String.valueOf(o));
                    }
                } else {
                    System.out.println(String.valueOf(obj));
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

DbSessionManager

Methods

getSystemToken

Description:

Method	Return values	Description
getSystemToken()	**String**	Get the authorization token of the user system.

> *There's a special super-user named system executed by OpenKM in the background.*
>
> *The authorization token of the user system is usually used to promote standard user's actions or cron jobs as a super-user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            String systemToken = DbSessionManager.getInstance().getSystemToken();
            okmDocument.delete(systemToken, "d50133e3-dbfa-4d01-a109-28785cd48f40"); // Delete action is execute
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

ConfigSrv

getString

Description:

Method	*Return values*	*Description*
getString(String key, String defaultValue)	***String***	*Find the config parameter (as string type) by pk with a default value*

Example:

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getString("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getText

Description:

Method	*Return values*	*Description*
getText(String key, String defaultValue)	***String***	*Find the config parameter (as text type) by pk with a default value*

Example:

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getText("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getBoolean

Description:

Method	*Return values*	*Description*
getBoolean(String key, String defaultValue)	***boolean***	*Find the config parameter (as boolean type) by pk with a default value*

Example:

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getBoolean("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getInteger

Description:

Method	*Return values*	*Description*
getInteger(String key, String defaultValue)	***Integer***	*Find the config parameter (as integer type) by pk with a default value*

Example:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getInteger("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLong

Description:

Method	*Return values*	*Description*
getLong(String key, String defaultValue)	***long***	*Find the config parameter (as long type) by pk with a default value*

Example:

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getLong("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getHtml

Description:

Method	*Return values*	*Description*
getHtml(String key, String	***String***	*Find the config parameter (as html type) by pk with a default*

defaultValue)		*value*

Example:

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getHtml("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPassword

Description:

Method	*Return values*	*Description*
getPassword(String key, String defaultValue)	***String***	*Find the config parameter (as password type) by pk with a default value*

Example:

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getPassword("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getList

Description:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Method	*Return values*	*Description*
getList(String key, String defaultValue)	***List<String>***	*Find the config parameter (as list type) by pk with a default value*

Example:

```
package com.openkm;

import com.openkm.db.service.ConfigSrv;
import com.openkm.util.ContextWrapper;

public class Test {

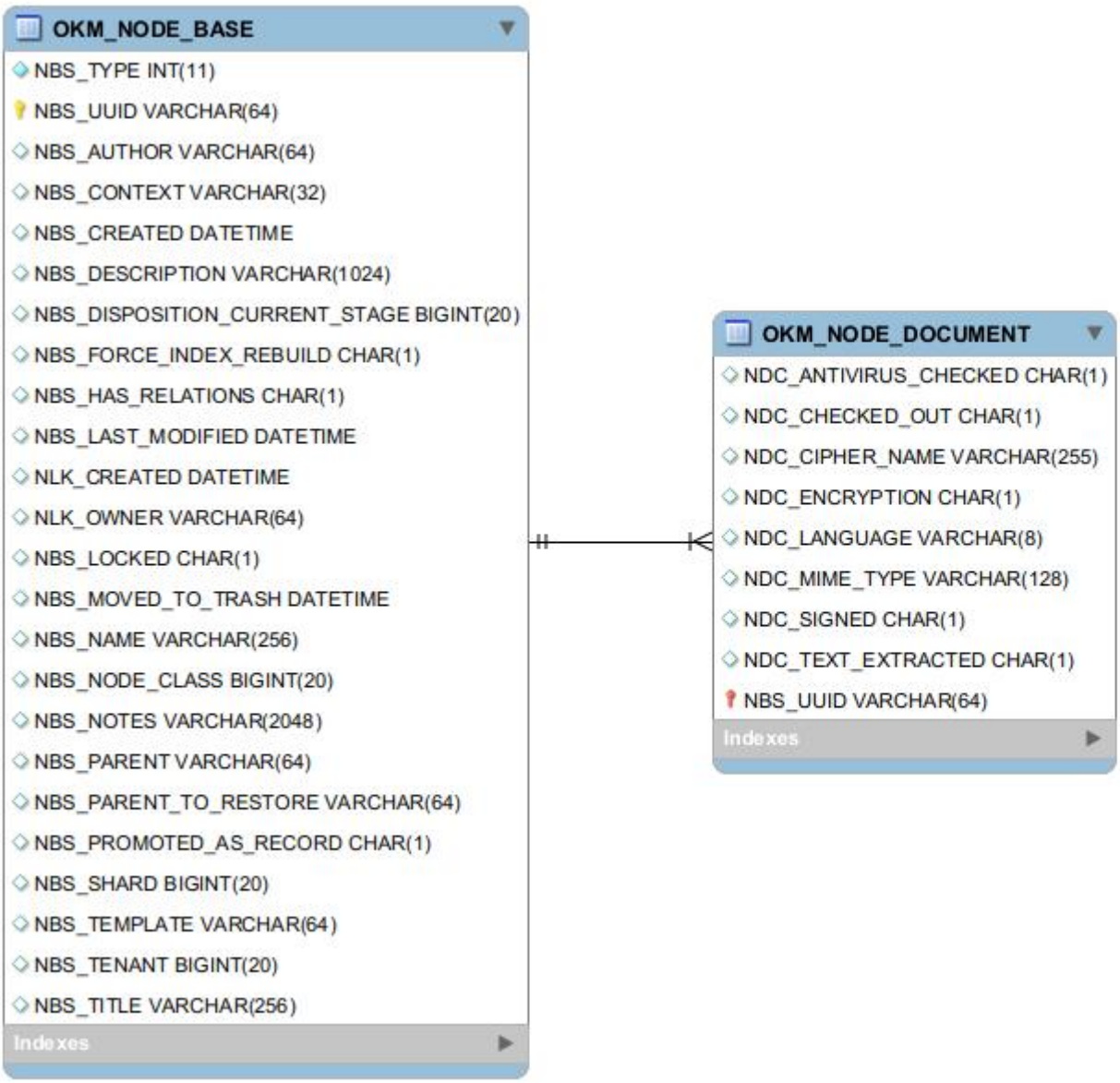
    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            for (String listElement : configSrv.getList("paramKey", "defaultValue")) {
                System.out.println(listElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Database description

Node types

These tables store the information about a node in OpenKM.



*The table named **OKM_NODE_BASE** stores the common fields (name, author, creation date, among others) for all the node types (Document, Folder, Mail or Record).*

*The field named **NBS_UUID** is the **unique node ID**. **NBS_UUID FIELD** is the **foreign key between parent table** (OKM_NODE_BASE) **and childs** (OKM_NODE_DOCUMENT, OKM_NODE_FOLDER, OKM_NODE_MAIL, OKM_NODE_RECORD).*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

There are other useful tables:

- ***OKM_NODE_DOCUMENT**: Store information about documents.*
- ***OKM_NODE_FOLDER**: Store information about folders.*
- ***OKM_NODE_MAIL**: Store information about mails.*
- ***OKM_NODE_RECORD**: Store information about records.*

SQL example queries

Query used to retrieve the information about every document:

```
select * from OKM_NODE_BASE nb
  inner join OKM_NODE_DOCUMENT nd on nd.nbs_uuid = nb.nbs_uuid
```

Query to get records what the name starts with 'test':

```
select * from OKM_NODE_BASE nb
  inner join OKM_NODE_RECORD nr on nr.nbs_uuid = nb.nbs_uuid
  where nb.nbs_name like 'test%'
```

To find a Document by UUID could use this query:

