

Documentation for OpenKM 8.2

Table of Contents

Table of Contents 2
OpenKM 8.2 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 questionnaire 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 Red Hat 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 the 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 the application 77
Check openkm.properties parameters 77
Database 79
Configure the 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 on Windows. 84
Migrate from Windows to Linux 84
Case-insensitive node names 84
Troubleshooting 85
Error "ASCII '\\0' appeared in the statement" when importing a database dump. 85
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 95
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 the 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 later 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 105
Can't connect to SQL Server 105
The connection was lost after some time. 106
Not able to log in to OpenKM 106
Unable to connect via TCP/IP to localhost SQL, port 1433, when using an 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
Alter SQL Server compatibility level 108
Configuring Hikari pool 109
Enable debugging 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 a 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 a remote backup 121
Do a remote backup to SMB or CIFS 122
Do a PostgreSQL backup 122
Configure crontab 122
Restoring a 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 156
Copy binary files 156
Configure service on 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 URLs 189
Configuring Lucene Analyzer 190
Configure an Analyzer 191
Configuring CADViewer 192
Troubleshooting 195
Client-side (browser) 196
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 certificate 204
Apache configuration 204

Certificate 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 ports 211
Change Tomcat ports 211
Configuring Tomcat session timeout 213
Configuring Tomcat time zone 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
OpenKM configuration 223
Parameters 223
Configuring safe trash purge 224
Configuring the search highlighter 225
Configuring Tomcat to work behind an Internet proxy 226
Linux 226
Windows 227
Step one 227
Step two 228
Configuring VidSigner 230
Change repository home 232
Understanding the contents of repository folders 232
Move entire repository location 233
Sample 1 233
Windows 233
Linux 233
Move repository folders individually 233
Sample 234
Windows 234
Linux 234
Troubleshooting 234
Windows mapped network disk is not working 234
Change default role names 235

Change the default roles 235
Edit the openkm.properties file 235
Update default roles to new names in the database 235
Enable extensions 236
LDAP configuration 238
OpenLDAP 238
Active Directory 238
Uncategorised examples shared by users 238
Tools 238
OpenLDAP example with URL base at login 239
LDAP Structure 239
Step 1 - configuration parameters 240
Step 2 - configure login 242
Another option to configure login 242
Active Directory example with referral enabled 244
LDAP Structure 244
Step 1 - configuration parameters 245
Step 2 - configure login 247
Another option to configure login 247
Active Directory example with login based on filtering users by roles 249
LDAP structure 249
Step 1 - configuration parameters 250
Step 2 - configure login 252
Another option to configure login 253
Active Directory basic configuration 255
LDAP structure 255
Step 1 - configuration parameters 255
Step 2 - configure login 257
Another option to configure login 258
Active Directory advanced configuration 259
LDAP structure 259
Step 1 - configuration parameters 259
Step 2 - configure login 261
Another option to configure login 262
Active Directory mixed configuration 264
LDAP structure 264
Step 1 - configuration parameters 264
Step 1 - configure login 266
Active Directory mixed configuration with LDAP synchronization 268
LDAP structure 268
Step 1 - configuration parameters 268
Step 2 - Simplified login 270
Step 2 - Configure login (old configuration) 270
LDAP best practices for filtering users and roles 273
Example based on a single LDAP group 273
Example based on the ROLE_USER and ROLE_ADMIN groups 273
ClearOS LDAP configuration 275
LDAP Structure 275
Step 1 - configuration parameters 276
Step 2 - create a Certificate 278
Step 3 - configure login 279
Another option to configure login 279

LDAP troubleshooting 281
How to do a nested group membership filter in Active Directory 281
No UserDetailsService registered error 281
javax.naming.PartialResultException error 281
Can't connect to LDAP server 282
Slow login or unable to log in 282
Force all users to be members of ROLE_USER. 283
org.springframework.beans.factory.parsing.BeanDefinitionParsingException 284
LDAP: error code 8 - BindSimple: Transport encryption required 285
LDAP error: Caused by java.security.cert.CertificateException: No subject alternative DNS name matching 286
Log Configuration 288
Logback-based Log Configuration 288
Third-party application execution 288
Enable log authentication 288
LibreOffice configuration 289
Linux 289
Debian & Ubuntu 289
Windows 289
Security configuration 291
Differences between simple and recursive 292
Extending security 293
Configuration for security changes which take a long time 293
Configuring Two-Factor Authentication 295
Compliance 295
Prerequisite 295
OpenKM configuration 295
Scan QR code 296
Google Authenticator use 296
Configuration 297
Register parameters 297
Configure parameters 297
Register metadata group 299
Crontab task 299
Register reports 300
Configuring CIFS 301
Database Authentication 301
OpenKM server 301
OpenKM Configuration 302
Windows client 303
Active Directory Authentication 304
Active Directory 304
OpenKM server in Active Directory configuration 305
Introduce the server into the domain 305
Redirect the required ports 306
Kerberos configuration file 306
cifs.login.config File 307
Enable Java login configuration. 308
OpenKM Configuration in Active Directory Configuration 308
Windows client in Active Directory configuration 310
Cache configuration 312
Configuring a digital email signature 313
More info 313
Configuring the anonymous user 314
Prometheus configuration 315
Setup validation 315

Running Prometheus with Docker 315
Configuring Content Replication 319
Configuration 319
Source Server 319
Target Server 319
Database Tables 319
How It Works 320
Troubleshooting 321
Server configuration 322
The packet for the query is too large (MySQL) 322
Unable to connect from another computer on 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 may 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
Red Hat 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 Add-in 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 the 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 a ZIP file 376
Antivirus checker 377
Re-indexing the whole Lucene repository 378
Execution timeout 379
Activity log actions 380
Dashboard activity 381
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 400
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

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

Backup MIME types 408
New MIME types 408
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 415
Using file system DTD 415
How metadata is stored in OpenKM 416
Metadata fields 418
Metadata Checkbox field 420
Checkbox 420
Metadata Iframe field 422
Metadata Script 424
Basic Example 425
JavaScript Example 425
Common Use Cases 426
Best Practices 426
Metadata Select field 427
Basic select example 429
Allow multiple choices 429
Based on SQL query 429
Based on className 430
Suggested options 430
Hierarchical relationship 431
Hierarchical relationship with SQL 432
Metadata Separator field 433
Metadata Suggestbox field 434
Based on filterQuery and valueQuery 435
Based on className 436
Metadata Text field 437
Metadata TextArea field 439
Metadata Validator element 441
Basic example 442
Regular expression 442
Metadata Option element 444
Basic select with selected option 444
Hierarchical relationship 445
Metadata Input field 446
Metadata VerticalPanel field 449
Metadata HorizontalPanel field 450
Metadata Security element 451
Basic grant example 451
Basic revoke example 452

Metadata internationalization 453
Label internationalization 453
Translations 453
Metadata definition 454
Manage users and roles 455
Manage users 456
Create a user 457
Edit a user 457
Delete a user 457
Change the user profile 458
Export the users 458
Manage user mail accounts 459
Create a new mail account 461
Edit a mail account 461
Delete a mail account 461
Browse emails on the mail server 461
Manage mail account filters 461
Create a mail account filter 463
Edit a mail account filter 463
Delete a mail account filter 464
Create a mail account rule 464
Edit a mail account rule 464
Delete a mail account rule 464
Manage roles 465
Create a role 465
Edit a role 466
Delete a role 466
Export the roles 466
Logged users 467
Real-time user activity log 467
Message queue 468
Send a message 469
Edit a message 469
Delete a message 469
Delete all messages in the queue 469
Refresh the message queue 469
Profiles 470
Create a new profile 470
Edit a profile 470
Delete a profile 470
Clone a profile 471
Profile General tab 472
General section 472
Misc section 472
Wizard section 474
Pagination section 475
Section General 476
Profile Components tab 478
Stacks section 478
Dashboard section 478
Tab section 479
Tab document section 480

Tab folder section 480
Tab mail section 481
Tab record section 481
Tab search section 482
Tab basic section 483
Tab advanced section 483
Profile Menu tab 485
Menu section 485
Menu bookmark section 485
Menu tool section 486
Menu help section 487
Menu file section 487
Menu edit section 489
Profile Notification tab 491
Events 491
Common 491
Document 491
Folder 493
Mail 493
Record 494
Profile Other tab 496
Toolbar section 496
File browser section 498
Relation types 499
Types 499
Bidirectional 499
Parent - child 499
Many-to-many 500
Create a relation 500
Edit a relation 500
Delete a relation 500
CSS 501
Create a new CSS 501
Edit a CSS 501
Delete a CSS 501
Clone a CSS 501
Download a CSS 502
Samples 502
New UI samples (KCenter) 502
Large font sample 502
Medium font sample 502
Arabic sample 502
Reports 504
Create a new report 504
Edit a report 504
Delete a report 505
Show report parameters 505
Execute a report 505
Report pack 506
Template reports 506
Authentication 506
Documents 506
Folders 509
Mails 511
Others 511

Subreports 512
Restrictions: 512
XML sections sample 512
Screenshots 513
Suggestions during development 515
Activity log 517
Auth actions 517
Document actions 518
Folder actions 519
Mail actions 521
Record actions 522
Property group actions 523
Node actions 524
Repository actions 524
Admin actions 524
Misc actions 534
Automation 536
Preliminary concepts. 536
Understanding Validators and Actions 538
Creating a rule 539
Editing a rule 539
Edit general parameters 539
Edit action and validation definitions 540
Deleting a rule 540
Built-in Automation Validator plugins 541
Built-in Automation Action plugins 546
Stamp 554
Create new stamp based on text 554
Create new stamp based on image 557
Edit stamp 559
Delete stamp 559
Test stamp 559
AI Prompt 560
Create AI prompt 560
Edit AI prompt 561
Delete AI prompt 561
Crontab 562
Create a new crontab job 564
Edit a crontab job 564
Execute a crontab job 564
Delete a crontab job 564
Built-in crontab tasks 565
OCR templates 568
Validation Control field description 568
Data Capture field description 570
Understanding OCR Validation Control and OCR Data Capture 571
Create a new OCR template 572
Edit an OCR template 575
Edit OCR template fields 575
Edit an OCR Validation Control 575
Edit an OCR Data Capture 576
Delete OCR template fields 576
Delete an OCR Validation Control 576

Delete an OCR Data Capture 576
Check OCR template 576
Check document recognition 577
Delete an OCR template 578
Enable OCR Data Capture 578
Built-in OCR Validation Control plugins 579
Built-in OCR Data Capture plugins 581
Languages 583
Create a new language 583
Edit a language 583
Delete a language 583
Edit language translations 584
Export language translations 584
Import language translations 584
Language translations 585
How to create a new translation 588
A sample of a new language based on Oromo from Ethiopia 588
Utilities 589
Admin permissions 591
Change user-level access 591
Cache stats 592
Clear all caches 593
Refresh cache stats 593
Enable cache stats audit trail 593
Disable cache stats audit trail 593
Check email 595
Check email 595
Check text extraction 596
Check text extraction 596
Clean conversion cache 597
Clean conversion cache 597
Create support zip 599
Custom actions 600
Database import 603
Import 603
Database query 605
Execute JDBC query 605
Execute Hibernate query 606
Allowed sentences 606
Download tokens 607
Delete token 607
Execute JAR 608
Hibernate stats 609
Information provided by Hibernate stats 609
Enable Hibernate stats 609
Clear Hibernate stats 610
Disable Hibernate stats 610
LDAP Search 611
List indexes 613

Search indexes 614
Enable show terms 615
Disable show terms 615
LogCat 616
Purge all logs 616
Download file as ZIP 616
Browse log file 616
Orphan Files 618
Filter action 618
Password encryption 619
Plugins 620
Profiling stats 623
Enable profiling stats 624
Disable profiling stats 624
Clear profiling stats 624
Refresh profiling stats 624
List of detailed calls 624
Purge data 625
Purge 625
Filter 625
Purge trash 626
Purge 626
Rebuild indexes 627
Rebuild 627
Repository checker 628
Check repository 628
Repository export 629
Export 629
Understanding output format 629
When metadata and history are not checked 629
When only metadata is checked 630
When only history is checked 631
When metadata and history are checked 631
Repository import 632
Import 633
Scripting 634
Execute a script as part of the application starting process 634
Execute a script as part of the application stopping process 634
Scripting in Automation 634
Shards 636
Shard actions 636
System properties 637
Workflow 639
Register workflow 639
Workflow forms definition 640
Basic XML skeleton 640
Workflow task form definition 640
Basic example 641
Using a file system DTD 642
Workflow fields 643

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

Workflow Button field 646
Workflow Download field 648
Basic download example 648
Dynamic download example 649
Workflow Validator element 650
Basic example 651
Regular expression 652
Workflow Node element 653
Basic download example 653
Workflow Upload field 654
Basic upload example 655
Document updated 655
Upload an example with dynamic parameters. 656
Workflow Checkbox field 657
Checkbox 657
Workflow Input field 659
Workflow Select field 661
Basic select example 662
Allow multiple choice 662
Based on SQL query 663
Based on className 663
Hierarchical relationship 664
Hierarchical relationship with SQL 664
Workflow Separator field 666
Workflow Suggestbox field 667
Based on filterQuery and valueQuery (without table attribute) 668
Based on className 669
Workflow Text field 670
Workflow TextArea field 672
Workflow Option element 674
Basic select with selected option 674
Hierarchical relationships 675
Workflow VerticalPanel field 676
Workflow HorizontalPanel field 677
User guide 678
Menu bar 679
Quick search box 679
Toolbar 679
Desktop 680
Navigation 680
Browser 680
Properties 680
Search 681
Dashboard 683
Administration 685
Status 685
Quick start 686
Logging in 686
Create a taxonomy 686
Add documents 688

Edit documents 689
Search documents 691
Essential fundamentals 693
Basic concepts 693
Understanding taxonomy, keywords, categories and metadata groups. 693
Taxonomy 694
Keywords 694
Categories 695
Metadata 695
Plan your data organization 696
Good practices 697
Working with the menu bar 698
File 698
Edit 702
Tools 704
Bookmark 708
Templates 709
Help 709
Working with the toolbar 710
Working with the desktop 714
Navigation 714
Browser 717
Pagination 721
Properties 722
Tabs on nodes of type document 723
Tabs on nodes of type folder 725
Tabs on nodes of type mail 728
Tabs on nodes of type record 730
Shortcuts 732
Working with search 734
Search panels 734
Search panel 734
Operators and wildcards 735
Default search 735
Advanced search 736
Control panel 738
Results 739
Working with Lucene queries 742
Term modifiers 742
Wildcard searches 742
Proximity searches 742
Range searches 743
Boolean operators 743
Working with Lucene queries 743
Term modifiers 744
Wildcard searches 744
Proximity searches 744
Range searches 744
Boolean operators 745
AND 745
+ 745
NOT 745
- 746
Grouping 746
Escaping special characters 746

AND 746
+ 746
NOT 747
- 747
Grouping 747
Escaping special characters 747
Working with metadata fields 747
Working with the dashboard 749
News 750
Workflow 750
Workflow lists 750
Workflow detail 751
User email signature 752
Activity log extension 753
Activity log 753
AutoCAD preview extension 754
Toolbar 754
Default preview 757
Digital Signature viewer extension 759
Verify a signed document 759
Document expiration extension 761
Add document expiration 761
Extra tab workspace extension 762
Configuration 762
Sample 763
External preview extension 764
Configuration 764
Sample 764
Configuration 764
Usage 765
External tab document extension 766
Configuration 766
Sample 767
Configuration 767
Usage 767
External Tab Folder extension 770
Configuration 770
Sample 771
Configuration 771
Usage 771
External tab mail extension 774
Configuration 774
Sample 775
Configuration 775
Usage 775
External tab record extension 778
Configuration 778
Sample 779
Configuration 779
Usage 779
External widget dashboard extension 782
Configuration 782
Sample 783
Configuration 783

Usage 783
HTML editor extension 785
Features 785
Edit HTML 786
Editor panel 786
Preview panel 787
Forum extension 788
Create new topic 788
Create reply 788
Update post 788
Delete post 788
Stamp extension 789
Document stamping 789
Workflow extension 790
Usage 790
Show workflows assigned to a node 790
Log 790
Graph 791
Using OpenKM from external applications 793
Microsoft Office Add-in 793
Digital Signature Client 793
Other applications 793
Update guide 794
Troubleshooting 794
Changelog 796
v8.2.3 796
Updater utility 797
Upgrade stages 797
Manual mode 798
Updater utility English video 800
Updater utility Spanish video 801
Tomcat update 802
Quick method 802
Linux 802
Preliminaries 802
Tomcat replacement 802
Windows 804
Preliminaries 804
Tomcat replacement 805
Manual update 808
Oracle Java to Temurin 809
Changes from version 8.1 811
Change 811
Development 812
OpenKM professional portable development environment 812
Javadoc 813
Personal development environment 814
Requirements 814
How to get the personal development environment 814
How to install 815
License activation 815
How to upgrade the development environment 815

Step 1 - OpenKM upgrade 815
Step 2 - OpenKM compilation libraries upgrade 816
Step 3 - Jbpm-console upgrade 817
Troubleshooting 817
Importing the OVA File into a Non–VirtualBox Virtual Machine 817
Configuring IDE from a clean Eclipse 819
Requirements 819
Software: 819
Hardware: 819
Download Eclipse 819
Configure Java 819
Configure Tomcat server 821
Maven support for Eclipse 827
Configuring Maven in your OS 828
Maven tips 828
Memory problems 828
Configure Eclipse to use local Maven 829
Eclipse tips 830
Debug Tomcat 830
Debug remote Tomcat server 830
Eclipse configuration 830
Eclipse known issues and troubleshooting 833
Maven raises error 206: The filename or extension is too long 833
Development tips 834
Tomcat application deployment 834
Sample deployment script 834
Profiling applications 836
Edit Tomcat startup scripts 836
Authentication 836
Monitoring Remote JVM Over SSH 837
HPROF 838
Application version compatibility table 839
Creating a JAR file with Eclipse 840
Images 840
Sample JAR file 843
Creating plugins 844
Register a new plugin 845
Creating your own AI prompt plugin 846
Method descriptions 846
Image to text sample 847
Extract metadata sample 849
Basic summarization sample 852
Creating your own Antivirus Analyzer plugin 854
Method description 854
Example of the ClamAV antivirus analyzer implementation 855
Creating your own Automation Action 856
Method descriptions 857
Understanding the env variable 858
Understanding the params variable 858
Description of available ParamType values: 859
Description of available Src values: 859
Detecting recursion 860
Example 860

Creating your own Automation Validation 863
Methods description 864
Understanding env variable 865
Understanding params variable 865
Description of available ParamType values: 866
Description of available Src values: 866
Example 867
Creating your own Crontab plugin 870
Method description 870
Example of the disposition cron implementation 871
Understanding the crontab job execution context 872
Creating your own Autocomplete Form plugin 873
Method description 873
Example of the default form values implementation 874
Metadata definition 874
Metadata Extraction using AI 875
Creating your own Default Form Values 876
Method description 876
Example of the default form values implementation 877
Metadata definition 877
Creating your own Document Converter 879
Example of the Document Converter implementation 880
Creating your own Form Validator plugin 882
Single field validation 882
Methods description 882
Example of the Form validator implementation 883
Full form validation 884
Methods description 885
Example of the Form validator implementation 886
Creating your own FormInterceptor plugin 889
Methods description 890
Example of the Form Validator implementation 890
Creating your own NodePaginator plugin 894
Method description 895
Example 895
Creating your own NodeProperties plugin (customize search result objects) 899
Method description 900
Example of NodeProperties implementation 900
Creating your own OCR Validation Control plugin 902
Method descriptions 902
Example: 903
Creating your own OCR Data Capture plugin 905
Method descriptions 905
Example 906
Creating your own Option Select Values plugin 908
Method descriptions 908
Example of the Option Select Values implementation 909
Creating your own Principal Adapter 910
Method descriptions 911
Example of the Principal adapter implementation 913
Creating your own Report plugin 916
Method descriptions 917

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

Example of the Report plugin implementation 917
Creating your own REST Plugin (extending the REST API) 920
Method description 920
Example of REST plugin implementation 921
Example of REST plugin implementation that returns a file 921
Objects declaration 922
String scenario 922
Complex Object scenario 923
Example of complex classes: 923
Advanced REST sample 924
Creating your own Search plugin 926
Method description 927
Example 927
Creating your own security access manager 929
Method descriptions 930
Example of Security access manager implementation 930
Creating your own Stack Metadata Adapter (deprecated) 933
Method descriptions 934
Example of the Stack Metadata implementation 935
Creating your own Suggestion Analyzer 943
Method descriptions 943
Example of a Suggestion Analyzer 944
Creating your own Suggestbox plugin 946
Method descriptions 946
Example of implementation 947
Creating your own Text Extractor 949
Method descriptions 949
Example of the Text Extractor implementation 950
Creating your own Version Number Adapter 951
Methods description 952
Example of the Version Number Adapter implementation 953
Creating templates 954
Creating LibreOffice templates 955
Preliminaries 955
Creating the template 955
Date format 956
Dynamic tables 957
Configuring sample 957
Creating a PDF template 959
Preliminaries 959
Creating the template 960
Date format 960
Sample configuration 960
Creating Text templates 961
Preliminaries 961
Creating the template 961
Date format 962
Sample configuration 962
Creating your own login module 963
PrincipalAdapter sample based on properties 965
Creating your own parameters 969

Sample 969
Creating your own password validator 970
Sample 971
Creating your own reports 973
Introduction to reports based on SQL 974
Create a data source 974
Testing the SQL sample report 975
A closer look at the JRXML file 978
Query section 978
Fields section 979
Remote image section 979
Using fields in report 979
Introduction to reports based on Java 980
Testing the Java sample report 980
Understanding the code sample 982
Using in the report 983
Reports with parameters 985
Understanding filtering 985
Params XML file 985
Using fields in the report 985
Java sample 987
SQL sample 988
Params form fields 990
Basic XML skeleton 990
Basic example 990
Using a file system DTD 991
Parameter fields 992
Params Validator element 994
Basic example 995
Regular expression 995
Params Input Field 997
Params Suggestbox field 999
Based on the table, filterQuery, and valueQuery 1000
Based on filterQuery and valueQuery (without table attribute) 1001
Based on className 1002
Params Checkbox field 1003
Checkbox 1003
Params TextArea field 1004
Params Select field 1005
Basic select example 1006
Allow multiple choice 1006
Hierarchical relationship 1006
Based on SQL query 1007
Based on className 1008
Suggested options 1008
Params Separator field 1010
Params Text Field 1011
Database stuff 1012
Web services 1013
RESTful 1014
Sample usage 1014

Folder samples 1016
Document samples 1016
Search samples 1017
Security samples 1017
Metadata samples 1018
Notes samples 1018
Conversion samples 1019
REST plugin samples 1019
Swagger 1020
How to access Swagger 1020
Playing with Swagger 1022
RESTClient sample 1027
Login 1027
Get roles 1028
PHP Samples 1030
Download sdk4php 1030
Preview sample 1030
Login 1030
Document samples 1031
createDocument 1031
deleteDocument 1032
getDocumentProperties 1033
getContent 1034
getDocumentChildren 1035
moveDocument 1036
copyDocument 1037
Folder sample 1038
createFolder 1038
getFolderProperties 1039
deleteFolder 1040
renameFolder 1041
moveFolder 1042
getFolderChildren 1043
copyFolder 1044
extendedFolderCopy 1045
getContentInfo 1046
purgeFolder 1047
setStyle 1048
createMissingFolders 1049
createFolderFromTemplate 1050
PropertyGroup samples 1052
addPropertyGroup 1052
removePropertyGroup 1053
getPropertyGroups 1054
getAllPropertyGroups 1055
getPropertyGroupForm 1056
getPropertyGroupFormByNode 1056
setPropertyGroupProperties 1058
hasPropertyGroup 1059
getRegisteredPropertyGroupDefinition 1060
getPropertyGroupSuggestions 1061
registerPropertyGroupDefinition 1063
getPropertyGroupProperties 1064
getPropertyGroupPropertiesByVersion 1065
Search samples 1066
find 1066
findPaginated 1067
Integrate preview with other web applications 1070
Troubleshooting 1073

Samples 1074
Scripting samples 1075
Script - Basic samples 1077
Update repository statistics 1077
Create some folders and set a property group 1077
Show the number of documents, folders, and the size for a given path 1077
Get path by UUID 1077
Show all documents used by every running workflow 1078
Script - Compact documents with size equal to 0 1079
Script - Convert TIF to PDF 1080
Script - Count all child subfolders 1081
Script - Exporter 1082
Script - Export repository folder to server as a ZIP file 1083
Script - Folder depth 1084
Images 1085
Execution 1085
Results 1085
Script - Import data from CSV file 1087
Image 1089
Register the metadata group 1089
Repository view 1089
Execution 1090
Script results 1091
Imported metadata 1091
Script - Purge all folders into a parent 1092
Script - Purge all users' trash 1093
Script - Put documents from a folder and its descendants into the pending text extraction queue 1094
Script - Read PDF form fields 1095
Script - Recursive character renaming 1096
Script - Recursive repository traversal 1097
Script - Recursive security change 1098
Script - Security change at low level 1100
Images 1103
Initial security status 1103
UUID of the folder 1104
Execution 1105
Log 1106
Rebuild Lucene indexes 1107
Security changes applied 1108
Script samples for automation 1110
Script for automation - Autonumeric 1111
Register metadata group 1112
Script for automation - Set unique document name based on metadata 1113
Register automation rule 1115
Crontab samples 1116
Crontab sample - Basic file system document importer 1117
Register a new plugin 1118
Images 1118
Crontab task 1118
Files in the application server 1119

Result 1119
Crontab sample - Document importer with metadata values 1121
Register a new plugin 1126
Images 1126
Register metadata definition 1126
Crontab task 1127
Files on the application server 1128
Result 1128
Mail notification 1128
Files imported into the repository 1129
Crontab sample - Generate HTML report 1131
Register a new plugin 1132
Crontab task - XML importer 1133
Register a new plugin 1138
Images 1138
Crontab task 1138
Register metadata definition 1139
Logfile folder 1140
Output folder 1140
Result 1140
Variables by Automation Event 1142
Available environment variables 1142
Document 1145
Create Document 1145
Update Document 1146
Delete Document 1147
Rename Document 1147
Purge Document 1148
Move Document 1148
Restore Version Document 1149
Download Document from UI 1149
Download Document from UI for preview 1150
Print Document 1151
Stamp Document 1151
Virus Detected Document 1152
Set Node Class Document 1152
Folder 1153
Create Folder 1153
Delete Folder 1153
Purge Folder 1154
Move Folder 1154
Rename Folder 1155
Export Folder as ZIP 1155
Export Folder as JAR 1156
Mail 1156
Create Mail 1156
Move Mail 1158
Download Mail from UI 1158
Download Mail from UI for preview 1159
Set Node Class Mail 1159
Import Mail Attachments 1160
Delete Mail 1160
Purge Mail 1160
Record 1161
Create Record 1161
Delete Record 1162
Purge Record 1162
Move Record 1163
Set Node Class Record 1163
Property Group (metadata) 1164
Add property group 1164
Get all property groups 1165

Set property group 1165
Remove property group 1165
Auth 1166
Create User 1166
User Login 1166
User Logout 1167
Task 1167
Create Task 1167
Update Task 1167
Delete Task 1168
Note 1168
Create Note 1168
Update Note 1169
Delete Note 1170
General 1170
Text extraction 1170
Export Nodes to ZIP 1171
Vault of stuff 1172
Convert Exchange Legacy DN 1173
Download document with direct link 1175
Import a document with a Bash script using REST 1176
Creating users from scripts 1177
API description 1178
OKMActivity 1180
Methods 1180
findByFilter 1180
countByFilter 1180
getActions 1181
OKMAuth 1183
Basics 1183
Methods 1183
login 1184
login 1184
grantUser 1185
revokeUser 1186
getGrantedUsers 1186
grantRole 1187
revokeRole 1188
getGrantedRoles 1189
getUsers 1189
getRoles 1190
getUsersByRole 1190
getRolesByUser 1191
changeSecurity 1191
overwriteSecurity 1192
createUser 1193
deleteUser 1193
updateUser 1194
createRole 1195
deleteRole 1195
updateRole 1196
assignRole 1196
removeRole 1197
getProfiles 1197
getUserProfile 1198
setUserProfile 1198
logout 1199
getUser 1200
setUserPermissions 1200
setRolePermissions 1201
getUserTenants 1202

setUserTenant 1202
isPasswordExpired 1203
resetPassword 1203
OKMBookmark 1205
Basics 1205
Methods 1205
add 1205
get 1206
remove 1206
rename 1207
getAll 1207
OKMDashboard 1209
Basics 1209
Methods 1209
getUserCheckedOutDocuments 1209
getUserLastModifiedDocuments 1210
getUserLockedDocuments 1211
getUserLockedRecords 1212
getUserLockedFolders 1213
getUserLockedMails 1214
getUserSubscribedDocuments 1215
getUserSubscribedFolders 1216
getUserSubscribedRecords 1217
getUserLastCreatedDocuments 1218
getUserLastCreatedDocumentsNotes 1219
getUserLastCreatedFolders 1220
getUserLastCreatedFoldersNotes 1221
getUserLastCreatedRecords 1222
getUserLastCreatedRecordsNotes 1223
getUserLastDownloadedDocuments 1224
getUserLastImportedMails 1225
getUserLastCreatedMailsNotes 1226
getUserLastImportedMailAttachments 1227
getUserDocumentsSize 1228
getUserSearches 1229
find 1230
getLastWeekTopDownloadedDocuments 1231
getLastMonthTopDownloadedDocuments 1232
getLastWeekTopModifiedDocuments 1233
getLastMonthTopModifiedDocuments 1234
getLastModifiedDocuments 1235
getLastCreatedDocuments 1236
getLastCheckedOutDocuments 1237
getLastCreatedFolders 1238
getLastCreatedRecords 1239
visitNode 1240
OKMDocument 1242
Basics 1242
Methods 1242
create 1242
createSimple 1243
delete 1244
getProperties 1244
getContent 1245
getContentByVersion 1246
getExtractedText 1247
getChildren 1247
rename 1248
setProperties 1249
setLanguage 1249
setTitle 1250
checkout 1251
cancelCheckout 1251
forceCancelCheckout 1252

isCheckedOut 1252
checkin 1253
checkin 1254
lock 1255
unlock 1255
forceUnlock 1256
isLocked 1257
getLockInfo 1257
purge 1258
move 1258
copy 1259
copy 1260
extendedCopy 1260
getVersionHistorySize 1262
isValid 1262
getPath 1263
getDetectedLanguages 1263
createFromTemplate 1264
updateFromTemplate 1265
setAnnotations 1266
getAnnotations 1267
getCheckedOut 1267
stamp 1268
stampText 1269
stampImage 1270
setDescription 1271
setNodeClass 1271
setDispositionStage 1272
createWizard 1273
getNumberOfPages 1273
getPageAsImage 1274
isConvertibleToPDF 1274
setIndexable 1275
mergePdf 1276
OKMFolder 1277
Basics 1277
Methods 1277
create 1277
createSimple 1278
getProperties 1278
delete 1279
purge 1279
rename 1280
move 1281
copy 1281
copy 1282
extendedCopy 1283
extendedCopy 1284
getChildren 1285
getContentInfo 1285
isValid 1286
getPath 1287
setStyle 1287
createMissingFolders 1288
addNodeClassChildren 1288
setDescription 1289
OKMImport 1290
Basics 1290
Methods 1290
createDocument 1290
createFolder 1291
createRecord 1291
createMail 1292
importFolder 1293

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

OKMMail 1295
Basics 1295
Methods 1295
getProperties 1295
createAttachment 1296
deleteAttachment 1296
getAttachments 1297
delete 1297
purge 1298
rename 1299
move 1299
copy 1300
copy 1300
extendedCopy 1301
extendedCopy 1302
getChildren 1303
isValid 1304
getPath 1304
sendMail 1305
setTitle 1305
sendMailWithAttachments 1306
importEml 1307
importMsg 1308
setNodeClass 1309
setDispositionStage 1309
setDescription 1310
getContent 1310
createWizard 1311
getExtractedText 1312
forwardEmail 1313
OKMNode 1314
Basics 1314
Methods 1314
getNodeByUuid 1314
getVersionHistory 1315
restoreVersion 1315
renameVersion 1316
purgeVersionHistory 1316
mayBePromotedAsRecord 1317
promoteAsRecord 1318
degradeRecord 1318
isElectronicRecordPath 1319
getElectronicRecordInPath 1319
getChildrenPaginated 1320
getChildrenByCategoryPaginated 1321
getBreadcrumb 1323
subscribe 1323
unsubscribe 1324
importZip 1324
unzip 1325
exportZip 1326
getNodesFromUuids 1326
evaluateZipDownload 1327
restore 1328
hasNodesLockedByOtherUser 1329
lock 1329
forceLock 1330
unlock 1330
forceUnlock 1331
isLocked 1332
getLockInfo 1332
setComment 1333
OKMNote 1334
Basics 1334

Methods 1334
add 1334
get 1335
delete 1335
set 1336
list 1337
getNoteHistories 1337
OKMNotification 1339
Basics 1339
Methods 1339
subscribe 1339
unsubscribe 1340
getSubscriptors 1340
notify 1341
OKMProperty 1342
Basics 1342
Methods 1342
addCategory 1342
removeCategory 1343
addKeyword 1343
removeKeyword 1344
setEncryption 1345
unsetEncryption 1345
setSigned 1346
isSigned 1347
OKMPropertyGroup 1348
Basics 1348
Methods 1348
addGroup 1348
removeGroup 1350
getGroups 1351
getAllGroups 1352
getAllGroups 1352
getUserGroups 1353
getProperties 1353
setProperties 1354
getPropertyGroupForm 1356
hasGroup 1357
getSuggestions 1357
registerDefinition 1358
getRegisteredDefinition 1359
getGroups 1360
getGroupProperties 1360
getProperties 1361
getSuggestBoxKeyValue 1362
getSuggestBoxKeyValuesFiltered 1362
OKMRecord 1364
Basics 1364
Methods 1364
create 1364
createSimple 1365
getProperties 1365
delete 1366
purge 1366
rename 1367
move 1368
copy 1368
copy 1369
extendedCopy 1370
isValid 1371
getPath 1371
getChildren 1372

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

lock 1372
unlock 1373
forceUnlock 1374
setTitle 1374
setNodeClass 1375
setDispositionStage 1375
addNodeClassChildren 1376
setDescription 1376
createWizard 1377
OKMRelation 1379
Basics 1379
Methods 1379
getRelationTypes 1379
add 1380
delete 1381
getRelations 1381
getRelationGroups 1382
addGroup 1382
addNodeToGroup 1383
deleteGroup 1384
getGroup 1384
setGroupName 1385
OKMReport 1387
Basics 1387
Methods 1387
list 1387
get 1388
execute 1388
execute 1389
OKMRepository 1391
Methods 1391
getRootFolder 1391
getTrashFolder 1391
getTrashFolderBase 1392
getTemplatesFolder 1393
getPersonalFolder 1393
getPersonalFolderBase 1394
getMailFolder 1394
getMailFolderBase 1395
getThesaurusFolder 1396
getCategoriesFolder 1396
purgeTrash 1397
getUpdateMessage 1397
getRepositoryUuid 1398
hasNode 1399
getNodePath 1399
getNodeUuid 1400
getAppVersion 1400
copyAttributes 1401
getMetadataFolder 1402
getChangeLog 1402
getAppLicense 1403
OKMSearch 1404
Basics 1404
Methods 1407
find 1407
find 1408
findPaginated 1409
findWithMetadata 1410
findWithMetadataPaginated 1411
findByQuery 1412
findByQuery 1413
findByQuery 1414

findByQueryPaginated 1415
findByQueryPaginated 1416
findByQueryPaginated 1417
findSimpleNodeBaseByQueryPaginated 1419
saveSearch 1420
updateSearch 1421
getSearch 1421
getAllSearches 1422
deleteSearch 1423
getKeywordMap 1423
getCategorizedDocuments 1424
getCategorizedFolders 1425
getCategorizedMails 1426
getCategorizedRecords 1426
getDocumentsByKeyword 1427
getFoldersByKeyword 1427
getMailsByKeyword 1428
getRecordsByKeyword 1428
getDocumentsByPropertyValue 1429
getFoldersByPropertyValue 1429
getMailsByPropertyValue 1430
getRecordsByPropertyValue 1431
findMailPaginated 1431
csvExport 1433
OKMStamp 1435
Methods 1435
calculateStampCoordinates 1435
calculateStampExpressions 1436
stampDocument 1437
stampTextCustom 1438
stampImageCustom 1439
stampImageCustom 1439
getAllStamps 1440
getStampTextByPk 1441
getStampImageByPk 1441
getStampBarcodeByPk 1442
calculateFlyImageDimensions 1443
getPersonalStamps 1443
getPersonalStampImage 1444
OKMStats 1446
Basics 1446
Methods 1446
getDocumentsByContext 1446
getDocumentsSizeByContext 1446
getFoldersByContext 1447
getMailsByContext 1448
getRecordsByContext 1448
OKMTask 1450
Basics 1450
Methods 1452
create 1452
update 1453
find 1454
delete 1454
getByProject 1455
getByType 1455
getAssigned 1456
getAssigned 1457
countAssigned 1458
getActive 1459
getActive 1460

countActive 1461
getFinished 1462
getFinished 1463
countFinished 1464
findStatus 1465
findProject 1466
findType 1466
findNote 1467
getStatuses 1467
getProjects 1468
getTypes 1468
getNotes 1469
createStatus 1470
createProject 1470
createType 1471
createNote 1471
updateStatus 1472
updateProject 1473
updateType 1473
updateNote 1474
deleteStatus 1475
deleteProject 1475
deleteType 1476
deleteNote 1476
getByNode 1477
getHistory 1477
getAssigned 1478
countAssigned 1480
getActive 1481
countActive 1482
getFinished 1483
countFinished 1485
getAssignedCount 1486
getActiveCount 1487
getNotified 1488
getFinishedCount 1489
OKMUserConfig 1491
Methods 1491
setHome 1491
getConfig 1491
OKMWorkflow 1493
Methods 1493
registerProcessDefinition 1493
deleteProcessDefinition 1493
getProcessDefinitionImage 1494
getProcessDefinitionForms 1495
getProcessDefinition 1496
runProcessDefinition 1496
runProcessDefinition (without node association) 1497
sendProcessInstanceSignal 1498
endProcessInstance 1499
deleteProcessInstance 1500
findProcessInstances 1500
findAllProcessDefinitions 1501
findLatestProcessDefinitions 1502
findLastProcessDefinition 1502
getProcessInstance 1503
suspendProcessInstance 1504
resumeProcessInstance 1504
findUserTaskInstances 1505
findPooledTaskInstances 1506
findTaskInstances 1506
setTaskInstanceValues 1507
getTaskInstance 1508
setTaskInstanceActorId 1508

startTaskInstance 1509
endTaskInstance 1510
getToken 1510
sendTokenSignal 1511
Common classes 1513
ArchiveUtils 1515
Methods 1515
createZip 1515
createJar 1515
ExecutionUtils 1517
Methods 1517
runScript 1517
runJar 1517
runCmd 1518
FileLogger 1519
Methods 1519
info 1519
warn 1519
error 1520
FileUtils 1521
Methods 1521
getFileName 1521
getFileExtension 1521
FormatUtil 1523
Methods 1523
formatSize 1523
parseSize 1523
formatDate 1524
ISO8601 1525
Methods 1525
parseBasic 1525
formatBasic 1525
GeneralUtils 1527
Methods 1527
uiNotify 1527
MailUtils 1528
Methods 1528
sendMessage 1528
sendMessage 1528
sendDocuments 1529
PathUtils 1531
Methods 1531
getParent 1531
getName 1531
getContext 1532
PDFUtils 1533
Methods 1533
merge 1533
stampImage 1534
stampText 1535
getNumberOfPages 1536
PrincipalUtils 1537
Methods 1537
getUser 1537
getTenantId 1537
StackTraceUtils 1539
Methods 1539

isCallingMe 1539
toString 1539
TemplateUtils 1541
Methods 1541
templateExists 1541
replace 1541
WorkflowUtils 1543
Methods 1543
findProcessInstancesByNode 1543
ShardUtils 1544
Methods 1544
setCurrentShard 1544
createShardDirs 1544
changeNodeShard 1545
getShardIds 1545
getShardSize 1546
getShardPaths 1546
getShards 1547
getShardFolders 1547
LegacySrv 1549
executeSQL 1549
executeHQL 1549
DbSessionManager 1551
Methods 1551
getSystemToken 1551
ConfigSrv 1552
getString 1552
getText 1552
getBoolean 1553
getInteger 1553
getLong 1554
getHtml 1554
getPassword 1555
getList 1555
Database description 1557
Node types 1557
SQL example queries 1558
Mark documents to be indexed again 1558
Node version 1558
SQL example queries 1559
Node metadata properties 1561
Node categories and keywords 1562
SQL example queries 1563
Advanced samples 1565
Custom actions in KCenter 1566
Sample based on a JSP 1566
Step 1 - Create a sample JSP file 1566
Step 2 - Register the custom action 1566
Step 3 - Enable the custom action in the profile 1567
Step 4 - Check the feature 1567
Full JSP sample 1567
Sample based on JavaScript 1569
Step 1 - Register the custom action 1569
Step 2 - Enable the custom action in the profile 1570
Step 3 - Check the feature 1570
Sample with validation 1570
Step 1 - Register the custom action 1571

Step 2 - Enable the custom action in the profile 1572
Step 3 - Check the feature 1572
Custom UI template 1573
General description 1573
Properties file 1573
Authentication 1574
Log file 1574
Remember me 1574
Roles management 1574
Bootstrap 1575
Packaging 1575
Application port 1575
Catalog information 1575
Description 1575
Installation 1575
Step 1: Download 1576
Step 2: Role creation 1576
Step 3: SQL data needed 1576
Step 4: Metadata 1576
Step 5: Automations 1577
Step 6: Catalog documents from UI 1577
Form UI template 1578
Packaging 1578
Installation 1578
Step 1 - Register metadata in OpenKM 1578
Step 2 - Upload template file 1578
Security Provider 1580
General description 1580
Properties file 1580
Custom Principal Adapter 1580
Step 1: Download 1580
Step 2: OpenKM configuration 1581
Step 3: Install the plugin 1581
Authentication Provider 1581
Step 1: Download 1581
Step 2: Install the patch 1581
Step 3: OpenKM configuration 1582
Set a unique document name based on metadata 1583
Tab UI content extension 1587
Packaging 1587
Installation 1587
Tab UI content extension using KCenter methods 1588
Step 1 - create a sample HTML file 1588
Step 2 - Configure the external tab document extension 1588

OpenKM 8.2

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 often more influenced by marketing than by 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 had some marketing success and is the most commonly used. However, we prefer the term **EDRMS**, perhaps with less success in the marketing world but stricter and more 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 holds to records.
- Manage records disposition.
- Preserve records throughout their life cycle.

Some of OpenKM's 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 it for inspections.
- Preserve **historical content**. Track a document's status (determined by its lifecycle phases).
- **Consolidate** the company's knowledge into a single access point.

The maxim 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 simple application; it 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. If you have any doubts, contact us. Usually, a problem can be resolved in several ways.

Your suggestions are always welcome; 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 it 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 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 "Command Prompt" 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

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

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 installing Installing 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 of RAM (>= 2 GB 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 inexpensive.

A 64-bit architecture is highly recommended.

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 your sales and marketing representatives. They should provide you with an estimate of the hardware requirements.

(See the questionnaire at the bottom of the page)

You can use your document and folder history to get some of these values.

Browser support

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

Currently supported browsers:

- 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 you to keep your browser 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 distributions and build the packages for them. However, any recent Linux distribution should be usable as long as the:

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

Windows (Known working configurations):

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

Databases

Known working configurations:

- MySQL
- Oracle
- PostgreSQL
- SQL Server



The application internally uses Hibernate; in fact, you should be able to run it with any supported Hibernate dialect; for more information, read Hibernate: SQL Dialects.

JVM

Although OpenKM can run with other JDKs, we recommend using OpenJDK 11. A 64-bit architecture is highly recommended.

LDAP

- Open LDAP
- Microsoft Active Directory

Small questionnaire



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

Try to give them as much detailed information as possible for a more accurate estimate 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 more detailed questions below are a guide to describe the document management problem to solve and may be helpful to estimate hardware requirements:

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 are 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 queries are expected per minute and per hour?
- What kind of documents do you manage, such as standard Office documents, PDFs, or less common documents like video and large TIFF files?
    - Describe them, especially those you know are not common in most 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 other software is 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 is i In most cases,
Ghostscript	• Interpreter for the PostScript language and PDF. • Needed by ImageMagick.	Usually, it is i
Tesseract	• An OCR engine. • It is used for the text extraction feature.	It is required w
FFmpeg	• Converts and streams audio and video.	
ClamAV or equivalent	• Antivirus.	Required whe

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 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 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 via SSH because it is fast and secure. Please don't install Java, Tomcat, or any other software unless you have already consulted 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 free disk space is 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 distribution (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 other 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

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)
```

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

Otherwise, set Java 8 as the default one:

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

If you prefer a different 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 in "/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 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 Red Hat and CentOS

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

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

This guide has been validated with the following:

- Red Hat 9.1
- Red Hat 8.7
- Oracle Linux 7.6
- Rocky Linux 9.1
- CentOS Stream 9

If you use a proxy to access the Internet, please see 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:

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

If you do not 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 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 as follows:

```
$ 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

Red Hat does not allow connections to port 8080 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



If you plan to configure OpenKM in cluster or sharding mode, please read the following sections before starting the installation process:

- Cluster configuration
- Sharding configuration

Check that you have enough disk space.

We recommend installing these tools:

- Notepad++ (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 which one you have installed, but we recommend the latest 1.8 release.

We also recommend setting the JAVA_HOME variable:

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

Install Java



Only perform these steps 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 already have it 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 is not the 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;
```

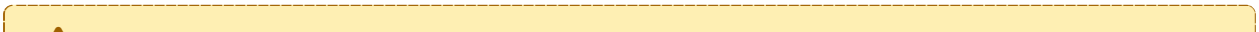
Installing Tomcat

Unzip Tomcat-9.0.76.zip to c:\

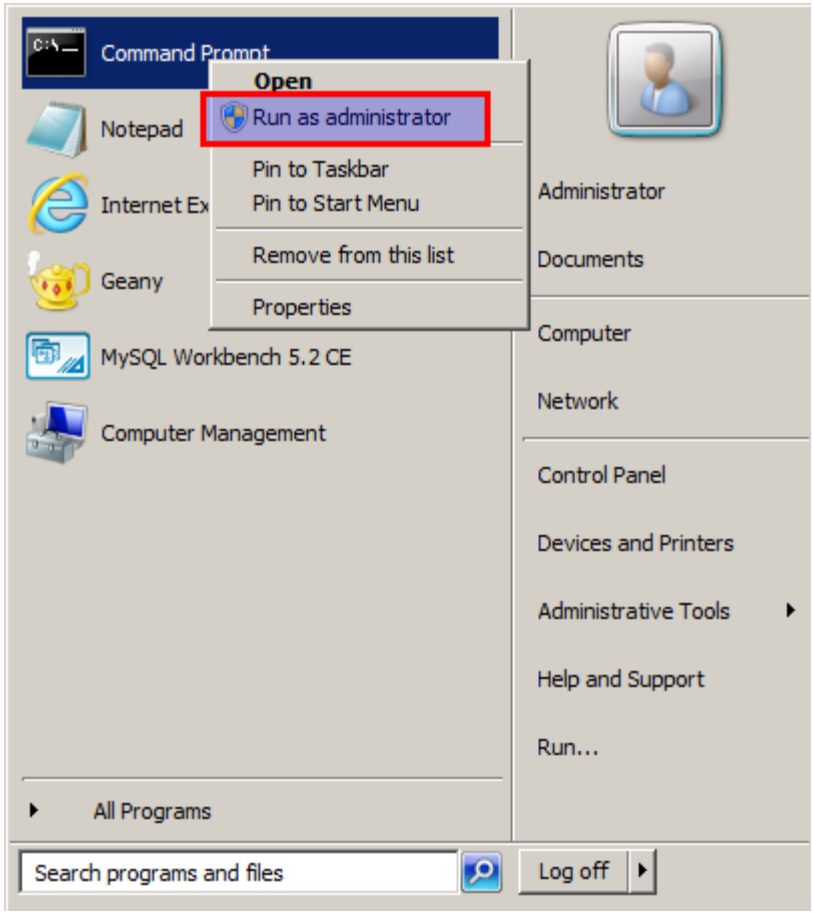
The Tomcat version numbering can vary when upgrading to a major version.

Configuring Tomcat as a service

Register for the service:



Open a Windows terminal as administrator.



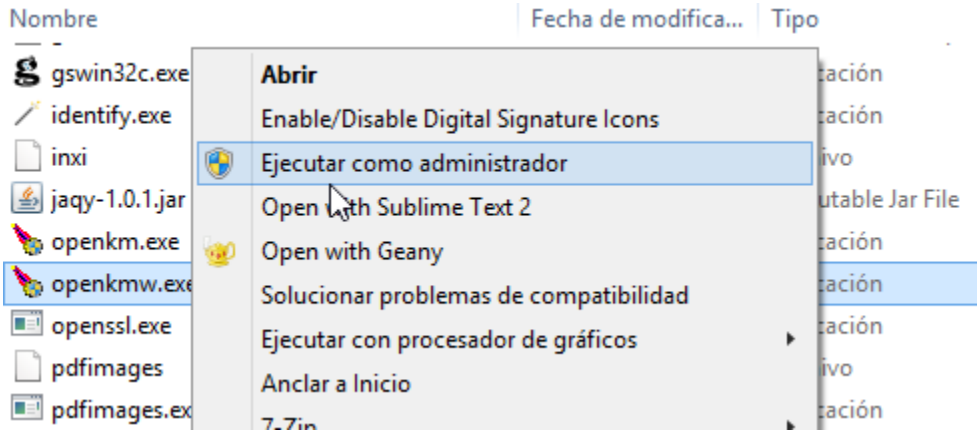

```
cd c:\tomcat-9.0.76\bin
service.bat install
```

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"
```

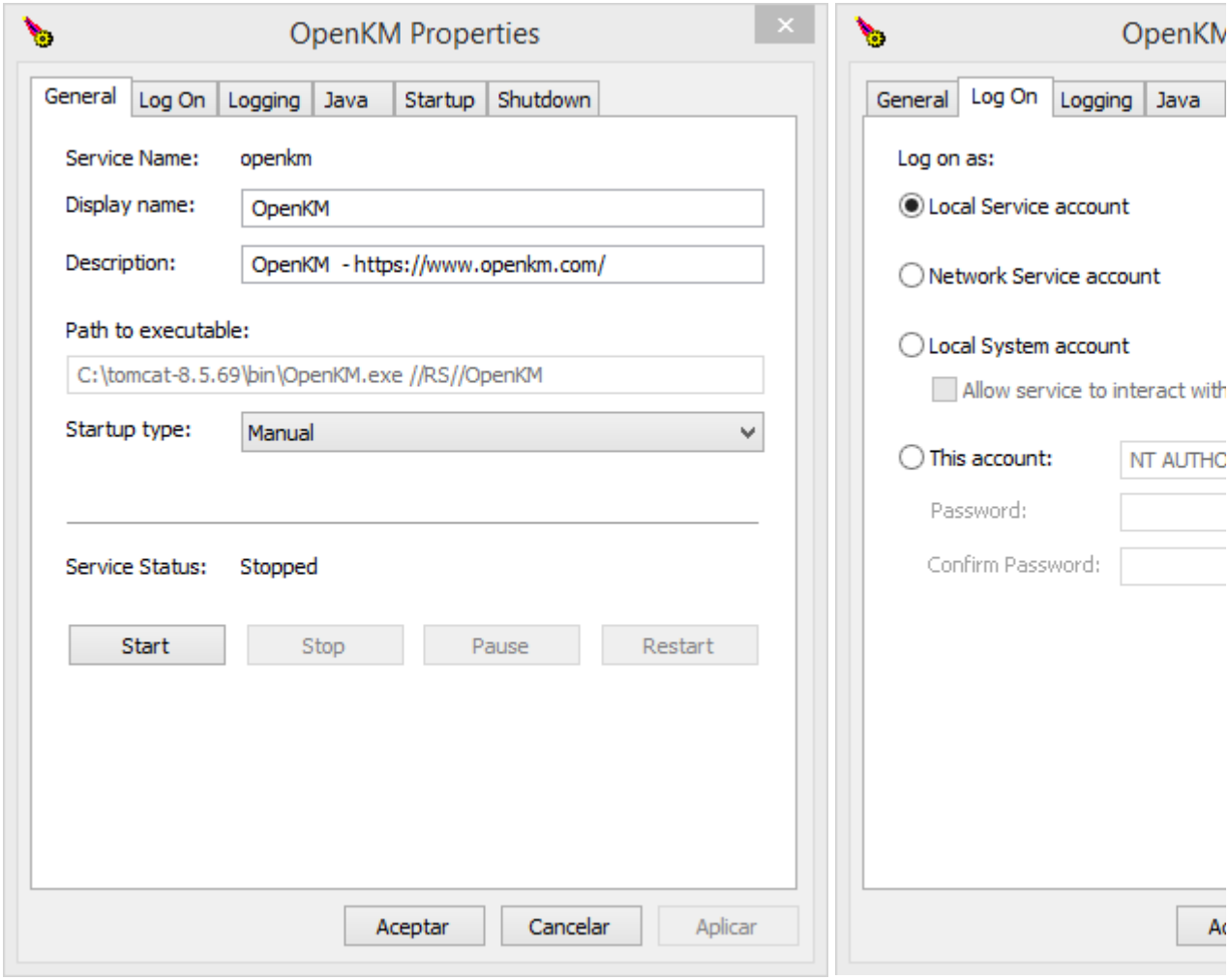
```
openkm //US//OpenKM ++JvmOptions="-Dfile.encoding=UTF-8"
openkm //US//OpenKM ++JvmOptions="-Dsun.jnu.encoding=UTF-8"
openkm //US//OpenKM ++JvmOptions="-Djava.awt.headless=true"
openkm //US//OpenKM ++JvmOptions="-Dsun.java2d.cmm=sun.java2d.cmm.kcms.KcmsServiceProvider"
```

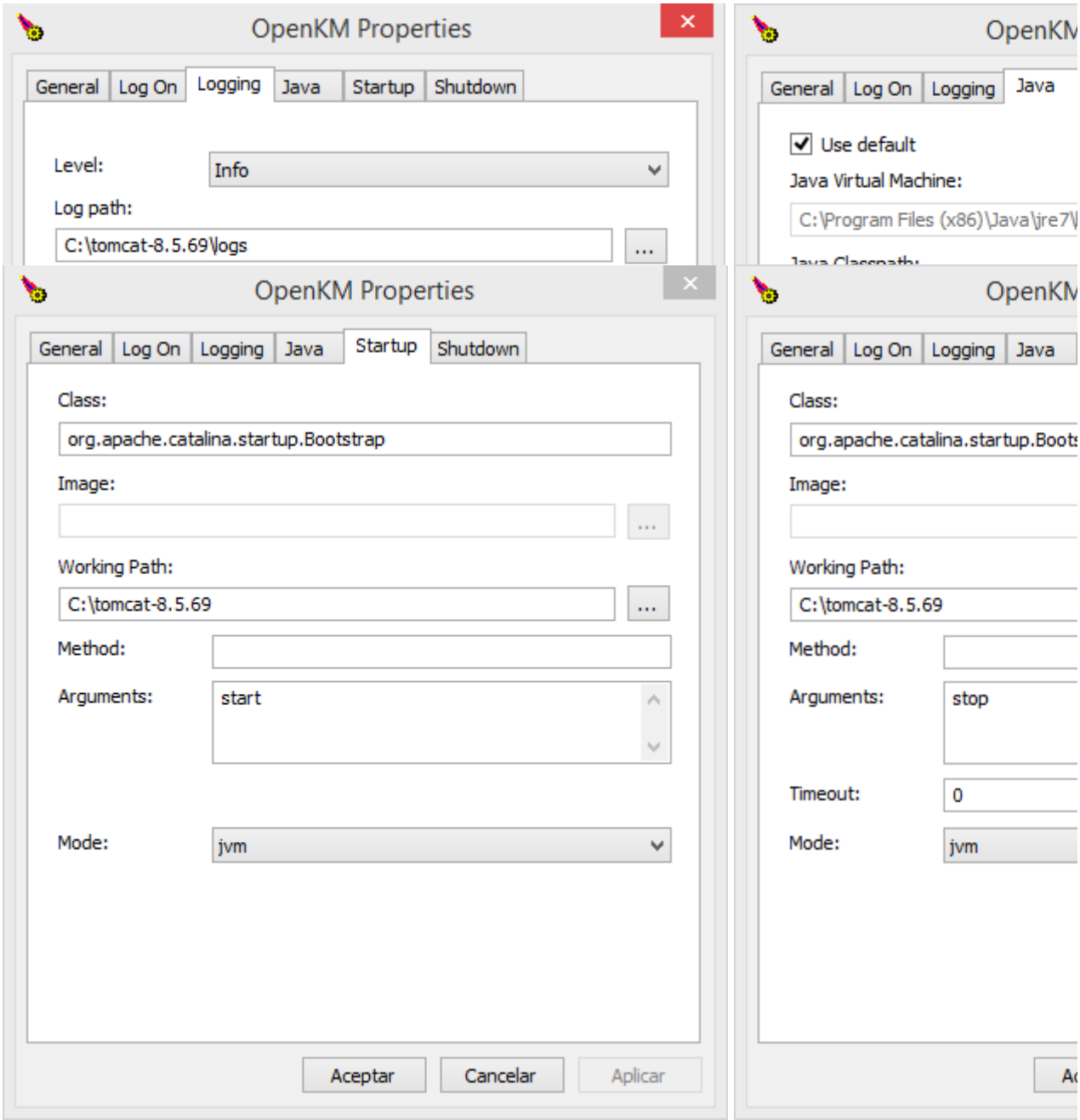
Execute the openkmw.exe as administrator:



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
stop
Timeout:
0

More information about configuring the Tomcat service on Windows is available at:
https://tomcat.apache.org/tomcat-9.0-doc/windows-service-howto.html

Check the service

From terminal

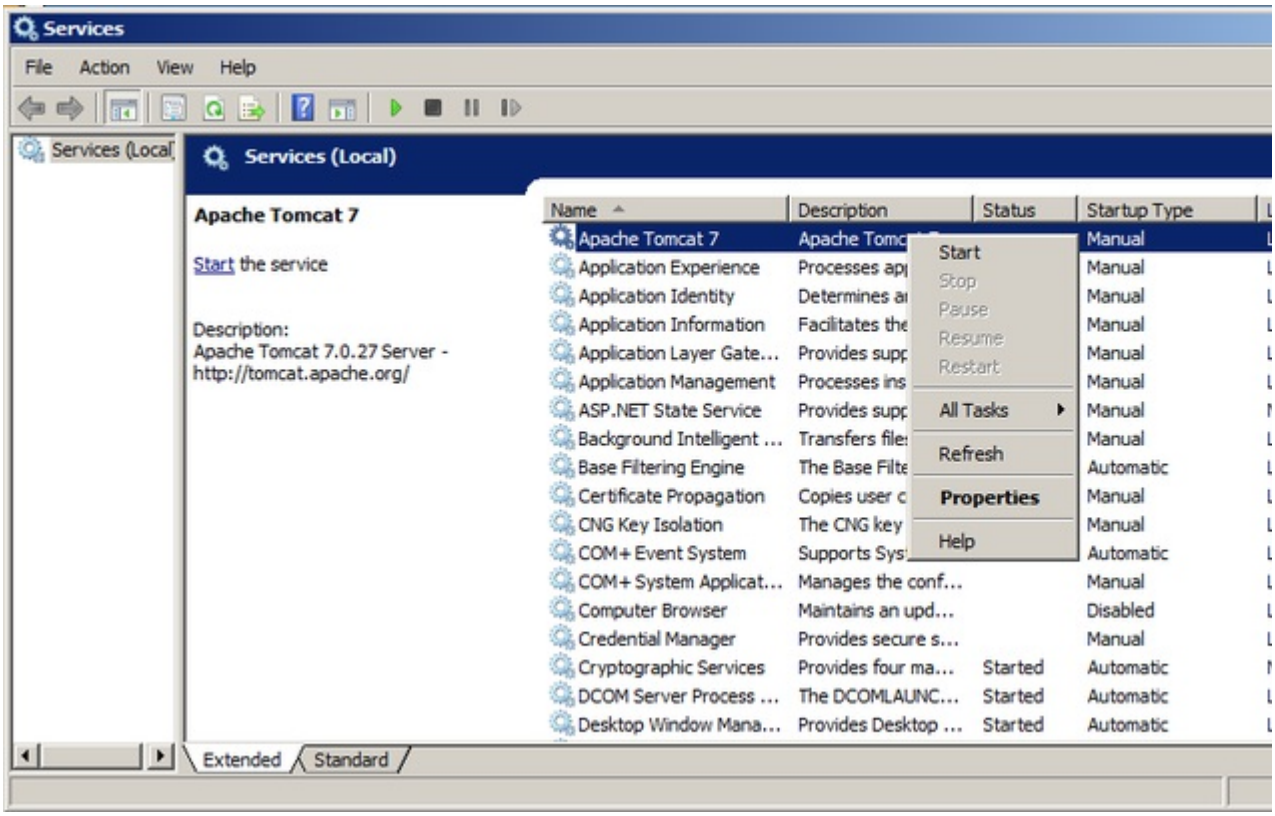
Start the service:

```
$ net start OpenKM
```

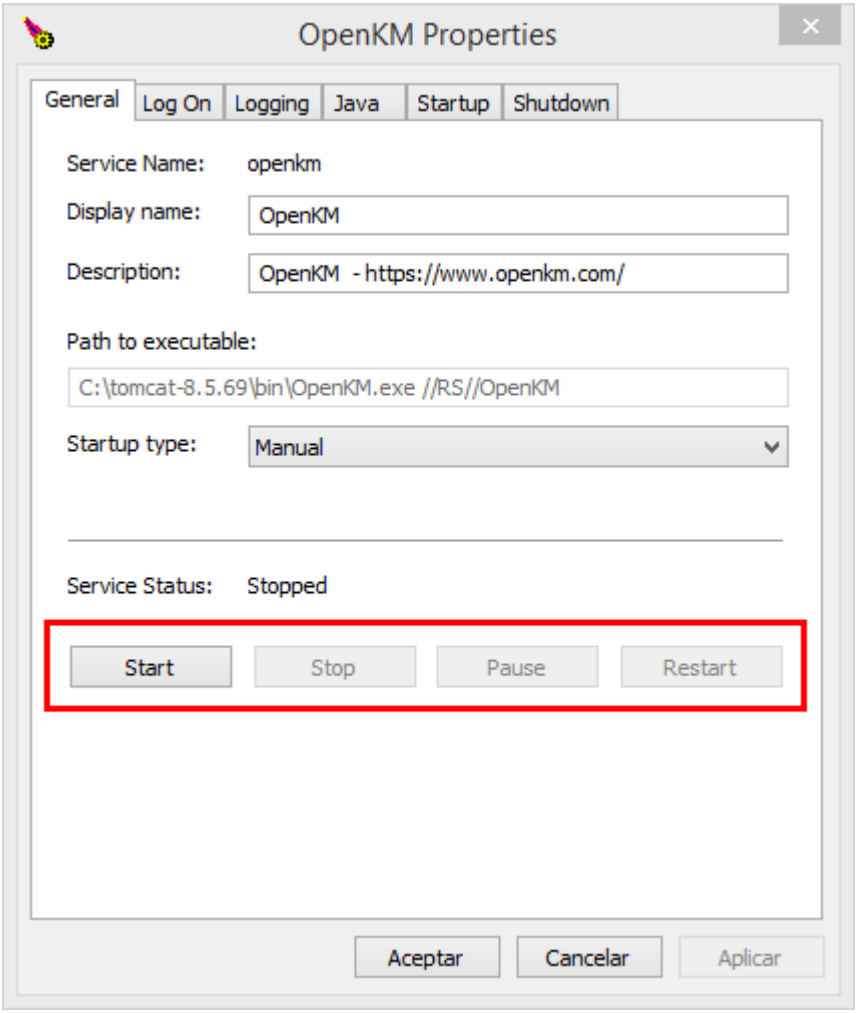
Stop the service:

```
$ net stop OpenKM
```

From the Windows service panel



From the openkmw tool



Configuring environment variables

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

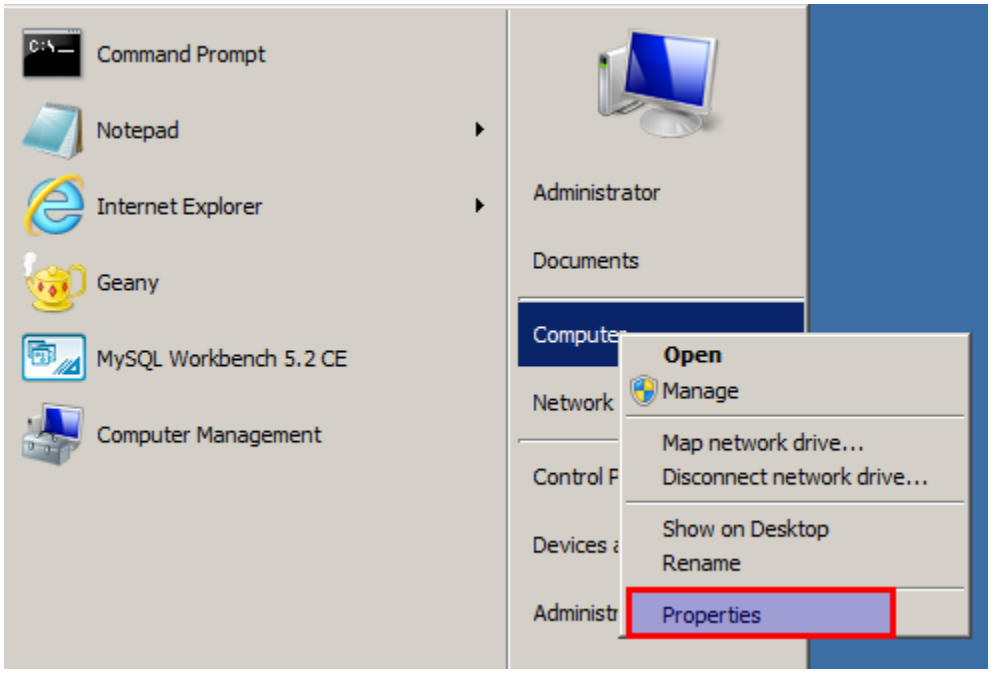
```
setx PATH "%PATH%;C:\tomcat-9.0.76\bin" /M
```

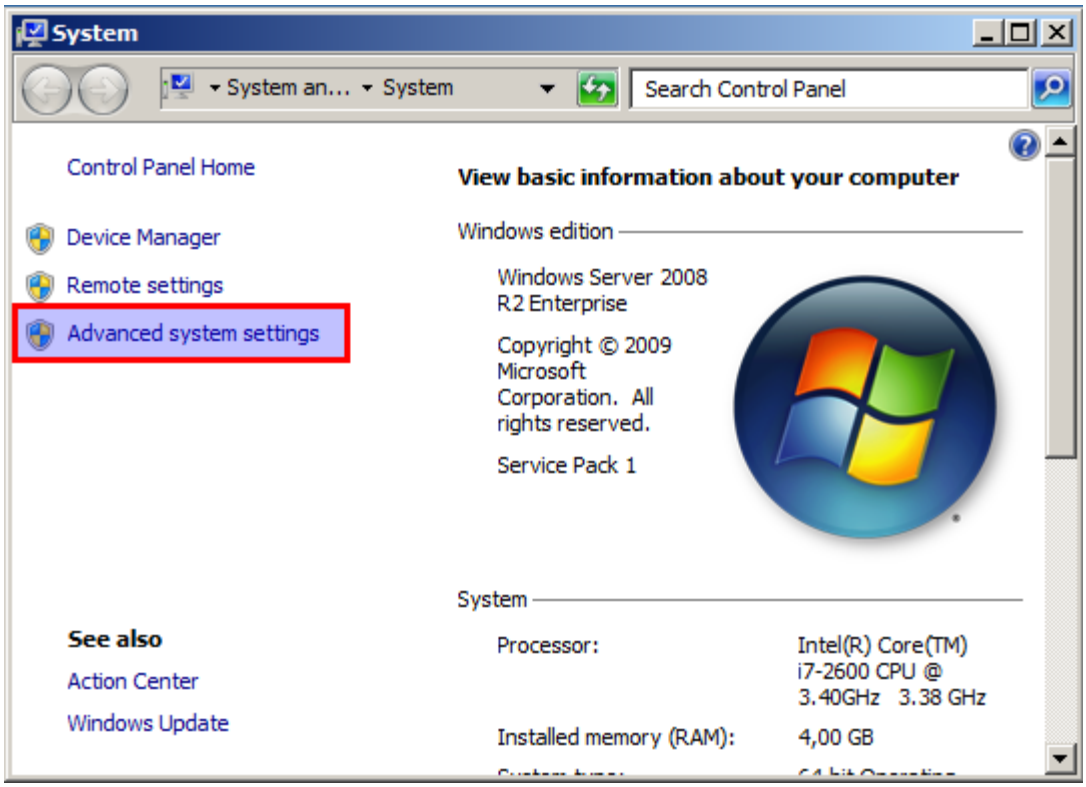
These operations can also be done from Windows environment variables:

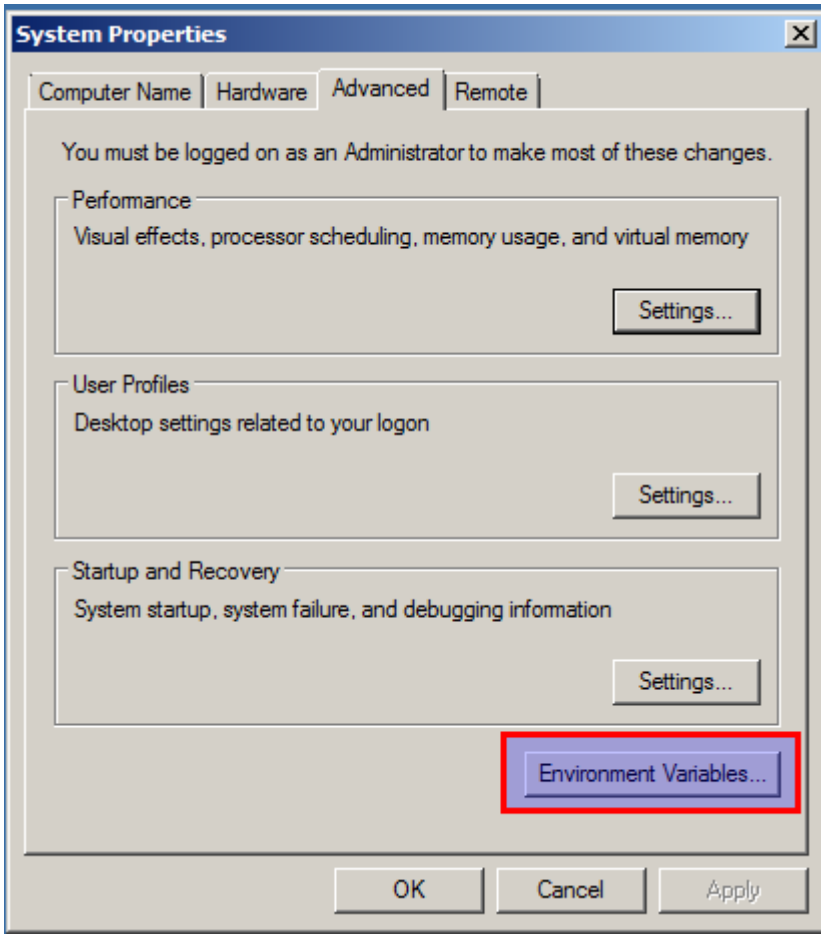


The following screens could be quite different depending on your Windows OS version.

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


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	The 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-bit builds need more hardware resources than 8-bit builds. 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 numbering may differ due to ImageMagick version upgrades).
GhostScript	No	The 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 **Extras.zip** file available 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 the 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 requires credentials to use it, so if you want to access it, you should 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, Red Hat, 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 Java 8 is installed, you can download the installer from http://download.openkm.com/.



The installer will perform the installation in the folder in which it runs, so before running, move the OKMInstaller.jar to the location where OpenKM should be installed. On Linux, it also changes the file's owner to a user called "openkm", so if you haven't created such a user, please create 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:

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

```
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, the default value is chosen. If you want a different one, type the value. In the end, the installer 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 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

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

The installer supports these databases:

- MariaDB (mariadb)
- MySQL (MySQL)
- PostgreSQL (postgresql)
- 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 the 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 for the OpenKM container to use.

After that, point your browser to http://localhost:8080/. The default user and password are "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 volume, 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 to use 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=u
    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 explains how to deploy OpenKM on a pristine Apache Tomcat.

Preliminaries

Take a look at the installation of the whole system for 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 where you want to install it.

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 distribution 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 in 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 content 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 the 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. If not, please check the documentation or create a support ticket.

First-time application startup:

```
$ start openkm
```



Use mTail or an equivalent program 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 the user **okmAdmin** and the 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 an SQL database. Several databases are supported, but they must be configured to work correctly.

Configure the database

- Configuring MySQL and MariaDB
- Configuring Oracle
- Configuring PostgreSQL
- Configuring SQL Server
- Configuring the 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. It was forked due to concerns about 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 |

If another default engine is in use, there are two options:

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

Change default engine

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

Then, under [mysqld], add

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

After changing the default in 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, you need to restart the database server.

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 restarting 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, the default values from my.ini will be used):

```
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';
```

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

Configure your openkm.properties



This configuration property should be set before creating the database. Once the database has been initialized, do not modify it, because doing so 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 to match the timezone configured in your system, you can first see the mappings for both MySQL and MariaDB like this:

```
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.datasource.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.



Configure the values of the attributes named:

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

It can only use the resource called JDBC/OpenKMDS.

Configure application login

Edit the file $TOMCAT_HOME/openkm.properties

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

Run application

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

At 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 on Windows.

Ensure your MySQL\bin folder is in your PATH environment variable.

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

If you have a 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 are 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
```

Case-insensitive node names

By default, the database is case sensitive. However, by executing:

```
ALTER TABLE OKM_NODE_BASE MODIFY NBS_NAME varchar(256) COLLATE utf8mb3_general_ci;
```

*the **NBS_NAME** column is modified so that node names are treated in a **case-insensitive** manner. This collation also ignores accent differences, so names with and without accents are considered equivalent.*

As a consequence, it will not be possible to have two documents such as " Más temas.txt" and "mas temas.txt", since for the table index they are interpreted as the same string and the uniqueness constraint will fail.

If this happens, the system may throw an error like:

```
ERROR 1062 (23000): Duplicate entry '483ab139-9c20-44a9-8534-ed124a8f37ba-NULL' for key
'IDX_NODE_BASE_PARNAM'
```

This means both values are considered identical and the unique index detects them as duplicates.

To identify problematic nodes, you can run the following query:

```
SELECT NBS_PARENT, NBS_NAME, COUNT(*) FROM OKM_NODE_BASE
GROUP BY NBS_PARENT, LOWER(NBS_NAME) HAVING COUNT(*) > 1;
```

Any duplicates found should be renamed so they stop being treated as equal.

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
```

If the file is in UTF-16 format, convert it to UTF-8. For example, use 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 should be read before changing the MySQL configuration. The documentation is based on experience shared by OpenKM server administrators.

Useful links:

- *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 @2.93 GHz*
- *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 the MySQL server's InnoDB engine with the default configuration parameters in the 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***



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

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

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

Considerations *The repository is 1 GB, 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****. On an x86 MySQL server, you cannot allocate more than 2 GB of RAM.*

Information collected from OpenKM forum posts.

Sample two

Server information

- *CPU: Intel Xeon E5606 @2.13GHz*
- *RAM: 4GB*
- *System: Windows Server 2008, 64-bit*
- *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*

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

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, it will be possible to increase the MySQL buffer and probably achieve better performance.

Information collected from OpenKM Forum post.

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

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 sufficient permissions for the new folder /new_path/mysql_data.

Optionally modify the path of the old MySQL data store folder:

```
$ mv /var/lib/mysql /var/lib/mysql_old
```

*Edit the **datadir** property in 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 old data store folder path to the new one:

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

to

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

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

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

Change the old data store folder path to the new one:

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

```
/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 sufficient permissions for the new folder /new_path/mysql_data.

Optionally modify the path of the old MySQL data store folder:

```
$ mv /var/lib/mysql /var/lib/mysql_old
```

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

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

Change the old data store folder path to the new one:

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

Optionally modify the path of the old MySQL data store folder:

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

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

Change the old data store folder path to the new one:

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

to

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

Start MySQL service:

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

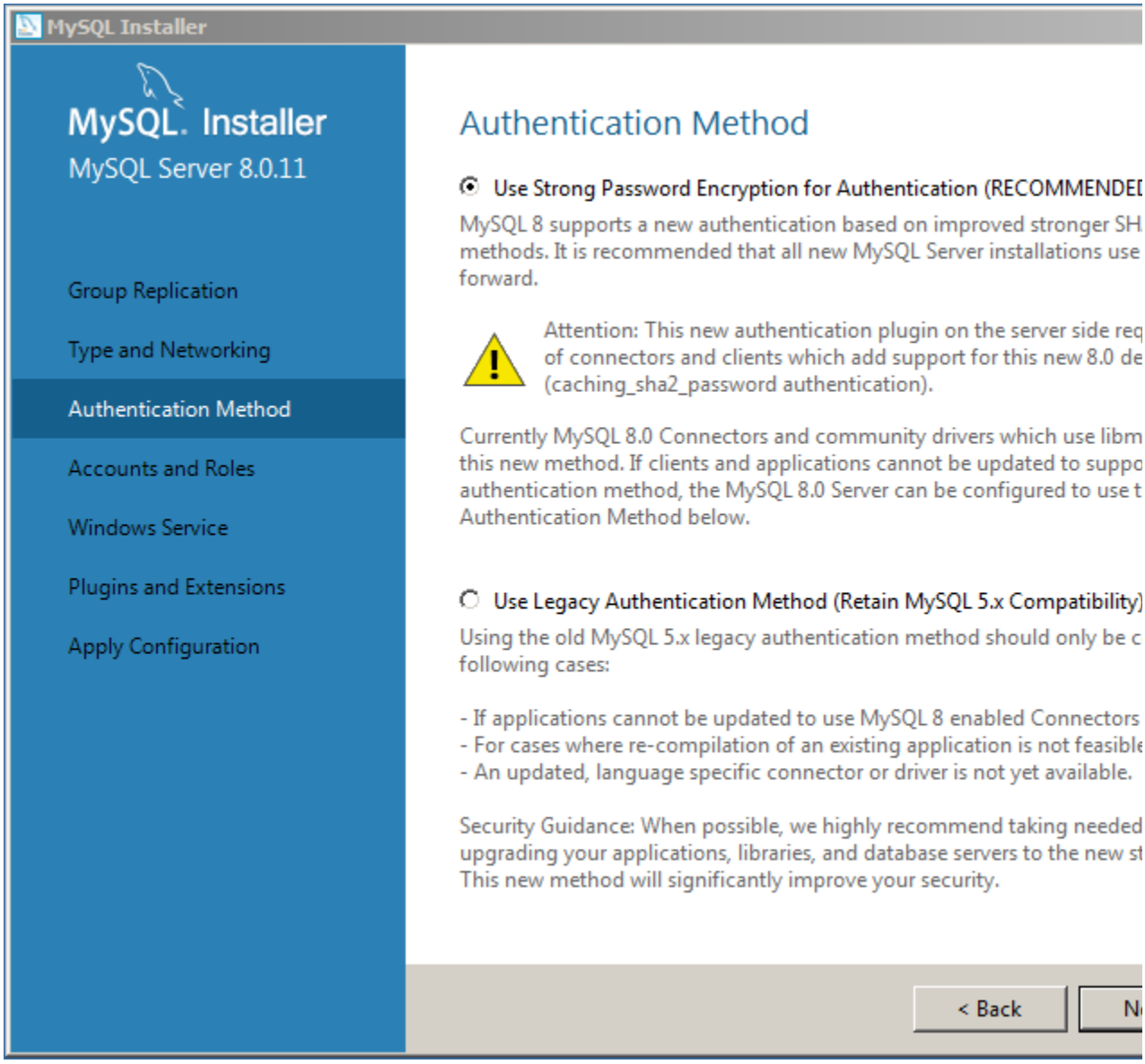
MySQL 8 troubleshooting

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 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 documentation and apply changes to your connection settings or to your server.xml to connect to the database. Usually, in this scenario, you will get an error like the one 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*
>
> *On Windows, it 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 shows 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 to connect to MySQL Server 8; you should make the following changes:

- *Replace the MySQL-connector-java-5.1.38-bin.jar in 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 in your $TOMCAT_HOME/conf/server.xml; they should be something like this:*

> ⚠ *In the latest version of the MySQL JDBC driver, the name of the class **com.mysql.jdbc.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 shows the error below or a similar one:

```
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 has an 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
```

Problem connecting to your database

If you get an error like this:

```
Caused by: com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException: Public Key Retrieval is not
```

You need to modify your file ***conf/server.xml*** *and add the following to the JDBC URL:* ***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;autoReconnect=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 creating the database. Once the database has been initialized, do not modify it because doing so can damage your installation.

If your OpenKM installation has already been configured with another database, you cannot switch databases by changing this property.



Choose the dialect that best fits 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*
```

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

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

Sample using a service name instead of SID:



The main difference between SID and service-name connections is that 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 host and port).*



If you encounter this error when starting 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 the application

During 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****.*

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 host 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

During 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 you with this task.