```
select * from OKM_NODE_BASE nb
  inner join OKM_NODE_DOCUMENT nd on nd.nbs_uuid = nb.nbs_uuid
  where nb.nbs_uuid = 'fc35ce80-57f2-4026-ae77-7a11bca042ed'
```

List of child nodes with standard security:

```
SELECT NB.NBS_UUID FROM OKM_NODE_BASE NB left outer join OKM_NODE_USER_PERMISSION UP ON I
WHERE NB.NBS_PARENT='586939f7-988a-430a-b3d1-4f8e75bd92e4' AND ((UP.NUP_USER='okmAdmin' and (U
and (RP.NRP_PERMISSION & 1)=1)) group by NB.NBS_UUID
```

Mark documents to be indexed again

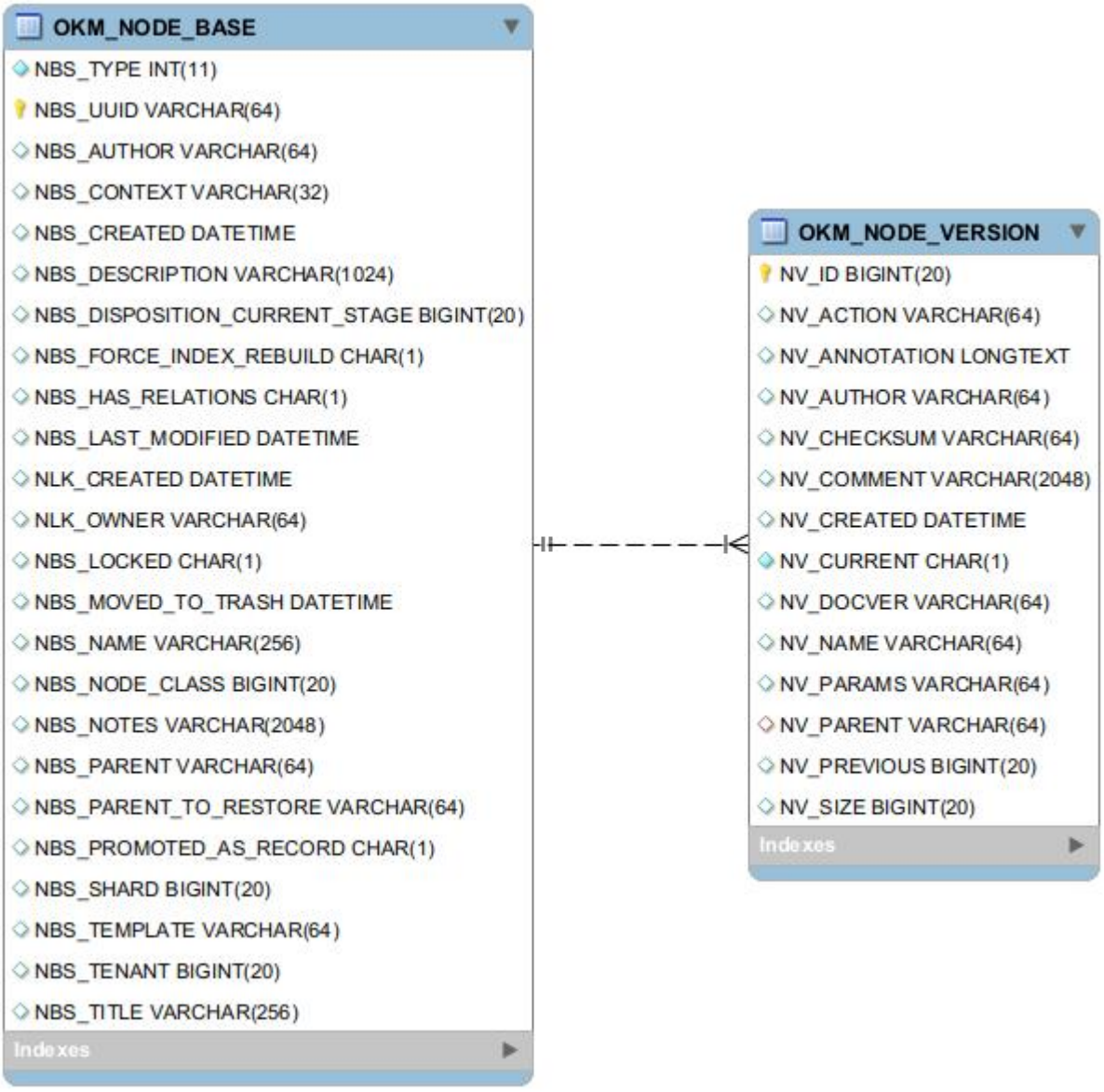
The column named NDC_TEXT_EXTRACTED in the table OKM_NODE_DOCUMENT takes control of what documents have been processed into the Text extraction queue and what are still pending to be processed (value 'T' means processed and 'F' means pending). If you want to reindex the whole repository again or some files you might execute a query like this one:

```
update OKM_NODE_DOCUMENT set NDC_TEXT_EXTRACTED='F'
```

Node version

These tables store information about node version history:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



*The table named **OKM_NODE_VERSION** stores version information about every version of each node.*

- *The field **NV_ID** is the id of a version.*
- *The value of the field named **NV_PARENT** is the **unique document ID** (the same value as NBS_UUID).*

SQL example queries

Query to get all changes of a document named 'test.pdf':

```
select nv.* from OKM_NODE_DOCUMENT nd
 inner join OKM_NODE_VERSION nv on nv.nv_parent = nd.nbs_uuid
 inner join OKM_NODE_BASE nb on nd.nbs_uuid = nb.nbs_uuid
 where nb.nbs_name = 'test.pdf';
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

This query prints the main info like this:

Column	Type	Description
NV_ACTION	DOCUMENT_CREATED FOLDER_CREATED MAIL_CREATED RECORD_CREATED DOCUMENT_CHANGED DOCUMENT_IMPORTED FOLDER_IMPORTED RECORD_IMPORTED MAIL_IMPORTED PROPERTY_GROUP_ADDED PROPERTY_GROUP_REMOVED PROPERTY_GROUP_CHANGED	What kind of changes were made.
NV_NAME	**String**	The name of the version (for example "1.1")
NV_DOCVER	**String**	Document content version id.

Query to get changes of a document named 'test.pdf' when its binary was updated:

```
select nv.* from OKM_NODE_DOCUMENT nd
 inner join OKM_NODE_VERSION nv on nv.nv_parent = nd.nbs_uuid
 inner join OKM_NODE_BASE nb on nd.nbs_uuid = nb.nbs_uuid
 where nb.nbs_name = 'test.pdf' and nv.nv_action = 'DOCUMENT_CHANGED';
```

The next query is useful to know where all the versions of a file named 'test.pdf' are stored in the datastore:



Only applicable to the default datastore configuration (case all the versions of the documents are stored in the file system).