PostgreSQL 9.6 and later

*PostgreSQL versions 9.6 and later have **changed some tables and columns in the schema**. Check that you are using the correct JDBC driver.*

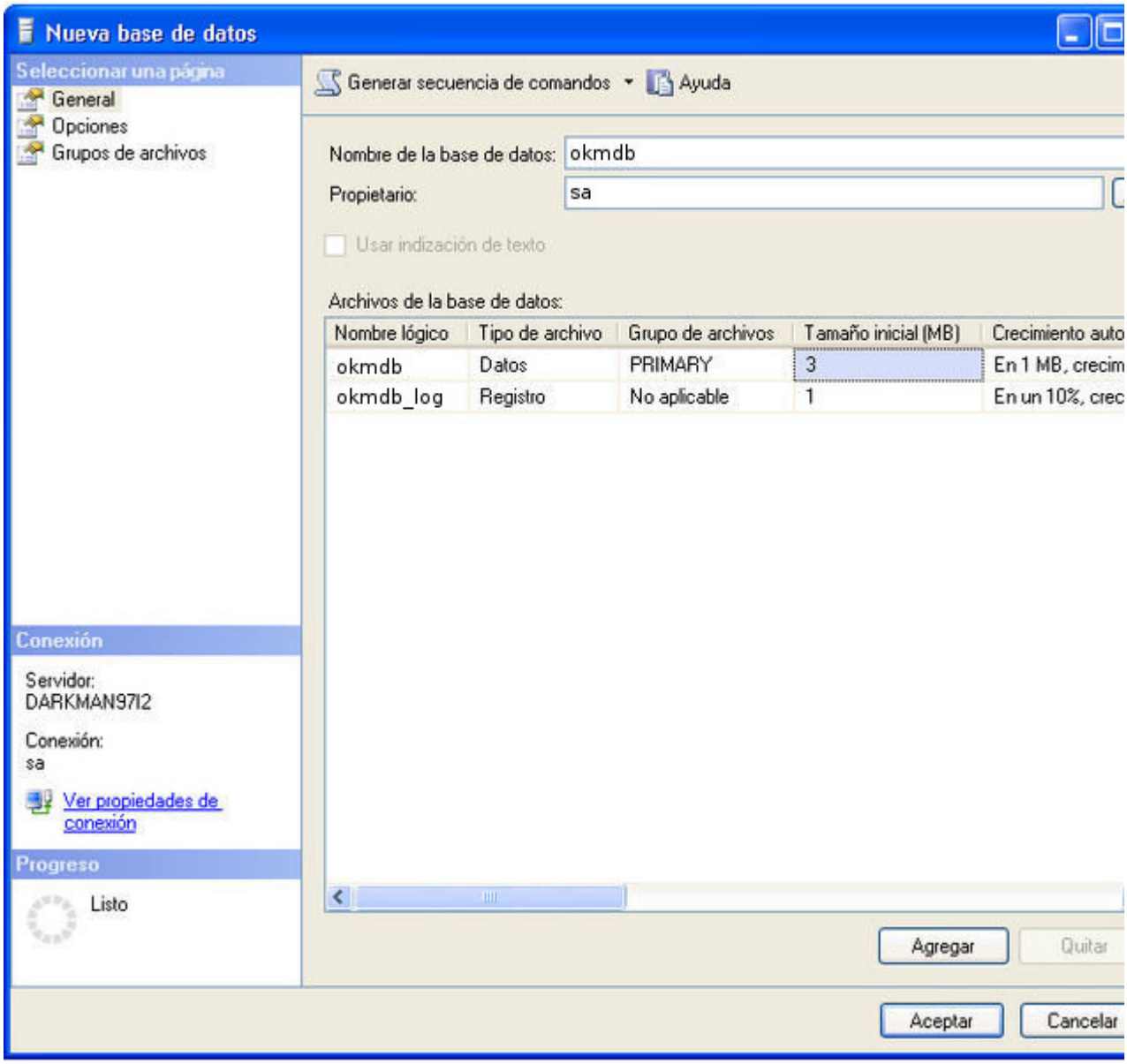
Read this related issue for more information: PostgreSQL - ERROR o.h.tool.hbm2ddl.SchemaUpdate - could not complete schema update

Configuring SQL Server

Database creation

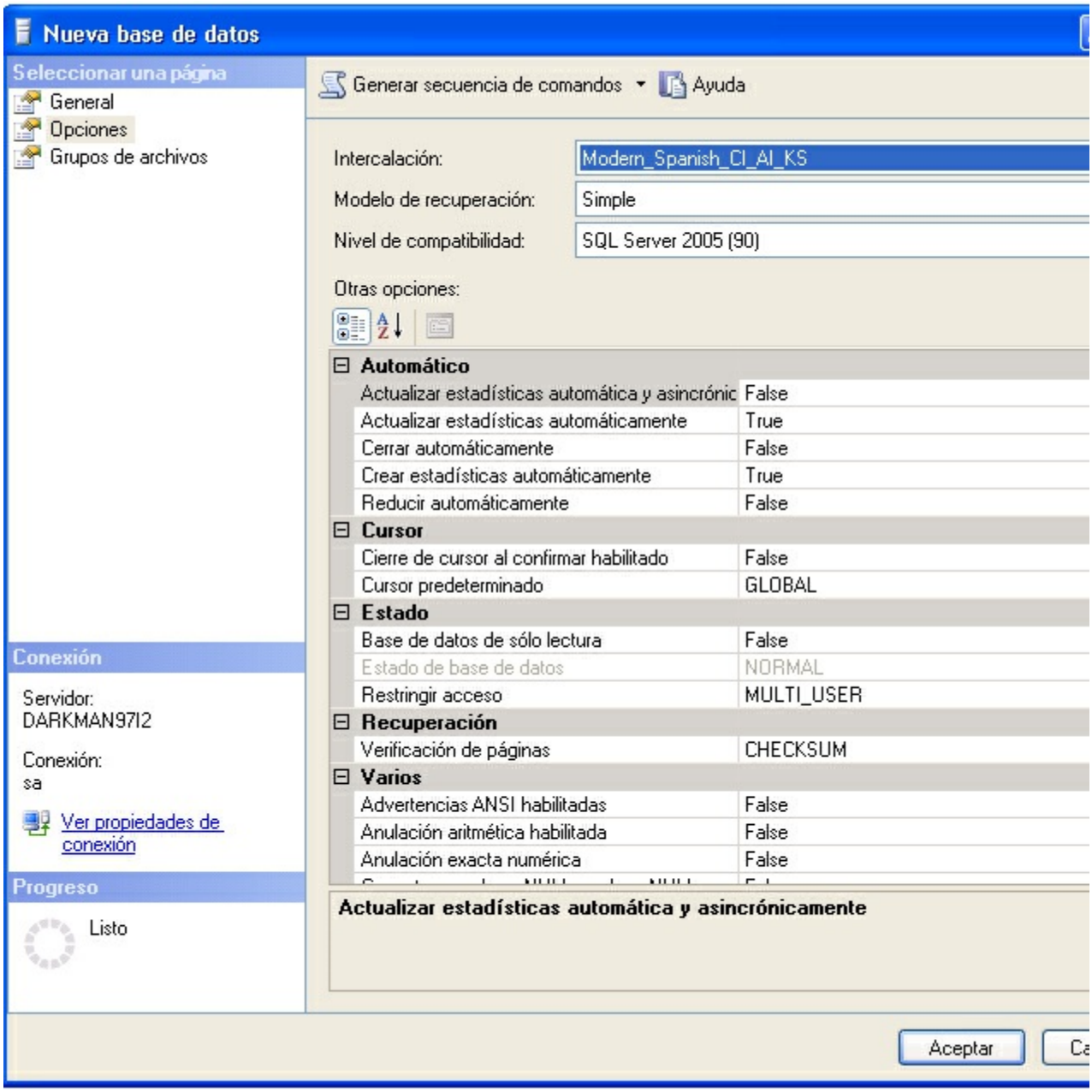
Create a database and user

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



Select the appropriate database collation.

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 correct OpenKM functionality, the database should be selected as **case-sensitive**.*

For more info, please check Collation and Unicode support.

Configure your openkm.properties

This configuration property should be set before creating the database. Once the database has been initialized, do not modify it because it can damage your installation.

If your OpenKM installation has already been configured with another database, you cannot switch to another

database by changing this property.



Choose the Hibernate dialect that best fits your current SQL Server 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 may also need to set this compatibility level from 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 host and port).*

Configure application login

Edit the file $TOMCAT_HOME/openkm.xml

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

Run application

During 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 the TOMCAT_HOME/bin folder.*

More info at:

- *Microsoft SQL Server documentation*

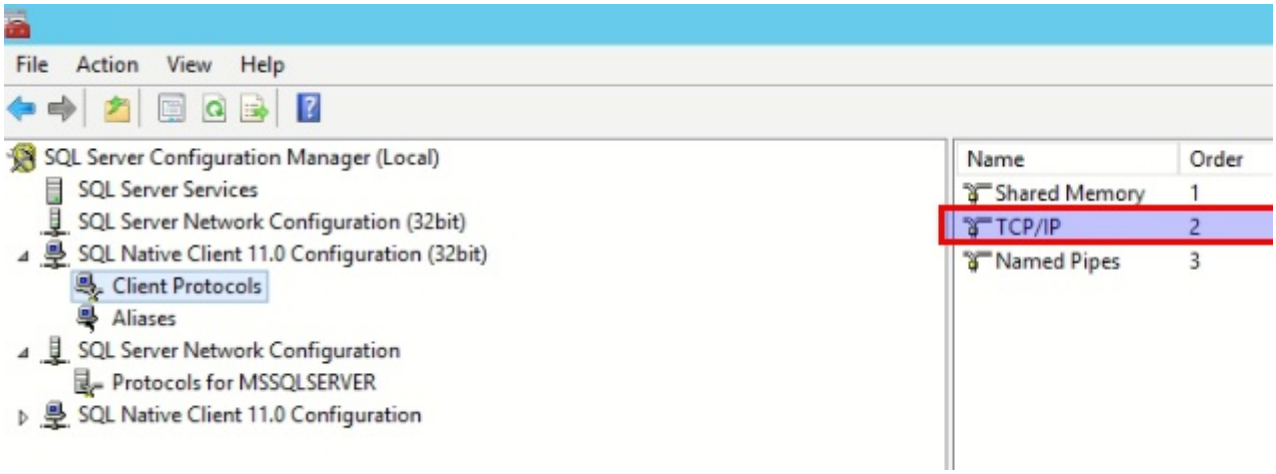
Troubleshooting

Can't connect to SQL Server

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



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



You configure the TCP/IP protocol in this section. Here you can see whether the 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 whereby, after some hours, the connection is closed (it might be an extra configuration parameter in the latest SQL Server editions that is enabled by default or similar). There is a new configuration attribute named ***autoReconnect*** *that solves this.*

```
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 log in to OpenKM

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



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

Unable to connect via TCP/IP to localhost SQL, port 1433, when using an 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 uses 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 by 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 the latest version of the 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;
```

Alter SQL Server compatibility level

To set SQL Server compatibility level to 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 uses 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 by setting your own values in the $TOMCAT_HOME/openkm.properties file.

Enable debugging

```
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 up-to-date 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 other information, folders, document structure, metadata, etc., are stored in a database.*

In that case:

1. *First, back up your database (make a SQL dump).*
2. *Then, back up 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. Note that, from OpenKM version 8.1.x, the **Lucene indexes path location** on 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. However, we do not recommend this method of backing up; we suggest incremental backup solutions for medium to large repositories.

Linux detailed steps

You can follow these steps to back up and restore if you have OpenKM configured with MySQL or MariaDB. For other databases, check the corresponding commands.

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 deleting 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 created previously. On Ubuntu, if using MySQL or MariaDB, 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 statements:

```
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 perform a repository check. See *Repository checker*.

Windows detailed steps

You can follow these steps to back up and restore if you have OpenKM configured with MySQL or MariaDB. For other databases, check the corresponding commands.

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 deleting 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 created 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 statements 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 perform 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 targets that are not fully trusted.

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 it manually. Installing it is as simple as:

> *On the official Borg releases page, 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 a 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 backup, all 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 want to unmount the folder:

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

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

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-efficiently over a pipe, like rsync. Thus, you can securely use rdiff-backup and ssh to back up a hard drive 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, the following default values are assumed:

- ***$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.

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

Explanation of configuration parameters:

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

OPENKM_HOME	*The OpenKM home folder.*
TOMCAT_HOME	*The Tomcat home folder.* *Usually in the OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually on the same server.*
BACKUP_DIR	*Backup destination. In the case of a remote server, the format 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 to increase the parameter value:*

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

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

Do a remote backup

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

> *The backup@server indicates the remote "server" accessed by the 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, you could modify only 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 a 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
```

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

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 a 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 one 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 a 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 **/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

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

- *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 on Debian/Ubuntu as simply 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 is not in the CentOS/Red Hat default repositories, so you need to install it 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-C
```

Command line basics

To make a backup, you can run:

```
$ 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-up 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 relative 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

These scripts assume the following default values:

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

Sample script



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



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

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

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

Explanation of configuration parameters:

Parameter	*Description*
HOST	*The server host name.*
DATABASE_PASS	*The database password.* *The database user used to perform the backup is always the root user.*
OPENKM_DB	*The application database name.* *It is usually named "okmdb".*
OPENKM_HOME	*The OpenKM home folder.*
TOMCAT_HOME	*The Tomcat home folder.* *Usually inside the OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually on 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*"
```

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

```
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 indicates that backups older than one month will be removed. Changing to "**3M**" or "**1Y**" indicates backups older than 3 months or 1 year, etc.*

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

Perform the remote backup

*Modify the **## Config 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 the **postfix** service on your server.

Troubleshooting deleting old backups

You have configured Duplicity to make a full backup every 7 days and to delete backups older than 14 days to save space. What happens 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 records incremental changes. Those files won't be deleted because, even though a backup may be older than 7 days, there are incremental backups that are newer than 7 days.

So, after two 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 those 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 one 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 **/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, the following default values are assumed:

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

Sample script



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



The script below performs a backup to a USB drive and the application's database is MySQL.

The 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 variable configuration under **## BEGIN CONFIG ##** section.*
- *Checking for the root user under **# Check root** section.*
- *Deleting older database dump under **# 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 **# Create backup** section.*
- *Start application (optional section) under **# Start tomcat** section.*
- *Umount disk (optional section) under **# Umount disk** section.*

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

Explanation of configuration parameters:

Parameter	*Description*
HOST	*The server hostname.*
DATABASE_PASS	*The database password.* *The database user that performs the backup is always the root user.*
OPENKM_DB	*The application's database name.* *It is usually named "okmdb".*
OPENKM_HOME	*The OpenKM home folder.*
TOMCAT_HOME	*The Tomcat home folder.* *Usually inside the OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually on 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 the "**postfix**" service on your server.*

Rsync and FTP

*rsync needs 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 when using this method.

Install curlftpfs and rsync:

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

Create mount point:

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

Mount remote directory:

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

Create local directory:

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

Rsync the mounted directory to the local directory:

```
$ 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 information.*



*Inside **/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

- *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

These scripts assume the following 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 performed by the root user.



The script below backs up to a USB drive, 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 variable configuration under the **## BEGIN CONFIG ##** section.*
- *Check for the root user under the **# Check root** section.*
- *Deleting older database backups under the **# Delete older local database backup** section.*
- *Stop the application (optional section) under **# Stop tomcat** section.*
- *Clean logs (optional section) under the **# Clean logs** section.*
- *Back up the database to the filesystem under the **# Backup database** section.*
- *Back up the repository and database dump to the backup destination under the **# Backup and purge old backups** section.*
- *Start the application (optional section) under the **# Start tomcat** section.*

Explanation of configuration parameters:

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

Parameter	*Description*
HOST	*The server host name.*
NOW	*Current date in the format 'dd-MM-yyyy'.*
DATABASE_PASS	*The database password.* *The database user used to perform the backup is always the root user.*
OPENKM_DB	*The application database name.* *It is usually named "okmdb".*
OPENKM_HOME	*The OpenKM home folder.*
TOMCAT_HOME	*The Tomcat home folder.* *Usually in the OPENKM_HOME folder.*
DATABASE_EXP	*The database dump folder.* *Usually on 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 ##
```

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

```
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 on your server.*

Cluster configuration

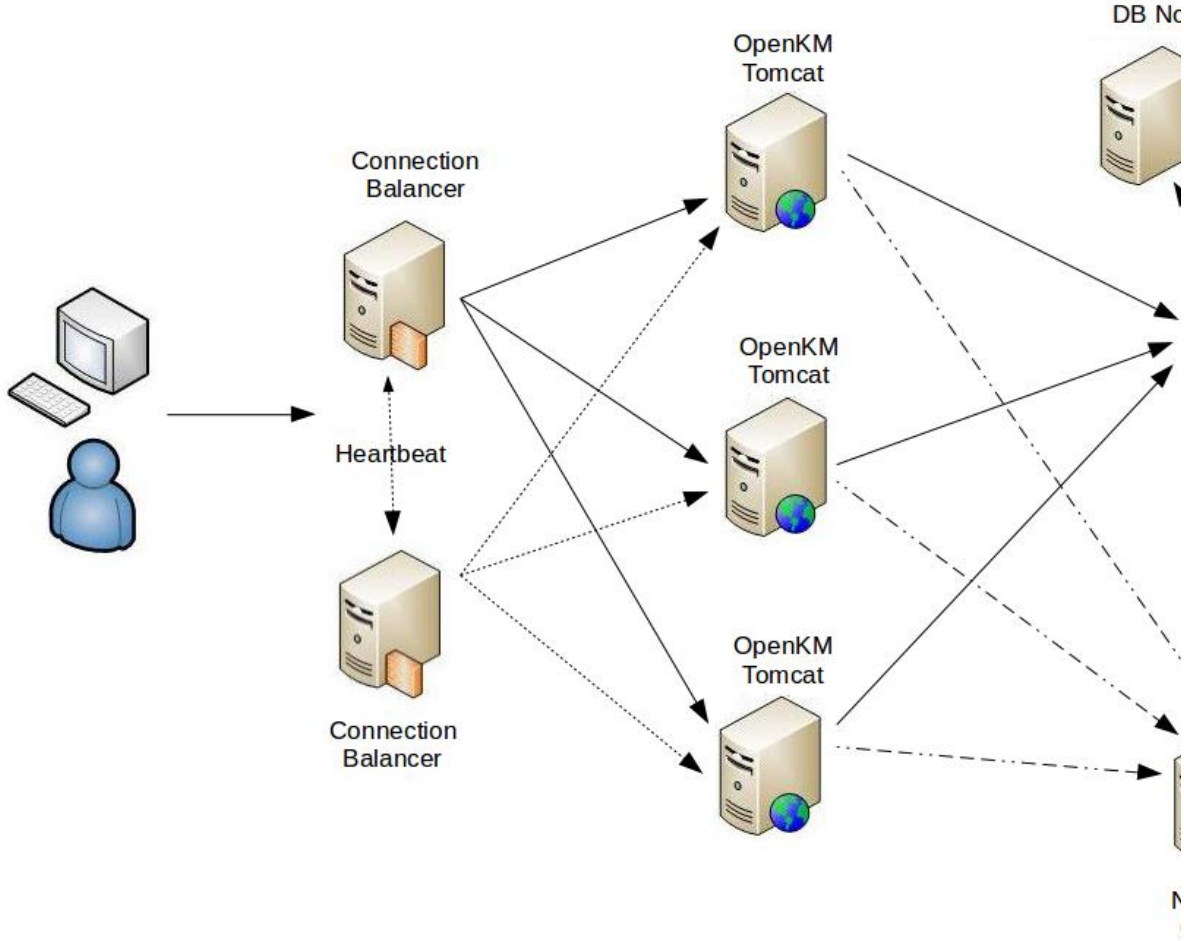
The updated cluster architecture includes enhancements over the previous one and fixes specific bugs found in earlier versions:

- ***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 architecture provides load balancing 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 user requests to these OpenKM servers. HAProxy can also be configured to detect which OpenKM server is down and avoid sending requests until it is up again.

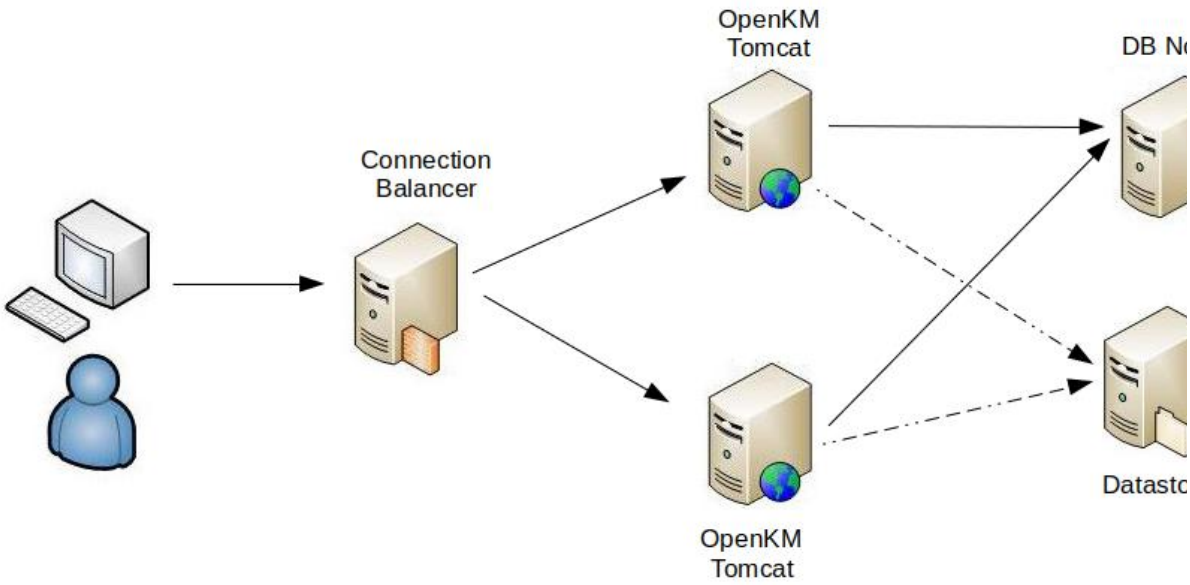
Optionally you can configure several replicated database servers (also handled by another HAProxy), which prevent the service from being interrupted in case a database node goes down. By default, OpenKM stores the documents on disk to maximize performance, so you need these files replicated across your OpenKM servers. An option is GlusterFS, a software solution to create distributed file systems, but NFS is quite good in most situations and easier to set up on 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 on 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. Use only one database for all instances. This resource is configured in the **TOMCAT_HOME/openkm.properties** file.*

If you look into the TOMCAT_HOME/repository directory, you can see the following:

- ***cache**: These are generated files used primarily for preview. If deleted, they will be recreated.*
- ***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**: This is where Lucene stores the indexes used for document and metadata search. If deleted, you can re-index the repository from Administration > Utilities.*

*You can place **cache** and **datastore** folders on another server as shared resources, according to this information. On average, 80% of user actions are document reads, and only 20% are writes (create or update documents). The problematic folder 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 to users and will dispatch requests between the slave nodes. The most important part is the HAProxy configuration. This configuration format works with HAProxy v1.5

```
frontend LB
```

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

```
  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 requests will be balanced using the round-robin algorithm between the two OpenKM servers. If any OpenKM servers fail, the requests will only be forwarded to the working server. At http://proxy/stats (default user: admin, password: admin), you will be able to see the nodes' 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 that it can run crontab tasks among other administrative 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 common 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 via 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 run this command (you should run it every time you modify this file):

```
$ 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 in 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 will 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 address.

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 in 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 on another disk.

> *The best way to understand sharding 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 the current shard. Shards 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 it is dangerous and may cause data destruction.*

> *A use case of sharding is to have a fast medium like an SSD (which is also expensive) in the main shard, and other slower and cheaper HDDs in the second shard. Documents will be moved from the fast main shard to the slower second one by a process which can identify less-accessed documents. This way the fast shard will be filled with frequently used documents, which will also 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 into 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 emails to OpenKM, otherwise you will have problems with document and email download, preview, and search.*

Sharding and Docker

*If you are using Docker to deploy OpenKM, we recommend externalizing the **openkm.properties** configuration file to be able to configure sharding.*

This is a sample Docker Compose definition for version 8.1.13 (change the 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 corresponding ***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

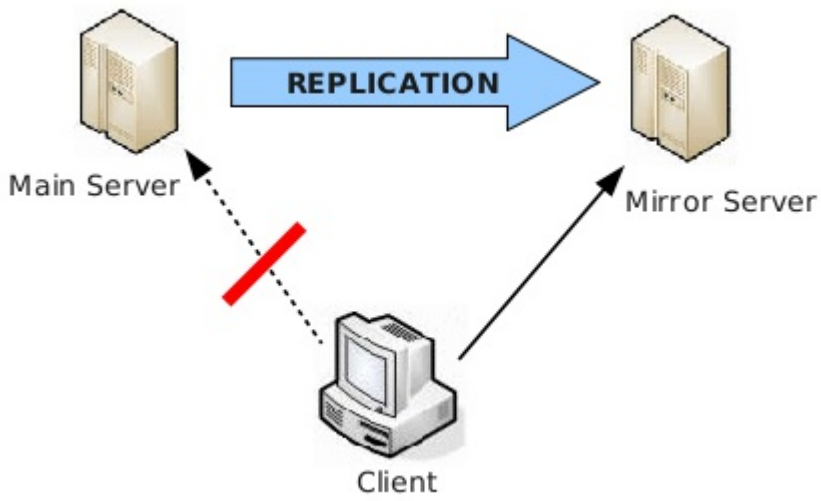
# Logging config
spring.output.ansi.enabled=always

# Extra
repository.datastore.home=/opt/tomcat/repository/${shard.id}/datastore
```

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 set up 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 requests.

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. A NAS solution improves 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. You can also deploy the database on a dedicated server (which should also be replicated). Some providers, like Oracle, offer specific database cluster solutions that 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:



In the sample, the main server is 192.168.1.101.

Both OpenKMs are installed in the OpenKM user's home (/home/openkm) for simplicity.

This script should be executed by the root user from the main server, and can be scheduled to run 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 performing this kind of action, we recommend creating a backup. Refer to the Backup and restore section.

Linux



In the following documentation, we will refer to the Installing on Ubuntu and Debian in some sections because they are the same actions.

Preliminaries

- *Ensure you **have enough disk space** on the new server.*
- *Check if you have **enough CPU power and RAM** on the new server.*
- ***Create the user** who is the **owner** of the OpenKM files ($TOMCAT_HOME folder). This user is usually named "**openkm**" (more info at Installing on Ubuntu and Debian), but a different user may have been chosen during installation.*
- *Update the OS.*

Stop OpenKM service

It is important that, while moving the data between the two servers, no changes are made on the old server. That's why it is a good practice to stop OpenKM.



*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

*The database connection is configured in the "**openkm.properties**" file (in the $TOMCAT_HOME folder). In the following 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 back up 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*** *on 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 a MySQL or 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 configured in another 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 keeping the same original folder path on the destination (otherwise you would need to perform extra configuration steps).*
- *Set the owner and permissions for the TOMCAT_HOME folder on 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 on the new server

- *Copy the file **/etc/init.d/openkm to the same location on the new server and register it 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 in the $TOMCAT_HOME/bin folder, so these were already 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 moved to the new server, you will only be able to access the administration user interface .

- ***If the new server has less RAM than the old server, check*** *the* ***$TOMCAT_HOME/bin/setenv.sh*** *file and adjust it accordingly.*
- *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 the administration interface*** *at http://localhost:8080/openkm/management/ and* ***run a repository checker*** *(this action may take a long time to complete).*

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 on **configuring** and **tuning** OpenKM.*

The application contains many Configuration parameters, and you probably 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.

You might also be interested in:

- *Preview parameters*
- *User interface configuration parameters*

If you have any issues, see the Troubleshooting section.

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

Antivirus configuration

OpenKM can be integrated with antivirus software that can execute an 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 in the default repositories, so you must install it from a non-default repository.

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
```

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

```
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.*



Initially, 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)*

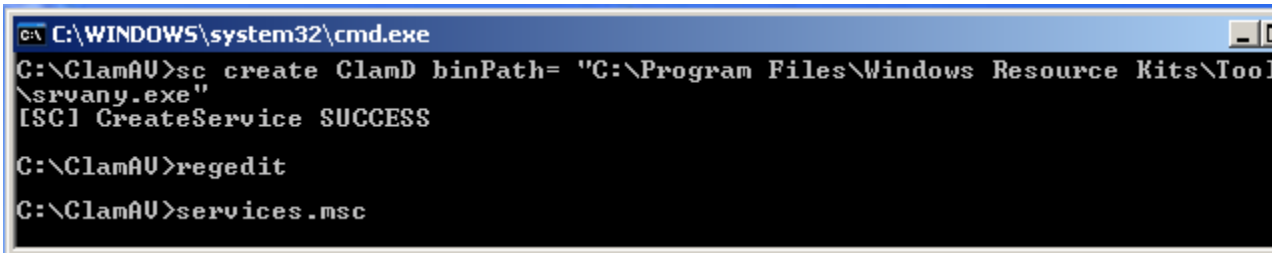
In case you need to create a freshclam.conf file, use this as a base:

```
# 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 set 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***

- *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 Advanced 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 with the actions performed by users on all the nodes. This table can easily contain several millions of rows, which can affect the performance of the database. You should be interested only in the most recent activity log data (for example, the last year) and keep the other data outside the database. Therefore, it is good practice to perform a cleaning process: export the oldest activity log data and then delete it from the database.



By controlling the size of the OKM_ACTIVITY log table, you can improve your database performance and reduce both the size of your database backups and the time required to create them.

By backing up OKM_ACTIVITY by range and then deleting it from the database, you keep your activity log information safe without affecting database performance.

You can always restore the OKM_ACTIVITY logs to another database for forensic analysis.

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** an 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:

```
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 the **postfix** service on your server.*

Windows

*Create a file named **clean_okm_activity.bat** in the 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** in the c:\backup folder:*

```
DELETE FROM OKM_ACTIVITY WHERE ACT_DATE < (NOW() - INTERVAL 365 DAY);
ANALYZE TABLE OKM_ACTIVITY;
```

You can use Windows Task Scheduler to schedule your task periodically.

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: a production MySQL server (A) and a backup MySQL server (B).

First of all, you must enable federated tables on the production server. Stop the MySQL production service and edit the MySQL configuration file (my.cnf or my.ini, depending on your OS). Go to the [mysqld] section and add the following:

```
[mysqld]
# To enable federated feature
federated
```

Start your production MySQL service and execute the following command from your preferred MySQL client tool:

```
SHOW CREATE TABLE okm_activity;
```

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

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 (note that we have **changed the table name**, replaced*

***ENGINE=innodb with ENGINE=FEDERATED**, and **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 a 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. Port 8080 may be blocked by a firewall; for this reason, it is recommended 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 the 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.

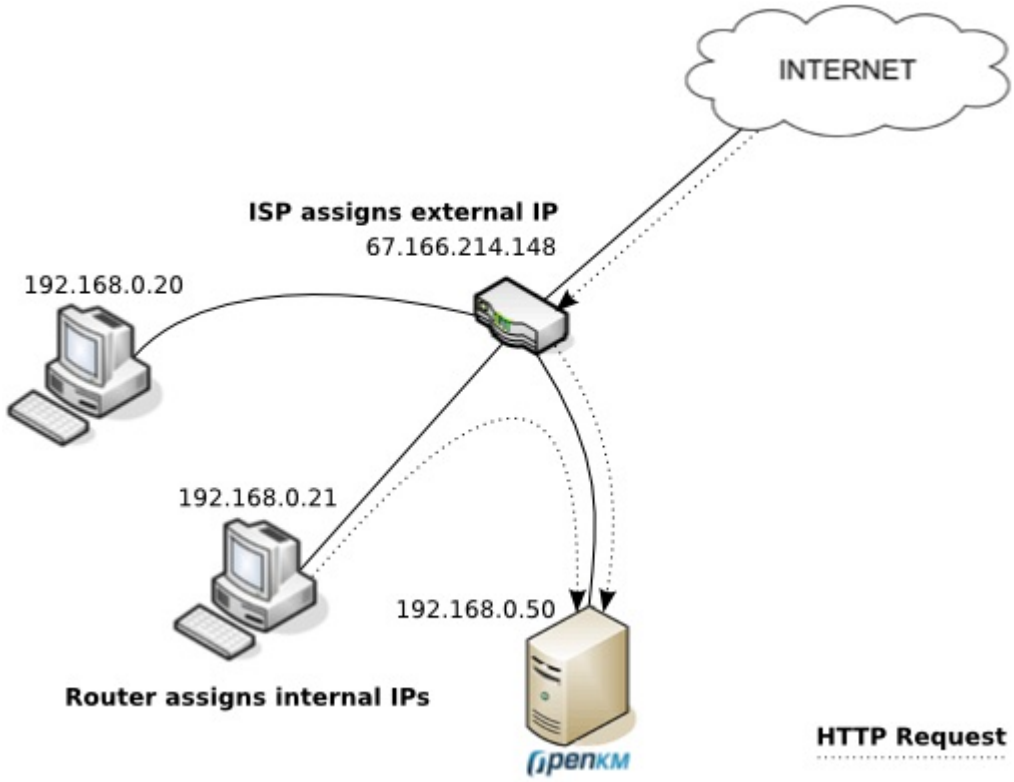
Internal IP vs external IP

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, which 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 as 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 the 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 the Internet. To resolve the Internet IP (67.166.214.148) your computer uses the public DNS. So, you need to configure a DNS server inside the LAN or modify every client's hosts file to resolve to the internal IP.*

Debian and Ubuntu

The first step is to install the required Apache software. On 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"
```

Editing apache2.conf is optional but sometimes necessary 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 the ***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 to your server IP or your domain.

The VirtualHost ServerName must be different from the ServerName in the main Apache configuration. Enable this site configuration:

```
$ cd /etc/apache2/sites-available/

$ sudo a2ensite openkm.conf
```

You have to explicitly enable proxy access by 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, you might be interested in removing the default Apache configuration:

```
$ a2dissite 000-default.conf
```

Check the configuration

Restart Apache:

```
$ sudo /etc/init.d/apache2 restart
```

Check access to your OpenKM installation at 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, 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 to your server IP or your domain.

Check the configuration

Restart Apache:

```
$ sudo /etc/init.d/httpd restart
```

If this command fails, try with:

```
$ sudo systemctl restart httpd
```

Check access to your OpenKM installation at 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 an error like **Permission denied: proxy: AJP: attempt to connect** appears in the log, solve it by executing the following command:*

```
$ /usr/sbin/setsebool httpd_can_network_connect 1
```

If it works, then make 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 the 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
CustomLog logs/openkm-access.log combined
</VirtualHost>
```

Change 192.168.10.230 to your server IP or your domain.

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 to your server IP or your domain.



Uncomment this configuration section:

```
# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
```

*Change the 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 these Apache modules:

```
$ 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)
```

It is possible that Tomcat is not accepting AJP connections. Please uncomment these lines in

$TOMCAT_HOME/conf/server.xml

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
```

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

```
<Connector protocol="AJP/1.3" address="127.0.0.1"
        port="8009" redirectPort="8443" secretRequired="false" />
```

*The **secretRequired** parameter is important, and if it's not present you need to add it.*

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 the Internet (for example https://issues.jboss.org/browse/JBAS-3861). As a result you can be infected by PerlBot. Also port 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 requests and forward them to the 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.

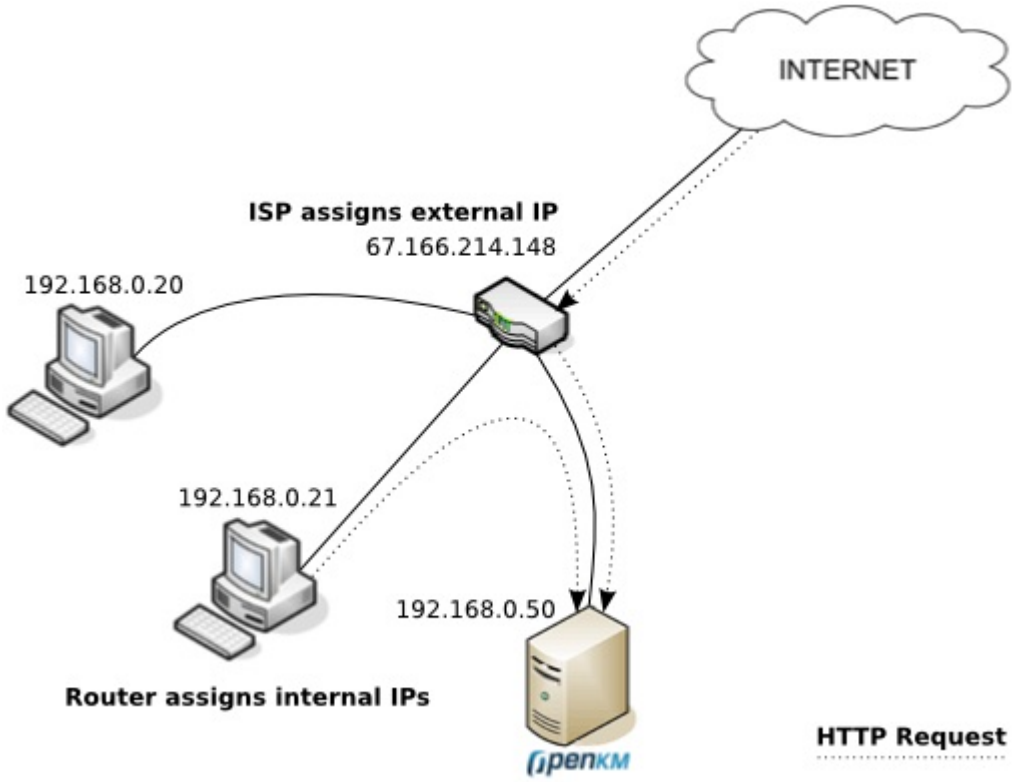
Internal IP vs external IP

Your OpenKM can be accessed from two different zones: Internet and LAN. This means that to access this server you need to use two IP addresses: 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 and 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 as 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 the Internet. To resolve the Internet IP (67.166.214.148) your computer uses the public DNS. So, you need to configure a local DNS server inside the LAN or modify every client's hosts file to resolve to the internal IP.*

Debian and Ubuntu

The first thing is to install the required Apache software. On 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"
```

Editing 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 the **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 a .pem file to .crt, use the command:

```
$ openssl x509 -in cacert.pem -out cacert.crt
```

You might also be interested in Configuring OpenSSL.

*Ensure port **443** is listed 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 to your server IP or your domain.

The VirtualHost ServerName must be different from the 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 this 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 access to your OpenKM installation at 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, you 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. If you are using Apache, you also need 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 it expires. It's a good practice to create a cron task to update it automatically:

```
@monthly /usr/bin/certbot renew
```

Also keep in mind that both ports 80 and 443 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 a self-signed certificate and not send it to a Certificate Authority. It is good practice to set a password on the private key, but in this case each time you restart the Apache service it will be required.

You might also be interested in Configuring OpenSSL.

During the process, two files will be generated 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 to your server IP or your domain.

Finally, you must modify ***SSLCertificateFile*** *and* ***SSLCertificateKeyFile*** *values in the 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 access to your OpenKM installation at *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 the log shows an error like ***Permission denied: proxy: AJP: attempt to connect****, solve it by executing the following command:*

```
$ /usr/sbin/setsebool httpd_can_network_connect 1
```

If it works, then make 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 these Apache modules:

```
$ 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)
```

It is possible that Tomcat is not accepting AJP connections. Please uncomment these lines in

$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 important, and if it's not present you need to add it.*

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*
- *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 a result, it can be infected by PerlBot. Also, port 8080 may be blocked by a firewall. For these reasons, it 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 the 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. If you have a multitenant configuration, please use Configuring Apache HTTP Reverse-Proxy.

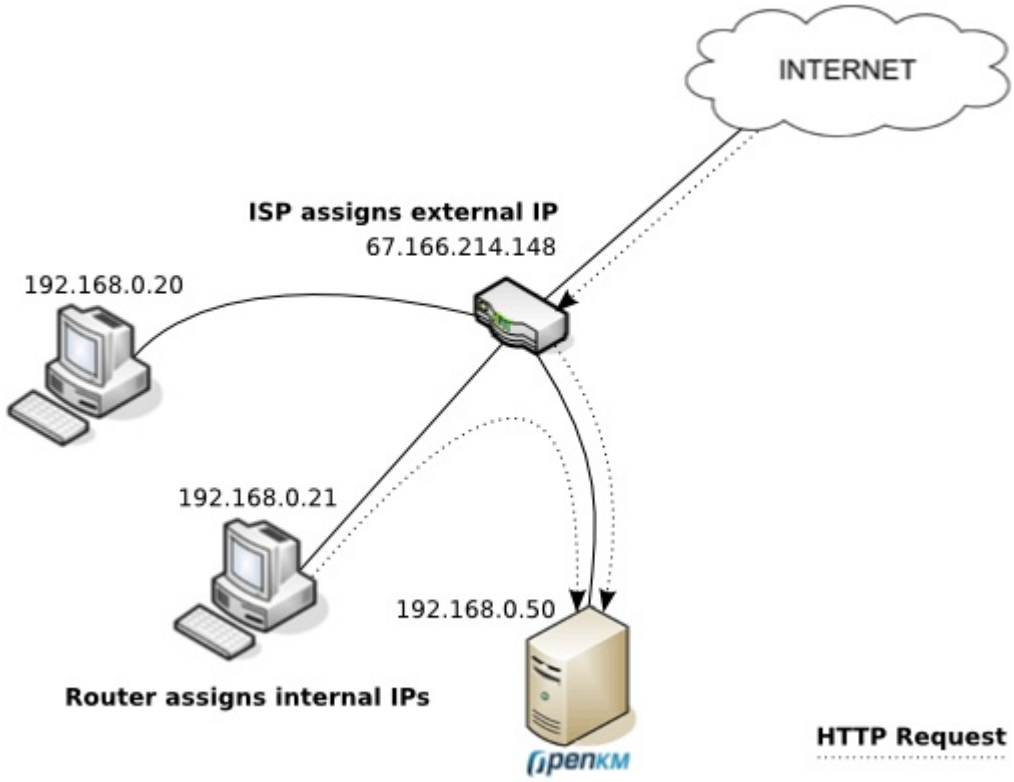
Internal IP vs external IP

Your OpenKM can be accessed from two different zones: Internet and LAN. This means that to access this server you need to use two IP addresses: external IP (Internet) and internal IP (LAN).

The internal IP address (also known as "local IP address") is the address which is assigned by your local network router and 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 as 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 the Internet. To resolve the Internet IP (67.166.214.148), your computer uses the public DNS. So, you need to configure a DNS server inside the LAN or modify every client's host file to resolve to the internal IP.*

Debian and Ubuntu

The first thing is to install the required Apache software. On 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 to your server IP or your domain name.

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 access to your OpenKM installation at 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 of 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 to your server IP or your domain name.

Check the configuration

Restart Nginx:

```
$ sudo /etc/init.d/nginx restart
```

Check access to your OpenKM installation at http://openkm.your-domain.com/.

If you can't access the 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.



If you see errors like this 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 commands related to SELinux:

Get all related HTTP configuration

```
$ getsebool -a | grep httpd
```

More information at [SELinux Booleans](#).

Troubleshooting

This is a different configuration which works better for multitenant configurations:

```
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 URLs

This page contains obsolete information and will be removed in the near future.

If a document contains links based on the old OpenKM version, those links will not work by default because the format has changed.

*For example, a link in version 6.4 may look like "**https://old.server.de/OpenKM/index.jsp?uuid=39448438-1f42-43fb-b33f-7b9ad5ac010c**"; to match version 7.1.x it should be converted to "**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 use the following 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 can obtain better search results by configuring a proper Lucene Analyzer.

*By default, OpenKM uses the **org.apache.lucene.analysis.standard.StandardAnalyzer**, which works fine with English and most languages, but you can get better search results by configuring a more specific analyzer for your 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 is available at the Lucene documentation site and the Guide to Lucene Analyzers.

If you want a non-accent-sensitive analyzer, you can try the AccentInsensitiveAnalyzer.

If you are working with oriental languages like Chinese or Japanese, you have several analyzers available. Read the Lucene documentation. You can also try ik-analyzer.

If you want only a whitespace-tokenized analyzer, you can try the WhitespaceAnalyzer.

If you have not configured the search analyzer before you start OpenKM for the first time, then Lucene indexes 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 the 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.backend.analyzer=org.apache.lucene.analysis.es.SpanishAnalyzer
```

If you want the accent-insensitive analyzer to ignore the distinction between accented and unaccented letters:

```
spring.jpa.properties.hibernate.search.backend.analyzer=com.openkm.search.lucene.analysis.AccentInsensitiveAna
```



- *The changes will take effect after restarting the application.*
- *Remember to rebuild the Lucene indexes after the application restarts.*

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 executables should already be 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 the bottom right, select the **JDBC** option, and paste the SQL insert into the box:*

```
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 works with local installations. It is the JavaScript CAD previewer.*
extension.cadviewer.autoxchange	***String***	*Path to the **autoxchange** utility. Used in thumbnail generation, preview, and PDF conversion.*
extension.cadviewer.linklist	***String***	*Path to the **linklist** utility. Extracts text and metadata from DWG documents.*
extension.cadviewer.dwglist	***String***	*Path to the **dwglist** utility. Gets XREFs from DWG documents. Also used in thumbnail generation, preview, and PDF conversion.*
extension.cadviewer.sibling.xrefs	***Boolean***	*When this parameter is set to **true** the AutoCAD files' references must all be in the same folder. If it is set to **false** the AutoCAD file references can be stored in any place in the taxonomy tree. In this case, keep in mind that you should avoid document name duplication. There is an automation called **CheckUniqueDocumentName** which can help you maintain this restriction.*

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

extension.cadviewer.debug	***Boolean***	*Set to **true** to view the commands executed during conversion in the log file. Only for debugging purposes.*
extension.cadviewer.remove.temp.files	***Boolean***	*Set to **false** to keep the temporary files created during conversion. Only for debugging purposes.*
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 corner where to place the QR code in the AutoCAD file.*
extension.cadviewer.qr.pos_y	***Integer***	*Percent Y from the lower-left corner where to place the QR code in the AutoCAD file.*
extension.cadviewer.qr.width	***Integer***	*QR code width as a percentage.*

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 as the executable, typically the **$TOMCAT_HOME/bin/cadviewer** folder.*
- ***dl_lic.key**: This is the license for the **dwglist** executable. This one should be placed in the same directory as the executable, typically the **$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 as the executable, typically the **$TOMCAT_HOME/bin/cadviewer** folder.*
- ***ll_lic.key**: This is the license for the **linklist** executable. This one should be placed in the same directory as the executable, typically the **$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 executing it from the command line. For example, this would perform a PDF conversion:

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

```
$ ./autoxchange -i=test.dwg -f=pdf
```

On Linux, ensure you have configured the LD_LIBRARY_PATH properly. On 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 a script which may help by setting the correct library path:

```
$ ./cad2svg test.dwg
```

The resulting "test.svg" file can be viewed with any browser and you will see if the license was set up correctly.

*To enable this extension, go to **Administration** > **Utilities** > **Database query**. At the bottom right, select the **JDBC** option, and paste the SQL insert into the box:*

```
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*



If you are using the old UI, you need to use an extension, but keep in mind this will be unsupported soon.

*To view external references for each AutoCAD document, you must set the value of the configuration key **extension.external.tab.document**. To do this, go to **Administration** > **Configuration** and set the appropriate value for this key:*

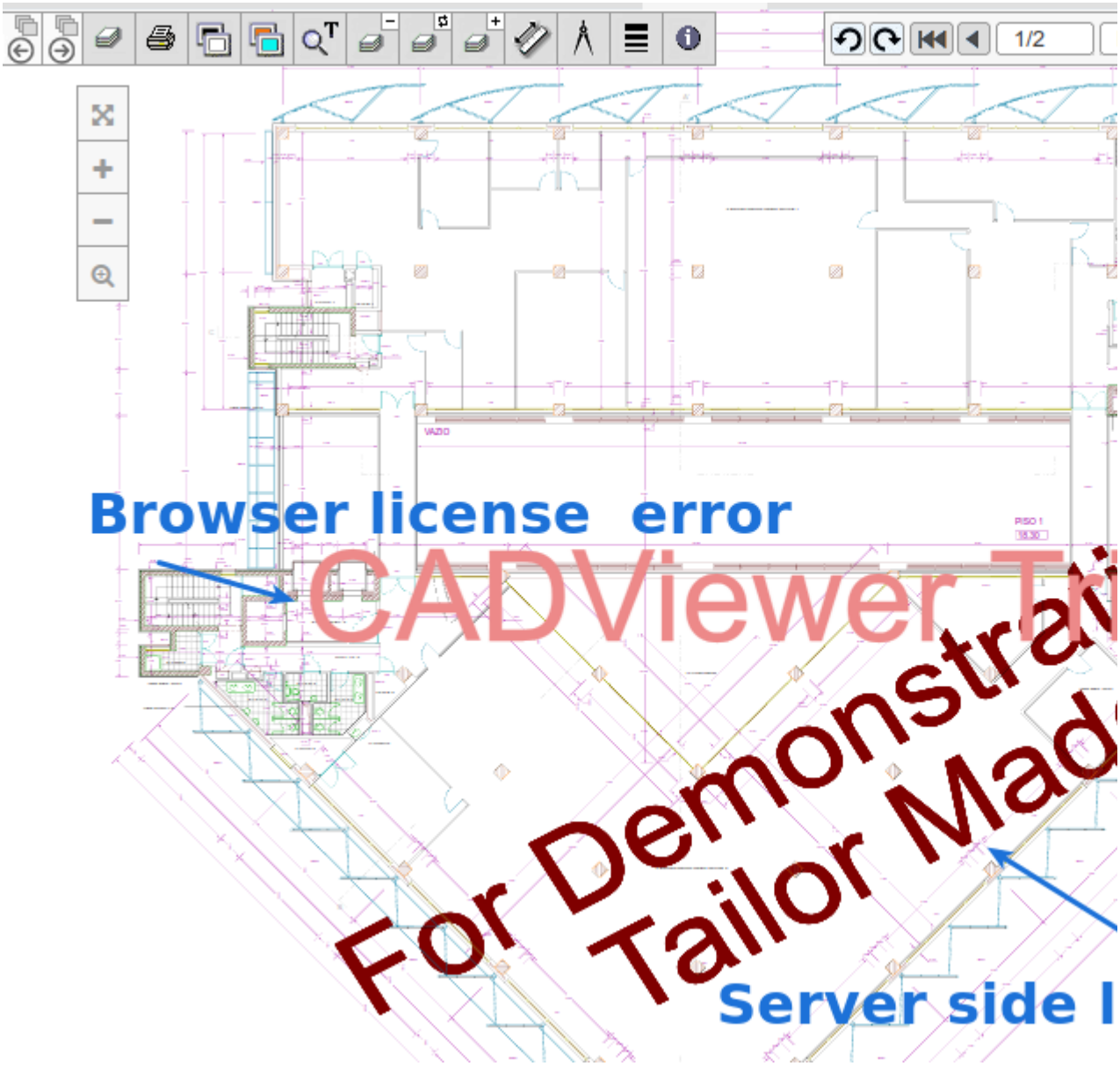
extension.external.tab.document = References;/openkm/frontend/XRefs

You can use this SQL statement 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 JavaScript previewer. The license set in ***extension.cadviewer.license.key*** *configuration property is incorrect.*

You can identify this issue by the "CADViewer Trial Version" watermark.

To resolve this issue, set the correct 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 on the server. The 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 correct ***axlic.key*** *license file to the OpenKM server. Once copied, go to* ***Administration*** *>* ***Utilities*** *>* ***Clean Conversion Cache*** *and clear the* ***SVG*** *cache.*

You can verify the license this way:

```
$ ax2023 -verify
```

This parameter is only available in recent AutoXchange versions.

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 the command line.

Tesseract

Check the list of supported languages at https://tesseract-ocr.github.io/tessdoc/Data-Files-in-different-versions.html

More information about installation on different OSes and Linux distros is available at https://tesseract-ocr.github.io/tessdoc/Installation.html

Tesseract is an open-source OCR engine adopted by Google. It works really well. OCR can natively read TIFF documents and has a high recognition rate with images of 300 dpi resolution that are 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 may want to install another language file.
	
	For example, to install the Spanish language pack:
	`$ 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	*On 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 may want to install another language file.* *For example, to install the Spanish language pack:* *$ yum install tesseract-langpack-spa* *You can check the installed languages this way:* *$ tesseract --list-langs* *On Red Hat, we recommend contacting Red Hat 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 the 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 well, a file named **text.txt** which 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** indicates that the Spanish language support must be used 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** indicates that the Spanish language support must be used while executing OCR.*

To see all Tesseract options, execute:

```
c:\Tesseract-ocr-3.0.2> tesseract.exe
```

*The parameter **${fileIn}** will be replaced internally by the image document to be processed.*

*The parameter **${fileOut}** will be replaced internally by a temporary text file.*

*You can set other additional parameters like the **-l spa** parameter in the example.*

Each parameter is separated by a single whitespace.

*Once configured, please go to **Administration** > **Utilities** > **Plugins**, choose **com.openkm.plugin.extractor.TextExtractor** from the select box 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. You 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 come in .zip or .oxt format. You can use both.

OCR rotate configuration

There's a configuration parameter that forces OCR on rotated documents.

The feature is useful, for example, for upside-down scanned pages, where the standard OCR direction will fail.

When the feature is enabled, the operation will be performed on all OCR extractions, which can have undesired effects on the 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 the 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 example was done with Ubuntu and Debian servers; for Red Hat and CentOS the path to Apache files is different.

*On Ubuntu and Debian the Apache path is **/etc/apache2**; on Red Hat and CentOS it is **/etc/httpd**.*

*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)
```

```
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 self-signed keys for our own Certification Authority (in the example they are valid for 10 years). We will need to specify certificate data and a password to encrypt, 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 permissions:

```
$ chmod 600 /opt/met-ca/private/ca.key
```

Create Apache certificate

Create a pair of keys. You will need to specify a password and certificate data. This will create the key file (server.key) and a certificate request (server.pem).

```
$ openssl req -new -sha1 -newkey rsa:2048 -nodes -keyout server.key -out server.pem
```

The CA should sign the certificate request. Copy server.pem to the requests directory and sign it.

```
$ openssl ca -config /etc/ssl/openssl.cnf -policy policy_anything -extensions ssl_server -out requests/server-signed.pem -infiles requests/server.pem
```

Apache expects PEM format; it is mandatory to 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 the SSL folder does not exist, create it). Change file permissions (server.key should be readable only by the root user; server.crt should be readable by everyone):

```
$ chmod 400 server.key
$ chmod 444 server.crt
```

If Apache is not listening on the SSL port, enable it by configuring /etc/apache2/ports.conf:

```
Listen 80
Listen 443
```

Enable Apache SSL support:

```
$ apt-get install libapache-mod-ssl
```

Tell Apache to load SSL modules:

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

```
$ a2enmod ssl
```

Configure website:

Link the default-ssl file to enable SSL:

```
$ ln -s /etc/apache2/sites-available/default-ssl /etc/apache2/sites-enabled/default-ssl
```

Edit /etc/apache2/sites-available/default-ssl (enable the 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 them with the CA, first create the pair of keys. You will need to specify a password and certificate data. This will create a key (user.key) and a certificate request (user.pem):

```
$ openssl req -new -sha1 -newkey rsa:2048 -nodes -keyout user.key -out user.pem
```

With the CA, you need to 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 the certificate to PKCS#12 to be recognized by the 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 in the user's browser.*

Certificate revocation

If the server has been compromised, the certificate should be revoked by the CA. To revoke a certificate, check the 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.

Create a list of revoked certificates (CRL)

```
$ openssl ca -config /etc/ssl/openssl.cnf -gencrl -crlexts crl_ext -md sha1 -out crl.pem
```

To send the list to the browser, it should be converted to DER format:

```
$ openssl crl -in crl.pem -out ca.crl -outform DER
```

It can now be sent to the browser.

Add CRL to Apache

Copy the CRL file to the Apache SSL folder:

```
$ cp /tmp/crl.pem /etc/apache2/ssl/ca.crl
```

Modify /etc/apache2/sites-available/default-ssl file and add SSLCARevocationFile:

```
SSLCARevocationFile /etc/apache2/ssl/ca.crl
```

Restart the 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 bound to 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 (the same as you would use to configure a 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 look 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 connection test fails, OpenKM won't start**:*

- *spring.mail.test-connection=true*

*If you get problems during Tomcat startup, you 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 means the user who appears in the "From" mail header.*

Testing configuration

Sometimes configuring mail can be complex and error-prone because there are many configuration parameters and many different mail servers. To ease this configuration, we have developed a utility called OKMMailChecker which helps you to test the mail configuration without the need to restart OpenKM, which takes time and makes it unavailable for users during these tests.

Usage

*This utility can be downloaded from https://download.openkm.com and should be placed in the OpenKM directory where the **openkm.properties** file 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, --attachment**: An optional file to be sent as an 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 Tomcat's memory utilization.

*Edit the file $TOMCAT_HOME/bin/setenv.sh and change the **JAVA_OPTS** parameter to increase the system memory managed by the JVM (Java Virtual Machine).*

OpenJDK 11

These are the default configuration file values. Tomcat will reserve 4096 MB (Xmx parameter), and the heap size in Java 11 is dynamically assigned by the JVM. For this configuration, your server needs at least 5.5 GB of RAM (It is good practice to leave 1 GB 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.ignoreunknownencoc
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 ports

By default, the Tomcat binding ports are 8005, 8080, and 8009.



If you have another Tomcat running on the same server, or another application like JBoss Application Server, these ports may already be in use. In this case, you should change the default ports.

Change Tomcat ports



The port change must be made while the Tomcat application is stopped.

Edit $TOMCAT_HOME/conf/server.xml and look for all Tomcat port properties.

Service	Default port	Example of new port
Shutdown 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 timeout

By default, the session timeout is set to 30 minutes.

To change the Tomcat session timeout, edit the $TOMCAT_HOME/conf/web.xml and change the values in 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 time zone

Tomcat uses the GMT time zone. If you want to change it, you should add something like the following to $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 time zone list.

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

Configuring openkm.xml



By default OpenKM does not use openkm.xml from the file system; it uses the openkm.xml file that is inside openkm.war

By default OpenKM comes with three authentication providers:

- *Superuser in memory.*
- *Database.*
- *Active Directory (LDAP).*

The configuration added to the ***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 an external openkm.xml file, you must change the configuration parameter named ***okm.authentication.config*** *in the* ***openkm.properties*** *file. The value of the parameter* ***okm.authentication.config*** *can 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 independent application to ensure security and stability.

Please take a look at JBPM Console for more information.

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

Configuring ONLYOFFICE

This guide helps 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 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****.*

WOPI must be enabled manually by editing this default.json and setting the "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 in **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 an error message like "**Error validating token IP where {} != {}**", or enter 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 it from the top main menu in the OpenKM user interface.*

*For information 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.

Sample URLs:

- *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 you 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.*

This guide is deprecated. Please contact support for more information.

Code installation

OpenKM has only been certified with Collabora Professional.

OpenKM configuration

Edit the following configuration parameters in OpenKM administration:

*If you are unsure about the Collabora IP, check the log file for an error message like " **Error validating token IP where {} != {}**", or enter the Docker container (**docker exec -it name_container bash**) and view 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 = 192.168.1.38
```

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 the Collabora Docker container.*
- ***online.office.ip**: Source IP used by Collabora to authenticate with OpenKM.*

Configuring safe trash purge

When a user deletes a node (document, folder, mail or record) from OpenKM, it goes to the user's 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 the trash is accidentally cleaned, the nodes can only be restored from a previous backup, usually by restoring a full database and repository backup, which is a tedious task.



The 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 an accidentally deleted node, then it will not be necessary to restore the entire application from backup.

*You can configure a directory in the server file system where an export of the trash will be placed, using the "**repository.purgatory.home**" configuration parameter. Each time before a node is permanently removed from the trash, the application exports it (using the Repository export feature) to the purgatory home folder, creating a subfolder structure "**YEAR/MONTH/DAY**" that can help the administrator 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***	*Sets the safe trash purge destination folder.* `/mnt/backup/trash`



It is good practice to back up the purgatory home folder, at least for a prudent period (1-2 months depending on your company policy).

Configuring the search highlighter

The search highlighter feature highlights text in the search view.

By default, the search highlighter is disabled. To enable it, you must add the parameter ***hibernate.search.highlighter*** *in the* ***openkm.properties***.

```
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 you to consider not configuring Internet connections on servers behind an Internet proxy.

It has some disadvantages, such as:

- *You will need to use your proxy credentials each time you upgrade the application with the OpenKM automatic updater tools.*
- *You need to configure JVM parameters with your Internet proxy settings for the application to be able to connect to the Internet.*

It is not very clear what the advantages are for this kind of configuration on Production Servers, which are not the same scenario as end users. ***Take it as is; only a suggestion****.*

Linux

The configuration below is based on setting proxy parameters in the JVM environment. Another option is configuring the proxy at the OS level; here is one of several options:

*Edit your "****/etc/bash.bashrc****" file as root and add these lines to 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 following lines:*

The http://proxy.url.com value must be your own proxy host URL.

*The "****user****" value must be the username used to connect to the proxy. If your* ***proxy does not require authentication you can remove the entire line*** *where these parameters are set.*

*The "****password****" value must be the password for the user connecting to the proxy. If your* ***proxy does not require authentication you can remove the entire line*** *where these parameters are set.*

```
# 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 look something like this*** *:*

```
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 the following configuration to use Google's 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 (the Google HTML code should be shown).*

```
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

The configuration below is based on setting proxy parameters in the JVM environment. Another option is configuring the proxy at the OS level; consider looking for this information on the Microsoft TechNet website.

Step one

Verify starting and stopping OpenKM from the command line.

- *First, stop the OpenKM service.*
- *Edit $T****OMCAT_HOME/bin/setenv.bat*** *file and add the following lines:*

The http://proxy.url.com value must be your own proxy host URL.

*The "****user****" value must be the username used to connect to the proxy. If your* ***proxy does not require authentication you can remove the entire line*** *where these parameters are set.*

*The "****password****" value must be the password for the user connecting to the proxy. If your* ***proxy does not require***

authentication you can remove the entire line *where these parameters are set.*

```
# 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 look something like this****:*

```
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 the following configuration to use Google's 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 (the Google HTML code should be shown).*

```
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 went well in step one, you can now configure parameters for the OpenKM service.

- ***Stop OpenKM***.
- *Go to **$TOMCAT_HOME/bin** and execute the following command lines:*



The command lines below do not register the parameters for DNS resolving. If you need them, 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 (the Google HTML code should be shown).*

```
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 the digital signature of a document.

Some configuration properties are needed:

Field / Property	*Type*	*Description*
vidsigner.user	***String***	*Name of the user provided 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 it if you want to sign the document on any device associated with the subscription. In this case, you can find the document on the device by enabling search mode in the configuration and searching by OpenKM user.*
vidsigner.notification.url	***String***	*The URL to which 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 the signature image will be shown. If it is set to zero, the signature will be invisible.*
vidsigner.sign.posX	***String***	*The X position in millimeters where the signature image should be placed.*
vidsigner.sign.posY	***String***	*The Y position in millimeters where the signature image should be placed.*
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 detailed log with all the information about the call to the VidSigner service.*

In addition, you have to create a user in OpenKM with the name and password provided by VidSigner for your subscription.

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

Also, you have to create two new Automations on the event ***Set metadata group*** *(on post) and associate the action* ***VidSigner Document To Sign*** *with the first and the action* ***VidSigner Document Signed*** *with the second.*

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 it to another location.*



This action must be done with the application stopped.

Understanding the contents of repository folders

*In "**$TOMCAT_HOME/repository**" there are four folders: **tmp**, **index**, **cache** and **datastore**.*

Folder	Backup	Description
tmp	*No*	*This folder contains some temporary files to be shared between several clustered OpenKM instances.*
index	*No*	*The folder which contains the Lucene search engine indexes. The content of that folder can be rebuilt.*  *Although it is not mandatory to back up this folder, we recommend doing so.*
cache	*No*	*The folder which contains the application's cached files. For example, for previewing documents, it is sometimes necessary to transform the original document to another 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 cleared without major problems.*  *Although it is not mandatory to back up this folder, we recommend doing so.*
datastore	*Yes*	*The folder which contains all binary data for the documents.*  *It is **mandatory** to back up this folder; otherwise the binary data of the documents could be lost if server hardware fails.*
extraction	*Yes*	*This folder contains the text extracted from the 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 "**$TOMCAT_HOME/repository**" to another location.*
- *Edit the "**$TOMCAT_HOME/openkm.properties**" file and add the property "**repository.home**".*
- *Start the application*

Sample 1

> *From OpenKM version 7.1.x and later, when the "**repository.home**" parameter is set, the file system location 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 to 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*

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

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 "**$TOMCAT_HOME/repository/{folder}**" to another location.*
- *Edit the "**$TOMCAT_HOME/openkm.properties**" file and add the appropriate 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 you not to use network storage (NAS or Samba) for storing the repository. Network issues will affect the repository; we consider this a dangerous configuration.

*There seem to be restrictions when using a network drive (mapped with the "**net use**" command) and running the application as a service. To solve this it is necessary to create a link between the UNC path and the file system.*

```
MKLINK /D C:\okm-repository \\192.168.1.36\eDocuments\okm-repository
```

In most cases it will be necessary to use a standard user to run the OpenKM service instead of the default. Take a look at the 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, the application comes with two connection roles named "**ROLE_USER**" and "**ROLE_ADMIN**". Any OpenKM user must be assigned one of these roles to log in.*

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, they will not be visible to users until you enable them on the Profile General tab.

By default, OpenKM comes with the extensions disabled.

Types of extensions:

- *Do not require additional software configuration.*
- *Require additional software configuration.*
- *Sample extensions for development.*

Available extensions are registered in the OKM_EXTENSION table. To enable any of them, you only need to insert them into this table.

Extensions that do not require additional software configuration:

- *Go to **Administration** > **Utilities** > **Database query**. At the bottom right, select the **JDBC** option, paste the SQL insert statements of the extensions you want to enable into the box, then click the **execute** button at the bottom right.*



*The **Stapling** extension is deprecated and has been superseded by **Relation**.*

*The **Online Editor** extension has been available since OpenKM version 8.1.15.*

```
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 the bottom right, select option **JDBC**, **paste into the box the SQL inserts** of the extensions you want to enable. Finally, click the **execute** button at the bottom right.*

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 are some sample extensions for development and testing purposes.

- *Go to **Administration** > **Utilities** > **Database query**. At the bottom right, select the **JDBC** option, paste the SQL insert statements of the extensions you want to enable into the box, then click the **execute** button at the bottom right.*

```
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 into two sections: the login process and retrieving users and roles from an LDAP server. It's usually better to start with LDAP configuration parameters, which are used by OpenKM to show users and roles in the administration area, 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 directory, to be shown in the user interface or to be able to log in to 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 during login*

Active Directory

- *Active Directory basic configuration*
- *Active Directory advanced configuration*
- *Active Directory example with referral enabled*
- *Active Directory example using 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 remain the same.*
- *User data and roles are now managed in OpenKM administration and stored in the database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to the 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 LDAP. This list is cached for 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clear 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**ogging in to OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps it will be necessary to restart the OpenKM service and you do not want to lose administrative 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 is restarted and can continue working in the administration. After this change, the users and roles lists will be empty in the administration. Until you successfully configure 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*

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, which is concatenated by default in all filter queries**.*
- *Roles (groups) filter base is **ou=roles** (real distinguished name is ou=roles,dc=some,dc=com). Any valid role should have it as a parent. ou=roles"/> really points to ou=roles,dc=some,dc=com"/>*
- *Users filter base is **ou=organization** (real distinguished name is ou=organization,dc=some,dc=com). Any valid user should have it as parent.*
- *User filter is uid={0}*
- *Finally, take into 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 updating the openkm.properties file, you must restart the OpenKM service for the changes to take effect.

Another option to configure login

In some cases, you might be interested in setting the configuration in the 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://192.168.0.13:389/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 remain the same.*
- *User data and roles are now managed in OpenKM administration and stored in the database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to the 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 domain works together and the ***follow*** *parameter is needed.*

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*

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*
- *cn=user3,ou=organization2,dc=company,dc=com*
- *cn=user4,ou=organization2,dc=company,dc=com*

*Any distinguished name includes **dc=company,dc=com** by default.*

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 for 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**ogging in to OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps it will be necessary to restart the OpenKM service and you will not want to lose administrative access.*

*The first action should be to modify the **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 is restarted and can continue working in the administration. After this change, the users and roles lists will be empty in the administration. Until the next parameters are successfully configured, these lists will remain empty.*

*Go to **Administration > Configuration parameters**:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter has been 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)*
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}))*

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 domain servers working together (balanced).
- Users defined in any Active Directory node will be able to log in because the base filter is defined as DC=company,DC=com.
- Any user authenticated in Active Directory can log in because there is no filtering clause.
- Groups read by OpenKM can be defined in any Active Directory node because the base filter is defined as DC=company,DC=com.

Apply changes to 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 updating the openkm.properties file, you must restart the OpenKM service for the changes to take effect.

Another option to configure login

In some cases, you might be interested in setting the configuration in the 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>
```

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 remain the same.*
- *User data and roles are now managed in the OpenKM administration and stored in the database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to the 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 any Active Directory-authenticated user who is a member of ROLE_USER or ROLE_ADMIN to connect.

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 dc=company,dc=com by default, except for users who are not members 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 LDAP. This list is cached by OpenKM for 30-45 minutes 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 **logging in to OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because you will need to restart the OpenKM service in the next steps and you do not want to lose administrative access.*

*The first action should be to modify **principal.adapter** parameter value and **restart OpenKM service**. Because the session ID is kept in the browser, you should not lose the login after the service is restarted and you can continue working in the administration. After this change, the users and roles list will be empty in the administration. Until the next parameters are successfully configured, these lists will be empty.*

- *Users who are members of ROLE_ADMIN or ROLE_USER can be created in any Active Directory node, because DC=company,DC=com is defined as the base filter, **principal.ldap.user.search.base=DC=company,DC=com** and the user search filter is **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 it has DC=company,DC=com as the base filter, **principal.ldap.role.search.base=DC=company,DC=com**.*
- *All Active Directory groups will be listed because no filter restriction has been applied: **principal.ldap.role.search.filter=(objectclass=group)**.*
- *The mail attribute used in this case is **userPrincipalName** (the most commonly used LDAP attribute for mail is **mail**)*

principal.ldap.mail.attribute=userPrincipalName.

Go to ***Administration*** *>* ***Configuration parameters****:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter has been 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 who are members of ROLE_ADMIN or ROLE_USER will be able to log in, because DC=company,DC=com is defined as the base filter, and* ***as user account filter.***
- *Groups read by OpenKM can be created in any Active Directory node, because DC=company,DC=com is defined as the base filter.*

Apply the changes to 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 updating the openkm.properties file, you must restart the OpenKM service for the changes to take effect.

Another option to configure login

In some cases you might be interested in setting the configuration in the 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 remain the same.*
- *User data and roles are now managed in OpenKM administration and stored in the database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to the 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 another 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 LDAP. This list is cached for 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clear the cache via Administration > Tools > Cache stats. In the second step, configure login; this configuration always works in real time.

Step 1 - configuration parameters

*We suggest l**ogging in to OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps it will be necessary to restart the OpenKM*

service and you do not want to lose administrative access.

*The first action should be to modify **principal.adapter** parameter value and **restart OpenKM service**. Because the session ID is kept in the browser, you should not lose the login after the service is restarted and you can continue working in the administration. After this change, the users and roles lists will be empty in the administration. Until you successfully configure the next parameters, these lists will remain empty.*

- *Users who are members of ROLE_ADMIN or ROLE_USER are created in the 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 the base filter, **principal.ldap.role.search.base=DC=company,DC=com**.*
- *All Active Directory groups will be listed because no filter restriction has been applied **principal.ldap.role.search.filter=(objectclass=group)**.*

Go toAdministration > 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)*

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

principal.ldap.username.attribute	***String***	*cn*
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.*

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

*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 updating the openkm.properties file, you must restart the OpenKM service for the changes to take effect.

Another option to configure login

In some cases, you might be interested in setting the configuration in the 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 remain the same.*
- *User data and roles are now managed in OpenKM administration and stored in the database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to the 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 LDAP. OpenKM caches this list for 30-45 minutes to prevent overloading the LDAP server. You can clear 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**ogging in to OpenKM with the admin URL** (for example http://localhost:8080/openkm/admin/index) because in the next steps it will be necessary to restart the OpenKM service and you do not want to lose administrative access.*

*The first action should be to modify the **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 is restarted and can*

continue working in the administration. After this change, the users and roles lists will be empty in the administration. Until you successfully configure the next parameters, these lists will be empty.

- *User members of ROLE_ADMIN or ROLE_USER are created in the 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 the base filter, **principal.ldap.role.search.base=DC=company,DC=com**.*
- *All Active Directory groups will be listed because no filter restriction has been applied **principal.ldap.role.search.filter=(objectclass=group)**.*

*Go to **Administration > Configuration parameters**:*

Field / Property	*Type*	*Description*
principal.adapter	***String***	*The parameter has been 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*

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

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}))*
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 to 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 updating the openkm.properties file, you must restart the OpenKM service for the changes to take effect.

Another option to configure login

In some cases, you might be interested in setting the configuration in the openkm.xml file. 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 remain the same.*
- *User data and roles are now managed in OpenKM administration and stored in the database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to the 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, to 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 LDAP. This list is cached for 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**ogging into OpenKM with the admin URL** (for example, http://localhost:8080/openkm/admin/index) because the next steps will require restarting the 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 you can continue working in the administration. After this change, the users and roles lists will be empty in the administration. Until you successfully configure the following parameters, these lists will remain empty.*



*We **suggest** using "**com.openkm.plugin.principal.DatabasePrincipalAdapter**" in the **principal. The adapter** configuration parameter value can be either "**com.openkm.plugin.principal.MixedPrincipalAdapter**" or the value shown in the table below.*

*If you use "**DatabasePrincipalAdapter**", the restrictions are:*

- *Before the user logs in, you must create all the user data in the database.*
- *The password you set in the database will not be used; the login process works only with an AD connection.*

- *User members of ROLE_ADMIN or ROLE_USER are created in 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 making the changes in the previous files, you must restart the OpenKM service for them to take effect.

Active Directory mixed configuration with LDAP synchronization

The objective of this kind of configuration is to authenticate against Active Directory and, at the same time, obtain roles, usernames, and email addresses from the database. When this configuration is enabled, a cron job named "Sync LDAP users" periodically synchronizes the Active Directory users' data (username, email addresses) 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 LDAP. This list is cached for 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clear the cache from Administration > Tools > Cache stats. In the second step, configure the login; this configuration always works in real time.

Step 1 - configuration parameters

*We suggest l**ogging into OpenKM with the admin URL** (for example, http://localhost:8080/openkm/admin/index) because the next steps will require restarting the OpenKM service, and you do not want to lose administrative access.*

*The first action should be to modify the **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 restarts, and you can continue working in Administration. After this change, the users and roles lists will be empty in the Administration. Until the following parameters are configured successfully, these lists will remain empty.*

*We **suggest** using "**com.openkm.plugin.principal.DatabasePrincipalAdapter**" in the **principal.adapter** configuration parameter value, or the "**com.openkm.plugin.principal.MixedLdapSyncedPrincipalAdapter**" value shown in the table below.*

*If you use "**DatabasePrincipalAdapter**", the restrictions are:*

- *Before a user logs in, you must create all the user's data in the database.*

- *The password you set in the database will not be used; the login process works only with an AD connection.*

- *Users who are members of ROLE_ADMIN or ROLE_USER are created in 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 the 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 the 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 making the changes in the previous files, you must restart the OpenKM service for them to take effect.

For versions older than 8.1.12, use the following 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 to be able to log in.

Example based on a 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.*
- *Assign roles and users as members of the OpenKM group.*
- *Goal: only users and roles that are members of the OpenKM group will be displayed in user interface lists.*

For filtering the 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 restrictions:

Modify openkm.properties with:

```
ldap.user.search.filter=(&(sAMAccountName={0})(memberOf=CN=OpenKM,CN=users,DC=company,DC=com))>
```

If you use openkm.xml, then use:

```
<module-option name="baseFilter">(&amp;(sAMAccountName={0})(memberOf=CN=OpenKM,CN=users,DC=comp
```

Example based on the ROLE_USER and ROLE_ADMIN groups

- *Create a group named **OpenKM (CN=OpenKM,CN=users,DC=company,DC=com)**. It will be used for filtering roles.*

- *Assign roles as members of the OpenKM group.*
- *Goal: only roles that are members of the OpenKM group will be displayed in user interface lists.*
- *Goal: only users who are members of the ROLE_USER or ROLE_ADMIN groups will be displayed in user interface lists.*

For filtering the 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=company,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 restrictions:

Modify openkm.properties with:

```
ldap.user.search.filter=(&(sAMAccountName={0})(objectClass=person)(|(memberOf=CN=ROLE_USER,CN=users,c
```

If you use openkm.xml, then use:

```
<module-option name="baseFilter">(&amp;(sAMAccountName={0})(objectClass=person)(|(memberOf=CN=ROLE_
```



*Some characters are restricted in XML; one of these is **&**. You should use **&** in places where you would normally 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 remain the same.*
- *User data and roles are now managed in OpenKM administration and stored in the database.*
- *There is a cron job called "Sync LDAP users" which synchronizes user data from LDAP to the database.*
- *The current configuration can be found at Active Directory mixed configuration with LDAP synced.*

You need to install the Certificate Manager and Directory Server modules.

- *Create an **okmAdmin** user in ClearOS.*
- *Create **ROLE_ADMIN, ROLE_USER** groups in ClearOS and populate them.*

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 includes 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 LDAP. This list is cached for 30-45 minutes by OpenKM to prevent overloading the LDAP server. You can clear 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 always works in real time.

Step 1 - configuration parameters

*We suggest l****ogging in to OpenKM with the admin URL*** *(for example http://localhost:8080/openkm/admin/index) because in the next steps it will be necessary to restart the OpenKM service and you do not want to lose administrative access.*

The first action should be to modify the ***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 is restarted and you can continue working in the administration. After this change, the users and roles lists will be empty in the administration. Until you successfully configure the next parameters, these lists will remain 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

By default, ClearOS 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. An easy way is to specify only 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 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 as follows:

```
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 the 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 updating the openkm.properties file, you must restart the OpenKM service for the changes to take effect.

Another option to configure login

In some cases, you might be interested in setting the configuration in the 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, use the following 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 Directory

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 this line to your openkm.xml:*

```
<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 cannot 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) for 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 result exception property.

*Go to **Administration > Configuration parameters**:*

Field / Property	*Type*	*Description*
ldap.ignore.partial.result.exception	***Boolean***	*true*

Solution 3:

Enable the 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 the server connection URL to ldaps://.

Field / Property	*Type*	*Description*
principal.ldap.server	***String***	*ldaps://192.168.0.6:389*

Slow login or unable to log in

It may be a problem with LDAP DNS name 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 is good practice to add all subdomains to the application server's hosts file (/etc/hosts for Linux or C:\Windows\System32\drivers\etc\hosts for Windows).

Example based on Active Directory (LDAP) with distinguished base name ***dc=company,dc=com*** *and server domain name* ***SBSSERVER****.*

```
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 available at the OpenKM forum.

If, after adding these subdomains, the login continues to take a long time, check with your IT security team if they have enabled micro-segmentation policies. This may be another reason why logging in takes so long.

To check connections made by the application during the login process to the AD, you can run the following command:

```
$ tcpdump -A tcp port 389
```

To capture only the request:

```
$ sudo tcpdump -i any port 389 > login-test-request.txt
```

Create a Wireshark dump (pcap) and convert it to a human-readable format:

```
$ 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 large AD user repositories, it is not feasible for the AD administrator to add a new group to all users; in this case, we can enforce via the XML login configuration that any logged-in user be a member of ROLE_USER. Add 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*

an incorrect 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 occurs when you try to connect with LDAPS (SSL encryption) and have not configured the trust certificate in the Java truststore.

*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**; to do so, follow the steps below:*

```
$ openssl s_client -showcerts -connect  your.ldap.server.com:636
```

The OpenSSL Windows binaries https://wiki.openssl.org/index.php/Binaries (tested version 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 assumed that $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 the trust store** in 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, you may see 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 logging utility. This is useful, for example when you need to get more information about something that is not working properly in OpenKM, such as third-party application configuration or Spring authentication. By default, only log messages at the INFO level are shown, but you can increase the amount of logging information by raising the log level to WARN or DEBUG. The log configuration file is ***$TOMCAT_HOME/logback.xml*** *Here you can change the log level for messages generated by a given class or package.*



When OpenKM detects 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 take effect.*

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 the **system.openoffice.program** parameter, you must 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 the following:

- *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.

Security configuration

OpenKM can work with several security managers. Depending on the selected value, the manner in which the result is evaluated differs. This application has separate managers, one for the search engine and another 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 the **com.openkm.cache.parentNodePermissions** cache.*

Rest of application manager:

> *Access manager works with a plugin architecture.*

Type	*Name*	*Description*
Default	***com.openkm.plugin.access.DbSimpleAccessManager***	*Default OpenKM security manager. Analyzes only the current node, not the entire hierarchy.*
Recursive	***com.openkm.plugin.access.DbRecursiveAccessManager***	*Analyzes the entire hierarchy.* > *Analyzing the entire hierarchy will take more time and may cause performance issues.*
Read recursive	***com.openkm.plugin.access.DbReadRecursiveAccessManager***	*Analyzes the entire hierarchy (only for read access).* > *Analyzing the entire hierarchy will take more time and may*

		cause performance issues.
Default file plan	***com.openkm.plugin.access.DbFilePlanAccessManagerWrapper***	*Analyzes based on document type and file plan restrictions.* *Should only be enabled when the file plan is enabled.*

*The **Default** option only evaluates the selected node. With **recursive**, the security is evaluated starting with the selected node up to the repository root (so, evaluating from 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. It should be used in combination with **am_more**.*

The application is configured by default with:

Field / Property	*Type*	*Description*
security.access.manager	***String***	*com.openkm.plugin.access.DbSimpleAccessManager*
security.search.evaluation	***String***	*lucene*

Differences between 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 that user. When the user performs a search, OpenKM will return these documents as search*

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

results. However, the user can't access the documents because when OpenKM tries to access a document, it will open every path location and will fail to open level-2a because the user has no rights. But these documents can be accessed using the API. You need to revoke permissions 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 deeper the node, the more parent nodes need to be evaluated.*

Extending security

*You can use extended 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 the **security.extended.mask** parameter.*



After the value is changed, refresh the browser to take effect in the user's view.

Values:

- *DOWNLOAD = 1024;*
- *START_WORKFLOW = 2048;*
- *COMPACT_HISTORY = 4096;*
- *PROPERTY_GROUP = 8192;*

For example, to enable download and property groups, you should use 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 becomes a problem when every change affects several hundred (or even thousands) of nodes. There's an alternative security configuration parameter named **security.mode.multiple** that enables multiple changes at a time, making it much 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 complete because they affect a lot of nodes. It's good practice to run them as background tasks. To support this, there's a configuration parameter named ***security.live.change.node.limit*** *used to set the limit of nodes that can be changed in "live" or "realtime" mode. By default, this value is set to 100, which means that any change that affects more than 100 nodes will automatically be 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 logins. In addition to the typical login and password, once the user is authenticated, they will be asked for an extra six-digit code generated by a mobile application called Google Authenticator.

Compliance

The two-factor authentication implements a two-step verification of HOTP/TOTP. Also known as one-time passwords.

It has been tested with:

- *The Google Authenticator*
- *The Microsoft Authenticator*

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 the Apple App Store.

> *https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2*

Once installed and configured, this application does not even need an Internet connection.

OpenKM configuration

> *Before anything, please read Two-factor authentication configuration parameters.*

Each user can configure their own account to use 2FA. To do so: ***Go to KCenter, at top right > Settings*** *>* ***Settings tab*** *>* ***Two-factor authentication****.*

If 2FA is enabled by the administrator, at the bottom of the User Configuration dialog you will see a new option ***2FA****.*

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

Depending on whether 2FA is enabled, you will see:

- *Disable button: if 2FA has already been configured for this user.*
- *Enable button: if 2FA has not been configured yet.*

If you click the Enable button, a new window with the following parameters will appear:

- *The current user ID to be configured.*
- *A QR code that you should scan with the* ***Google Authenticator*** *application.*
- *A text box where you must enter the verification code generated by* ***Google Authenticator*** *(after the QR code has been scanned).*

Once you have completed these steps, please click the Accept button.

Scan QR code

To scan the QR code, open ***Google Authenticator*** *on your mobile and perform the following steps:*

1. *Click on Configure account.*
2. *Select* ***Scan code***.
3. *Point your mobile device at the QR image on the screen.*
4. *Once the image is properly scanned, the account for OpenKM is added.*
5. *Now you will see a square with a number (code) which you will have to enter when required.*



These generated codes are time-based: they expire every 60 seconds. If you pay attention to the ***Google Authenticator*** *screen, you will see how the code changes periodically.*

Google Authenticator use

Once ***2FA*** *is configured on your OpenKM account, the next time you log in to OpenKM, after your username and password are verified, a new screen will appear requiring you to enter a code obtained from the* ***Google Authenticator*** *application you previously configured.*

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 advise users about 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 on the 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.* *Newly uploaded document with an expiration*
expiration.notification.template	***String***	*Mail body.* *Newly uploaded document ${docUrl} Expiration date: ${date} *
expiration.expiration.alert.days	***Integer***	*Set the number of days before expiration when the expiration alert message will be sent.* *15* *Default document expiration sets a unique value for all documents. This is the default logic, but you can implement 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.expired.reminder.days	***Integer***	*Set the number of days for sending reminder notifications for documents that expired more than that many days ago.* *15* *Default document expiration sets a unique value for all documents. This is the default logic, but you can implement your own.*
expiration.expired.reminder.subject	***String***	*Mail subject.* *Reminder alert for expired documents*

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

expiration.expired.reminder.template	***String***	*Mail body.* *Document ${docUrl} expired at ${date} *

These parameters are set in Administration > the 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/p
<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.group_notify" type="multiple" className="com.openkm.plugin.form.
      <validator type="req"/>
    </select>
    <select label="Notify expiration alert" name="okp:expiration.group_alert" type="multiple" className="com.openk
      <validator type="req"/>
    </select>
    <select label="Notify expirated" name="okp:expiration.group_expired" type="multiple" className="com.openkm.
      <validator type="req"/>
    </select>
    <select label="Notify expirated reminder alert" name="okp:expiration.group_expired_reminder" type="multiple" cla
      <validator type="req"/>
    </select>
  </property-group>
</property-groups>
```

Crontab task

- *Copy jar files into the $TOMCAT_HOME/plugins folder.*
- *Restart OpenKM (this will register crontab tasks).*

Task	*Expression*	*Description*
InitialNotification	***0 */1 * * * **	*After a metadata group is added to the node, the "initial notification" email is sent.* *The crontab task is executed every minute, looking for new nodes with a metadata group.*
AlertExpirationNotification	***0 30 0 * **	*Sends a daily email alert for documents that will expire soon.*

	*	
ExpirationNotification	***0 0 0 * * **	*Sends a daily email alert for documents that expired today.* *This crontab task should be executed before the **AlertExpirationNotification** task.*
AlertExpiredReminderNotification	***0 0 1 * * **	*Sends a daily email alert for documents that expired x days ago.*

More information on Crontab.

Register reports

Report	*Description*
ValidDocuments.jrxml	*Lists all valid documents.*
ExpiredDocuments.jrxml	*Lists all expired documents.*
NextExpirationDocuments.jrxml	*Lists documents that will expire soon.*

More information on Reports.