Take a look at Datastore for more information.

```
select nv.nv_docver from OKM_NODE_DOCUMENT nd
 inner join OKM_NODE_VERSION nv on nv.nv_parent = nd.nbs_uuid
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

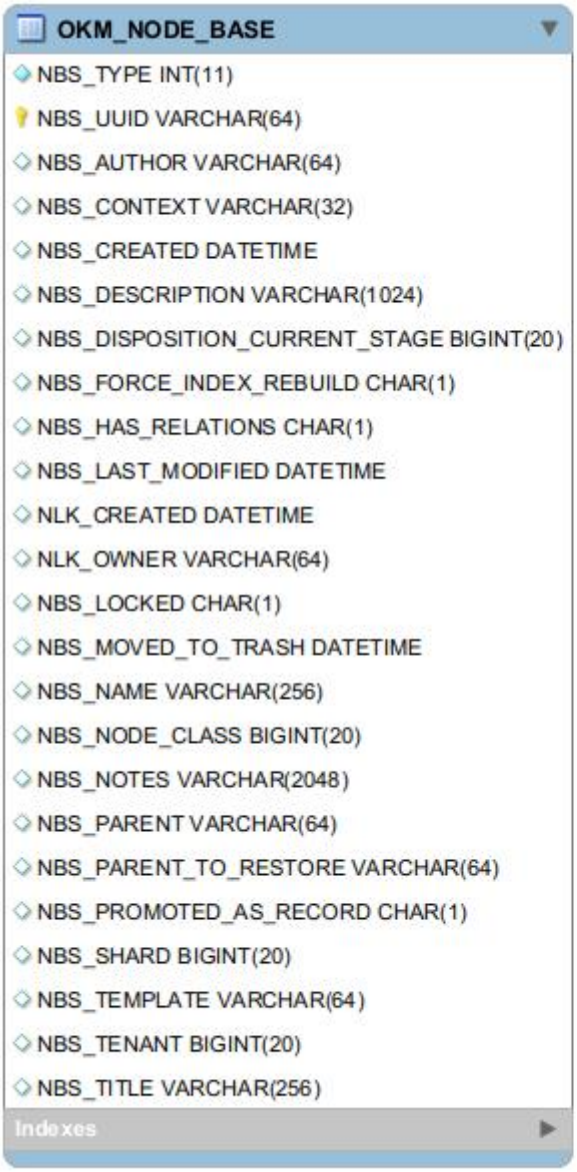
```
inner join OKM_NODE_BASE nb on nd.nbs_uuid = nb.nbs_uuid
where nb.nbs_name = 'test.pdf'
```

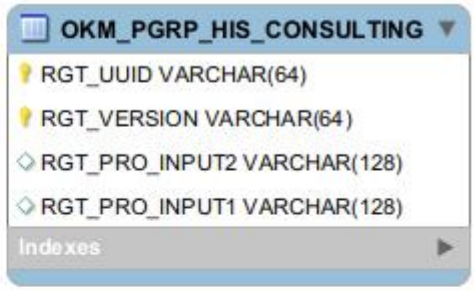
*For example '**f0d7f0a1**-b337-4917-b592-56e163b81cde'. This means your file is located in the physical drive in the path (based in first eight characters of this id):*

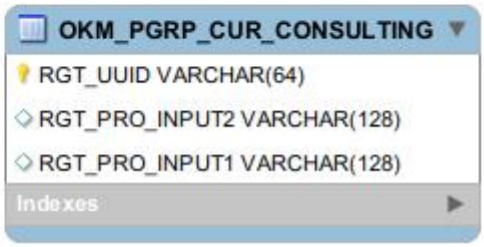
*/home/user/tomcat_home/repository/datastore/**f0/d7/f0/a1**/f0d7f0a1-b337-4917-b592-56e163b81cde*

Node metadata properties

Each metadata group uses two tables to store information, take the next as a sample:







Sample:



Take a look at Metadata XML definition for more information.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

After registering the previous metadata definition will be created two tables with these columns:

- ***OKM_PGRP_CUR_CONSULTING** (used for current metadata values)*
  - ***RGT_UUID**: ID of the node which has this metadata*
  - ***RGT_PRO_INPUT1**: Value of property okp:consulting.input1*
  - ***RGT_PRO_INPUT2**: Value of property okp:consulting.input2*
  - ***RGT_PRO_SEPARATOR**: Value of property okp:consulting.separator*
- ***OKM_PGRP_HIS_CONSULTING** (used for the history of the metadata values)*
  - ***RGT_UUID**: ID of the node which has this metadata*
  - ***RGT_VERSION**: ID of the related version*
  - ***RGT_PRO_INPUT1**: Value of property okp:consulting.input1*
  - ***RGT_PRO_INPUT2**: Value of property okp:consulting.input2*
  - ***RGT_PRO_SEPARATOR**: Value of property okp:consulting.separator*

Query to get all the metadata values from the document named 'test.pdf' with uuid 'e0518d0e-8312-4df9-991c-263f16b061bf' and their changes:

```
SELECT * FROM OKM_PGRP_HIS_CONSULTING where RGT_UUID = 'e0518d0e-8312-4df9-991c-263f16b061bf
```

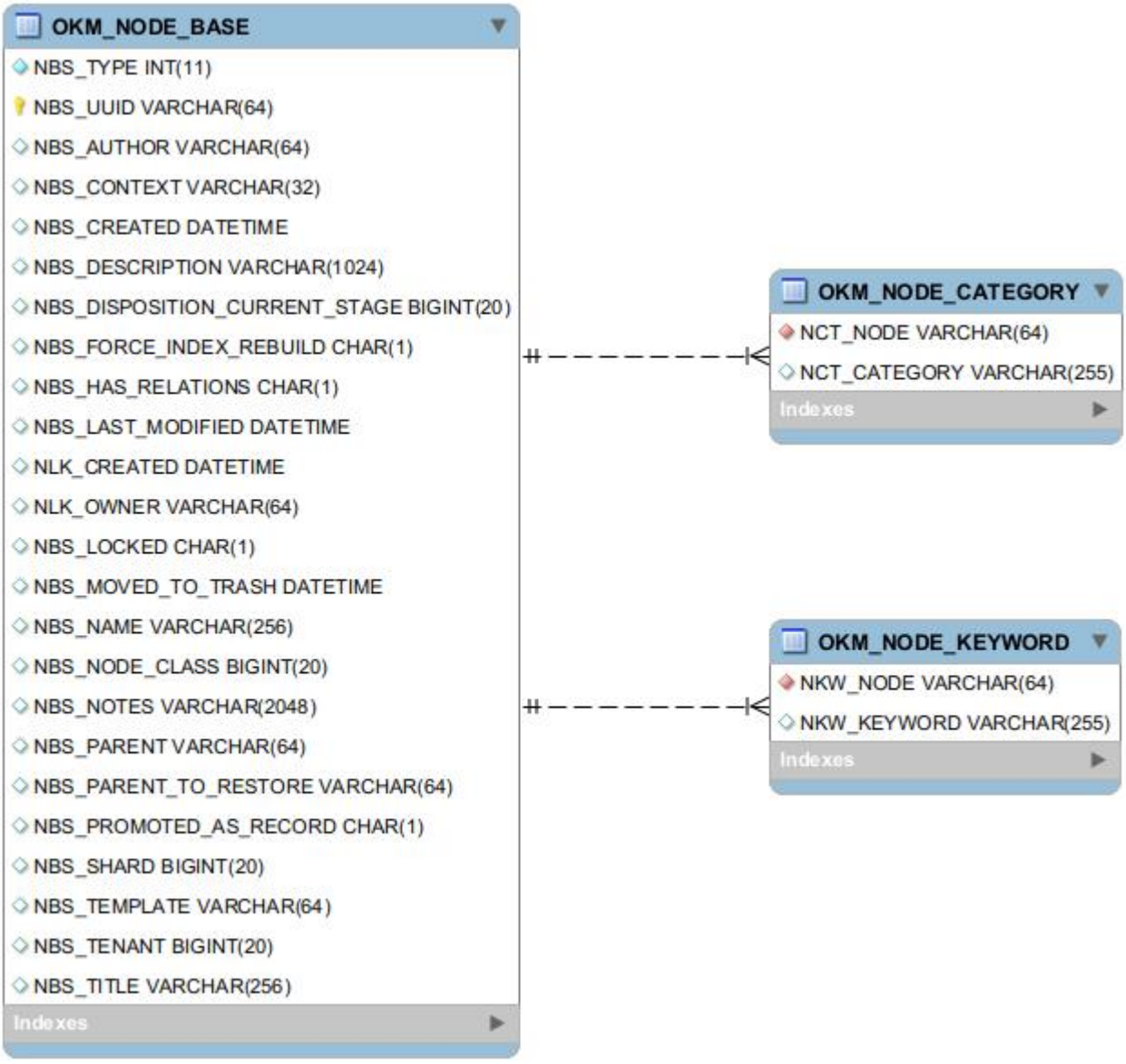


This query will print all metadata values for each version. Usually, the actual version of these metadata values are the ones having the biggest RGT_VERSION value.

Query to get changes of a document named 'test.pdf' when its metadata properties was updated:

```
select nv.* from OKM_NODE_DOCUMENT nd
 inner join OKM_NODE_VERSION nv on nv.nv_parent = nd.nbs_uuid
 inner join OKM_NODE_BASE nb on nd.nbs_uuid = nb.nbs_uuid
 where nb.nbs_name = 'test.pdf' and nv.nv_action = 'PROPERTY_GROUP_CHANGED';
```

Node categories and keywords



*The value of the field named **NCT_NODE** is the **unique document ID** (the same value as NBS_UUID).*

*The value of the field named **NKW_NODE** is the **unique document ID** (the same value as NBS_UUID).*

SQL example queries

Query to extract all keywords associated with a node named 'test.pdf':

```
select nk.* from OKM_NODE_BASE nb
  inner join OKM_NODE_KEYWORD nk on nk.nkw_node = nb.nbs_uuid
  where nb.nbs_name = 'test.pdf'
```

Query to get all the categories for a node named 'test.pdf':

```
select nb.nbs_uuid, nb.nbs_name, nc.nct_category
 from OKM_NODE_BASE nb
 inner join OKM_NODE_CATEGORY nc on nc.nct_node = nb.nbs_uuid
 where nb.nbs_name = 'test'
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Advanced samples

The samples below demonstrate building your user interface based on an external application.

Field / Property	Description
Custom actions in KCenter	Demonstrates how to create a new custom menu option in the KCenter UI.
Custom UI template	Complete wizard demonstration. You have a video on youtube where it is shown how this sample works https://www.youtube.com/watch?v=do-uVuk816M
Form UI template	Demonstrates how to create a document from a third-party application form.
Security Provider	Create your security provider.
Set unique document name based on metadata	Create a unique name and unique id field based on metadata
Tab UI content extension	Demonstrates how to add content in the KCenter UI with the tab extensions.
Tab UI content extension using KCenter methods	Demonstrates how to access KCenter methods from an external application with javascript

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Custom actions in KCenter

The following sample demonstrates how to create a new custom action in the KCenter.

Sample based in a JSP

Step 1 - Create a sample JSP file

Create the following sample JSP file and copy it to the webapps/openkm folder with name ***test.jsp***

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page isErrorPage="true" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<html>
<head>
  <script type="text/javascript" src="<%=request.getContextPath() %>/kcenter/api/common.js"></script>
  <script type="text/javascript" src="<%=request.getContextPath() %>/kcenter/api/modal.js"></script>
</head>
<body>
  <h1>Quick sample about how to create a custom action in the KCenter</h1>

  <p><b>UserId:</b> <%=request.getParameter("userId")%></p>
  <p><b>Uuid:</b> <%=request.getParameter("uuid")%></p>
  <p><b>Encrypted token:</b> <%=request.getParameter("token")%></p>

  <button onclick="_closeCustomFeatureModal()">Close popup</button>
</body>
</html>
```

Step 2 - Register the custom action

Create a new custom action with these values. More information at [Custom actions](#).

Field	Value
Name	test
Label	Label test
Icon	mdi-copy-all
Description	
Document	false

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Folder	*true*
Record	*false*
Mail	*false*
URL	*../test.jsp* *Another option:* *http://localhost:8080/openkm/test.jsp*
Action	*popup-wizard*
Location	*node-toolbar*
Width	*50%*
Height	*50%*
Show close	*true*

Step 3 - Enable the custom action in the profile

- *Go to Administration > Profiles*
- *Edit a profile and **enable the custom action** in the **Misc tab**.*

Step 4 - Check the feature

- ***Refresh** the **KCenter UI***
- ***Select a folder** in the browser view.*
- *In the **contextual menu** or in the **toolbar menu** is shown a **new menu option** named "**Extra**"*

Full JSP sample

The next JSP sample allows everybody to unlock a node.

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
<%@ page import="com.openkm.util.ContextWrapper" %>
<%@ page import="com.openkm.util.StackTraceUtils" %>
<%@ page import="com.openkm.api.OKMDocument" %>
<%@ page import="com.openkm.module.db.DbNodeModule" %>
<%@ page import="com.openkm.bean.Document" %>
<%@ page import="com.openkm.principal.PrincipalUtils" %>
<%@ page import="com.openkm.module.db.stuff.DbSessionManager" %>
<%@ page import="org.apache.commons.lang3.StringUtils" %>

<html>
<head>
  <link rel="stylesheet" type="text/css" href="<%=request.getContextPath() %>/kcenter/css/bootstrap/bootstrap.css" />
  <link rel="stylesheet" type="text/css" href="<%=request.getContextPath() %>/kcenter/css/mdb/mdb.css" />
  <link rel="stylesheet" type="text/css" href="<%=request.getContextPath() %>/kcenter/css/okm.css" />
  <script type="text/javascript" src="<%=request.getContextPath() %>/kcenter/api/common.js"></script>
  <script type="text/javascript" src="<%=request.getContextPath() %>/kcenter/api/modal.js"></script>
  <script type="text/javascript" src="<%=request.getContextPath() %>/kcenter/api/browser.js"></script>
  <script>
    function closeButton() {
      _refreshUI();
      _closeCustomFeatureModal();
    }
  </script>
</head>
<body>

<%
  String error = "";
  boolean isCheckedOut = false;
  boolean isLocked = false;

  try {
    String uuid = request.getParameter("uuid");
    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
    DbNodeModule dbNodeModule = ContextWrapper.getContext().getBean(DbNodeModule.class);
    boolean isDocument = okmDocument.isValid(null, uuid);

    if (isDocument) {
      Document doc = okmDocument.getProperties(null, uuid);
      if (doc.isCheckedOut()) {
        isCheckedOut = true;
      } else if (doc.isLocked()) {
        isLocked = true;
      }
    } else {
      isLocked = dbNodeModule.isLocked(null, uuid);
    }

    if (isLocked) {
      long tenantId = PrincipalUtils.getTenantId();
      String systemToken = DbSessionManager.getInstance().getSystemToken(tenantId);

      dbNodeModule.forceUnlock(systemToken, uuid);
    }
  } catch (Exception e) {
    error = StackTraceUtils.toString(e).replaceAll("\n","<br/>");
    e.printStackTrace();
  }
%>

  <div class="d-flex justify-content-center w-100 my-2">
    <% if (StringUtils.isEmpty(error)) { %>
      <% if (isCheckedOut) { %>
        <p class="text-danger">Can not unlock a document in edition.</p>
      <% } else if (!isLocked) { %>
        <p class="text-danger">The node is not locked.</p>
```

```
    <% } else { %>
      <p class="text-success">The node has been unlocked</p>
    <% } %>
  <% } %>
</div>

<div class=" d-flex justify-content-center w-100 my-2">
  <button type="button" class="btn btn-success btn-sm my-0 m-0" onclick="closeButton()">
    Tancar
  </button>
</div>

<% if (StringUtils.isNotEmpty(error)) { %>
  <div class="d-flex justify-content-center w-100 my-2">
    <p class="text-danger">Error found:</p>
    <%= error %>
  </div>
<% } %>

</body>
</html>
```

Sample based in a Javascript

Step 1 - Register the custom action

Create a new custom action with these values. More information at Custom actions.