See also Profiles for configuring reports in the user interface menu.

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

Configuring CIFS

*The aim of this configuration is to allow viewing OpenKM as a remote drive using the **CIFS** protocol. Below the two ways of configuring it are described:*

- *Authentication using the 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 cannot be started with root permissions, the ports mentioned below must be above 1024 (the default system ports are 138, 137, 139 and 445), so it is necessary to redirect the original ports to new ones. To do this, 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 purposes, it is necessary to access CIFS from the local server machine. In this case, the above configuration is insufficient and it is necessary 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*** *>* ***Configuration*** *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 on the authentication adapter values, the proper a should be used:* *com.openkm.integration.cifs.authenticator.Datab* *com.openkm.integration.cifs.authenticator.LdapC*
system.cifs.debug	***Boolean***	*Property used to enable detailed logging 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 indicate the name of the repository*	*By default the name is **repository***



After changing these properties, OpenKM must be restarted.



*If this is not a fresh installation of OpenKM, it is important to reset user passwords in OpenKM Administration Configuration (no matter if it is the same password) in order to generate the **MD4 version** needed for authentication.*

Windows client

*It is necessary to modify a **Local Security Policy**. To do so, go to **Control Panel** → **Administrative Tools** → **Local Security Policy** → **Local Policies** → **Security Options**:*

*Set the option **Network security: LAN Manager authentication level** to: **Send LM & NTLM: Use NTLMv2 session security if negotiated.***

You can also do it by launching the following command from the 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 available drive letter*
- *Folder: \\server_ip\repository OR \\server_name\repository*
- *Connect using different credentials (check this if the Windows username and password are different from the OpenKM user's).*

Then click the Finish button.

If 'Connect using different credentials' was checked, a new window to enter the corresponding credentials will appear.

You can also do it by launching the following command from the 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 a wrong password error, try resetting the password of the user in OpenKM Administration Configuration.

Active Directory Authentication

In this case, it is assumed that the system was previously integrated with Active Directory. It will be necessary to configure:

- *Active Directory*
- *OpenKM Server*
- *OpenKM configuration*
- *Windows client*

Active Directory

It is necessary to create a user that OpenKM, using a private key, will use to validate the credentials of the incoming CIFS request. To do this, follow these steps:

1. *Create a Computer Object that will be the one corresponding to the OpenKM server.*
2. *Once created, modify its properties by enabling the 'Trusted for delegation' property.*

1. *Trusted for delegation. This step is important because, if not enabled, it will not be possible to negotiate between client and server.*

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 maps the service (application server) to a domain user. This user must be used only and exclusively for this purpose because if this user is used to log on to another computer, it will not be able to be used to log on to the application server. Once the user is created, its properties should be modified to prevent password changes and to disable account expiration. Finally, select the Account tab and enable* ***Do not require Kerberos preauthentication*** *in the Account Options selection. For this example, a user called* ***servercifs*** *is created.*

a. *Select the Account tab and enable the Do not require Kerberos preauthentication option in the Account Options section.*

1. *Generate the keytab file. This file is an encoded copy of the private key of a user in Kerberos. It is used by the web application as an authentication proxy for 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 : Previously created user.*
  3. *cifs.keytab: file where the private key exported for use by the Tomcat server is stored.*

2. *A Service Principal Name (SPN) needs to be set up with CIFS and a server name; for example: tomcat.example.org where the Tomcat servlet container is running. This is used with a domain user and its keytab which 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 the Tomcat servlet container is running and servercifs is the previously created user.

Usually the previous command "ktpass" should create the service cifs/tomcat.example.org; if it is already created, an error will be raised indicating it is already present.

OpenKM server in Active Directory configuration

The following steps must be followed:

Introduce the server into the domain

The first thing to do is to include the application server in the domain if it was not done previously. To do this, 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/hosts 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 cannot be started with root permissions, the ports mentioned below must be above 1000 (the default system ports are 138, 137, 139, 445), so it is necessary to redirect the original ports to new ones. To do this, 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 purposes, it is necessary to access CIFS from the local server machine. In this case, the above configuration is insufficient and it is necessary 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 the 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/krb5.conf".

In this file Domain, KDC server and allowed algorithms must be specified. The most important algorithms are ***arcfour-hmac-md5*** *and* ***rc4-hmac*** *which Microsoft Active Directory offers.*



Preserve most of the krb5.conf default configuration and make only the suggested changes 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 the following 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 corresponds to the property* ***system.cifs.ldap.login.entry***
- *The principal property corresponds to* ***system.cifs.ldap.principal***

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

- *The keytab property is the secure location 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 which is usually placed 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 previous section.
```

> *The default value **${java.home}/lib/security** can be changed to any other location where the previously created file cifs.login.config has been stored.*

OpenKM Configuration in Active Directory Configuration

*To configure it: go to **Administration** > **Configuration** and change the following properties:*

Field / Property	*Type*	*Description*	*Value*
system.cifs.server	***Boolean***	*Property to activate or deactivate CIFS*	*true*
system.cifs.authenticator.class	***String***	*Class used by CIFS to authenticate against OpenKM*	*com.openkm.integration.cifs.authenticator.LdapCIFSAuthen* *Depending on the authentication adapter values, the proper should be used:* *com.openkm.integration.cifs.authenticator.Data* *com.openkm.integration.cifs.authenticator.Ldap*
system.cifs.debug	***Boolean***	*Property used to enable detailed logging of*	*false*

		CIFS functionality	
system.cifs.domain.name	***String***	*Domain or workgroup 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 purposes*	*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*
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 for sessions*	*By default **1139***
system.cifs.port.smb	***Integer***	*Port number used by SMB*	*By default **1445***
system.cifs.shared.name	***String***	*Property to indicate the name of the repository*	*By default the name is **repository***



After changing these properties, OpenKM must be restarted.

Windows client in Active Directory configuration

*It is necessary to modify a **Local Security Policy**. To do so, go to **Control Panel** → **Administrative Tools** → **Local Security Policy** → **Local Policies** → **Security Options**:*



*Set the option **Network security: LAN Manager authentication level** to: **Send LM & NTLM: Use NTLMv2 session security if negotiated.***



You can also do it by launching the following command from the command line:

REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "LmCompatibilityLevel" /t "REG_DWORD" /d "1" /f

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

Then, to connect to the remote machine, go to Computer → Map Network Drive:

- *Drive: Any available drive letter*
- *Folder: \\server_name\repository*
- *Connect using different credentials (check this if the Windows username and password are different from the OpenKM user's).*

Then click the Finish button.



If 'Connect using different credentials' was checked, a new window to enter the corresponding credentials will appear.



You can also do it by launching the following command from the 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, they will be written to disk. These elements will be removed from this cache if they are unused for 5 minutes. Otherwise, they will remain forever. If you want to force a refresh, you have to change the ***timeToLiveSeconds*** *to 3600 and they will be refreshed every hour.*

You can check the status of these caches by going to ***Administration*** *>* ***Utilities*** *>* ***Cache stats****. In this part you can enable and view the cache usage stats and even reset a given one. More info at Cache stats.*

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

Configuring a digital email signature

A digital signature attached to an email message increases security by providing the recipient with assurance that you, and no one else, signed the contents of the email message. Your digital signature, which includes your certificate and 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 S/MIME 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 the anonymous user

Behind the anonymous user, there's a standard user named "public," which automatically logs in to the application using a unique URL.

*This **feature must be enabled in 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 the user can access (**this is mandatory to create the user's trash folder, etc.**).*

*The "**public**" user **must have READ access** to the main root nodes (okm:root, okm:templates, etc.).*

***The share preview option will only allow the document node type** with "**ROLE_PUBLIC**" **read access**.*

Optional steps:

- *Create a profile for the "public" user. Disable all the features you consider the user should not have access to.*

Finally, share the public autologin URL, which will be something like http://yourhost:8080/openkm/public/

In case you want to log out the public user, please use a URL like http://yourhost:8080/openkm/kcenter/#/logout

Other usages:

- *Share a specific document link http://yourhost:8080/openkm/public/uuid (where the uuid is the UUID of the node)*
- *Share a specific document preview link http://yourhost:8080/openkm/public/preview/uuid (where the uuid is the UUID of the document)*

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

Prometheus configuration

Prometheus is an open-source monitoring system that has some interesting features for event monitoring and alerting. You can find a more detailed description at https://prometheus.io/.

This endpoint has been included in OpenKM since v8.1.10. To enable it, you need to include these two lines in the ***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 only be accessible to users with the ROLE_ACTUATOR or ROLE_ADMIN roles.

Setup validation

In order to verify that 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 username and password match, and the user has the 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 to use 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 appropriate Docker Compose file:

```
version: '3.2'

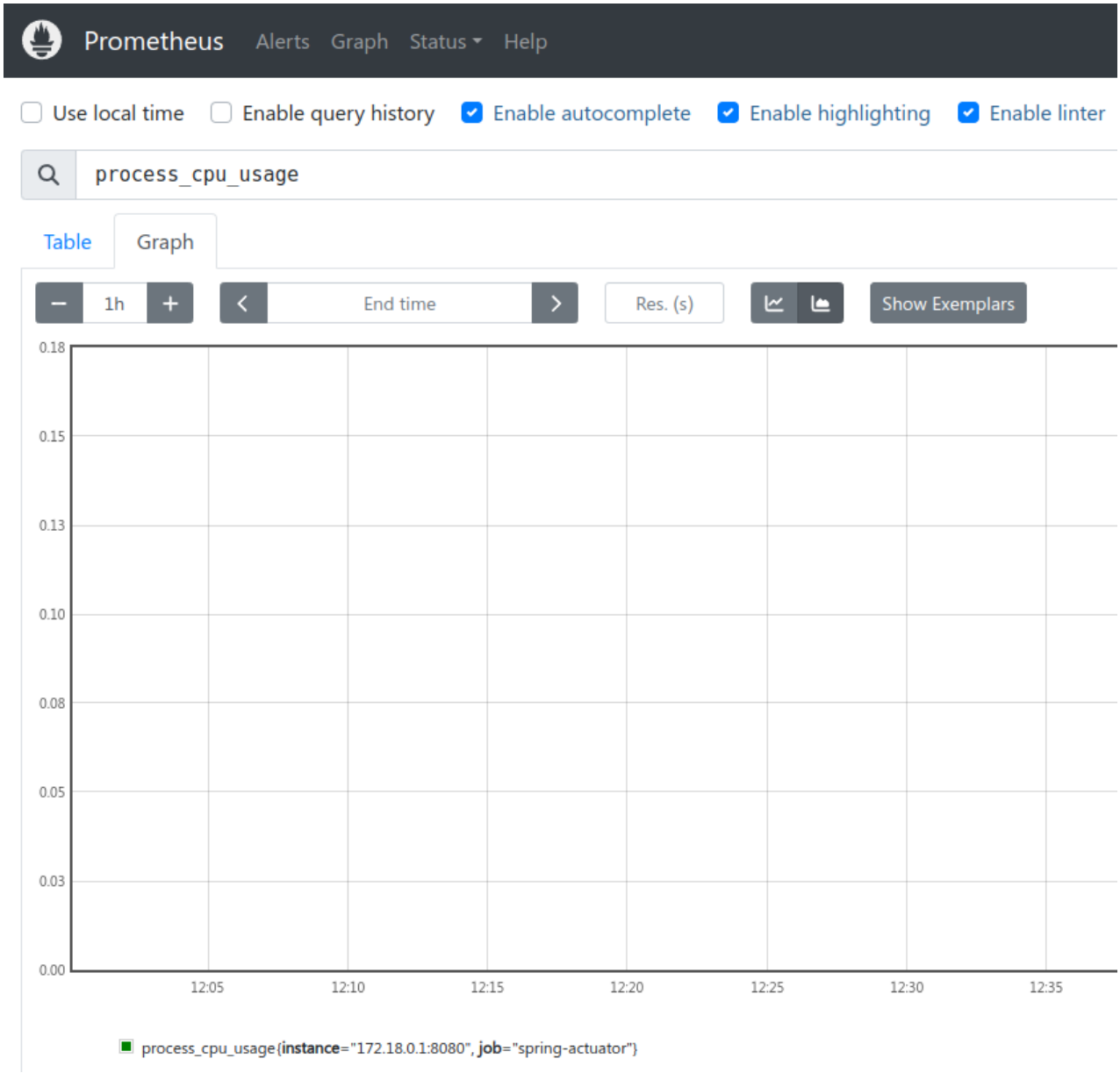
services:

  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:

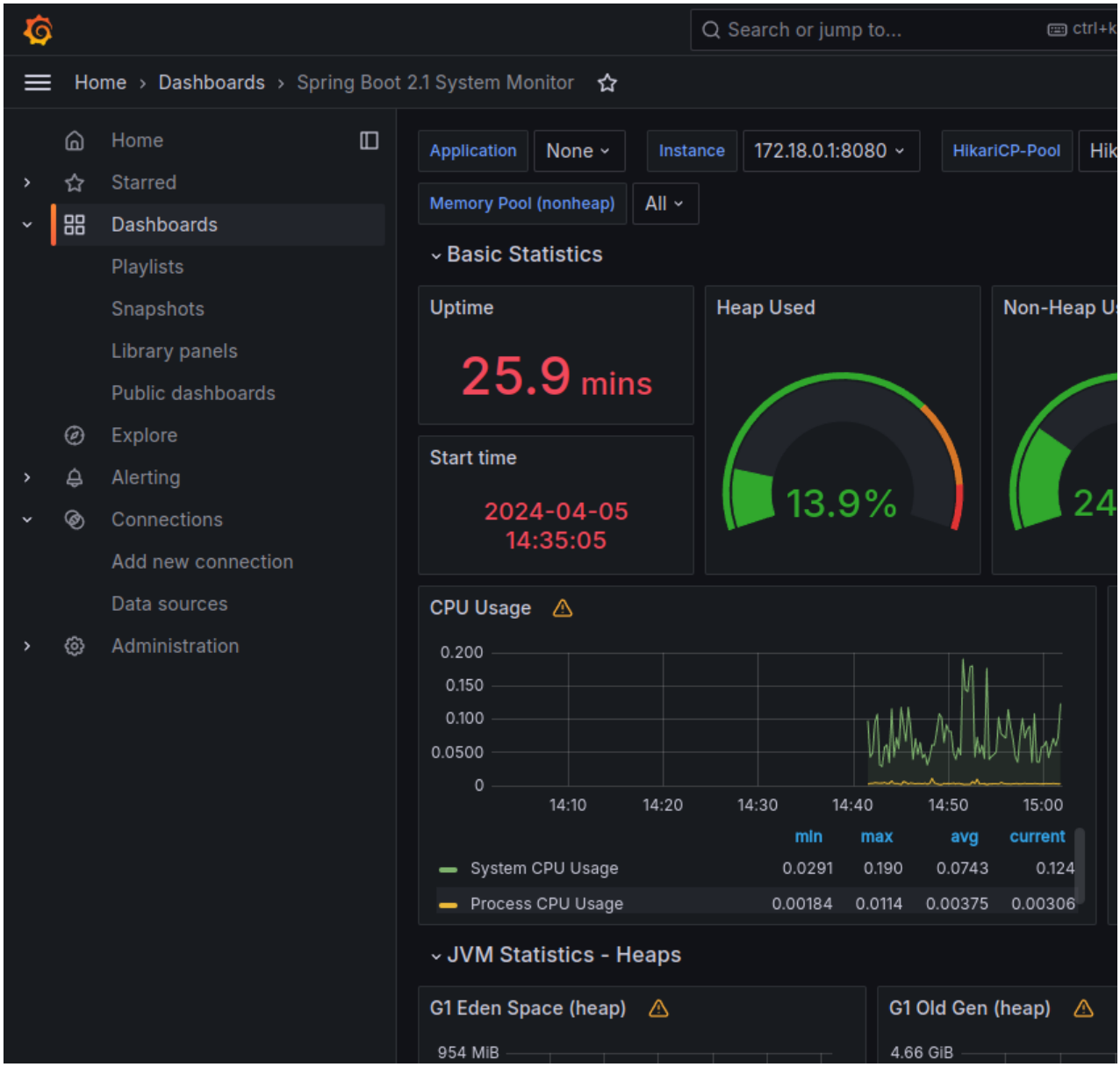
```
$ docker-compose up -d
```

Once started, Prometheus will be accessible at http://localhost:9090/ and you can see a graph of application CPU usage filtered by "process_cpu_usage":



Grafana will be available at http://localhost:3000/ and to log in, use "admin" as the default username and password. Go to "Data sources" and add a Prometheus data source using the URL http://172.18.0.1:9090. To create a sample dashboard, go to Dashboard and click the "Create Dashboard" button. After that, click "Import dashboard" and choose the "Spring Boot template" which has the id 11378. Use the previously created Prometheus data source.

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



More information is available on the Prometheus and Grafana websites.

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

Configuring Content Replication



*Content replication allows documents and folders to be automatically replicated from a **source server** to a **target server**. Replication is unidirectional and based on scheduled background processing.*

*The replication feature requires configuration on **both servers**.*

Configuration

Source Server

Configure the folder that will be monitored for replication events. Only changes inside this folder (and its children) will generate replication events.

OpeenKM configuration property	*Description*
content.replication.monitored.folder	*Set the source uuid folder*

Target Server

Configure the destination folder and the connection details to the source server. All replicated content will be created under the target folder.

OpeenKM configuration property	*Description*
content.replication.source.url	*Set the source server openkm url*
content.replication.source.username	*Set the source user that will access to the documents in the source server*
content.replication.source.password	*Set the source user password*
content.replication.target.folder	*Set the folder's uuid where the content will be replicated*

Database Tables

The replication mechanism relies on the following database tables:

Table	*Server*	*Description*
OKM_CONTENT_REPLICATION_EVENT	*Source*	*Stores all replication events generated when content is*

		created, modified, or deleted inside the monitored folder.
OKM_CONTENT_REPLICATION_STATE	*Target*	*Tracks the processing status of replication events (**PENDING**, **DONE**, **ERROR**).*

How It Works

The replication flow is executed automatically and follows these steps:

1. ***Source Server***

   When a document or folder is created, modified, or deleted inside the monitored folderan event is stored.

2. ***Target Server***

   Every 5 minutes, ContentReplicationScheduler

   - *Fetches new events from the source server and tryes to replicate the action in the target server*
   - *Stores them as ContentReplicationState entries with status **PENDING***
   - *Processes each event*
   - *Marks events as **DONE** or **ERROR***
   - *Automatically retries failed events on the next execution*

3. ***UUID Preservation***

   Documents and folders are created using the same UUID as the source ensuring consistency between servers.

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 content 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 packet 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 distributions like Ubuntu, this configuration file is located at /etc/mysql/my.cnf.*

Unable to connect from another computer on the network

Ensure the other computers have access to the server.

Ensure your server firewall allows remote connections to the destination port.

If you expose the Tomcat port 8080, ensure it is binding to your IP; the " ***address****" parameter value should be "****0.0.0.0****" in $TOMCAT_HOME/conf/server.xml:*

```
<Connector port="8080" address="0.0.0.0" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

After any modification to server.xml, you must restart Tomcat for the changes to take effect.

Error downloading document as PDF

This error is usually caused by LibreOffice not being installed or by an incomplete package installation.

IMAP SSL certificate error

First, obtain the certificate with the openssl command, and then 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 is available at Connecting+to+SSL+services.

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 does not require authentication.

Problems with character encodings

If your language has non-ASCII characters and they are displayed as garbage, you have a problem with encodings.

*Check the $TOMCAT/bin/setenv.sh or $TOMCAT/bin/setenv.bat file. You should 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

This is caused because bindDN has a size limit for each LDAP query.

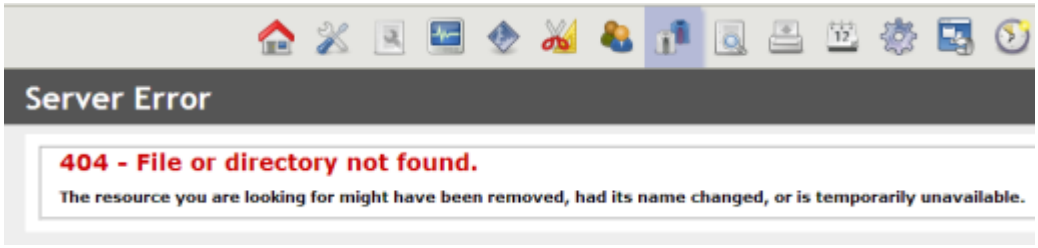
In OpenLDAP, the solution is to increase the limit of bindDN; for example (to change the size limit for the user used by the OpenKM LDAP connection):

```
limits dn.exact="cn=reader,ou=accounts,ou=admins,dc=example,dc=com" size=100000
```

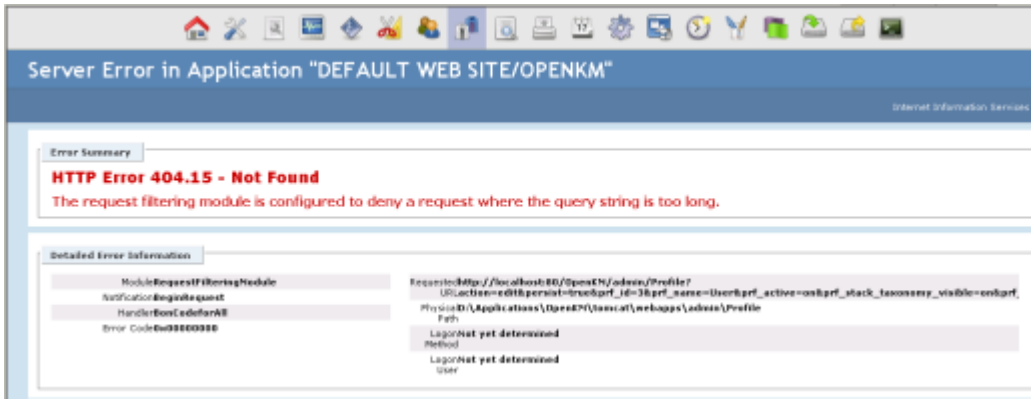
See also OpenLDAP Limits.

Behind the IIS proxy, you may get error 404 or 404.15

When using IIS as a web server, by default it limits the length of the query string sent to the server. If the limit is exceeded, you will see an error screen on the client side:

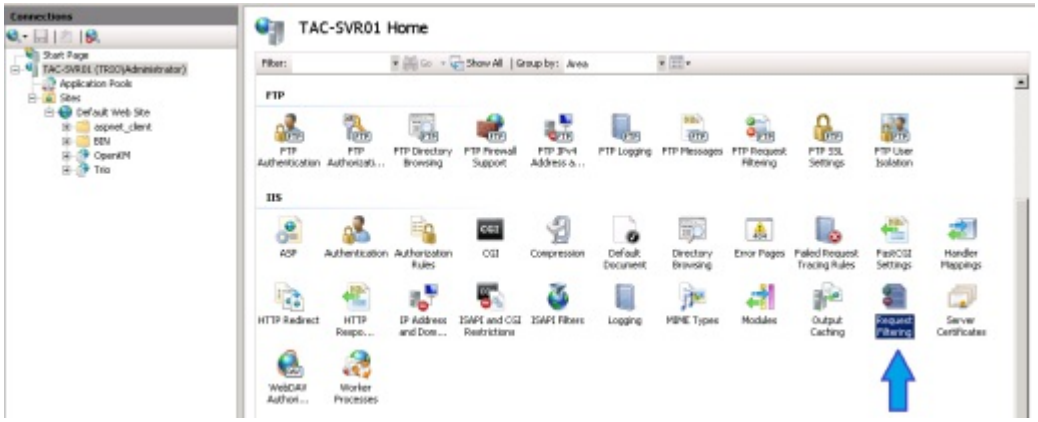


On the server side, we get a more detailed error:



To solve this issue, we need to increase the maximum size of the query string on the web server:

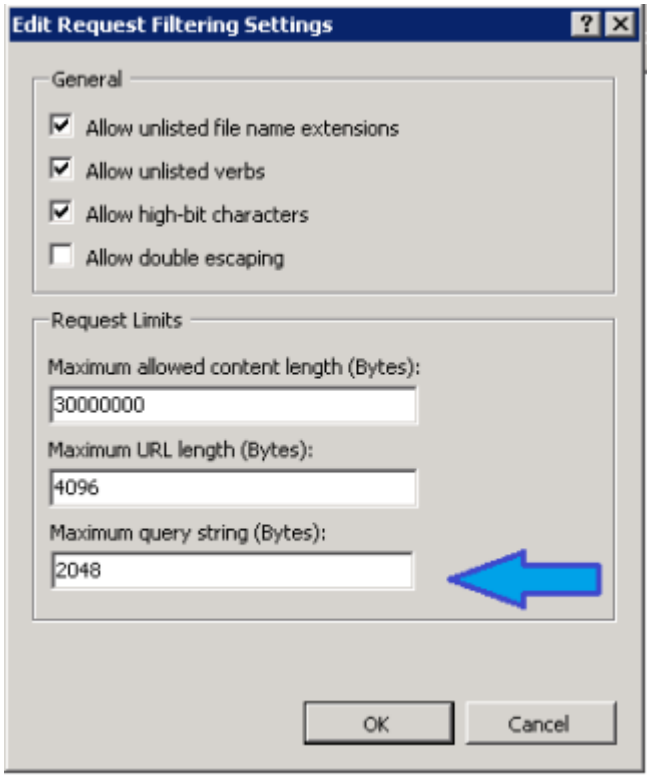
- ***Open Internet Information Services (IIS) Manager***.
- *In the* ***Connections pane****, go to the* ***connection*** *>* ***site*** *>* ***application*** *or directory for which you want to modify your request filtering settings.*
- *In the* ***Home pane, double-click Request Filtering***.



*Click **Edit Feature Settings...** in the Actions pane.*



Change the maximum query string length:



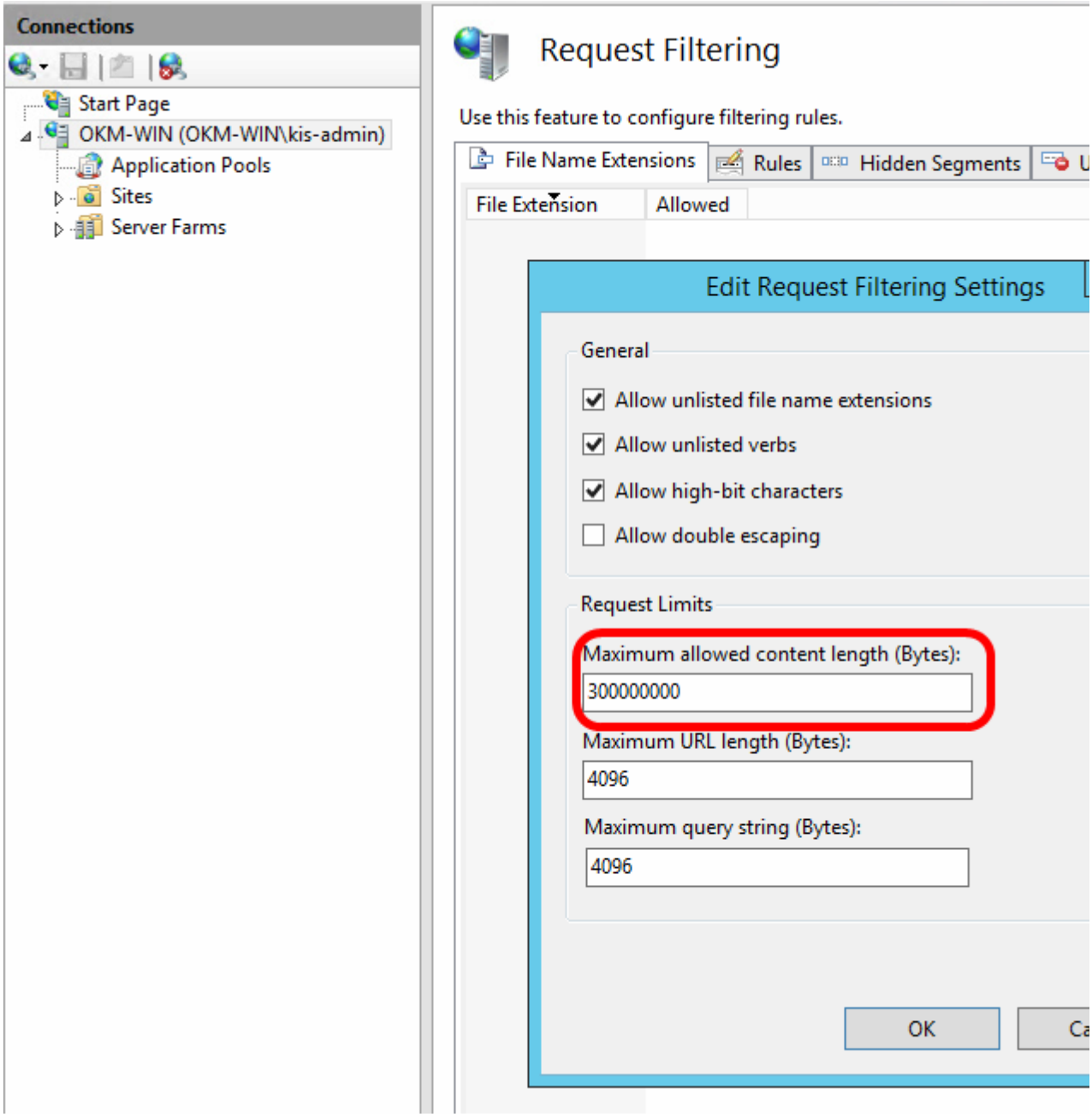
More information is available at the [OpenKM forum](#).

IIS gets IOException: Internal Server Error while uploading file or HTTP 413 Request Entity Too Large Error

This error may appear when uploading a large document to OpenKM.

Solution one

Change the "Maximum allowed content length (Bytes)" in your IIS configuration settings:



Solution two

- ***Go to the webapps*** *folder*

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

- ***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, 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 you should treat it 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.

Red Hat and Oracle Linux issues

OpenKM not starting as a service automatically

*If in your OS the OpenKM service does not start automatically, this may be 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 at the path (/etc/systemd/system).***

```
$ nano /etc/systemd/system/openkm.service
[Unit]
Description=OpenKM - https://www.openkm.com/
After=network.target
[Service]
Type=forking
User=openkm
WorkingDirectory=/home/openkm/tomcat
ExecStart=/bin/bash /home/openkm/tomcat/bin/startup.sh
ExecStop=/bin/bash /home/openkm/tomcat/bin/shutdown.sh
[Install]
WantedBy=multi-user.target
```

***** *ExecStart contains a sample value; the correct path is the location of the* ***startup.sh*** *file.*

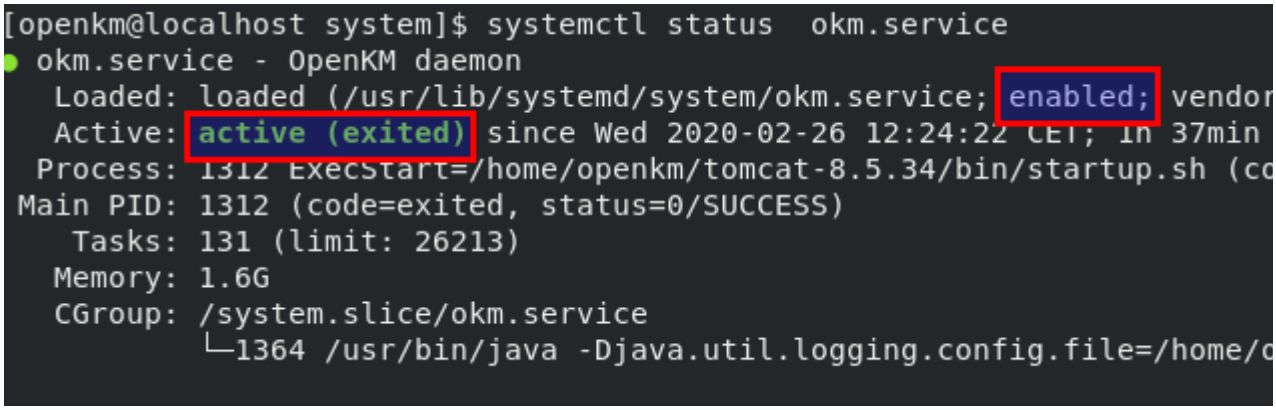
- ***Refresh, enable the service, and activate the service.***

```
$ systemctl daemon-reload
$ systemctl enable openkm.service
$ systemctl start openkm.service
```

With the following command, we can check the status of the service:

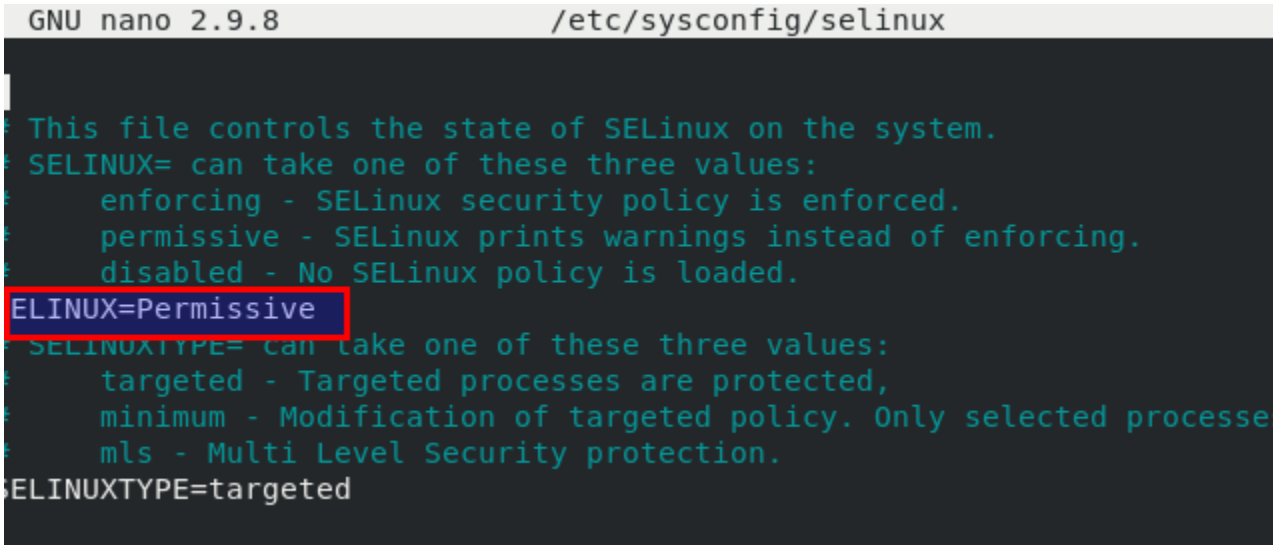
```
$ systemctl status openkm.service
```

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, you 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:

```
$ gm convert <params>
```

More info at GraphicsMagick convert documentation.

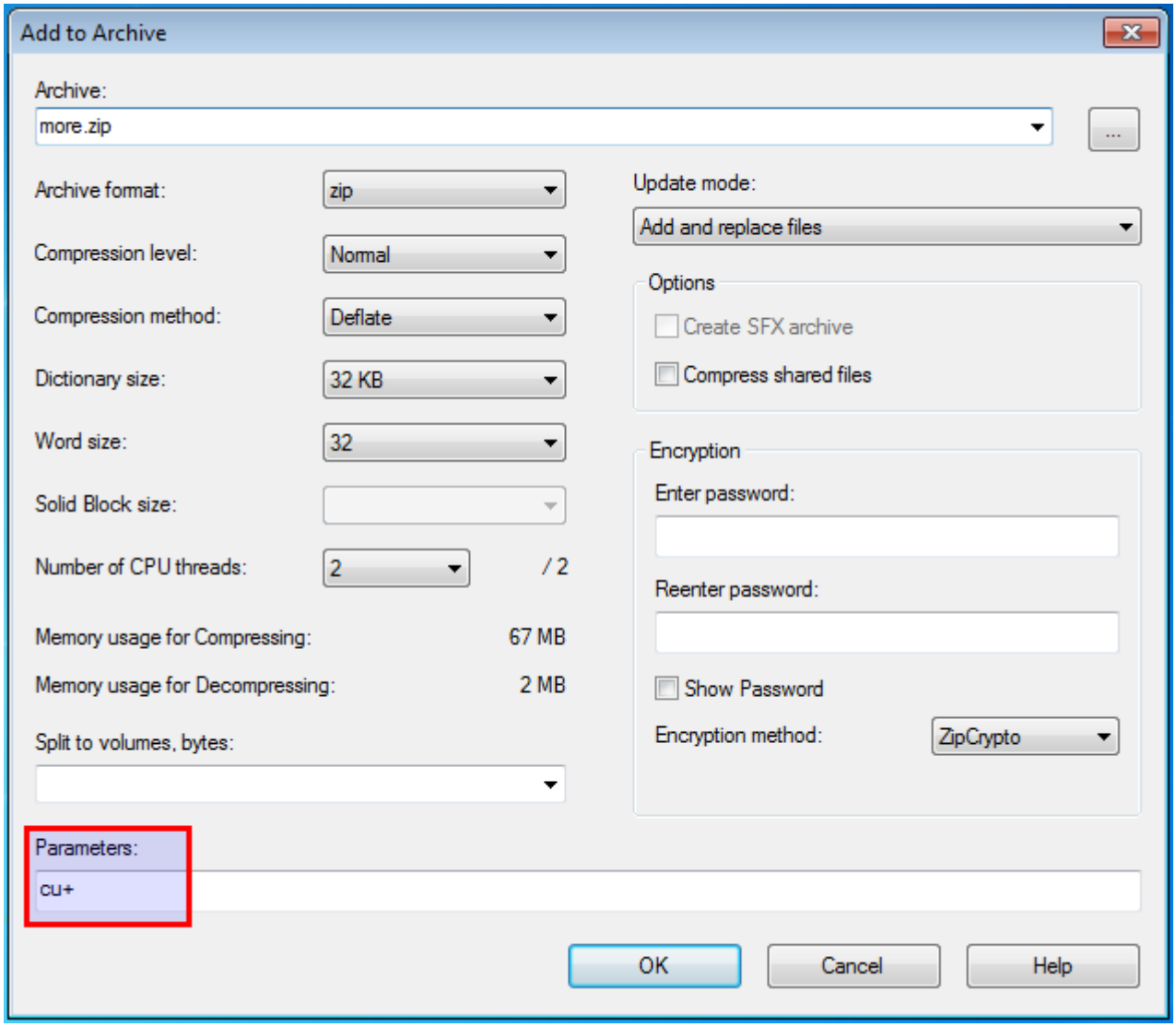
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Client configuration

ZIP filenames with wrong charset

When you download an OpenKM as a ZIP, the included files may have the wrong character set. This only happens with non-Latin languages like Greek, Chinese, or Arabic. This issue has been fixed since OpenKM 4.1 and exported ZIP archives can be opened without problems with WinZip 14 or higher. However, Unicode support in ZIP files is a little tricky and may cause problems. To ensure you don't have problems, try using 7-Zip, which seems to handle these Unicode filenames.

Another problem you may have is when you want to import a ZIP into OpenKM. In this case you can also get incorrect filenames. Filenames should be encoded in Unicode, and to ensure this 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 Linux port of the 7-Zip

compression tool:

```
$ sudo aptitude install p7zip-full p7zip
```

Microsoft Office Add-in disappears after Office restart

This could be a problem with the default template (normal.dot). To solve it, try this:

- *Open normal.dot*
- *Go to menu Tools / Personalize / Toolbars*
- *Open the OpenKM toolbar*
- *Save normal.dot*

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

Best practices for reporting issues

To help you quickly identify and resolve errors, please try to follow the guidelines below:

- *Provide a **detailed description** of the issue; this will help us understand your problem.*
- *When possible, **make screenshots of all the steps you followed**. The idea is to be able to reproduce the error or at least better understand the context in which it occurred.*
- *Videos are also welcome.*
- *Share the stack trace from the openkm.log file.*

Because the support website has restrictions on document size, share with us a download link using online file-sharing tools such as:

- *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 Windows a common location might be "**c:\tomcat-9.0.76\logs**".*

To watch how openkm.log changes in real time, you can use a Windows tool like mtail.

*You have a copy of the mtail tool in **c:\tomcat-9.0.76\extras**\.*

Sample of stack trace capture



When you capture the stack trace, it is a good practice to include some extra lines before and after the error.