Field	*Value*
Name	*test*
Label	*Label test*
Icon	*mdi-assistant-direction*
Description	
Document	*true*
Folder	*false*
Record	*false*
Mail	*false*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

URL	
Action	*run-script*
Location	*node-toolbar*
Width	*50%*
Height	*50%*
Show close	*true*

Script	*const kcenterApp = window.kcenterApp; // kcenter app*
	const uuid = '1aff985f-9243-4c1a-8d43-482026f3fa93'; // uuid to jump
	kcenterApp.$store.dispatch('browser/refreshByUuid', uuid, { root: true }); // internal function to jump
	
	Accessing webservices sample:
	
	kcenterApp.$ws.node.getNodeByUuid('095cdb6e-c28e-410c-a6e3-19e23bd4b3ef').then(response => {
	console.log('here1');
	console.log(response);
	console.log('here2');
	});

Step 2 - Enable the custom action in the profile

- *Go to Administration > Profiles*
- *Edit a profile and* ***enable the custom action*** *in the* ***Misc tab****.*

Step 3 - Check the feature

- ***Refresh*** *the* ***KCenter UI***
- ***Select a document*** *in the browser view.*
- *In the* ***toolbar menu,*** *a* ***new menu icon*** *will appear.*

Sample with validation

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Step 1 - Register the custom action

Create a new custom action with these values. More information at Custom actions.

Field	*Value*
Name	*test*
Label	*Label test*
Icon	*mdi-assistant-direction*
Description	
Document	*true*
Folder	*false*
Record	*false*
Mail	*false*
URL	
Action	*run-script*
Location	*node-toolbar*
Width	*50%*
Height	*50%*
Show close	*true*
Script	`console.log('The document have the group okg:consulting');`

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Validation script	*Available since version 8.1.15.*
	
	The return statement must be a ***boolean variable or a promise****, like the sample below.*
	
	async function main() {
	console.log('execute main');
	const kcenterApp = window.kcenterApp;
	const currentNode = kcenterApp.$store.getters['browser/currentNode'];
	const isValid = await hasGroup(currentNode.uuid);
	console.log('isValid: ' + isValid);
	return isValid; // Return is valid
	}
	
	async function hasGroup(uuid) {
	const response = await kcenterApp.$ws.propertyGroup.getGroups(uuid);
	console.log(response); // Groups of the node
	const found = response.some(group => group.name === 'okg:consulting');
	return found;
	}
	
	// Return the promise (will be executed in the UI) or a boolean value
	return main();

Step 2 - Enable the custom action in the profile

- *Go to Administration >* *Profiles*
- *Edit a profile and* ***enable the custom action*** *in the* ***Misc tab****.*

Step 3 - Check the feature

- ***Refresh*** *the* ***KCenter UI***
- ***Select a document*** *in the browser view.*
- *In the* ***toolbar menu,*** *a* ***new menu icon*** *will appear.*
- *The* ***menu option*** *is* ***enabled*** *when the* ***document have the okg:consulting*** *metadata assigned.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Custom UI template

Developer information for Sample Spring Boot base project with OpenKM authentication and support for JSP, JSTL & REST.

To learn more about Spring Boot, please visit https://projects.spring.io/spring-boot/

There is also a very good tutorial about Spring Boot at https://spring.io/guides/gs/spring-boot/

> *You can download the application (**code-samples.x.x.zip**) from OpenKM Download center.*

General description

Properties file

*You have to create a file called **custom.properties** inside **$CATALINA_HOME** folder.*

> *This file is located in the root folder of the project. However you could also download from here: custom.properties*

*All application properties will be read from this file. If you need to change this path you must modify the class **MainAppConfig***

For more information: https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html (section 24.3)

These properties are:

Property	Description
openkm.url	*OpenKM url*
anonymous.user	*Anonymous user name* > *Used to get web services to OpenKM if no user is provided.*
anonymous.password	*Anonymous user password*
admin.user	*Admin user name* > *Used to get web services to OpenKM if no user is provided.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

admin.password	*Admin user password*
logback.file	*Logback file name*



Other properties in this file should not be modified unless you know what are you doing. More information about properties at https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

Authentication

This project is configured to authenticate against an OpenKM instance. It works using OpenKM SDK through web services.

OpenKM instance URL is defined in the ***openkm.url*** *configuration property.*

Log file

You must copy file ***logback-custom.xml*** *to* ***$CATALINA_HOME*** *folder. This file is used by log system.*



This file is located in the root folder of the project. However you could download from here: logback-custom.xml.

To change log level for a particular package:

```
<?xml version="1.0" encoding="UTF-8"?>
<included>
  <logger name="com.openkm" level="INFO"/>
</included>
```

Remember me

This sample customization uses H2 database to store remember me data. A database file called ***custom.mv.db*** *will be created into the user home directory.*

There is a file called ***import.sql*** *which contains SQL sentences that are executed when application starts. In this particular case it only contains one sentence to create the database table to allow the "remember me" feature.*

Roles management

In file ***menu.jsp*** *you can see how the roles are managed by the application:*

```
<sec:authorize access="hasRole('CATALOG')">
```

Using this code an user is able to see or not to see this menu option, depending on his roles. However, if you does not want to allow a user to access a resource you can con figure it in ***WebSecurityConfig*** *file.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

*Spring Security 4 automatically prefixes any role with ROLE_*

Bootstrap

The application uses Bootstrap in order to make a beautiful and responsive design. More information about Bootstrap at http://getbootstrap.com/

Packaging

*This application could be packaged as well as a **jar** and **war**. In this pom configuration the artifact will be a war.*

If you need to change packaging please see this url: http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#build-tool-plugins-maven-packaging (section 66.2)

Application port

If you packaged the application as a jar you could change the port.

To do that please visit: http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#cli-default-import-statements (section 72.4)

Catalog information

Description

This sample contains a catalog system for OpenKM:

- *Using OpenKM a user could upload files to a folder **custom.pending.folder** (we will se what is this folder)*
- *An automation we will add okg:type metadata to this document*
- *Using the catalog system of this application we will add some metadata to this document*
- *An automation we will move this document to other folder*

The main classes of the catalog system are:

- ***CatalogController**: app routing of the catalog system*
- ***CatalogServiceImpl**: services for the catalog system*
- ***WSCacheDAOImpl**: manages the webservices to communicate OpenKM and this app*

Installation

You must read the following steps in order to configure the application correctly:

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Step 1: Download

Download the application from OpenKM Download center.

> *This will downlaod a zip file contaning a Maven project. For more information about Maven: https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html*

Step 2: Role creation

*Create **ROLE_CATALOG** and add to the user used in the application.*

See Manage roles section for more information.

Step 3: SQL data needed

In order to execute SQL sentences go to Administration > Utilities > Database query and enter these queries:

```
CREATE TABLE CP_DOCUMENT_TYPE(ID INT NOT NULL, TYPE VARCHAR(100) NOT NULL, METADATA_GR
CREATE TABLE CP_AGENCY(ID INT NOT NULL, NAME VARCHAR(100) NOT NULL);
INSERT INTO CP_DOCUMENT_TYPE (ID, TYPE, METADATA_GROUP) VALUES (1, 'Consulting', 'okg:consulting
INSERT INTO CP_DOCUMENT_TYPE (ID, TYPE, METADATA_GROUP) VALUES (2, 'Marketing', 'okg:marketing')
INSERT INTO CP_AGENCY (ID, NAME) VALUES (1, 'Web');
INSERT INTO CP_AGENCY (ID, NAME) VALUES (2, 'Desktop');
```

> *SQL sentences could be different depending on your databse.*

*The parameter **custom.pending.folder** contains the uuid of the folder to read documents to catalog. You need also to create this folder in the Taxonomy manually:*

```
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('custom.pending.folder', 'string', '1
INSERT INTO OKM_CONFIG (CFG_KEY, CFG_TYPE, CFG_VALUE) VALUES ('custom.document.type', 'list', '');
```

*Update the values of the property **custom.document type**:*

```
1;okg:consulting;Consulting
2;okg:marketing;Marketing
```

Step 4: Metadata

```
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
"http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
 <property-group label="Document type (custom app)" name="okg:custom">
     <select label="Type" name="okp:custom.type" type="simple"
         optionsQuery="select ID, TYPE from CP_DOCUMENT_TYPE order by TYPE asc" />
 </property-group>
 <property-group label="Consulting" name="okg:consulting">
    <input label="Date" type="date" name="okp:consulting.date" />
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
    <input label="Input label 1" name="okp:consulting.input1" dbColumnSize="256"/>
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label 2" name="okp:consulting.input2" dbColumnSize="512"/>
  </property-group>
  <property-group label="Marketing" name="okg:marketing">
    <select label="Agency" name="okp:marketing.agency" type="simple"
      optionsQuery="select ID, NAME from CP_AGENCY order by NAME asc" />
    <input label="Date" type="date" name="okp:marketing.date" />
  </property-group>
</property-groups>
```

Right now only these tipes are supported: Input, TextArea, Select and Checkbox

Step 5: Automations

You have to create two automations like the following (see Automation section for more info):

Automation create
*Go to OpenKM administration, select the option automation and new rule, fill the name "**Move Consulting**", select the event "* *and check the box "Active" at post.*
*Go to OpenKM administration, select the option automation and new rule, fill with name "**Move Marketing**", select the event "* ***group**" and check the box "Active" at post.*

Step 6: Catalog documents from UI

- *Set in the browser the URL http://localhost:8080/custom-2.1*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Form UI template

Developer information for Sample Spring Boot base project with OpenKM authentication and support for JSP, JSTL & REST.

To learn more about Spring Boot, please visit https://projects.spring.io/spring-boot/

There is also a very good tutorial about Spring Boot at https://spring.io/guides/gs/spring-boot/

*You can download the application (**code-samples.x.x.zip**) from OpenKM Download center.*

Packaging

If you are using Eclipse it is mandatory you configure Lombok (see https://projectlombok.org/setup/eclipse)

*This application could be packaged as well as a **jar** and **war**. In this pom configuration the artifact will be a war.*

If you need to change packaging please see this url: http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#build-tool-plugins-maven-packaging (section 66.2)

Installation

Step 1 - Register metadata in OpenKM

- *Go to Administration > Metadata and register the metadata definition:*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Sample Template" name="okg:tpl">
    <input label="Name" name="okp:tpl.name"/>
    <input label="Birth Date" name="okp:tpl.birth_date" type="date" />
    <select label="Language" name="okp:tpl.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
  </property-group>
</property-groups>
```

- *Enable "Sample templates" metadata group in the profiles.*

Step 2 - Upload template file

You can also download the template.html

- *Create file named template.html with content:*

```
<html>
 <body>
  <h1>Sample template</h1>
  <table>
   <tr>
    <td><b>Name</b></td>
    <td>${okp_tpl_name!}</td>
   </tr>
   <tr>
    <td><b>Birth Date</b></td>
    <td>${okp_tpl_birth_date?string("yyyy-MM-dd")}</td>
   </tr>
   <tr>
    <td><b>Language</b></td>
    <td>${okp_tpl_language!}</td>
   </tr>
  </table>
 </body>
</html>
```

- *Upload the **template.html** file to "**/okm:templates/template.html**"*
- *Assign metadata group "Sample template" to the document "**/okm:templates/template.html**"*

Step 3 - Deploy the application



*There's a file named **application.properties** into the project where are set the OpenKM connection parameters.*

*By default application should be **run or deployed into the OpenKM server** and exist a user named **okmAdmin** with password **admin** as are set in the properties file shown below:*

```
# Custom properties
openkm.url=http://localhost:8080/openkm
openkm.user=okmAdmin
openkm.password=admin
openkm.template=/okm:templates/template.html
```

- *Copy the **forms.war** into your **$TOMCAT_HOME**/webapps*

Step 4 - Create a document from UI

- *Set in the browser the URL http://localhost:8080/forms/*
- *Fill the fields shown in the screen and click on submit button*
- *Check that into OpenKM has been created a document in **/okm:root** in this format: **{creation_date}-template.html***

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Security Provider

This section is deprecated from version 8.1.12

Developer information for OpenKM authentication through custom Authentication provider (vía patch) and custom Prinpipal adapter (via plugin).

To learn more about OpenKM authentication providers, please visit Configuring openkm.xml.

To learn more about OpenKM plugins, please visit Register a new plugin.

*You can download the application (**code-samples.x.x.zip**) from OpenKM Download center.*

General description

Properties file

*You have to modify your called **openkm.properties** inside **$CATALINA_HOME** folder.*

These properties have to be added/modified in this way:

Property	Value
okm.authentication.database	*false*
okm.authentication.supervisor	*false*
okm.authentication.ldap	*false*
okm.authentication.config	*file:/home/openkm/tomcat-8.5.69/openkm.xml* *This is the path where is your tomcat.*

Custom Principal Adapter

You must read the following steps in order to configure this piece correctly:

Step 1: Download

Download the application from OpenKM Download center.

This will downlaod a zip file contaning a Maven project. For more information about Maven: https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Step 2: OpenKM configuration

Go to the configuration thru the OpenKM administration ui and search by this property ***principal.adapter*** *and modify its value by* ***com.openkm.plugin.principal.CustomPrincipalAdapter***

Step 3: Install the plugin

Once you build the maven project (security-provider), you have to copy ***security-provider-jar-with-dependencies.jar*** *into the path:* ***$CATALINA_HOME/plugins***

You can add/update more users and roles modifying the class ***com.openkm.security.utils.Users.java***

Authentication Provider

You must read the following steps in order to configure this piece correctly:

Step 1: Download

The application is the same you downloaded from OpenKM Download center.

Step 2: Install the patch

Once you build the maven project (security-provider) copy these two classes:

- *You have to copy the class* ***com.openkm.security.provider.TestCustomAuthProvider.class*** *from* ***security-provider.jar*** *into the path:* ***$CATALINA_HOME/webapps/openkm/WEB-INF/classes/com/openkm/security/provider***
- *And copy* ***com.openkm.security.utils.Users.class*** *from* ***security-provider.jar*** *into the path:* ***$CATALINA_HOME/webapps/openkm/WEB-INF/classes/com/openkm/security/utils***

If you added/updated more users and roles, you also have to modify that class ***com.openkm.security.provider.TestCustomAuthProvider.java***

In addition, you have to configure a patch to preserve that classes across future OpenKM updates. To do this, create a folder called ***$TOMCAT_HOME/patch****. Create an script called* ***prepare.sh*** *with this content:*