```
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
```

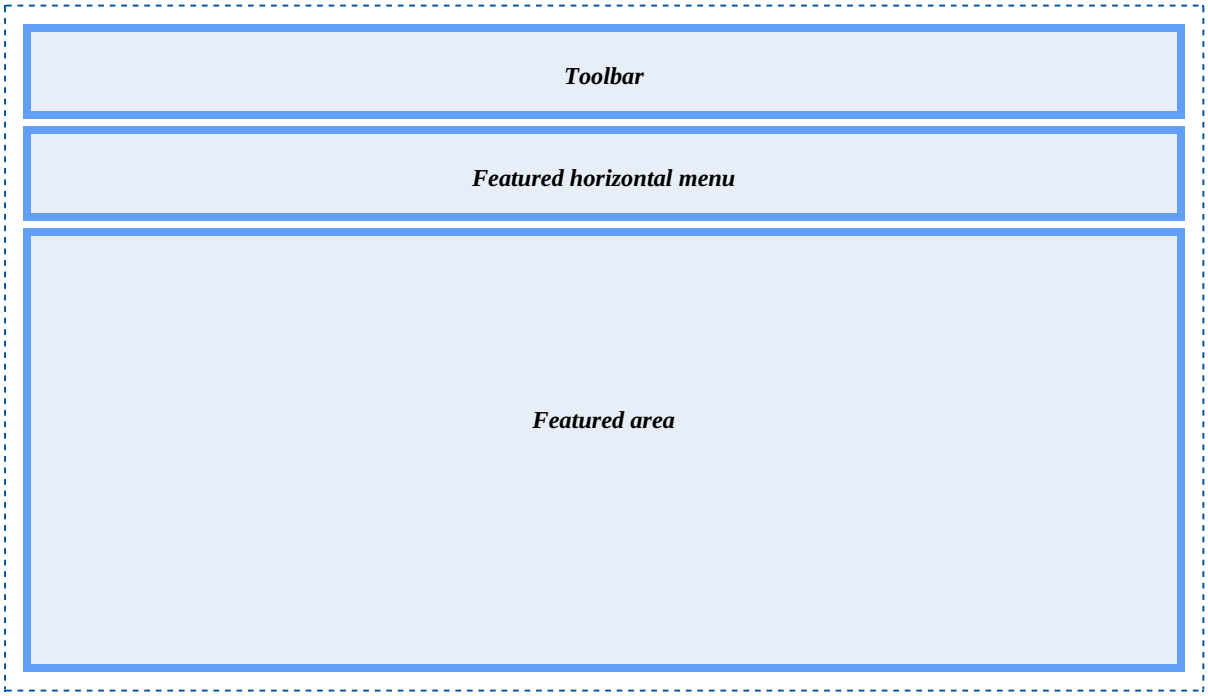
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

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 are listed here. You can modify, remove, or add a new one.*
	***Statistics***	*Shows information about* ***the repository size****. You 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 types.*
	***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 the* ***OKM_ACTIVITY*** *table.*
	***Automation***	*Sets automatic tasks based on application events like DOCUMENT_CREATION, among others.*
		Sets text or image stamp definitions.

	***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 such as indexing documents and antivirus analysis.*
	***OCR templates***	*Sets OCR template definitions to automate document data capture.*
	***Languages***	*Adds, modifies, or removes languages and manages 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 are allowed).*
- *List (each broken line is considered 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 the* ***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 top-right corner to see the action buttons in the table.*
- *Click the* ***New configuration parameter button***.

Check Configuration

This action will check whether some conflicting parameters are correctly configured (pdf2swf, convert tool, OCR engine, or LibreOffice path).

> *Although a parameter may pass this check, that does not mean it is 100% correct. Please take it as a light check that 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.*

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

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 the body of mail notifications.* *Message: ${notificationMessage}* *User: ${userId}* *<#list documentList as doc>Document: ${doc.path}*
notification.message.subject	***Text***	*This parameter is used in the subject of mail notifications. It helps users identify that these emails are coming from OpenKM.* *OpenKM - NOTIFICATION*
subscription.message.enabled	***Boolean***	*Whether the subscription is evaluated and the message is sent. By default it is enabled, but in case of massive document import we recommend disabling it to avoid performance issues.*
subscription.message.body	***HTML***	*This parameter is used in the body of mail user subscription notifications.* *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 the subject of mail subscription notifications. It helps 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 threads for text extraction.*
text.extraction.batch	***Integer***	*It indicates the number of documents being processed on each indexing cycle.* *When parameter "**text.extraction.concurrent**" is enabled, it is good practice to set it to a multiple of the "**text.extraction.threads**" value.*
text.extraction.threads	***Integer***	*Number of concurrent threads that will be used for text extraction.* *It's good practice to set this number to be less than or equal to the number of 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 threads for mail extraction.*
mail.extraction.batch	***Integer***	*It indicates the number of emails being processed on each indexing cycle.* *When parameter "**mail.extraction.concurrent**" is enabled, it is good practice to set it to a multiple of the "**mail.extraction.threads**" value.*
mail.extraction.threads	***Integer***	*Number of concurrent threads that will be used for mail extraction.* *It's good practice to set this number to be less than or equal to the number of CPU cores you want to use for mail extraction.*
system.imagemagick.convert	***String***	*Path to ImageMagick convert tool.*

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

system.ocr	***String***	*Path to the OCR text extractor engine.*
system.openoffice.program	***String***	*Path to the LibreOffice executable.*
logo.login	***File***	*Login page logo.* *It automatically tries to resize the image to 316px × 74px (suggested background #F1F3F5).*
logo.mobile	***File***	*Mobile login page logo.* *It automatically tries to resize the image to 161px × 38px (suggested background #F1F3F5).*
logo.report	***File***	*Default logo used in Jasper Reports.* *It automatically tries to resize the image to 150px × 35px (suggested background #FFF000).*
logo.tiny	***File***	*Small logo at the top right corner in the frontend UI.* *It automatically tries to resize the image to 51px × 13px (suggested background: transparent).*
logo.favicon	***File***	*Small icon on the left side of the browser URL.* *The uploaded file must be a valid icon (16px × 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 compiled 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 obtained 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 the pdfimages extraction 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 to the Ghostscript conversion tool.* `/usr/bin/gs`

Windows

Field / Property	*Type*	*Description*
system.imagemagick.convert	***String***	*Path to ImageMagick convert tool.* `C:\openkm\bin\convert.exe`
system.ocr	***String***	*Path to the OCR text extractor engine.* `C:\openkm\tools\Tesseract-OCR\tesseract.exe ${fileIn} ${fileOut}`
system.ffmpeg	***String***	*Path to FFmpeg audio and video conversion tool.* `C:\openkm\bin\ffmpeg.exe`
system.html2pdf	***String***	*Path to html2pdf conversion 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 the pdfimages extraction 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 Ghostscript 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 the mail attribute from an object of type person with sAMAccountName attribute value {0} - this parameter indicates the userId - from node cn=users,dc=company,dc=local and its 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.**** *is used to retrieve the user name.*
- ***principal.ldap.mail.**** *are used to retrieve users' email 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, arguments are injected into option filters. For example, to get the user's email, the application uses parameter {0} to set the argument value - userId - in the filtering options.

Queries that use parameters for 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 as of version 8.1.12* *Class used to connect with LDAP.*
system.login.lowercase	***String***	*By default, it is empty. When connecting to AD (Microsoft Active Directory), this must be set to "**true**", which forces all users to log 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 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 to manually handle these referrals as a 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**", it tries to retrieve all users from the roles list.*
principal.ldap.user.attribute	***String***	*User attribute.*
principal.ldap.user.search.base	***List***	*List of node bases.*
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 node bases.*
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 as of version 8.1.12* *com.openkm.plugin.principal.LdapPrincipalAdapter*

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

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 to use 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.* *It automatically tries to resize the image to 316px × 74px (suggested background #F1F3F5).*
logo.mobile	**String**	*Mobile logo.* *It automatically tries to resize the image to 161px × 38px (suggested background #F1F3F5).*
logo.report	**String**	*Report logo.* *It automatically 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 the* `${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 be downloaded in live mode. If this number is exceeded, the application will notify the user that ZIP creation will be executed in background mode; after it is finished, it will notify the user by email with a download URL. Value **0** indicates an **unlimited number of files**.* *Downloading a few selected files and folders can take time. The application on the server side creates a temporary folder where all data is packed into a ZIP file and then forwarded to the end-user interface.* *100*
download.zip.live.max.file.size	***String***	*Sets the maximum total size of files allowed to be downloaded in live mode. If this size is exceeded, the application will notify the user that ZIP creation will be executed in background mode. After it finishes, it will notify the user via email with a download URL. Value **0** indicates **unlimited size**. Specify size units with KB, MB, GB, or TB; otherwise, the value will be considered bytes.* *Downloading a few selected files and folders can take time. The application on the server side creates a temporary folder where all data is packed into a ZIP file and then forwarded to the end-user interface.* *100MB*
min.search.characters	***Integer***	*Sets the minimum number of characters required to execute a search. If fewer characters are entered, the search button will not be enabled.*

		3

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

AI configuration parameters

This feature is available starting from version 8.1.16.

*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.*

The functionality must be enabled in the user's Profiles for it to be available.

From Administration > AI Prompt, prompts can be defined.

You should only activate one AI model.

*It is possible to **integrate with other AI service 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*

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

		gemini-1.5-flash
gemini.api.key	***String***	*The API key*

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

Workflow configuration parameters

Explain what these parameters are for:

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 be shown.* *After changing this value, you should restart the OpenKM application for it to take effect.*
workflow.adapter.login	***String***	*Login credentials are 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 if 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***	*Determines whether 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 that is used in a workflow. By default, it is set to "true".* *true*

Notification configuration parameters

When a user is subscribed to a document, any change to the document will trigger an email notification. These email messages are created using predefined templates, and these default templates for creating 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' email field v disabled, it will use the default from email address for all emails sent by the appl 'from' email address).* *When the mail server restricts changing the default 'from' mail field (the general@mail.com account to send a mail with the 'from' mail fi configuration is set to prevent spam), then this parameter must be se* *true*
notification.message.subject	**Text**	*The subject of the message sent when a user has chosen to notify about something* *OpenKM - NOTIFICATION*
notification.message.body	**HTML**	*Body of the message sent when a user has chosen to notify about something else.* *ReplaceMessage: ${notificationMessage}* *User: ${userId}* *<#list documentList as doc>Document: ${doc.path}* *</#list>*
subscription.message.subject	**Text**	*Subject of the message sent automatically due to some event. This happens becau* *OpenKM - ${eventType} - ${documentPath}*
subscription.message.body	**HTML**	*Body of the message sent automatically due to some event. This happens because* *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* *OpenKM - Task Manager Notification: ${message}*
task.message.body	***HTML***	*Body of the message sent when there's some change to the task assigned to a user* ***Task:*** *${tasksubject}* ***Start:*** *${taskstart!}* ***End:*** *${taskend!}* ***Project:*** *${taskproject!}* ***Status:*** *${taskstatus!}* ***Note:*** *${tasknote!}*
download.message.body	***HTML***	*Body of the message sent when a user shares a document with others for downloc* *Your archive is ready. Please download from ${url}*
download.message.subject	***String***	*The message's subject is sent when a user shares a document with others for dov* *OpenKM - DOWNLOAD*
download.token.validity	***Integer***	*Range of days during which 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" c* *<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>*

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

```
</tr>
<tr>
<td>
<div style="border-top: solid 1px #88888b; min-height: 5px; max-height
</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:
</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.*
- ***${userName}*** *displays the name of the user who caused the event.*
- ***${userId}*** *shows the user ID of the user who caused the action.*
- ***${notificationMessage}*** *shows the notification message.*

For ***task.message.body*** *property:*

- ***${tasksubject}*** *shows the subject of the task.*
- ***${taskstart}*** *shows when the task started.*
- ***${taskend}*** *shows when the task ended.*
- ***${taskproject}*** *shows the task project.*
- ***${taskstatus}*** *indicates the status of the task.*
- ***${taskknote}*** *shows 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 the Jackrabbit implementation.*

More information about how to create 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***	*Number format for numeration. 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 it to "%02d" to create versions like "01", "02", etc.*

Restarting the application *is required after changing the value of* ***version.numeration.adapter****.*

Also note that if there are already documents in the repository, changing the Numeration Adapter may cause problems because they are not compatible with each other.

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 that you set the correct numeration before you start using the application. *Otherwise, a migration of existing numeration will be needed, 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 there are already documents in the repository, changing the* ***Numeration Adapter*** *may cause problems when they are not compatible with each other.*

If you change the version numeration adapter, you also need to update the existing numeration; for example, migrating 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

Password validation enables the application to set a couple of rules that passwords must meet. By default, the application performs no password checking (***validator.password*** *property is set by default to* ***com.openkm.validator.password.NoPasswordValidator*** *).*

To enable the validator, validator.password must be changed to ***com.openkm.validator.password.CompletePasswordValidator*** *or create 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.* *A numeric value greater than 0.*
validator.password.max.length	***String***	*Maximum length allowed.* *A numeric value greater than 0.*
validator.password.min.lowercase	***String***	*Minimum number of lower-case characters.* *A numeric value greater than 0.*
validator.password.min.uppercase	***String***	*Minimum number of upper-case characters.* *A numeric value greater than 0.*
validator.password.min.digits	***String***	*Minimum number of digits.* *A numeric value greater than 0.*
validator.password.min.special	***String***	*Minimum number of special characters.*

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

		A numeric value greater than 0.

Sample configuration

- *Must have at least 8 characters.*
- *Must have at least 1 upper-case character.*
- *Must have at least 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.min.special	***String***	

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

Switch the application to read-only mode

It can force the whole repository into read-only mode. This is useful when performing maintenance tasks, or during migration when you wish to keep the system available to your users but want to ensure nobody can change anything.

Field / Property	*Type*	*Description*
system.readonly	*Boolean*	*Switch the whole repository to read-only. By default, it 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 **automatically switch to read-only mode** until the process is finished. Do not change it manually; wait until it is 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 superuser ID. By default the superuser ID is okmAdmin.* *okmAdmin*
user.assign.document.creation	***Boolean***	*By default, when a user creates a node, they are added to the node with full permissions. You can disable this behavior by setting this parameter to false. By default, the parameter is set to true.* *true*
user.password.expiration	***Integer***	*Indicates the number of days to force users to change their password. A value of 0 indicates the feature is disabled.* *0*
user.password.reset	***Boolean***	*Sometimes the user loses or forgets their password. When this option is enabled, users on the landing page have the option to reset their password and be notified with a new one by email. By default this option is set to false.* *false*
system.antivir	***String***	*Path to the antivirus.* *An antivirus can decrease performance, taking several seconds to analyze each document and consuming a lot of CPU resources.*
security.access.manager	***String***	*Sets the security access manager that determines how 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* *Take a look at Security configuration*
security.search.evaluation	***String***	*Sets the security search manager and determines how security is evaluated by the 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, starting workflows, adding, removing or modifying property groups (metadata), or compacting history. The default value is empty.* *Values:* • *DOWNLOAD = 1024;* • *START_WORKFLOW = 2048;* • *COMPACT_HISTORY = 4096;* • *PROPERTY_GROUP = 8192;* *For example, to enable download and property groups, set the mask to 9216 (1024+8192).* *9216* *Take a look at Security configuration*
security.live.change.node.limit	***Integer***	*When security changes affect more than the value set in this property, the security changes are applied as a background task.* *100* *Take a look at Security configuration*

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

system.login.lowercase	***Boolean***	*By default it is empty. If connecting to AD (Microsoft Active Directory), it must be set to "**true**", which forces all users to log in with lowercase user IDs. The reason is OpenKM is case-sensitive and Microsoft Active Directory is not.*
principal.adapter	***String***	*You must restart the 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 user names and role names on creation.* ***^[a-zA-Z0-9_]+$*** *Sample regular expression for Arabic:* ***^[\u0600-\u06FF]+$*** *This parameter can only be used in combination with the "**principal.adapter**" parameter value "**com.openkm.plugin.principal.DatabasePrincipalAdapter**".*
browser.password.autocompletion	***Boolean***	*By default it is true (allowed autocompletion). This parameter allows disabling password autocompletion in the login frame. This is a security measure on untrusted clients.*

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

default.security.recursive.role	***String***	*Set the role that is used to identify which users are able to set recursive security.* *ROLE_SECURITY_RECURSIVE*
secure.file.delete	***Boolean***	*Enables safe file deletion from the file system. The process ensures the document can be recovered later from the file system.* *false*
security.login.failed.attempts	***Integer***	*Number of failed login attempts before the user is locked. A value of 0 indicates unlimited failed attempts and the user will never be locked.* *0*
default.task.manager.admin.role	***String***	*Set the role that is used to identify which users can manage all OpenKM tasks.* *ROLE_TASK_MANAGER_ADMIN*
default.version.admin.role	***String***	*Set the role that is used to identify 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 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***	*A 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.workspace** , it is considered an **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 (a pair 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 the 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 show the tab content in the properties tab based on an external URL request when a folder node is selected.

Field / Property	Type	Description
extension.external.tab.folder	**List**	*Each row is a pair of values declaring tab content; 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 show tab content in the properties tab based on an external URL request when a mail node is selected.

Field / Property	Type	Description
extension.external.tab.mail	**List**	*Each row is a pair of values declaring tab content; 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 show tab content in the properties tab based on an external URL request when a record node is selected.

Field / Property	Type	Description
extension.external.tab.record	**List**	*Each row is a pair of values declaring tab content; 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 perception of the user interface.

When documents are uploaded, they are automatically added to the ***Text extraction queue***. *You can see the Text extraction queue at* ***Administration > Statistics > Text extraction queue***.

*There's a crontab task named "**Text extractor worker**" at **Administration > Crontab** that controls the indexing cycle.*



When a document is uploaded, it will not be searchable by content until it is processed by the Text extraction queue.



If the application needs to index a large number of files per day, or you have imported a considerable number 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 require an OCR engine), can use a lot of CPU (up to 100%).

Field / Property	*Type*	*Description*
text.extraction.batch	***Boolean***	*Indicates the number of documents to be processed on each indexing cycle.* *When the parameter "**text.extraction.concurrent**" is enabled, it is good practice to set it to a multiple of the "**text.extraction.threads**" value.* *20*
text.extraction.concurrent	***Boolean***	*Enable or disable the concurrent text extraction threads feature.* *True*
text.extraction.threads	***Integer***	*The number of concurrent threads 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 "**Mail extractor worker**" at **Administration** > **Crontab**, which controls the indexing cycle.*

> *When a mail is uploaded, it will not be searchable by content until it is processed by the Mail extraction queue. Furthermore, attachments won't appear until they are processed.*

Field / Property	*Type*	*Description*
mail.extraction.batch	***Boolean***	*Indicates the number of emails to be processed on each indexing cycle.* *When the parameter "**mail.extraction.concurrent**" is enabled, it is good practice to set it to a multiple of the "**mail.extraction.threads**" value.* *20*
mail.extraction.concurrent	***Boolean***	*Enable or disable the concurrent mail extraction threads feature.* *True*
mail.extraction.threads	***Integer***	*The number of concurrent threads 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 a ZIP file

*It is not good practice to download high volumes of files as ZIP files. In most cases, consider using 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 then be forwarded to the user interface. To prevent blocking the user interface while waiting for process completion, there are two parameters that will notify the user when they have 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 be downloaded 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 be downloaded 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. The 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 perception of the user interface. The antivirus checker can work in live mode (as part of the uploading process: the 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 degrade the user interface and the end user's experience. Take into consideration that to analyse some documents, the antivirus can take 3 seconds or more, which will be added to the upload time.

If the application needs to analyse a large number of files per day, or you have imported a considerable number 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, it is set to **false**.* *There's a **crontab** task (**Administration > Crontab**) named **Antivirus Checker Worker** that controls the background antivirus checker.* *When the background check is disabled, the antivirus is processed during document upload. That can add extra time to the uploading process (in some cases, the antivirus may take more than 3 seconds to check a document). When you perform a large import of files, the total time to complete the operation can be multiplied by 3Ã— or more with this option disabled, because the antivirus check runs as part of the upload*

		process. Our suggestion is to set it to enabled. *true*
background.antivirus.check.threads	***Integer***	*The number of concurrent threads used for antivirus.* *This number should be less than or equal to the number of hardware cores.* *2*
background.antivirus.check.batch	***Integer***	*Indicates the number of documents to be processed on each indexing cycle.* *When the parameter "**background.antivirus.check**" is enabled, it is good practice to set it to a multiple of the "**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 to rebuild 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, it is set to "**flush**". Possible values are "**mass**", "**flush**", and "**delayed**". Take a look at the table below to see the details of each value.*

*You can use the "**hibernate.indexer.batch.size.load.objects**" configuration parameter to indicate to Hibernate how many objects should be handled at a time. To avoid **OutOfMemory** problems, the repository needs to be re-indexed in batches. If the value of this property is too low, performance will be poor, but if it is too high, you can have **OutOfMemory** problems.*

Mode	*Description*
Mass Indexer	*This mode will re-index 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 and 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 it is 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 programs to process documents; for example, it extracts text with an OCR engine, analyses documents with antivirus software, and transforms documents to other formats, among other actions. Sometimes these

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

*processes can take a long time or, for some reason, do not finish correctly, and the process continues running on the OS consuming resources. To prevent this, the parameter "**system.execution.timeout**" sets the maximum allowed execution time for external applications.*

Field / Property	Type	Description
system.execution.timeout	**Integer**	*Set the maximum allowed execution time 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 user activity, but sometimes these actions do not need to be logged and can 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 information 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 actions see Activity log.

Dashboard activity

The OpenKM dashboard displays a global view of application activity. To provide this, the application needs to process a large amount of information, which can decrease overall performance (for example, to get the most recently uploaded documents, OpenKM might execute slow database queries). Therefore, we suggest that very large repositories disable dashboard activity.

Field / Property	*Type*	*Description*
dashboard.user.activity	***Boolean***	*Set to false to disable the feature.*

Lucene

The Lucene search engine can be disabled or configured in several ways. Depending on the configuration, the behavior of the application and overall performance might change.



*These configuration parameters go 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 it is set to "write-sync", which means that an entity modification will not complete until the Lucene index has been properly updated. In the case of "async",*

		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 continue working without waiting for the process to end.*

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

LibreOffice configuration parameters

As part of the application startup, OpenKM executes a LibreOffice service. It's used internally for conversion purposes, for example, to convert DOC files to PDF.



The LibreOffice service can use a lot of hardware resources (CPU at 100%).

Field / Property	*Type*	*Description*
system.libreoffice.program	***String***	*Path to the LibreOffice executable.*
system.libreoffice.dictionary	***String***	*The dictionary used by the search engine tokenizer.*

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

CSV configuration parameters

The CSV ("Comma-Separated Values") 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 Values), such as search results, user lists in administration, and others. The format of the output CSV file can be configured with several parameters to suit your needs.

Field / Property	*Type*	*Description*
csv.format.delimiter	***String***	*Set the delimiter used between columns. By default the character "**;**" is used.*
csv.format.ignore.empty.lines	***Boolean***	*Specify whether empty lines should be processed. By default the value is **false**.*
csv.format.quote.character	***String***	*Set the quote character used to enclose column values. By default it is set to the " character.*
csv.format.skip_header	***Boolean***	*Specify whether the first row contains column names and should not be processed as values. By default it is set to **true**.*
csv.format.comment.indicator	***String***	*Set the character used to indicate the beginning of a comment. By default it 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 CPUs). An aggressive antivirus policy can decrease the performance of the application and affect end users' experience with the user interface.

Antivirus analyzer can work in two modes (depending on the value of the ***background.antivirus.check*** *parameter):*

- *Live mode.*
- *Batch mode.*

> *Antivirus checker in live mode can dramatically decrease end users' experience of the user interface. Take into consideration that to analyse some documents the antivirus can take up to 3 seconds or more, which will be added to the upload time.*

Field / Property	*Type*	*Description*
system.antivir	***String***	*Set the antivirus path.* */usr/bin/clamscan*
background.antivirus.check	***Boolean***	*Enable or disable the background antivirus checker. By default it is set to* ***false****.* *There's a* ***crontab*** *task (****Administration > Crontab****) named* ***Antivirus Checker Worker*** *that takes control of the background antivirus checker.* *For more information Built-in crontab tasks.* *When background checking is disabled, the antivirus runs while uploading any document. This can add extra time to complete the upload process (in some cases more than 3 seconds may be needed for the antivirus to check a document). For example, when you perform a large import of files, the total time to complete the operation can be multiplied by 3x or more with this option disabled. Our suggestion is to enable it.*
background.antivirus.check.threads	***Integer***	*Number of concurrent threads that will be used by the antivirus.* *This number should be less than or equal to the*

		number of hardware cores.
background.antivirus.check.batch	***Integer***	*Indicate the number of documents to be processed in each indexing cycle.* *When the parameter "**background.antivirus.check**" is enabled, it is good practice for it to be a multiple of the **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 in the $TOMCAT_HO*
spring.jpa.properties.hibernate.search	**Boolean**	*This parameter can be used to disable Hibernate Search comp* *able to search documents or folders.* *This parameter must be set in the $TOMCAT_HO*
system.ocr.rotate	**String**	*This parameter forces OCR to process rotated documents.* *The feature is useful, for example, in upside-down* *standard OCR direction will fail.* *The parameter is a collection of degrees separated by the char* *90;180;270;* *More information at Configuring OCR engine.*
system.ocr.spell.check	**Boolean**	*When enabled, the text extracted by OCR is validated by a spe* *The application must have an enabled dictionary;* *"**system.openoffice.dictionary**" in the LibreOffice*
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 limit for search results. For example,*

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

		never get more than 500 results from any query. *500* *The reason the number of results is limited is that huge number of results, especially when the queri methods.*
system.keyword.lowercase	***Boolean***	*This parameter is used to force all keywords to lowercase.* *The application is case sensitive; that means a us "keyone" and "KeyOne", and to the application th To prevent this problem, it is always a good idea lowercase.*
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 parameter to "**UTF-8**".* `URIEncoding="UTF-8"` *ISO-8859-1*
mail.session.debug	***Boolean***	*Enable mail session debugging. It's useful for detecting proble sending mail.*
mobile.theme	***String***	*There are five mobile themes ("a", "b", "c", "d", "e").* *More information about themes at jQuery mobile* *b*
plugin.debug	***Boolean***	*When enabled, shows debug information about the applicatior*
principal.hide.connection.roles	***Boolean***	*Hide "**ROLE_USER**" and "**ROLE_ADMIN**" roles from the a*

		*By default the hidden roles are set by the values o "**default.admin.role**" parameters. Usually "**ROLE** "**ROLE_ADMIN**".*
principal.database.filter.inactive.users	***Boolean***	*Deprecated since version 8.1.12* *Hide inactive users from the available users list.*
restrict.file.name	***String***	*Restrict file name extensions that are not allowed.* *Each file name extension is separated by the char* **~;*.bak*
restrict.file.mime	***Boolean***	*Restrict file name extensions based on existing ones at **Admin***
restrict.duplicated.documents	***Boolean***	*It is not allowed to upload the same document more than once* *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 enabled, the **Administration** > Statistics results take les*
repository.content.checksum	***Boolean***	*When enabled, the document checksum is calculated for each*
system.pdf.force.ocr	***Boolean***	*Force processing of all PDF files with the OCR engine.*
stack.metadata.hidden.properties	***List***	*Filter metadata fields in the metadata stack view on the fronte*
show.records.in.tree	***Boolean***	*When enabled, show record nodes in the tree stack view on the*
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 it is included in the **catdoc** package.* `/usr/bin/xls2csv ${fileIn}`
system.strings	***String***	*Configure the Strings tool.* `/usr/bin/strings ${fileIn}`
system.zbar	***String***	*Configure the barcode extractor tool.* `/usr/bin/zbarimg`
system.profiling	***Boolean***	*The Profiling stats feature provides low-level information to th application performance problems.* *You can also enable profiling from **Administratio***
system.ftp.port	***Integer***	*FTP port.* *On Linux, only the root user can bind to ports like a good idea to bind FTP to another port.*
system.ftp.server	***Boolean***	*When enabled, the application enables the FTP server feature*
user.keywords.cache	***Boolean***	*When enabled, the keywords of nodes are cached.* *The feature provides better performance with less*
user.item.cache	***Boolean***	*When enabled, the application caches the number of nodes ow* *The feature provides better performance with less calculating 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*

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

search.suggestion.minimal.results	***Integer***	*The minimum number of search results required to provide su*
search.suggestion.returned.words	***Integer***	*The number of suggested words. By default it is 3.*
checkin.comment.mandatory	***Boolean***	*Force users to set a comment when uploading a new documen*
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 the 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:/o`
		`/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 that are granted to users witho webservices.*
rss.news	***Boolean***	*Enable or disable news on 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 by the OpenKM server and refre means any change to this parameter will take effe the OpenKM service.*
rss.news.visible	***Integer***	*Number of visible news items at the same time.* *1*
trash.retention.days	***Integer***	*It indicates the maximum number of days nodes in the trash w automatic purge is executed. Default value is **0**, which indicat* *There's a Crontab task named "Purge Old Trash action.* *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 the 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 searching for documents in email attachmer* *false*
property.group.element.default.access	***Boolean***	*When set to true, all the metadata fields are shown to users; w fields are shown.* *Sample scenarios:* • *When the value is set to **true,** all the **metadata fields a** scenario it makes sense to use **revokeUser** and **revoke*** • *When the value is set to **false,** all the **metadata fields** scenario it makes sense to use **grantUser** and **grantRo*** *Additional information is available at Metadata S* *true*

Experimental configuration parameters

Property	*Type*	*Description*
unit.testing.folder	***String***	*Folder path used for unit testing.* */okm:root/okmTesting*
unit.testing.user	***String***	*Username for unit testing.* *okmAdmin*
unit.testing.password	***String***	*Password for unit testing user.* *admin*
unit.testing.url.base	***String***	*Base URL for unit testing.* *http://localhost:8080/openkm/*
extension.digicert.base.path	***String***	*OpenKM base folder path.* */okm:root/invoices*
extension.digicert.keystore.file	***String***	*Keystore file path.* */home/openkm/certificate.p12*
extension.digicert.keystore.password	***String***	*Keystore password.*
extension.digicert.keystore.type	***String***	*Keystore type (JKS or PKCS12).* *JKS*
extension.digicert.private.key.alias	***String***	*Certificate alias.* *b138bef79c0104378101b77b71922c61_9463f621-75bc-4*
extension.digicert.signature.algorithm	***String***	*Certificate signature algorithm.* *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 uploaded.* */okm:root/invoices/upload*
extension.digicert.validate.properties	***String***	*Metadata group fields to validate.* *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 news on the login page.*
rss.news.box.width	***Integer***	*Width of the news box on the login page.* *300*
rss.news.max.size	***Integer***	*Maximum number of news items retrieved from the news server.* *10* *News are cached by the OpenKM server and refreshe any change to this parameter will take effect tomorro OpenKM service.*

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

rss.news.visible	***Integer***	*Number of visible news items at the same time.* *1*
check.node.deletion.integrity	***Boolean***	*Enable or disable node deletion integrity check. When enabled, pr in use by workflows or have children in use (for folders or records should only be disabled when performing mass node deletion oper*
okm.content.security.policy	***String***	*By default, embedding the application in an iframe is only allowed required by the web UI. Default value:* *frame-ancestors 'self'* *To enable additional domains, use the following example:* *frame-ancestors 'self' http://example.com* *The configuration parameter must be set in the **openk** OpenKM service must be restarted 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 for all origins, methods, and headers* *[{"path":"/rest/**", "origins":["*"], "methods":["*"], "headers":[["*"], "methods":["*"], "headers":["*"]}]* *To enable REST API access for specific domains only, use the foll* *[{"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** OpenKM service must be restarted to take effect.* *The String value is a list of objects in JSON format. T relevant because internally the first match will be app "/**" should always be applied at the end.* *More information about CORS at: https://developer.m US/docs/Web/HTTP/CORS*

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

ldap.ignore.partial.result.exception	***Boolean***	*When using Active Directory integration, PartialResultException* *log by default. Setting this to true suppresses these warnings in th* *true*
okm.authentication.external.authentication	***String***	*Enable OpenKM external authentication. This feature allows the* *by third-party providers like OAuth, Azure Cloud, and others.* *okm.authentication.external.authentication=http://localhost*
rest.client.call.timeout	***String***	*Maximum timeout in seconds for REST requests made by OpenKM* *(e.g., jbpm-console).* *15*
optimized.thumbnail.generation	***Boolean***	*Enables optimized thumbnail generation method. Recommended w* *features.* *true*
security.database-query.enabled	***Boolean***	*Enables DatabaseQuery functionality in the administration interfa* *administrators can execute custom database queries.* *false*
security.scripting.enabled	***Boolean***	*Enables Scripting functionality in the administration interface. Wh* *execute custom scripts.* *false*
security.script-validation.enabled	***Boolean***	*Enables validation of scripting statements executed in Scripting. W* *validated before execution to prevent potentially harmful operatio* *true*
security.sql-validation.enabled	***Boolean***	*Enables validation of SQL statements executed in DatabaseQuery* *validated before execution to prevent potentially harmful operatio* *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***	*Tailor-made AutoXchange AX2020 tool. This tool is used to convert a given* */home/openkm/tomcat-9.0.76/bin/ax2020*
extension.cadviewer.linklist	***String***	*Tailor-made LinkList tool. This tool is used to extract text and annotations* */home/openkm/tomcat-9.0.76/bin/linklist*
extension.cadviewer.dwglist	***String***	*Configures the DWGList tool. This tool is used to extract XRefs from a given* */home/openkm/tomcat-9.0.76/bin/dwglist*
extension.cadviewer.sibling.xrefs	***Boolean***	*When this parameter is set to **true** , the AutoCAD files' references must all b* *in mind that you should avoid document name duplication. There is an auto*

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

FTP configuration parameters

Field / Property	*Type*	*Description*
system.ftp.server	***Boolean***	*Enables FTP service. By default, it is disabled. You have to restart OpenKM after changing this parameter.* *false*
system.ftp.port	***Integer***	*The port on which the FTP service listens. By default, it is 2121, although the default FTP port is 21 because it's a reserved port and OpenKM must be run as an administrator to be able to use this port. You have to restart OpenKM after changing this parameter.* *2121*
system.ftp.data.port.range	***String***	*Port range used on the server for data transmission. You should also open these ports in your firewall. By default, the range is 60000 to 60500 and they must 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 connections are proxied. Empty by default.*
system.ftp.debug	***Integer***	*Set the FTP debug flags. The greater the number, the more debug information is shown.* *0*

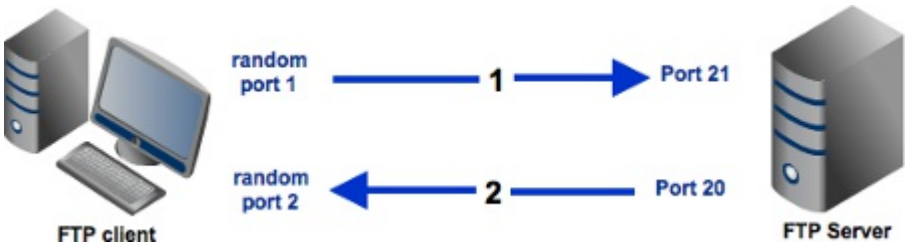
If you change any of these configuration parameters, the changes won't take effect until OpenKM has been restarted.

*CIFS's FTP works better in **Passive Mode**. Therefore, it might not work properly with file browsers such as Nautilus or Nemo.*

Active Mode

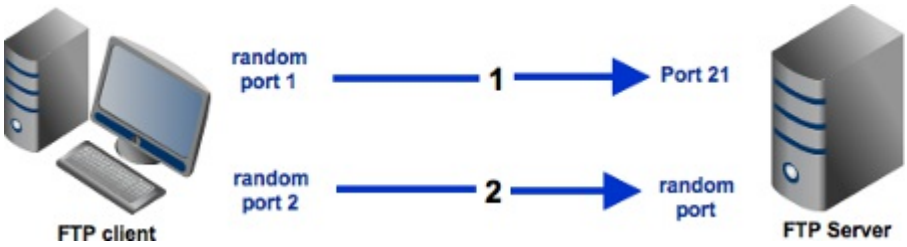
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

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, it would be logical for the server administrator to adapt to the situation and open a selection of ports to support passive mode configurations.



More information 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 default 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 will see an option in the top right corner called "Tenant configuration". If you click this option you will have access to these specific tenant configuration properties. By default it is empty, so you can only see an empty list.*

By clicking 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 tenant (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 must be accessed via a specific 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 aliases, you have to create the subdomain in your DNS server and have it resolve to the server's IP address. 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 record series, file organization, active file locations, file transfer instructions, file retention, 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 the disposition of 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**" in all tabs (Tab Document, Tab Folder, Tab Mail, Tab Record).*
- *In the "**Components tab**" enable the "**Business 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 for recording.*
- *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.*
- *Distinguishing and assigning responsibility for certain types of documents.*

In the file plan, every document must have the following:

Field	*Type*	*Sample*
Name	***String***	*Common records*
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 role	***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 in 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 document MIME types. Every MIME type has one or several file extensions which are used to match the uploaded document's MIME type.

> *Based on the MIME type associated with the uploaded document, the application chooses the associated text extractor. The text extractor works only for a few MIME types; it can easily be extended.*
>
> *For example, if you upload a "sample.doc", its 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 a list of common document 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 the 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 of multiple extensions, each one should be separated by the character ",".* *epub*
Image	*Image file.*

	The image size should be 16x16 pixels.
Search	*When checked, it will be shown in the application's user interface in the search view MIME type list.*
Live Edit	*When checked, this MIME type will be edited using Live Edit when enabled.*

- *Click **Create button**.*

Edit MIME Type

- *Click **Edit icon**  .*
- *In the **Edit MIME type view**, make changes in the fields.*
- *Click **Edit button**.*

Delete MIME type

- *Click **Delete icon**  .*
- *In **Delete MIME type view** click **Delete button**.*

Backup MIME types

- *Click the **Export MIME type icon** at the top right  .*
- *After a few seconds the application will download an SQL file.*

New MIME types

- *Go to the upper right 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 statistics view, graphical information about repository size and its distribution across the distinct root nodes (taxonomy, personal, trash, etc.) is shown. The basic server information, like memory and disk usage, is also shown.

To understand how your repository is growing and make hard disk estimates, these pie charts are useful:

- *Documents by context.*
- *Document sizes by context.*
- *Folders by context.*
- *Emails by context.*
- *Records 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 store their personal - not shared - documents)*
- *Mail (the /okm:mail context, where user emails are stored by default)*
- *Trash (the /okm:trash context, which stores deleted documents, 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 permanently deleted, you should empty your trash or individually purge the documents. Remember that each user has their own trash and is responsible for purging it. There's a feature ***Administration > Utilities > Purge trashes*** *that helps with it.*

Check the status of the Pending task queue

Pending tasks are only used in some application sections where a user action can take longer than usual.

- *Click the menu option* ***Pending task queue*** *at the top right.*
- *The status of the tasks is shown in the* ***Pending task queue view****. This is a list of the tasks 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 deep structure, these actions will be promoted to pending tasks. Meanwhile the user can do other things while they are executed in the background. If the number of documents to process is huge, these tasks may take hours to complete.

Check the status of the Text extraction queue

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

When you upload a document to the application, its content is not extracted immediately but is enqueued for extraction.

- *Click the menu option **Pending text extractor queue** at the top right.*
- *The status of the text extractor process is shown in the **Pending text extractor queue view**. This is a list of the documents to be processed or in progress.*

More information is available at Performance configuration parameters.



Sometimes you might be interested in processing all the documents again in the text extraction queue (you have introduced a new text extractor, etc.), you can find more information at Database description about how to do it.

Check the status of the Antivirus check queue

When you upload a document to the application, the antivirus analysis can be done just in time or enqueued for later.



This option is only shown when the application has been configured to perform antivirus analysis on a batch queue.

- *Click the menu option **antivirus check queue** at the top right.*
- *The status of the antivirus analysis process is shown in the **Antivirus check queue view**. This is a list of the documents to be processed or in progress.*

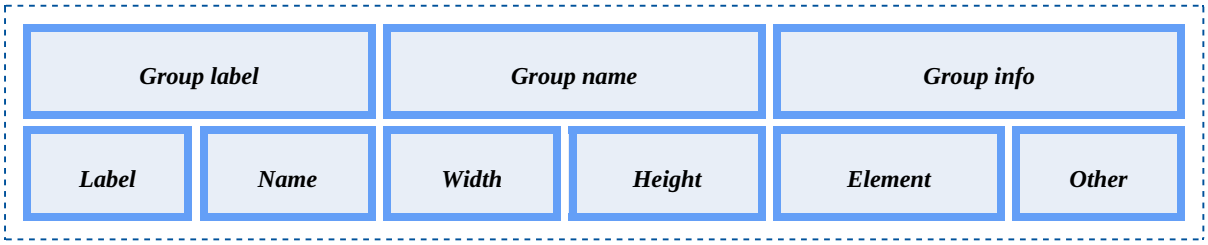
More information is available at Performance configuration parameters.

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

Metadata

The registered metadata groups are shown in the Metadata view. In this view, you can create, modify, or remove metadata definitions.

Property groups are shown in a table structure. The first row (three columns) corresponds to the general property group definition. The following rows are field definitions in the property group.



Description:

Property	*Description*
Group label	*The group label is shown in the user interface on the Property group tab.*
Group name	*Unique group name identifier.* *Two property groups cannot have the same name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*
Group info	*Shows information about:* • *Visibility (the property group is hidden from the user interface).* • *Read-only (the property group cannot be modified from the user interface).*
Field label	*Field label is shown in the user interface on the Property group tab.*
Field name	*Unique property group field identifier.* *Two property group fields cannot have the same name.* *Please use only letters, numbers, and underscores: "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 is used to render multiple values in the field in the user interface view.* *Values:* • *CheckBox* • *Input* • *Iframe* • *Select* • *Separator* • *SuggestBox* • *Text* • *TextArea*
Others	*Shows extra information about the field.*

Edit metadata

To add, modify, or delete metadata, there is only one operation for all: edit the metadata XML definition.

- *Click on*  ***Edit metadata***.
- *In the Edit view,* ***modify the XML***.
- *At the bottom right, click the* ***Edit button***.

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

Metadata XML definition

The application metadata 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 the latest DTD on our website http://www.openkm.com/dtd/property-groups-**{major}.{minor}**.dtd where **{major}** and **{minor}** 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.

*Once the metadata is registered, remember to enable it in the user profile at **Administration** > **Profiles**, **General** tab, and **Metadata Groups**.*

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 a DTD. In the previous example, the **property-groups-3.10.dtd** is used.*
- *The XML begins with the <**property-groups**> tag and ends with the </**property-groups**>. In this hierarchy, the metadata groups are defined.*

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 identifies the **beginning and end** of a new **metadata group definition**.*
- *Each metadata group has a **label** - used by the user interface - **Consulting**.*
- *Each metadata group has a **unique name**, **okg:consulting**.*
- *The metadata group name must always start with "**okg:**" (**O**pen**KM** **G**roup).*

- *Do not use special characters in the name.*

Attribute	*Description*	*Required*
label	*The text shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two metadata groups cannot 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" rather than "camelCase" for better column names.*	*true*
visible	*Show or hide the metadata.* *Allowed values are:* • *true.* • *false.* *By default, the metadata are visible.*	*false*
readonly	*Metadata values cannot be modified from the user interface; they can only be modified via the API.* *Allowed values are:* • *true.* • *false.* *By default, the metadata are modifiable.*	*false*
defaultValueClassName	*Set a custom Java class for setting initial default values.*	*false*

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

Basic example

Available field types:

- *Checkbox*
- *Input*
- *Iframe*
- *Select*
- *Separator*
- *Suggestbox*
- *Text*
- *TextArea*

More information about field type elements can be found at Metadata fields.

> *It's good practice to use the metadata group name as the basis for creating metadata field names. For example, **okp:consulting.code** uses **okg:consulting** as its base (changing **okg** to **okp**). This helps to determine which metadata group is associated with 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, the DTD file must be accessible in some other way; for example, you can store a copy on 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 the DTD file.*
>
> *It can be a file system path (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 in OpenKM

When you register a new metadata definition in OpenKM, the metadata values are stored in two separate tables: one table for current metadata values and another for historical 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, two tables will be created with these columns:

- ***OKM_PGRP_CUR_CONSULTING** (used for current metadata values)*
  - ***RGT_UUID**: ID of the node that 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 historical metadata values)*
  - ***RGT_UUID**: ID of the node that 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 named **RGT_UUID** contains the value of the node UUID.*



Be careful of Oracle's 32-character limitation for table and column names.

We recommend "snake_case" rather than "camelCase" for better column names.