```
#!/bin/bash

if [ -f "openkm.war" ]; then
 echo "Patching WAR..."
 jar uf openkm.war WEB-INF/classes/com/openkm/security/provider/TestCustomAuthProvider.class
 jar uf openkm.war WEB-INF/classes/com/openkm/security/utils/Users.class
 echo "Done!"
else
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
 echo "Not able to patch file"
fi
```



Remember to check executable permissions of the prepare.sh script.

Step 3: OpenKM configuration

You need this content into your ***openkm.xml***

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
 xmlns:security="http://www.springframework.org/schema/security"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans.xsd
 http://www.springframework.org/schema/security
 http://www.springframework.org/schema/security/spring-security.xsd">

 <!-- Security delegated to classes -->
 <security:authentication-manager alias="authenticationManager">
 <security:authentication-provider ref="testAuthProvider" />
 </security:authentication-manager>

 <beans:bean id="testAuthProvider" class="com.openkm.security.provider.TestAuthProvider">
 </beans:bean>
</beans:beans>
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Set unique document name based on metadata

Each time a new document is uploaded, the Automation action sets a unique document name based in metadata values.

- *Property **okp:data.id** stores a unique auto incremental value.*
- *Property **okp:data.code** stores a project code value.*
- *Property **okp:data.customer** stores a customer code.*
- *Property **okp:data.description** stores a document description.*
- *When a new document is uploaded the user must fill all the fields except **okp:data.id** which is automatically set by OpenKM.*
- *When Metadata group is changed - event triggered- is executed automation code which generates **okp:data.id** and renames the document based on mask **projectCode-autonumericId-clientCode-description.documentExtension**.*

The tables used in this sample is defined as:

```
-- LastID table
create table UNIQUE_ID (UN_ID INT, primary key(UN_ID));

insert into UNIQUE_ID (UN_ID) values (1);


-- Customer table
create table CUSTOMER (CUS_ID varchar(2),CUS_NAME varchar(32), primary key(CUS_ID));


insert into CUSTOMER (CUS_ID, CUS_NAME) values ('01', 'Customer 01');
insert into CUSTOMER (CUS_ID, CUS_NAME) values ('02', 'Customer 02');
insert into CUSTOMER (CUS_ID, CUS_NAME) values ('03', 'Customer 03');
```

Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
"http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
   <property-group label="Datos" name="okg:data">
      <input label="Id" type="text" name="okp:data.id" width="200px" readonly="true"/>
      <input label="Project code" type="text" name="okp:data.code" width="200px">
         <validator type="req"/>
         <validator type="num"/>
         <validator type="maxlen" parameter="6"/>
         <validator type="minlen" parameter="6"/>
      </input>
      <select label="Customer" name="okp:data.customer" type="simple"
            optionsQuery="select CUS_ID, CUS_NAME from CUSTOMER order by CUS_NAME">
         <validator type="req"/>
      </select>
      <input label="Description" type="text" name="okp:data.description" width="200px">
```

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
        <validator type="req"/>
        <validator type="maxlen" parameter="150"/>
      </input>
  </property-group>
</property-groups>
```

Automation action:



Take a look at Creating your own Automation Action

```
package com.openkm.plugin.automation.action;

import com.google.gson.Gson;
import com.openkm.core.DatabaseException;
import com.openkm.db.bean.*;
import com.openkm.db.service.LegacySrv;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.plugin.automation.Action;
import com.openkm.plugin.automation.AutomationException;
import com.openkm.plugin.automation.AutomationUtils;
import com.openkm.util.FileUtils;
import com.openkm.util.PathUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.stream.Stream;

@PluginImplementation
public class SetUniqueName extends BasePlugin implements Action {
 private static final Logger log = LoggerFactory.getLogger(SetUniqueName.class);
 private static final String PROPERTY_GROUP_NAME = "okg:data";
 private static final String PROPERTY_FIELD_ID_NAME = "okp:data.id";
 private static final String PROPERTY_FIELD_CODE_NAME = "okp:data.code";
 private static final String PROPERTY_FIELD_CUSTOMER_NAME = "okp:data.customer";
 private static final String CODE_PATTERN = "%06d";


 private static final ArrayList<AutomationRule.EnumEvents> EVENTS_AT_PRE = Stream.of(AutomationRule.Enum

 private static final ArrayList<AutomationRule.EnumEvents> EVENTS_AT_POST = new ArrayList<>();

 @Autowired
 private AutomationUtils automationUtils;

 @Autowired
 private LegacySrv legacySrv;

 @Autowired
 private DbDocumentModule dbDocumentModule;

 @Override
 public List<AutomationRule.EnumEvents> getValidEventsAtPre() {
  return EVENTS_AT_PRE;
 }
```

```
@Override
public List<AutomationRule.EnumEvents> getValidEventsAtPost() {
 return EVENTS_AT_POST;
}

@Override
public void executePre(Map<String, Object> env, Object... params) throws AutomationException {
 // Pre stage, here the metadata still has not been created and it is set the unique code in the map of values
 NodeBase node = automationUtils.getNodeBase(env);
 String propertyGroupName = automationUtils.getPropertyGroupName(env);
 try {
  if (node instanceof NodeDocument && PROPERTY_GROUP_NAME.equals(propertyGroupName)) {
   // Set the property in the map of values
   Map<String, String> proceedingProperties = automationUtils.getPropertyGroupProperties(env);
   proceedingProperties.put(PROPERTY_FIELD_ID_NAME, getUniqueCode());
   // Rename document based in metadata values
   String currentName = PathUtils.getName(node.getPath());
   String currentExtension = FileUtils.getFileExtension(currentName); // Extension must not be replaced
   // Getting values
   String id = proceedingProperties.get(PROPERTY_FIELD_ID_NAME);
   String code = proceedingProperties.get(PROPERTY_FIELD_CODE_NAME);
   String jsonValueOfCustomer = proceedingProperties.get(PROPERTY_FIELD_CUSTOMER_NAME);
   Gson gson = new Gson();
   List<String> customerValues = gson.fromJson(jsonValueOfCustomer, List.class);
   String customerValue = customerValues.get(0);
   // Setting the new name
   String newName = code + "-" + id + "-" + customerValue;
   if (!currentExtension.isEmpty()) {
    newName += "." + currentExtension;
   }
   dbDocumentModule.rename(null, node.getUuid(), newName);
  }
 } catch (Exception e) {
  log.error(e.getMessage(), e);
  throw new AutomationException(e.getMessage(), e);
 }
}

private synchronized String getUniqueCode() throws DatabaseException {
 Integer maxCode = 0;

 StringBuilder query = new StringBuilder("SELECT UN_ID FROM UNIQUE_ID");

 for (List<String> row : legacySrv.executeSQL(query.toString())) {
  maxCode = new Integer(row.get(0));
 }

 // increment value
 legacySrv.executeSQL("UPDATE UNIQUE_ID SET UN_ID = UN_ID +1");

 return String.format(CODE_PATTERN, maxCode);
}

@Override
public void executePost(Map<String, Object> env, Object... params) throws AutomationException {
}

@Override
public String getName() {
 return "Set Unique Name";
}

@Override
public String getParamType00() {
 return Automation.PARAM_TYPE_EMPTY;
}
```

```
@Override
public String getParamSrc00() {
 return Automation.PARAM_SOURCE_EMPTY;
}

@Override
public String getParamDesc00() {
 return "";
}

@Override
public String getParamType01() {
 return Automation.PARAM_TYPE_EMPTY;
}

@Override
public String getParamSrc01() {
 return Automation.PARAM_SOURCE_EMPTY;
}

@Override
public String getParamDesc01() {
 return "";
}

@Override
public String getParamType02() {
 return Automation.PARAM_TYPE_EMPTY;
}

@Override
public String getParamSrc02() {
 return Automation.PARAM_SOURCE_EMPTY;
}

@Override
public String getParamDesc02() {
 return "";
}
}
```

Create automation rule. Go to Administration > Automation and register a new rule:

- *Rule-based on "**add metadata**" event at the "**pre**" stage.*
- *Set action named "**SetUniqueName**" in the actions.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Tab UI content extension

Developer information for Sample Spring Boot base project with OpenKM authentication and support for JSP, JSTL & REST.

To learn more about Spring Boot, please visit https://projects.spring.io/spring-boot/

There is also a very good tutorial about Spring Boot at https://spring.io/guides/gs/spring-boot/



Packaging



*This application could be packaged as well as a **jar** and **war**. In this pom configuration the artifact will be a war.*

If you need to change packaging please see this url: http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#build-tool-plugins-maven-packaging (section 66.2)

Installation

You will find the instruction in the latest README.md in the demotab folder.

Tab UI content extension using KCenter methods

The following sample demonstrates how to access KCenter API methods from an external application with JavaScript.

Step 1 - create a sample HTML file

Create the following sample HTML file and copy it to the webapps/openkm folder with the name ***test.html***

```
<html>
  <head>
    <script type="text/javascript" src="kcenter/api/common.js"></script>
    <script type="text/javascript" src="kcenter/api/browser.js"></script>
  </head>
  <body>
    <h1>Quick sample about how do request to exposed KCenter API</h1>

    <button onclick="_refreshUI()">
      Refresh the UI in the browser view
    </button>
  </body>
</html>
```

Step 2 - Configure the external tab document extension

More information about the extension at the External tab document extension

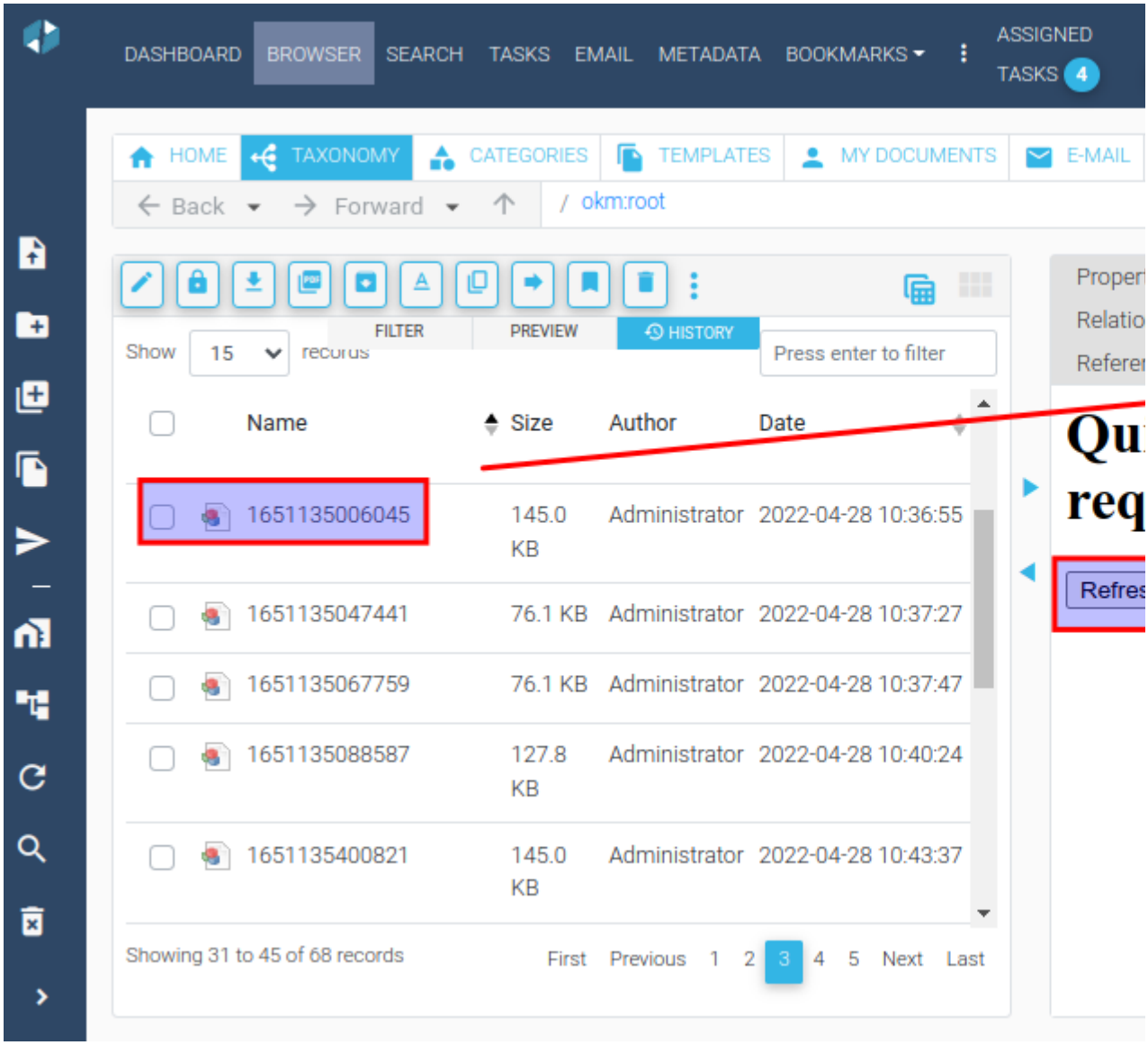
More information about enabling extensions at Enable extensions (remember to enable the extension in the profile and refresh the KCenter UI).

- *Configure extra tab document extension with the following values:*

Field / Property	*Type*	*Description*
extension.external.tab.document	***List***	*Integration;/openkm/test.html*

- ***Select a document*** *from the browser view.*
- *Click on the* ***Integration tab****.*
- *Click on " the* ***Refresh the UI in the browser view"*** *button. The KCenter UI will be refreshed.*

Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


DASHBOARD
BROWSER
SEARCH
TASKS
EMAIL
METADATA
BOOKMARKS
ASSIGNED TASKS 4
HOME
TAXONOMY
CATEGORIES
TEMPLATES
MY DOCUMENTS
E-MAIL
Back
Forward
/ okm:root
FILTER
PREVIEW
HISTORY
Show 15 records
Press enter to filter
Name
Size
Author
Date
1651135006045 145.0 KB Administrator 2022-04-28 10:36:55
1651135047441 76.1 KB Administrator 2022-04-28 10:37:27
1651135067759 76.1 KB Administrator 2022-04-28 10:37:47
1651135088587 127.8 KB Administrator 2022-04-28 10:40:24
1651135400821 145.0 KB Administrator 2022-04-28 10:43:37
Showing 31 to 45 of 68 records
First Previous 1 2 3 4 5 Next Last