*For example, this definition would create the table field **RGT_PRO_RETURNPLACE**:*

```
<input label="Return place" name="okp:sample.returnPlace" />
```

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

*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
Input	128 characters
Select	128 characters
SuggestBox	128 characters
TextArea	256 characters

*If you have larger metadata values, you should increase the column size 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 when you add a new file. After the column has been created in the database, changing the value of the "**dbColumnSize**" attribute will not update the column size. If you want to update the column size after creation, you should apply the changes directly using a database client.*

```
<?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.*
Script	*Metadata Script*	*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 child elements, validator elements, or option elements. This hierarchy is permitted:

- *Metadata group*
  - *Checkbox*
    - *Validator*
  - *Input*
    - *Validator*

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

- *Iframe*
- *Script*
- *Select*
    - *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	*A unique field identifier.* *Two metadata fields 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**" as the group name and "**okp:consulting**.checkbox" as the field name, changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "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, the 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 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 a size of 128 characters. Because your data length might be greater than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a detailed explanation of metadata storage in the OpenKM database.*	
defaultAccess	*Default security access policy. By default, it 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 the selected web URL into an iframe object. The uuid and propertyGroup will be passed as query parameters in the URL.

Attribute	*Description*	*Required*
label	*The text shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two metadata fields can't have the same name.* ***Name must be unique****.* *The name must start with "**okp:**".* *It's good practice to use a metadata group name as the basis for the metadata field name. For example, if "**okg:consulting**" has been used for the group name, the field name "**okp:consulting**.iframe" can be obtained by changing "**okg**" to "**okp**". That helps to identify the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
url	*Set the URL of the web page to be loaded in the iframe.* *Two parameters are set for 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 in the iframe will be http://www.openkm.com?uuid=03ced012-63cc-4feb-ac26-bb12ded86f73&propertyGroup=okg:consulting* *It's good practice for the URL to start with http:// or https://.*	*true*
width	*The width of the HTML element. It is advisable to set the width attribute to 100% so that it adjusts 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 Script

Feature available since version 8.1.20

The Script field allows executing custom JavaScript code to implement dynamic behaviors in metadata forms.

This field can interact with form elements such as:

- *Input*
- *CheckBox*
- *Select*
- *TextArea*

Attribute	Description	Required
label	*The text is shown as a label in the user interface.* *Although this field has a label attribute, it is typically not displayed in the UI since scripts execute in the background. However, it's useful for administrative purposes and documentation.*	*true*
name	*Unique field identifier.* *Two metadata items 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, using "**okg:consulting**" for the group name results in the field name "**okp:consulting.script**" by changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
source	*Path to the JavaScript file to be executed. The file must be located in the plugins directory.* *The JavaScript file must be placed in the **plugins** directory of the OpenKM installation. For example: **/opt/openkm/tomcat/plugins/metadata_script.js*** *The script is loaded and executed after the form is rendered, allowing it to access and manipulate form elements.*	*true*

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

Interacting with form fields:

The script can interact with other metadata fields using their names as the element IDs. Form fields that can be accessed include:

- ***CheckBox*** *- Checkbox fields*
- ***Input*** *- Text input fields (including date and link types)*
- ***Select*** *- Single and multiple select fields*
- ***TextArea*** *- Text area fields*

*Additionally, each form field row can be accessed using the prefix "**row-**" followed by the field name.*

Basic Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.15//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.15.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" />
    <input label="Script label" name="okp:consulting.script" source="metadata_script.js" />
  </property-group>
</property-groups>
```

JavaScript Example

*Example of a script file (**metadata_script.js**) that demonstrates interaction with form fields:*

```
(function() {
  alert('loading');
  const input = document.getElementById('okp:consulting.input1');
  const row = document.getElementById('row-okp:consulting.input1');
  if (input) {
    input.value = 'modified';
    // Trigger the input event so Vue detects the change
    input.dispatchEvent(new Event('input', { bubbles: true }));

    // Add event listener to detect changes
    input.addEventListener('input', function(event) {
      if (event.target.value.length === 10) {
        alert('Text lenght greather than 10 characters');
      } else if (event.target.value.length === 11) {
        row.style.display = 'none'; // hide the field
      }
    });
  }
  alert('loaded');
})();
```

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

Common Use Cases

The Script field is particularly useful for:

- ***Dynamic calculations*** *- Automatically calculating field values based on other fields*
- ***Conditional visibility*** *- Showing or hiding fields based on user input or conditions*
- ***Field synchronization*** *- Keeping related fields synchronized automatically*
- ***Date calculations*** *- Computing dates based on other date fields (e.g., reminder dates)*
- ***User notifications*** *- Displaying alerts or messages based on field values*

Best Practices

Important considerations when using Script fields:

- *Scripts are executed after the form is fully rendered, ensuring all DOM elements are available*
- *Use* ***document.getElementById('field-name')*** *to access form fields by their name*
- *Use* ***document.getElementById('row-field-name')*** *to access the entire row containing a field*
- *When modifying field values programmatically, dispatch an 'input' event with* ***bubbles: true*** *to ensure Vue.js detects the change*
- *Wrap your code in an IIFE (Immediately Invoked Function Expression) to avoid polluting the global namespace*
- *Scripts are removed from the DOM when the form component is destroyed, preventing memory leaks*
- *Test your scripts thoroughly, as errors can affect the entire metadata form functionality*

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

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 fields 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 good practice. For example, if "**okg:consulting**" has been used as the group name, use "**okp:consulting**.select" by changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "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, width values must be specified as percentages, 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	*This 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 a size of 128 characters. Because your data length might be greater than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a detailed explanation of metadata stored in the OpenKM database.*	
defaultAccess	*Default security access policy. By default, it is set to "grant" so all users will see the property.*	

Child elements:

- *Option. Pairs 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 case a hierarchical relationship is set.*
- ***Validator*** *(more information at Metadata Validator element)*
- ***Security*** *(more details at Metadata Security element)*

***** *There are 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. They cannot be combined; 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 an SQL query. This SQL statement should be put in 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"
```

```
                    "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 option values. In 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**: Gets recommendations based on document content (use String.contains).*
- ***com.openkm.form.suggestion.DocumentContentTokenizerSuggestion**: Gets suggestions based on document content (use StringTokenizer).*
- ***com.openkm.form.suggestion.DocumentTermsSuggestion**: Gets recommendations 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 option values 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**" in 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 obtain 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 items 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, the metadata group name "**okg:consulting**" has been used as the basis 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 underscores: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* *In the case of KCenter, the UI must use width values based on percentages, 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>
```

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	*A unique field identifier.* ⚠ *Two metadata fields 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**" as the group name; the field name "**okp:consulting**.suggestbox" is formed by changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the 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 insert the user input.*	*
valueQuery	*A SQL query to obtain the value of an identifier. Use {0} to insert the identifier.*	*
className	*A class is used to get the selected options.*	*
filterMinLen	*The minimum number of characters in the input to begin filtering 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 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 a size of 128 characters. Because your data length might be greater than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a detailed explanation of metadata storage in the OpenKM database.*	
defaultAccess	*Default security access policy. By default, it 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 are three options to get suggested box values (using **one of them is mandatory**):*

- *filterQuery + valueQuery*
- *className*

No other combination is allowed.

Based on filterQuery and valueQuery

*Now **filterQuery** and **valueQuery** can be regular SQL queries.*

You should only use {0} as the parameter for the user's input 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 3.10//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 retrieve a list from external applications or for other purposes, your class can be built to obtain 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.* *Since version 7.1.41, the label can render HTML. In the case of an 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	*A unique field identifier.* *Two metadata fields 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 good practice. For example, if the metadata group name is "**okg:consulting**", the corresponding field name can be "**okp:consulting**.text" by changing "**okg**" to "**okp**". This helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* *For KCenter, the 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 items 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**" as the metadata group name to create the field name "**okp:consulting.textarea**" by changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "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, the 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*
readonly	*The 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 detailed explanation of metadata storage in the OpenKM database.*	
defaultAccess	*Default security access policy. By default, it 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, properly formatted information 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	*Email value.*
url	*URL value.*
maxlen	*Maximum allowed length.*
minlen	*Minimum required length.*
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	*Create your own 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 a 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 among these predefined validators. For example, imagine you want to restrict an input to lowercase ASCII characters:

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 distinct value to select.

Attribute	Description	Required
label	*The text shown as the label in the user interface.*	*true*
value	*Unique value identifier.* ⚠ *Two options cannot have the same value. **Value must be unique**.* ⚠ *It's recommended to use only characters that match this regular expression: [a-z_] (a to z in lowercase and underscore, no spaces)*	*true*
parentValue	*When present, it specifies the option's parent.* ⚠ *The parent must be present in another select.*	*false*
selected	*When present, it specifies that an option should be pre-selected when the form is rendered.*	*false*
cssClass	*CSS class.* ℹ *More information about creating your own styles is available at CSS.*	*false*

Parent:

- *Select. For more information, please see: Metadata Select field.*

Basic select with selected option

- *The attribute selected=true, when present, specifies that an option should be pre-selected when the page loads.*

```
<?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">
```

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

```
<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**" has a parent relation with "**okp:consulting.parent**". It is defined by **parentElement="okp:consulting.parent"**.*
- *The option value "**benefits**" of the field "**okp:consulting.children**" has a parent value "**directives**" in the 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 items 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, using "**okg:consulting**" for the group name results in the field name "**okp:consulting.input**" by changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
type	*Type of value. Allowed types:* • *text.* • *date.* • *link.* • *folder.*	*true*
value	*Specifies a default value for this component.* *There is a special case for date types. Data are stored in the format "**yyyyMMddHHmmss**". Be aware of this format when using the OpenKM API.*	*false*
width	*The width of the HTML element.* *In the case of KCenter, the UI must use width values based on percentages, for example:*	*false*

	• *width="50%" will work* • *width="100px" will be ignored in the KCenter UI.*	
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 whether 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, making 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 a size of 128 characters. Because your data length might be greater than 128 characters, the administrator should update the default column size.* *Take a look at the Metadata XML definition documentation section for a*	

	detailed explanation of metadata stored in the OpenKM database.	
defaultAccess	*Default security access policy. By default, it is set to "grant", so all users will see the property.*	

Child elements:

- ***Validator*** *(more information on Metadata Validator element)*
- ***Security*** *(more information on 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 fields cannot 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**" as the basis 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 underscores: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* *In the case of KCenter, the UI must use width values based on percentages; 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 allows you to sort components inside.

Attribute	*Description*	*Required*
name	*Unique field identifier.* *Two metadata fields can't have the same name.* ***Name must be unique****.* *The name must start with "****okp:****".* *It's good practice to use a metadata group name as the basis for a metadata field name. For example, if the metadata group name* ***okg:consulting*** *has been used, the corresponding field name becomes* ***okp:consulting.separator*** *by changing* ***okg*** *to* ***okp****. That helps to identify the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
width	*The width of the HTML element.* *In the case of KCenter UI, you must use width values based on %, for example:* • *width="50%" will work* • *width="100px" will be ignored by 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 element.

*There's a configuration parameter named "**property.group.element.default.access**" that, when it is set to true, shows all the metadata fields to users; when it is set to 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, it makes sense to use **revokeUser** and **revokeRole**.*
- *When **property.group.element.default.access** is **disabled**, none of the **metadata fields are shown by default**. In this scenario, it makes sense to use **grantUser** and **grantRole**.*

For more information about this parameter, go to Administration > Configuration parameters.

Remember that members and the administrator role (usually ROLE_ADMIN) are not affected by these kinds 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	*Sets access for a user.*
grantRole	*Sets access for members of a role.*
revokeUser	*Revokes access for a user.*
revokeRole	*Revokes access for members of a 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 access 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>
```

Administrators (users with the ROLE_ADMIN role) will always see all fields.

Basic revoke example

Revoke action has priority over the grant action.

- *Revoke access to a field for a specific user: the user* ***testUser*** *does not have access to the metadata field named* ***okp:consulting.input1***, *while other users who have access to the metadata group still have access to it. The* ***defaultAccess*** *property is not present, so the default policy ("****grant****") is used, which 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>
```

Administrators (users with the ROLE_ADMIN role) will always see 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 the following:

- *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 to 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 read-only 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 on the top menu*  ***Authentication button*** *and then click on the* ***Users link***.

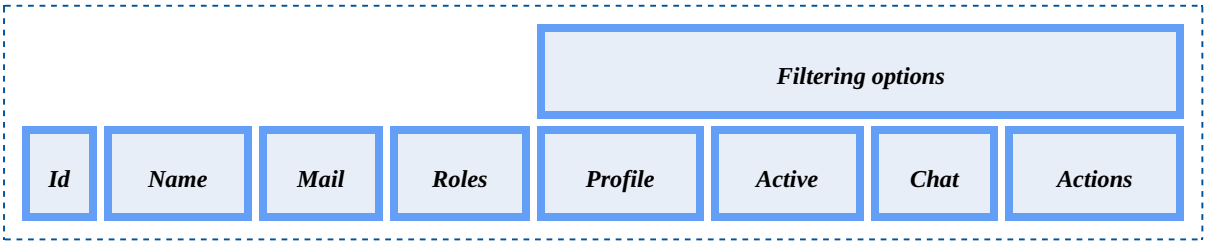
By default, the application manages users and roles in its own database.

It is also possible to connect the application to an external directory (LDAP directory, for instance).

The login module can be extended to 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 read-only mode. Almost all actions are restricted.

Users are shown in a table structure.



User parameters:

Field	*Description*	*Required*
Id	*The id that the user will use to log in.* *Two users cannot have the same id.*	*true*
Name	*User name.*	*true*
Mail	*User email address.*	*true*
Roles	*Roles - groups - of which the user is a member.* *The user must be in at least one of the 'ROLE_ADMIN' or 'ROLE_USER' roles.*	*true*

Profile	*Each profile is a set of options available in the user interface.*	*true*
Active	*Sets whether the user is enabled.*	*true*
Chat	*Indicates if the user is connected to the chat.* *Although a user can have the application open in several browser tabs, only one of them will be logged into the chat.*	*false*
Action	*Available actions:* • *Edit a user.* • *Delete a user.* • *Set the profile.* • *Set the user tenant.* • *Configure mail accounts. More information here Manage user mail accounts.* • *Show the activity log of the user.* • *Delete the user's 2FA secret*	*
** Mail account and Twitter account are optional.*		

Create a user

- *On the top right of the table, click on the* ***New user icon***.
- ***Fill all the fields***. *Remember that all users must be in at least one of the '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 accounts view, the mail user accounts are shown. These mail accounts are used by the application to periodically import mail into the application repository.

***To manage user mail accounts**, **from the Users view** click on*  ***Email account icon**.*

Mail account hierarchy

- *Mail account*
  - *Filter*
    - *Rules*

- *The mail account sets the data connection to the mail server to retrieve mail.*
- *Filters are optional and set the application location - folder - where to store mail.*
- *Rules are optional and set a couple of rules that control which mails are valid to import.*

When a filter is not set, by default the application will store mail in the /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 the mail protocol value is IMAP or IMAPS. In the case of POP3 you can leave it empty; otherwise it needs to be set to INBOX.*	*false*
Mail mark seen	*Sets the seen status on the mail server.* *Available options:* • *Keep (keep as is).* • *Yes (mark as seen)* • *No (mark as not seen).*	*true*
Mail mark deleted	*Sets mail to be deleted.* *When checked, the mail will be marked for deletion from the mail server.*	*true*

Active	*Sets whether the mail account is enabled or not.*	*true*
Actions	*Available actions:* • *Edit a mail account.* • *Delete a mail account.* • *Set mail account configuration filters.* • *Browse emails on 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***.

Browse emails on the mail server

- *Click on the* ***Server list icon***.

> *In this view, mail can be imported directly.*

Manage mail account filters

In the mail account configuration, rules can be set to automatically store mails in different application folders.

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

For example, filters can help automatically catalog mails based on a "subject" value or "from" value, among others.

Mail account filters are shown in a table structure.



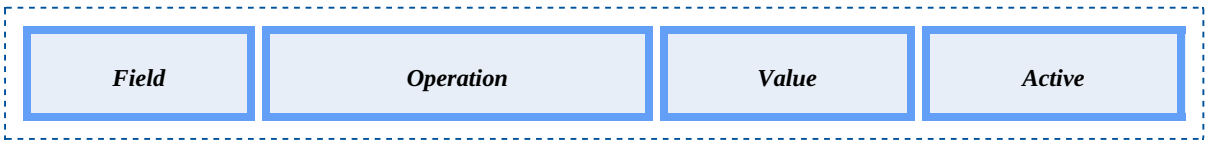
Mail account filter parameters:

Field	*Description*	*Required*
Folder	*Sets the destination folder.*	*true.*
Grouping	*Enables automatic cataloging by **year/month/day**.* *To prevent having too many mails in the same folder, we recommend enabling mail grouping.* *The year/month/day folders are automatically created by the application.*	*false.*
Active	*Sets whether the filter is enabled or not.*	*true.*
Actions	*Available actions:* • *Edit a filter.* • *Delete a filter.* • *Show rules.*	

At minimum, a filter must set a destination folder; it can also have additional rules.

A rule can only be set in a filter.

Mail account 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	*Sets the operation that will be done for the evaluation process.* *Available options are:* • *Contains* • *Equal*	*true.*
Value	*Sets the value that will be used for the evaluation process.*	*true.*
Active	*Sets whether the rule is 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***.
- *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 on the top menu*  ***Authentication button*** *and then click on the 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 to 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 read-only 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	*Indicates whether the role is enabled or not.*	*true*
Action	*Available actions:* • *Edit a role.* • *Delete a role.* • *Set a tenant for a role.*	

Create a role

- *At the top right of the table, click on the* ***New role icon***.

- ***Fill in 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***.

> *The ROLE id cannot be modified to prevent repository security inconsistencies. This operation can only be performed at the database level.*

Delete a role

- *Click on the* ***Delete role icon***.
- *Click on the* ***Delete button***.

Export the roles

- *At 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 on the top menu*  ***Authentication button*** *and then click on the submenu* ***Logged Users link***.

Logged users are shown in a table:

User	***Session ID***	***Remote IP***	***Remote Host***	***Creation***	***Last Access***

Real-time user activity log

- *Click on*  ***Activity log icon***.
- *A table will be shown 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 send messages across the user interface.

Helps administrators send a message to all logged-in users to indicate the system will be halted and automatically log out the users 5 minutes after they receive 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 other 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.*

The queue of messages is shown in a table structure:

Date	***Action***	***Message***	***To All***	***To Users***	***To Roles***

Message parameters:

Field	*Description*	*Required*
Date	*Date of the message.* *Automatically set by the application.*	*true*
Action	*Available values:* • *Logout (force users who receive it to log out)*	*false*

Message	*The text of the message.*	*true*
To All	*When present, the message is sent to all users.*	*
To Users	*When present, indicates a set of users to send the message.*	*
To Roles	*When present, indicates a set of users who are members of any of these roles.*	*
** At least one of the options.*		

Send a message

- *Click on the*  ***New message icon***.
- ***Fill in 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 in the queue

- *In the top right, click on*  ***Clear button***.



This operation cannot be reverted. The entire message queue will be deleted.

Refresh the message queue

- *In 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 administrators set the available interface features for each user.

For example:

- *Restricts the user interface, hiding search and dashboard tabs and almost all toolbar buttons.*
- *Restricts the user interface, showing only the search view.*

The application, and especially the user interface, comes by default with several features and extensions.

Not all users need access to everything, especially at the beginning when a lot of features can cause confusion for the end user.

Our suggestion is to go from fewer to more features as a natural process of learning the application.

Profiles are shown in a table structure.

Name	Active	Actions

Profiles are divided into 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 whether the profile is enabled.*

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 notifications on the upload popup.*
Notify external users	*Enables or disables sending notifications to external users.* *By default, the application is restricted to users' email addresses.*

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

	For example, sending documents to an external email address.
Increase version	*When enabled, users can choose whether to upload a document as a minor or a major version.*
Add document 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	*Sets 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 the top-right quick search in the UI.*
Recursive security change	*Enable or disable the checkbox for applying security changes recursively by default.*
Copy related information	*Enable or disable copying attributes when 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 one.



If an automation definition is present, these parameters have no effect.

The automation feature covers these cases and more.

Field	*Description*
Metadata	*Sets metadata that will be required 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 to be 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 categories.*
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 for 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 in the file explorer view.*
Page list	*Sets the distinct values that determine the number of elements shown per page.* *For example, the value "10;25;50;100" sets the possible number of elements per page to 10, 25, 50, or 100.* *From the user's view, they can choose one of these options.* *When pagination is enabled, at least one value must be set.* *Use the character ";" to separate the distinct values.*
Type filter	*When enabled, a type filter is shown, which helps the user show only documents, folders, records, or mails at the same time. ****
Show folders by default	*Show or hide folders by default. ****

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

Show documents by default	*By default, shows or hides documents. **
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.) must be active; otherwise they are not taken into consideration and all types are shown.*

Section General

Shortcuts are key combinations for a faster way of accomplishing a task.

Field	*Description*
Rename	*Executes the rename action on the selected node.* `F2`
Copy	*Executes the copy action on the selected node.* `CTRL+SHIFT+C`
Cut	*Executes the 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`
	Shows a pop-up for user interface testing.

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

Testing	*CTRL+G*
Log	*Shows the user interface log.* *CTRL+Z*
Create Folder	*Executes the create folder action on the selected node.* *CTRL+N*
Refresh	*Executes the 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 action.* *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 thesaurus in the stack panel.*
Templates	*Shows or hides the 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 in the 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 a user logs into the application.* *Available options:* • *Desktop* • *Search* • *Dashboard* • *Administration* *You must set an enabled view.*
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.* *Administration view is only visible to users with "ROLE_ADMIN".*

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 document version download.
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 logs into 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 the metadata 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 the advanced search view.*
Export ZIP	*Enables 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 performing searches.

Field	Description
Folder	*Shows or hides the folder selector.*
Category	*Shows or hides the category selector.*
Type	*Shows or hides the node type.* *In the search view, four types are shown:* • *Document*

	• *Folder* • *Mail* • *Record*
Document type	*Shows or hides the document type list.* *Contains a list of MIME types.* *To set the element to be shown in the list see MIME types*
Notes	*Shows or hides 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 take precedence over toolbar options or contextual menu options.

Hierarchy:

- *Menu*
    - *Toolbar*



Preference hierarchy. Upper levels take control over lower levels. If an upper option is disabled, it will also be disabled at the lower level.

For example, when the download menu option is disabled, it will not be shown on the toolbar even if it is enabled there.

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 the manage bookmarks menu option.*

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

Add bookmark	*Shows or hides the add 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 tries to open the default home.* *The default home is /okm:root, which can be changed by the 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, it helps the administrator find possible UI bugs.*
Administration	*Shows or hides the administration menu option.* *Administration opens in 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 (Optical 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.* *For example, this option is shown in the contextual menu in 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 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 remove 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 that occurred on a user's subscriptions. These notifications can be sent as an immediate notification email or with all the messages grouped in a single email delivered once a day.

Grouped notifications are sent by a crontab task named "Send Grouped Notifications".

Notifications are shown in a table structure:

Immediate	***Grouped***

*By default **these two groups are empty and no notifications will be sent**. If you want to work with notifications, you have to choose which events you want to subscribe to.*

Events

Common

Notification	*Description*
COMMON	*Common event.*
COPY_ATTRIBUTES	*Copy attributes*

Document

Notification	*Description*
CANCEL_DOCUMENT_CHECKOUT	*Cancel the editing of a document.*
CHECKIN_DOCUMENT	*Update a document.*
CHECKOUT_DOCUMENT	*Mark the document for editing.*
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 permanently removed from the 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 permanently removed from the 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 permanently removed from the repository.*
RENAME_MAIL	*Change 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.*

	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 permanently removed from the repository.*
RENAME_RECORD	*Change 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

Sets 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 document menu option.*
Update	*Shows or hides the update document menu option.*
Cancel Edit	*Shows or hides the cancel edit 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 modified date column.*
Author	*Shows or hides the author column.*
Version	*Shows or hides the version column.*
Column 1 to Column 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 set distinct relations between nodes (documents, folders, mails, records). In relation types the relation type definitions are set.

There are three types of relations:

- *Bidirectional*
- *Parent-child*
- *Many-to-many*

Relation types are shown in a table structure.



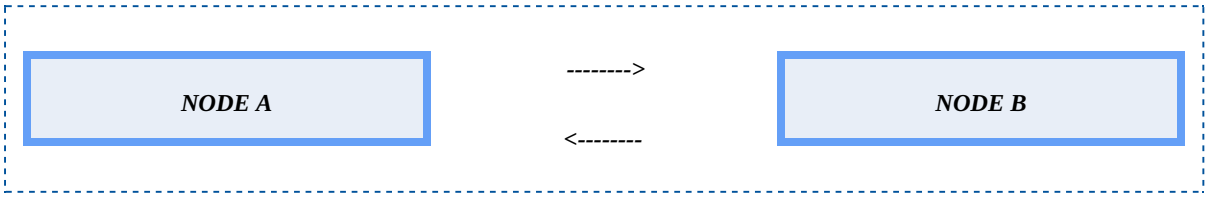
Types

Bidirectional

This relation is always set between two nodes.

A is linked to B as B is linked to A. The relation that links both elements is the same.

An example of this relation type is the "Language translation" relation. A is a translation of B, as B is a translation of A.

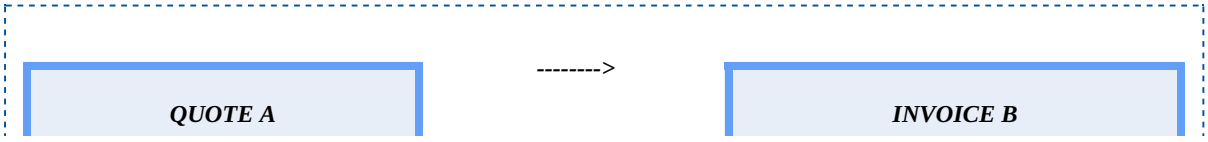


Parent - child

This relation 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 different.

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.

An example of this relation type is "Stapling". A, B, C, etc. are stapled.

Create a relation

- *In the top right, click on* ***New relation type icon***.
- ***Choose one type*** *(bidirectional, parent-child, or many-to-many).*
- ***Fill in the fields***.
- *Click the* ***Create*** *button.*

For the bidirectional type, you need to fill the "Title" field.

For the parent-child type, you need to fill the fields "Title parent to child" and "Title child to parent".

For many-to-many, you need to fill the "Title" field.

Edit a relation

- *Click on* ***Edit a relation icon***.
- ***Modify the fields***.
- *Click the* ***Edit*** *button.*

Delete a relation

Only relations that are not used can be deleted. If some nodes are using that relation type, then the relation cannot be deleted.

- *Click on* ***Delete a relation icon***.
- *Click the* ***Delete*** *button.*

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

CSS

The application allows you to set your own CSS - Cascading Style Sheets - in the user interface.

The CSS applied to the user interface is set in the Profile General tab section.

This helps the administrator change the look and feel of the application.

For example, you can change the font size for Cyrillic or logographic languages.

The CSS is shown in a table structure.

Name	Context	Active	Actions

Create a new CSS

There cannot be two CSS entries with the same name.

- *At the top right, click on the **New CSS icon**.*
- ***Fill in the fields**.*
  - *Name.*
  - *The context can be frontend or mobile.*
  - *Active.*
- *Click the **Create button**.*

Edit a CSS

- *Click on **Edit CSS icon**.*
- ***Modify the fields**.*
- *Click the **Edit button**.*

Delete a CSS

- *Click on **Delete CSS icon**.*
- *Click the **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 the* ***Clone button***.

Download a CSS

- *Click on*  ***Download CSS icon***.
- ***After a few seconds, a CSS file will be downloaded***.

Samples



OpenKM 7.1.x ***comes with two UIs****, the classical UI (frontend) and a new one named KCenter.*

Because the technologies used in both UIs are quite different, changes to the CSS should take into account 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 in OpenKM version 7.1.31 and later.



For Arabic, several fonts can be used:

- *"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;
}

/* verticalbar */
.okm-headerIcon-rtl {
 padding-bottom: 125px !important;
}
```

Reports

The application comes with Jasper reports engine. This helps administrators create their own reports.

The JasperReports Library is one of the world's most popular reporting engines. It can use data 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 on how to build your own reports is available at Creating your own reports.

More information about the Jasper report engine version included with the application is available at Application version compatibility table.

*Keep in mind that any font used in the reports should be installed on the OpenKM server. For example, if you are using Arial (the sample provided reports use this font), you should ensure the 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 Red Hat or CentOS, the package is **msttcore-fonts**. You need to restart the server after this package is installed.*

If you are not sure 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

- *At the top right, click on* ***New report icon***.
- ***Fill in the fields***.
- ***Choose a report file*** *(*.rep or *.jrxml)*
- *Click the* ***Create*** *button.*

Edit a report

- *Click on*  ***Edit report icon***.
- ***Modify the fields***.
- ***Optionally choose a report file*** *(*.rep or *.jrxml)*
- *Click the* ***Edit*** *button.*

Delete a report

- *Click on*  ***Delete report icon***.
- *Click the* ***Delete*** *button.*

Show 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 the report is executed.*

- *Click on*  ***Report params icon***.

Execute a report

- *Click on the*  ***Execute report icon.***
- ***Fill in the fields*** *(when the report comes with parameters).*
- *Choose a report output format*
  -  *PDF format.*
  -  *RTF format.*
  -  *CSV format.*

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

Report pack

Most of these reports are based on the Activity log events audit trail.

For more information Activity log.

Template reports

Name	Uses Activity log	Requires parameters	Description
JavaReportPattern.jrxml	No.	No.	Template pattern that uses Java to retrieve data.
SQLReportPattern.jrxml	No.	No.	Template pattern that uses SQL to retrieve data.
subreport-sample.rep	No.	No.	Subreport sample. *Support for subreports is available starting from version 8.1.15.*

Authentication

Name	Uses Activity log	Requires parameters	Description
UsersLogin.rep	Yes.	Yes.	User logins, filtered by date range.
UsersLogout.rep	Yes.	Yes.	User logouts, filtered by date range.
UsersSessionExpiration.rep	Yes.	Yes.	Users whose sessions expired, filtered by date range.

Documents

Name	Uses Activity log	Requires parameters	Description
DocumentCanceledCheckout.rep	Yes.	Yes.	Lists documents for which editing was

			canceled, filtered by date range.
DocumentCheckin.rep	*Yes.*	*Yes.*	*Lists documents that were updated, filtered by date range.*
DocumentCheckout.rep	*Yes.*	*Yes.*	*Lists documents that are being edited, filtered by date range.*
DocumentCreate.rep	*Yes.*	*Yes.*	*Lists documents that were created, filtered by date range.*
DocumentDelete.rep	*Yes.*	*Yes.*	*Lists documents that were deleted, filtered by date range.*
DocumentGetChildren.rep	*Yes.*	*Yes.*	*Shows when users listed documents in a folder or record. Results are filtered by date range.* *To list documents in a folder, an internal method named "getChildren" is called.*
DocumentContent.rep	*Yes.*	*Yes.*	*Shows when users retrieved the content of a document. Results are filtered by date range.* *For example, when a user downloads or previews a document (among other actions), they are retrieving the document's content.*
DocumentContentByVersion.rep	*Yes.*	*Yes.*	*Shows when users retrieved a specific version of a document's content. Results are filtered by date range.*
DocumentGetProperties.rep	*Yes.*	*Yes.*	*Shows when a user retrieved the properties of a*

			document. Results are filtered by date range. *For example, when a document is selected, an internal method named "getProperties" is called to fill the contents of the properties tab in the user interface.*
DocumentGetVersionHistory.rep	*Yes.*	*Yes.*	*Shown when users retrieve the list of all versions of a document. Results are filtered by date range.*
DocumentGetPropertyGroupProperties.rep	*Yes.*	*Yes.*	*Shows when the fields of a metadata group have been retrieved.* *For example, when a metadata group is shown in the user interface, an internal method named "getPropertyGroupProperties" is called to fill the metadata group values.*
DocumentLock.rep	*Yes.*	*Yes.*	*Lists documents that were updated, filtered by date range.*
DocumentMove.rep	*Yes.*	*Yes.*	*Lists documents that were moved, filtered by date range.*
DocumentPurge.rep	*Yes.*	*Yes.*	*Lists documents that were purged, filtered by date range.* *When a document is purged, it will permanently disappear from the application repository.*
DocumentRenamed.rep	*Yes.*	*Yes.*	*Lists documents that were renamed, filtered by*

			date range.
DocumentUnlock.rep	*Yes.*	*Yes.*	*Lists documents that were unlocked, filtered by date range.*
DocumentsAccessibleByUserOrdered.jrxml	*No.*	*No.*	*Retrieves a list of available documents (security filtered) ordered by folder name.* ⚠ *Optimized for CSV format.* *Not recommended for large repositories; it can take a long time.*
DuplicatedFiles-v-7.1.23.jrxml	*No.*	*No.*	⚠ *This report can only be used with version 7.1.23 or higher.* *List of duplicated files in the application repository, based on document checksum.*
DuplicatedFiles.jrxml	*No.*	*No.*	⚠ *This report can only be used with version 7.1.22 or lower.* *List of duplicated files in the application repository, based on document checksum.*
ListDocumentsWithChecksum.rep	*No.*	*Yes.*	*List of all documents in some folders and subfolders with checksum values.*

Folders

Name	*Uses Activity log*	*Requires parameters*	*Description*
FolderCopy.rep	*Yes.*	*Yes.*	*Lists folders copied, filtered by date range.*
FolderCreate.rep	*Yes.*	*Yes.*	*Lists folders created, filtered by date range.*

FolderDelete.rep	*Yes.*	*Yes.*	*Lists folders deleted, filtered by date range.*
FolderGetChildren.rep	*Yes.*	*Yes.*	*Shows when users listed folders in a folder or record. Results are filtered by date range.* *To list folders in a folder, an internal method named "getChildren" is called.*
FolderGetContentInfo.rep	*Yes.*	*Yes.*	*Shown when the internal method named "getContentInfo" has been executed.* *To perform some operations, it is necessary to calculate folder depth, for example before starting a folder ZIP download.* *There's an internal method named "getContentInfo" that does it.*
FolderGetProperties.rep	*Yes.*	*Yes.*	*Shows when a user retrieved the properties of a folder. Results are filtered by date range.* *For example, when a folder is selected, an internal method named "getProperties" is called to fill the contents of the properties tab in the user interface.*
FolderMove.rep	*Yes.*	*Yes.*	*Lists folders that were moved, filtered by date range.*
FolderPurge.rep	*Yes.*	*Yes.*	*Lists folders that were purged, filtered by date range.* *When a folder is purged, it will permanently disappear from the application repository.*
FolderRename.rep	*Yes.*	*Yes.*	*Lists folders that were renamed, filtered by date range.*
FolderSecurity.rep	*No.*	*Yes.*	*Lists folders and subfolders showing security settings regarding roles and/or users.*

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

FolderGetSize.rep	*No.*	*Yes.*	*Lists folders and subfolders showing the size of each one.*

Mails

Name	*Uses Activity log*	*Requires parameters*	*Description*
MailCreate.rep	*Yes.*	*Yes.*	*Lists mails that were created, filtered by date range.*
MailGetProperties.rep	*Yes.*	*Yes.*	*Shown when a user retrieves the properties of a mail. Results are filtered by date range.* *For example, when a mail is selected, an internal method named "getProperties" is called to fill the contents of the properties tab in the user interface.*
MailPurge.rep	*Yes.*	*Yes.*	*Lists mails that were purged, filtered by date range.* *When a mail is purged, it will permanently disappear from the application repository.*
MailRename.rep	*Yes.*	*Yes.*	*Lists mails that were renamed, filtered by date range.*
MailMove.rep	*Yes.*	*Yes.*	*Lists mails that were moved, filtered by date range.*
MailCopy.rep	*Yes.*	*Yes.*	*Lists mails that were copied, filtered by date range.*
MailGetChildren.rep	*Yes.*	*Yes.*	*Shows when users listed mails in a folder or record. Results are filtered by date range.* *To list mails in a folder, an internal method named "getChildren" is called.*

Others

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

Name	Uses Activity log	Requires parameters	Description
TenantUserQuota.jrxml	No.	No.	Document size (in MB) by user and tenant.
WorkflowWorkload.jrxml	No.	No.	Each workflow task and the time spent on it.
RegisteredUsers.jrxml	No.	No.	Registered users.

Subreports

Quick guide on 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 the subreportExpression.*
- *Remove any file path references from the subreportExpression.*

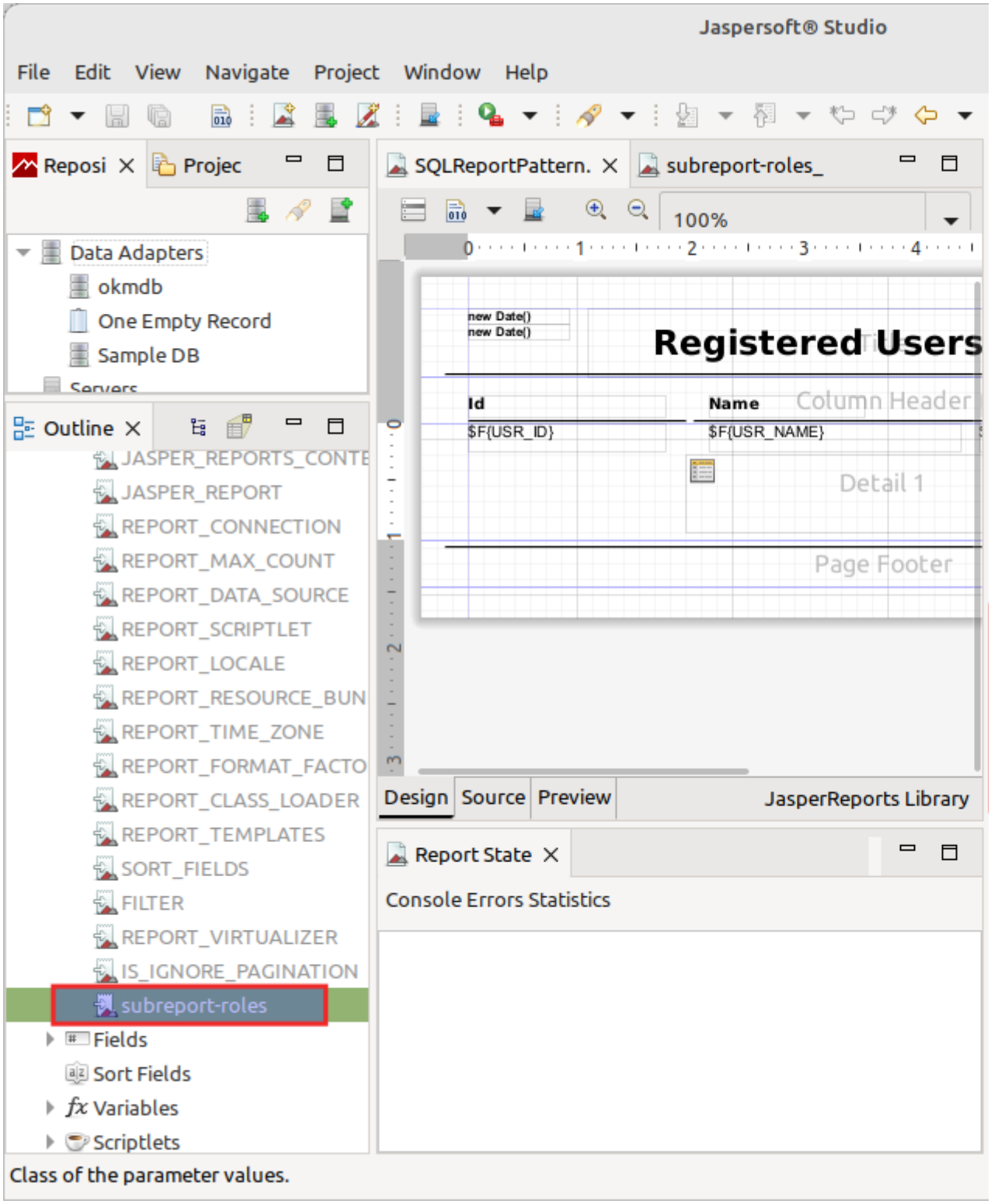
How it works in the background:

- *The OpenKM core loads the subreport from the REP (ZIP file).*
- *It adds it to the params map with the correct name.*
- *JasperReports finds it via the parameter reference instead of a file path.*

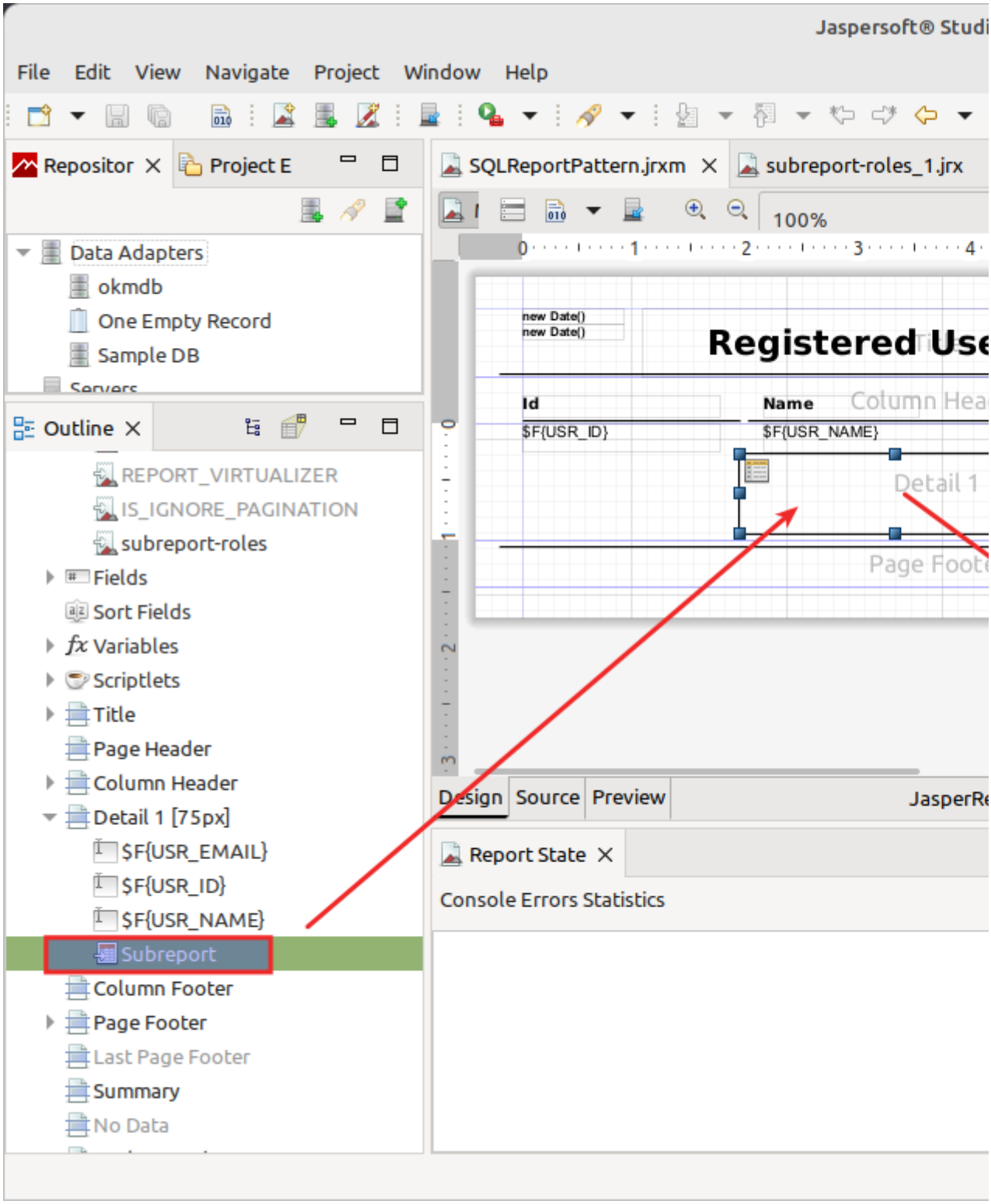
Screenshots

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

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

Studio but will likely not work in OpenKM (we do not recommend using paths).

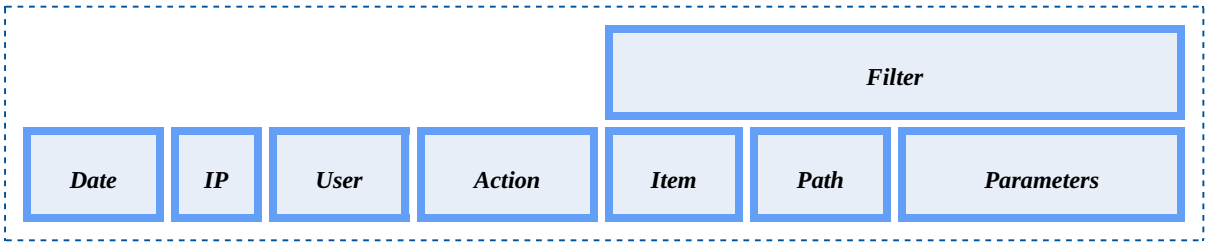
During development, you will likely want to test the report from the Jasper Studio application before deploying it to production. For this reason, we recommend setting the report path directly (file system path) during this phase of development, and once completed, modify it to use dynamic access via 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. The audit trail is primarily 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 logged in.*
LOGOUT	*User logged out.*
SESSION_EXPIRATION	*User session expiration.*
GRANT_USER	*Adds a user grant on a node.*
REVOKE_USER	*Removes a user grant on a node.*

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

GRANT_ROLE	*Adds a role grant on a node.*
REVOKE_ROLE	*Removes a 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 editing.*
CHECKIN_DOCUMENT	*Updates the document.*
CHECKOUT_DOCUMENT	*Edits the document.*
CREATE_DOCUMENT	*Creates a new document.*
DELETE_DOCUMENT	*Deletes a document.*
GET_CHILDREN_DOCUMENTS	*Retrieves a list of documents 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 permanently 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	*Sets the node class.*
UNLOCK_DOCUMENT	*Unlocks a document.*
DOWNLOAD_DOCUMENT	*Downloads a document.*
DOWNLOAD_PREVIEW	*Previews a document.*
SET_DOCUMENT_ANNOTATIONS	*Sets annotations on a document.*
GET_DOCUMENT_ANNOTATIONS	*Gets annotations of a document.*
SET_DOCUMENT_DISPOSITION_STEP	*Sets 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 list of folders in a node.* *List of folders in a folder or record.*
GET_FOLDER_CONTENT_INFO	*Counts nodes in a folder (documents, folders, records, and mails contained within).*
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 permanently removed from the application.*
RENAME_FOLDER	*Renames a folder.*
SET_FOLDER_STYLE	*Sets folder style.* *Default folder icons can be changed.*
EXPORT_FOLDER_AS_JAR	*Folder contents are 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 permanently removed from the application.*
RENAME_MAIL	*Renames a mail.*
MOVE_MAIL	*Moves a mail.*
COPY_MAIL	*Copies a mail.*
GET_CHILDREN_MAILS	*Retrieves a list of mails in a node.* *List of mails in a folder or record.*
DOWNLOAD_MAIL	*Downloads a mail.*
SEND_MAIL_LINK	*Sends a mail with a node link.*
SEND_MAIL_ATTACHMENT	*Sends a mail with document attachments.*

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

CREATE_MAIL_ATTACHMENT	*Creates a mail with document attachments.*
DELETE_MAIL_ATTACHMENT	*Deletes a mail with document attachments.*
GET_MAIL_ATTACHMENTS	*Retrieves a mail with attachments.*
SEND_MAIL_LINK	*Sends a mail link.*
DOWNLOAD_MAIL	*Downloads a mail.*
SET_MAIL_NODE_CLASS	*Sets the node class.*
SET_MAIL_DISPOSITION_STEP	*Sets 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 permanently removed from the 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 a list of records in a node.* *List of records in a folder or record.*
SET_RECORD_TITLE	*Sets the title.*
SET_RECORD_NODE_CLASS	*Sets the record node class.*
ADD_RECORD_NODE_CLASS_CHILDREN	*Adds node class children.*
SET_RECORD_DISPOSITION_STEP	*Sets the disposition step.*

Property group actions

Field / Property	*Description*
ADD_PROPERTY_GROUP	*When adding metadata to a node (document, folder, mail, or record).*
REMOVE_PROPERTY_GROUP	*When removing metadata from a node (document, folder, mail, or record).*
SET_PROPERTY_GROUP_PROPERTIES	*When updating metadata for 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 the properties of a single metadata group of a node (document, folder, mail, or record).*

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

Node actions

Field / Property	Description
SUBSCRIBE_NODE	When a user subscribes to a node (document, folder, or record).
UNSUBSCRIBE_NODE	When a user unsubscribes from a node (document, folder, or record).
NODES_TO_ZIP	Exports nodes to a ZIP file.
RESTORE_NODE_VERSION	Restores a version of a node (document, folder, or record).

Repository actions

Field / Property	Description
PURGE_TRASH	Trash is purged. All related nodes are permanently removed from the application.

Admin actions



These actions are only performed by users with Administration grants.

Field / Property	Description
ADMIN_ACTIVITY_LOG	The administrator uses the 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.

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

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 parameter.*
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.*

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

ADMIN_CRONTAB_EXECUTE	*Executes a crontab task.*
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 a database update 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.*

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

ADMIN_LANGUAGE_DELETE	*Deletes a language.*
ADMIN_LANGUAGE_EDIT	*Edits a language.*
ADMIN_LANGUAGE_IMPORT	*Imports a language.*
ADMIN_LOGCAT_DOWNLOAD	*Downloads log files.*
ADMIN_LOGCAT_LIST	*The administrator uses the logcat tool.*
ADMIN_LOGCAT_PURGE	*Deletes or purges log files.*
ADMIN_LOGCAT_VIEW	*The administrator uses the 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.*

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

ADMIN_MAIL_FILTER_RULE_DELETE	*Deletes a mail account rule.*
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 a message.*
ADMIN_MIME_TYPE_CREATE	*Creates a new MIME type.*
ADMIN_MIME_TYPE_DELETE	*Deletes a MIME type.*
ADMIN_MIME_TYPE_EDIT	*Edits a 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.*

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

ADMIN_OCR_TEMPLATE_UPDATE	*Updates an existing OCR template.*
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 contents.*
ADMIN_REPOSITORY_REMOVE_CURRENT	*Deletes all folder content.*
ADMIN_REPOSITORY_REMOVE_MIXIN	*Removes a mixin from a repository.*
ADMIN_REPOSITORY_SAVE	*Saves repository configurations or changes.*
ADMIN_REPOSITORY_SEARCH	*Performs 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.*

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

ADMIN_ROLE_DELETE	*Deletes a role.*
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.*

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

ADMIN_STAMP_TEXT_DELETE	*Deletes a stamp text.*
ADMIN_STAMP_TEXT_EDIT	*Edits a stamp text.*
ADMIN_TASK_INSTANCE_ADD_COMMENT	*Adds a 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 the actor on a workflow task instance.*
ADMIN_TASK_INSTANCE_START	*Workflow task instance started.*
ADMIN_TASK_INSTANCE_SUSPEND	*Workflow task instance suspended.*
ADMIN_TASK_INSTANCE_VARIABLE_ADD	*Adds a variable on a workflow task instance.*
ADMIN_TASK_INSTANCE_VARIABLE_DELETE	*Deletes a 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.*

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

ADMIN_TOKEN_SIGNAL	*Workflow task signal sent from the administration view.*
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 automation.

Misc actions

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

Field / Property	*Description*
MISC_OPENKM_START	*Starts the application.*
MISC_OPENKM_STOP	*Stops the application.*
MISC_STATUS	*Application check.* *Several internal checks to ensure the 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.

Preliminary concepts.

Each rule is composed of:

Field	Description
Order	*Indicates the order of the evaluation.* *Rules with lower order.*
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:



	Pre indicates rule execution before an internal event is executed. *Post indicates rule execution after the internal event is executed.* *For example, a rule based on the "Document creation" event.* • *On the "pre" stage, the rule is triggered when the document has not yet been created.* • *On the "post" stage, the rule is triggered after the document has been created.*
Evaluation	*Whether all automation validations should evaluate to true or just one of them.* *For example, if you have two validations like "NameMatches" and "PathContains", if the evaluation is set to "A conditions must be met. If it is set to "OR", only one needs to be true.*
Exclusive	*When enabled, indicates no other rules should be executed.*
Active	*Enables or disables the rule.*
Validators	*List of validators. Each validator is a condition that must be complied with.* *Validations are also ordered. It's a good practice to place first the validations that are more likely t* *When empty, always comply.*
Actions	*List of actions. An action is something that will be done to the node that caused the execution of the rule.* *Node types:* • *Document* • *Folder* • *Mail* • *Record* *An example of an action can be renaming a document immediately after creation.*

Understanding Validators and Actions

The application provides a complete list of available validators and actions in the Plugins view. The application's plugin

architecture helps administrators add new features to the application.

By default, the application comes with built-in plugins:

- *Built-in Automation Validator plugins.*
- *Built-in Automation Action plugins.*

You can also 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 definitions.*
>
> *Creating a rule has two steps. First define the general fields, then the action and validation definitions.*

- *In the top right, click on*  ***Create new rule***.
- ***Fill in the fields***.
- *Click on the* ***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 the* ***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***.

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

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, so the validation result is **valid**. The value of the parameter folder name is **test**. Node with path **/okm:root/folder1** triggered the event. The folder **/okm:root/folder1/test** doesn't exist, so the validation result 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 types are:* • *PNG* • *GIF* • *JPEG* • *BMP* • *TIFF* • *PDF*
NameContains	*Post*	*Name*	*Valid when the node name contains the name value.* *The value of the parameter name is **test**.* *Node with path **/okm:root/folder/sometest** triggered the event.* *Folder **sometest** contains the value **test**, so the validation result is **valid**.* *The value of the parameter name is **test**.* *Node with path **/okm:root/folder/somefolder** triggered the event.*

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

			*Folder **somefolder** does not contain the value **test**, so the validation result is **not valid**.*
PathContains	*Post*	*Path*	*Valid when the node path starts with the specified path.* *The value of the parameter path is **/okm:root/folder1**.* *Node with path **/okm:root/folder1/test/temporary** triggered th event.* ***/okm:root/folder1/test/temporary** starts with path **/okm:root/folder1**, so the validation result 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**, so the validation result 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 th event.* ***/okm:root/folder1/test/temporary** contains a folder named **folder1**, so the validation result 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**, so the validation result 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 contain a folder named ***folder1***, so the validation result 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 the event.* *The parent path* ***/okm:root/folder1*** *contains a folder named* ***folder1***, *so the validation result 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***, *so the validation result 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 contain a folder named* ***folder1***, *so the validation result 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 contain a folder named* ***folder1***, *so the validation result is* ***not valid***.
isUser	*Pre, Post*	*User id*	*Validates whether the authenticated user matches the user provided as a parameter.*

WasCreatedFromTemplate	*Pre,* *Post*	*Node uuid*	*Checks if the node was created from a template.*

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 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.
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	*Adds 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*		*If a document with the same name exists in the whole repository, an exception is raised.*
CompactDocumentHistory	*Post*		*Compacts the document history, removing all older versions.*
ConvertToPDF	*Post*		*Converts a document to PDF.*
ConvertToPDFA	*Post*		*Converts 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 file plan to automatically create the destination path of a node.*
DelCadReferences	*Pre*		*Deletes the OpenKM document references of an AutoC document.*
DeleteNode	*Post*		*Deletes a node that is moved into the user's trash node*
DocumentCreationNameToLowerCase	*Pre*		*Forces the name and the extension of the document to lowercase.*
DownloadAsDocumentVersion	*Post*		*When downloading a document, the document is renai to include the version number at the end.*
DownloadControl	*Post*		*Registers the action that triggered a download in the activity log.* *Available download causes:* • *Folder or document download as a ZIP.* • *Document download from the user interface.* • *Document download from the user interface t preview.* • *Document download from the user interface t print.* • *Folder exported as JAR.* • *Email download from the user interface.*
ExecuteScripting	*Pre* *Post*	*Code*	*Executes a script.*

ForceGrantActualUser	*Post*	*Permission*	*Grants permission to the user who caused the event or node that triggered the event.* *The super user makes the security change*
ForceRevokeActualUser	*Post*	*Permission*	*Revokes permission to the user who caused the event o the 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 th event.*
GrantUser	*Post*	*User name* *Permission*	*Grants a user permission to the node that triggered th event.*
MoveToAZFolder	*Post*	*Folder destination* *Folder depth;*	*Creates a folder hierarchy based on the name of the n and depth, and moves the node into it.* *Document named "Testing.pdf"* *Folder destination = /okm:root/move/into* *With depth = 3* *The document will be moved to /okm:root/move/into/T/E/S/Testing.pdf*
MoveToFolder	*Post*	*Folder destination*	*Moves the node to a new location.*
MoveToYearMonthDayFolder	*Post*	*Folder*	*Moves the node to a new location with a year/month/d*

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

		destination	*subfolder structure.*  *Document named "Testing.pdf"* *Folder destination = /okm:root/move/into* *Current date = 2020-05-08* *The document will be moved to /okm:root/move/into/2020/05/08/Testing.p*
NewCadReferences	*Post*		*Creates the OpenKM document references of an Auto document.*
OCRDataCapture	*Post*		*Captures data from the document.*
OMRDataCapture	*Post*		*Captures data from the document.*
PurgeNode	*Post*		*Purges a node. The node is removed from the reposito*
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*		*Revokes all users granted access to the node that triggered the event.*
RemoveCategory	*Post*	*Category*	*Removes the category from the node that triggered the event.*
RemoveKeyword	*Post*	*Keyword name*	*Removes 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 access to the node that trigg the event.*
Send Mail	*Post*	*Role names* *User names* *Message*	*Sends mail to the selected users and roles.* *The message format is the same as used in notificatior messages. For more information, take a look at Notification configuration parameters.*
SetAutoStartToWizard	*Pre, Post*		*Sets the autostart for 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 of the document node that triggered event.* *The language value is based on ISO 639-* *For example, for Spanish the value is "es" for English "en", etc.*
SetRelationsFromXRef	*Post*		*Sets the relations from xref.*
SetSummarization	*Post*		*Sets the document summarization.* *The KEA Summarization service must be enabled.*
SetTextExtracted	*Post*	*String*	*Replaces the original extracted text content of the document node that triggered the event.*
SetTextExtractor	*Pre*	*ClassName*	*Replaces the default text extractor.*
StampDocument	*Post*	*Stamp ID* *Stamp type*	*Stamps 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*		*Marks a document as signed by VidSigner.*  *A VidSigner account is required.*
VidSignerDocumentToSign	*Post*		*Sends a document to be signed by VidSigner.*  *A VidSigner account is required.*

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 stamps:

- *Based on images.*
- *Based on text.*
- *Based on barcodes.*

- ***To manage text stamps** click on the top menu **Stamp button** and then click on the submenu **Text Stamp**.*
- ***To manage image stamps** click on the top menu **Stamp button** and then click on the submenu **Image Stamp**.*
- ***To manage barcode stamps** click on the top menu **Stamp button** and then click on the submenu **Barcode Stamp**.*

Stamps are shown in a table structure.

Id	Name	Description	Text	Script	Color	Users	Active

Create new stamp based on text

When using a stamp based on text, the text is converted to an image and then stamped into a document.

Field descriptions:

Field	Description
Name	*Stamp name.*

Description	*Stamp description.*
Text	*The text to be shown on the stamp.*

Script	*When set, the result of the script will be the stamped text.*
	
	List of variables that can be used directly:
	
	Variable / ***Description***
	document — *Document object of the document node.*
	For more information see the 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 font file must be copied into the $TOMCAT_HOME/lib folder.
Layer	*Position of the stamped image.* *Available values:* • *Under content.* • *Over content.*
Opacity	*The text opacity. The range of allowed values is from 0 to 1.* *An **Opacity** equal to 0 means the text or image is transparent and you won't see it. Set it to 0.5 to see semi-transparent text or to 1 to be opaque.* *Also try changing the layer to stamp over or below the default text layer.*
Size	*Text size.*
Align	*Text alignment.* *Available values:* • *Left.* • *Center.* • *Right.*
Rotation	*Text rotation. The range of allowed values is 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 the Expr. X and Expr. Y input fields, you can enter more than a simple number. Currently the following macros are defined:

- *PAGE_WIDTH*
- *PAGE_HEIGHT*
- *PAGE_CENTER*
- *PAGE_MIDDLE*

So to center a stamp on the page you can use:

Expr. X	*PAGE_CENTER*
Expr. Y	*PAGE_MIDDLE*

- *Click on*  ***Create new Stamp icon***.
- ***Fill in the form***.
- *Click on the* ***Create button***.

Create new stamp based on image

Field descriptions:

Field	*Description*
Name	*Stamp name.*
Description	*Stamp description.*
Image	*The image to be added into the document.*
Script	*When set, the result of the script will be the stamped text.* *List of variables that can be used directly:*

Variable	*Description*

	document *Document object of the document node.*
	For more information see the 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	*Position of the stamped image.* *Available values:* • *Under content.* • *Over content.*
Opacity	*The image opacity. The range of allowed values is 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 users.*
Users	*List of users who can use the stamp.*

In the Expr. X and Expr. Y input fields, you can enter 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 on 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 in the form***.
- *Click on the* ***Create button***.

Edit stamp

- *Click on*  ***Edit icon***.
- ***Modify the form***.
- *Click on the* ***Edit button***.

Delete stamp

- *Click on*  ***Delete icon***.
- *Click on the* ***Delete button***.

Test stamp

- *Click on*  *the icon.*



Stamps that use scripting with the document variable cannot be tested from the administration interface.

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

AI Prompt

The AI prompt feature helps you create your own AI prompts.

This feature is available starting from version 8.1.16.

This feature remains hidden unless an AI model has been activated beforehand. Read about it at AI configuration parameters.

The functionality must be enabled in the user's Profiles for it to be available.

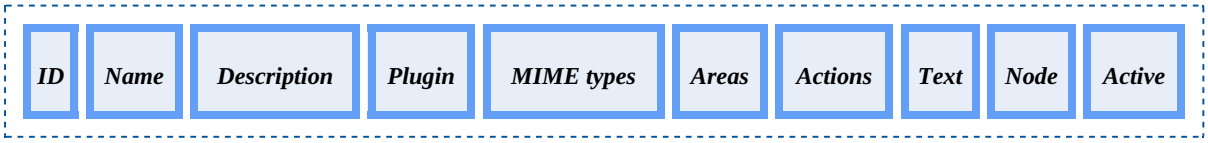
You can set two types of AI prompts:

- *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 prompts are shown in a table structure.



Create AI prompt

Field descriptions:

Field	Description
Name	*AI prompt name.*
Description	*AI prompt description.*
Prompt	*The AI prompt to be executed on 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.*
Areas	*Indicate in which section of the user interface the AI prompt will be displayed.* *Areas available:* • *Anywhere* • *Email composition*
Actions	*Actions are additional operations that can be performed with the result of executing an AI prompt.* *Actions available:* • *Add a note.* • *Set a description.* *Actions will only be available if the AI prompt has been executed on a node.*

- *Click on*  ***Create new AI prompt icon***.
- ***Fill in the form***.
- *Click the* ***Create button***.

Edit AI prompt

- *Click on*  ***Edit icon***.
- ***Modify the form***.
- *Click the* ***Edit button***.

Delete AI prompt

- *Click on*  ***Delete icon***.
- *Click the* ***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 the crontab view, there is a set of jobs that are executed periodically at fixed times or intervals.

Crontab jobs typically automate application maintenance or administration.

Crontab is shown in a table.



Crontab parameters

Field	Description	Required
Name	*Job name.*	*true*
Expression	*Sets when the job is executed.* *It takes into account 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, the task is delayed until the backup process is completed.*	*true*
Class name	*More information at Creating your own Crontab plugin.*	*true*
Mail	*When present, an email is sent as part of the task once the execution has been completed. You can use several email addresses separated by a comma (",").*	*false*
Last begin	*Shows the last time a process started.*	-
Last end	*Shows the last time a process was completed.*	-
Active	*Enables or disables execution of the process.*	*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) matches 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”€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”€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”€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”€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 during nine-to-five weekdays.*

"0 0 0 25 12 ?" = every Christmas Day at midnight

Create a new crontab job

A crontab job is based on a plugin architecture.

More information at Creating your own Crontab plugin.

Edit a crontab job

- *Click on the*  ***Edit crontab job icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Execute a crontab job

- *Click on the*  ***Execute crontab job icon***.

Delete a crontab job

A crontab job is based on a plugin architecture deployed in $TOMCAT_HOME/plugins. You 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, see Statistics.
Check disk space	0 0 0 * * *	Daily execution at 00:00	Sends an email alert to the administrator when disk space is low.
Check Repository Integrity	0 0 0 * * SAT	Executes each Saturday at 00:00	Checks repository integrity and generates a report.
Lucene Delayed Indexer	0 */5 * * * *	Executes every 5 minutes.	Indexes 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 * * * *	Executes every 5 minutes.	Process the mail extractor queue.
Mail Extractor Issues	0 0 0 * * *	Daily execution at 00:00	Processes mail extractor issues.
Metadata Refresh	0 0 0 * * *	Daily execution at 00:00	Refreshes metadata.
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 * * * ****	*Executes every 5 minutes.*	*Execute the pending task.* *For more information, see Statistics.*
Purge Activity Log	*0 0 0 * * ****	*Daily execution at 00:00*	*Deletes the activity log.*
Purge Dashboard Activity	*0 0 0 * * ****	*Daily execution at 00:00*	*Cleans dashboard activity logs older than one month.*
Purge Download Preview Tokens	*0 */5 * * * ****	*Execution every 5 minutes.*	*Deletes expired preview tokens that have not been consumed.*
Purge Empty Repository Directories	*0 0 0 * * ****	*Daily execution at 00:00*	*Purge empty folders in the repository*
Purge Node Versions	*0 0 0 * * ****	*Daily execution at 00:00*	*Purges all document history and keeps only the current version in the application.*
Purge Old Trash Nodes	*0 0 0 * * ****	*Daily execution at 00:00*	*Purges old nodes in the trash.*
Purge Orphan Repository Files	*0 0 0 * * SUN*	*Executes at 00:00 every Sunday*	*Purges the files in the datastore that are not in the application database.*
Purge Pending Downloads	*0 0 0 * * ****	*Daily execution at 00:00*	*Deletes 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*	*Deletes temporary printing files.*
Purge Tomcat Logs	*0 0 0 * * ***	*Daily execution at 00:00*	*Deletes 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*	*Updates repository statistics.*
Send Grouped Notifications	*0 0 0 * * ***	*Daily execution at 00:00*	*Sends daily email notifications grouped into a single message.*  *For more information, see the Profile Notification tab.*
Sync LDAP users	*0 0 0 1 1 **	*Executes at 00:00 on January 1st of every year.*	*Synchronizes AD user data to the OpenKM database.*
Text Extractor Issues	*0 0 0 * * ***	*Daily execution at 00:00*	*Collects text extraction issues and sends an email warning to the administrator.*
Text Extractor Worker	*0 */5 * * * ***	*Executes every 5 minutes.*	*Processes the text extraction queue.*  *For more information, see Statistics.*
User Mail Importer	*0 */30 * * * **	*Executes every 30 minutes.*	*Imports the emails from registered users' mail accounts.*
SyncLdapUsers	*0 0 0 1 1 **	*Executes once a week*	*Synchronizes users and roles from AD to the OpenKM database.*

OCR templates

OCR Templates allow creating zonal OCR templates which allow recognising and extracting structured 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 the stages of OCR template execution:



On recognition, the application based on the OCR template's definition tries to identify the document. A document will be considered recognised only if an OCR template matches 100%.

During data extraction, the values are extracted from the document.

When saving data, the dynamic values obtained in the data extraction stage are saved into metadata groups. Optionally, they are saved as static values defined on the OCR template (normally used to identify the document type).

There are two field types in OCR templates:

- *Validation Control fields.*
- *Data Capture fields.*

***Validation Controls fields** are used only in the **recognition** stage.*

***Data Capture fields** are used in the **data extraction** stage and optionally can also be used in the **recognition** stage.*

Allowed MIME types for the recognition stage are:

- *PNG*
- *JPG*
- *GIF*
- *BMP*
- *PDF*

Validation Control field description

Validation Control fields are used only in the recognise stage to help identify the document by checking a document section's

pattern.

Each Validation Control field is composed of:

Field	*Description*	*Required*
Name	*Name of the field.*	*true.*
Order	*Order of evaluation*	*true.*
Type	*List of **OCR Validation Controls**. Each validator is a condition that must be met.*	*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 command line. *When present it will be executed in place of the system.ocr configuration*	

OCR	*parameter.* *More information Configuration parameters.* *For example to force digits capture.* ***/usr/bin/tesseract ${fileIn} ${fileOut} digits***	*false.*
Empty value allowed	*When checked, indicates an OCR empty text value is also valid in the document identification process.*	*false*
***** These parameters are used in the Validation Control. They 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 data extraction stage but can also be used in the recognise stage to help identify a document by checking a document section's 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 **OCR Data Capture**.*	*true.*
Property	*Metadata field where the captured data will be stored.*	*true.*
Pattern	*A Java regex pattern.*	*
Rotation	*Indicates the image rotation that must be applied.* *Available image rotations:* • *-270* • *-180*	*true.*

	• *-90* • *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 recognise	*When it is set to true, it indicates that the field will also be used in 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 the Plugins view. The application's plugin architecture helps administrators add new features to the existing application.

By default, the application comes with built-in plugins:

- *Built in OCR Validation Control plugins.*
- *Built in OCR Data Capture plugins.*

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

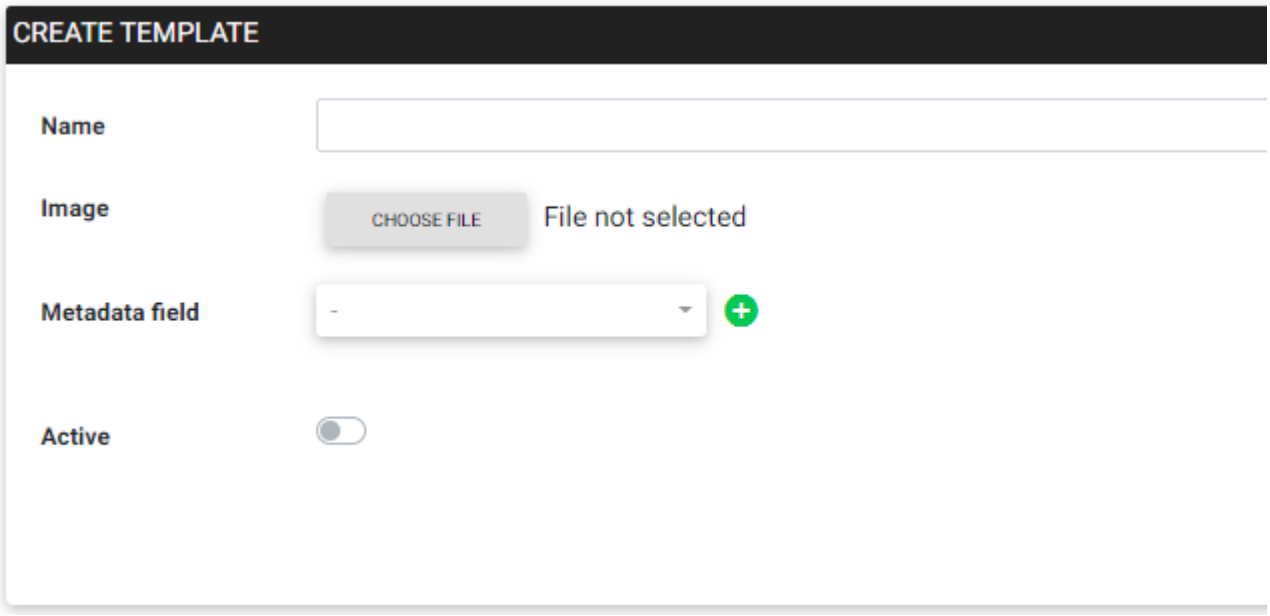
You can also create your own plugins:

- *Creating your own OCR Validation Control plugin*.
- *Creating your own OCR Data Capture plugin*.

> *For more information about how to Register a new plugin.*

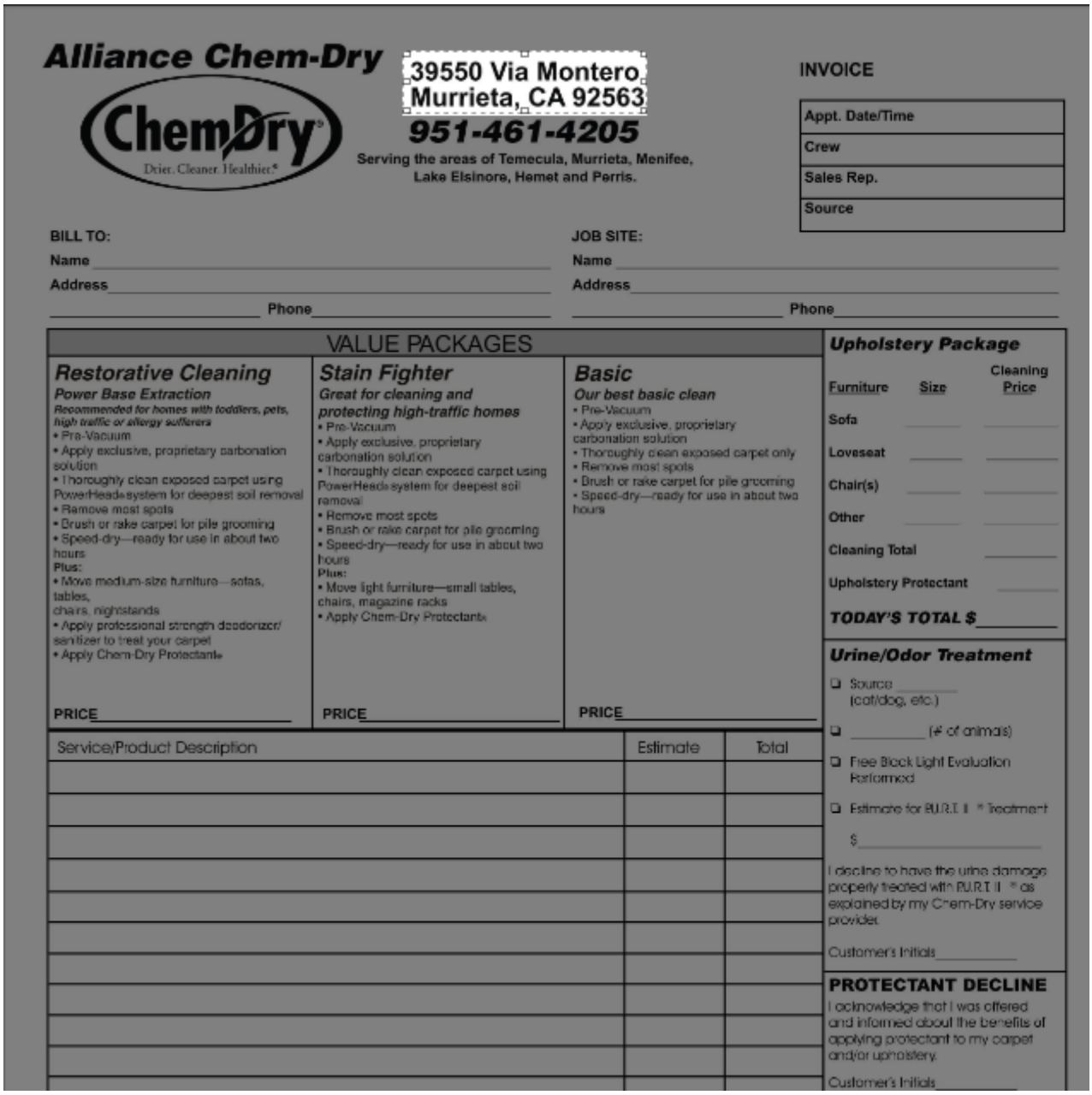
Create a new OCR template

- *On the top right click* ***New OCR template icon***.
- ***Fill in the fields***.
  - *Metadata field (static metadata values, normally used to identify document type).*
- *Choose the file (allowed MIME type).*
- *Click on the* ***Create button***.



- *Add Validation Control fields as needed.*
  - *Validation Controls*
    - *Click on* ***Fields***.
      - *To create a new Validation Control:*
        - *Click on* ***New Validation Control Field icon***.
        - ***Fill in the fields***.
        - ***Choose the area with the mouse on the left image*** .
        - *Click on the* ***Create button***.

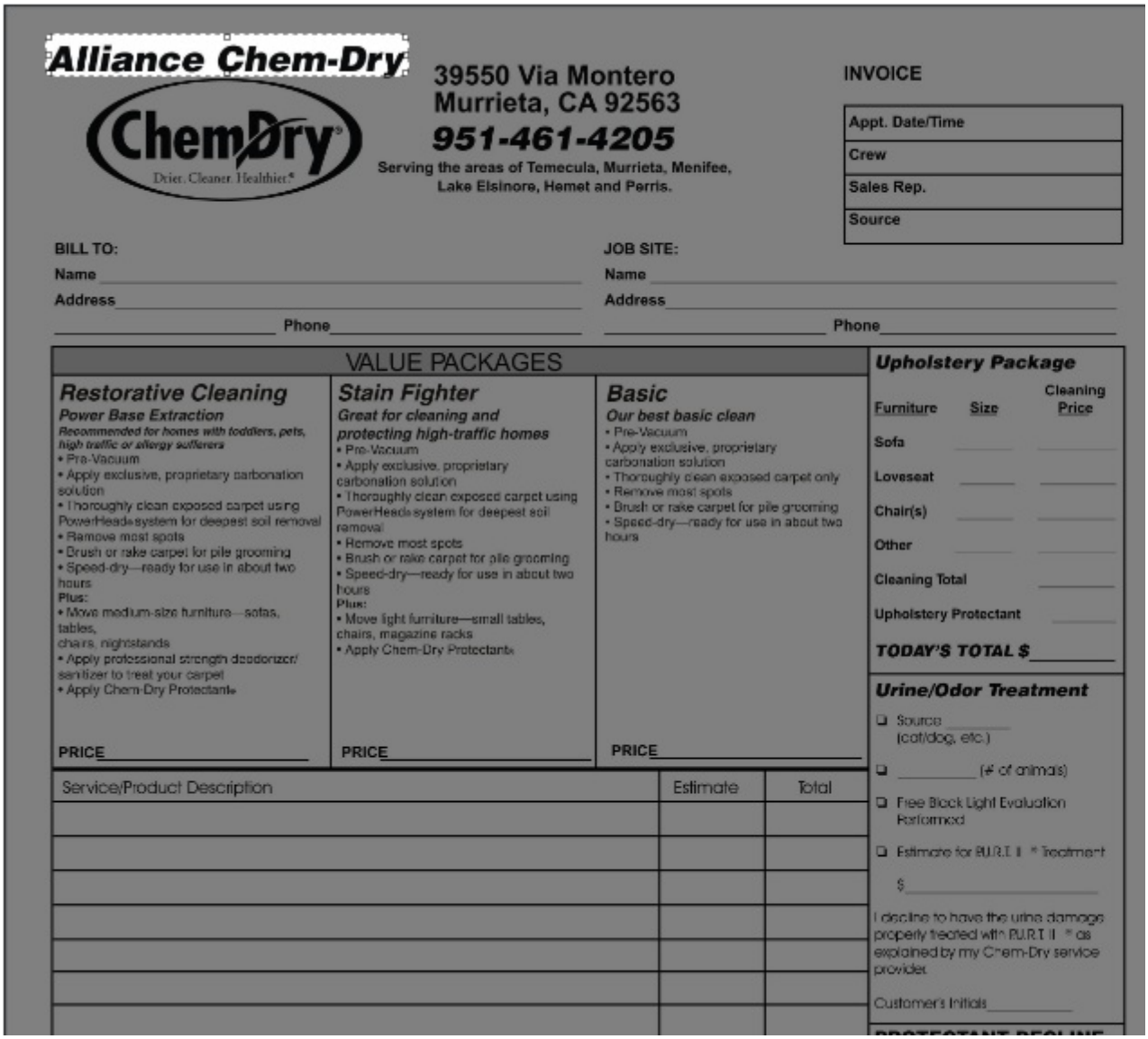
- *To edit a Validation Control:*
  - *Click on* ***Edit a Validation Control icon***.
  - ***Modify the fields***.
  - *Click on the* ***Edit button***.
- *To delete a Validation Control:*
  - *Click on* ***Delete a Validation Control icon***.
  - *Click on the* ***Delete button***.



- *Add Data Capture fields as needed.*

- *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 the mouse on the left image*** .
    - *Click on the* ***Create button***.
  - *To edit a Data Capture:*
    - *Click on*  ***Edit a Data Capture icon***.
    - ***Modify the fields***.
    - *Click on the* ***Edit button***.
  - *To delete a Data Capture:*
    - *Click on*  ***Delete a Data Capture icon***.
    - *Click on the* ***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 an OCR template

- *Click on* ***Edit OCR template icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Edit OCR template fields

Edit an 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 the* ***Edit button***.

Edit an OCR Data Capture

Click on ***Fields***.

- *Click on* ***Edit a Data Capture icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

Delete OCR template fields

Delete an OCR Validation Control

Click on ***Fields***.

- *Click on* ***Delete a Validation Control icon***.
- *Click on the* ***Delete button***.

Delete an OCR Data Capture

Click on ***Fields***.

- *Click on* ***Delete a Data Capture icon***.
- *Click on the* ***Delete button***.

Check OCR template

- *Click on the* *Check icon and the OCR template will be checked.*

The Capture Data field shows the result of the text extraction by the OCR.

The Validation Control field shows the result of the compliance validation (true or false).

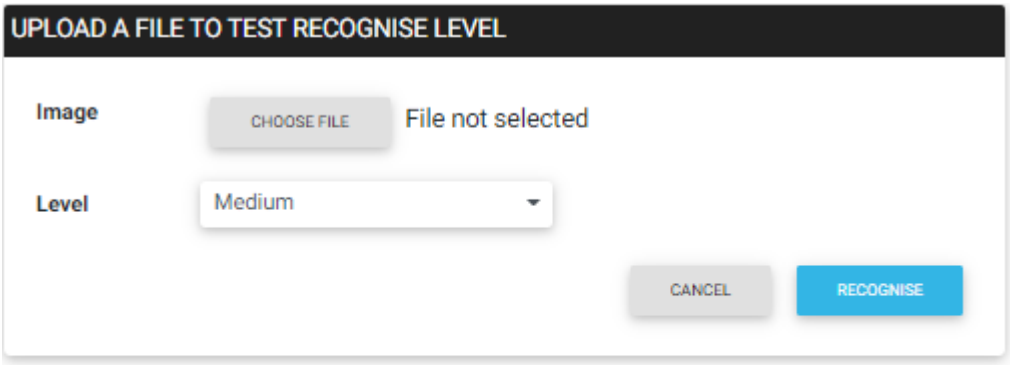
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

OCR TEMPLATE DEFINITION CHECK: Test

Field	Class	Type	Zone
Title	String	Data Capture field	
Street name	String contains	Validation Control field	

Check document recognition

- *At the top right, click on* ***Recognise icon***.
- ***Choose the file to be recognised*** *(allowed MIME type).*
- ***Choose the recognition level***.
- *Click on the* ***Recognise button***.



Name	Confidence level	Control	Recognised
Test	50 %	(1 of 2)	✕

The recognise level determines the speed used to identify a document. By default, the application uses the slow level.

- *In the fast level, it considers only the Validation Control fields.*
- *In the medium level, it also requires that the Data Capture fields are not empty.*
- *In the slow level, it also verifies that the Data Capture fields are correct (comply with patterns, etc.).*

Delete an OCR template

- *Click on*  ***Delete OCR template icon***.
- *Click on the* ***Delete button***.

Enable OCR Data Capture

Feature	*Description*
Profile General tab	*When OCR data capture is enabled from the Profile General tab, the OCR Data Capture process will be executed automatically for each newly uploaded document.* *If an automation definition is present, 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.*

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

Built-in OCR Validation Control plugins

Name	*Value*	*Pattern*	*Description*
String contains	*Optional*	*Optional.*	*Returns the evaluation of a "contains" operation for 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 once.* *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 once and the matched pattern contains the value of the value field.* 🐞 *In case the pattern and value fields are empty, it returns false.*
String equals	*Optional.*	*Optional.*	*It returns the evaluation of equality operations between a value or a pattern and 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 once.* *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 once 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 similarity operations between some values or a pattern*

String			*and the text extracted by OCR.* *The "similar operation" counts the number of changes that must be done to transform the expected value into the text extracted by OCR. This is the **distance between the two 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 value field is mandatory and cannot 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 value set in the 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 once and the distance between the matched text and the value of the value field is less than the value set in the ocr.control.similar.distance parameter.* *The value field is mandatory and the pattern field 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.*	*Returns a text value.* *In case the pattern is empty, it returns the text extracted by OCR.* *In case the pattern is not empty, it returns the text extracted by OCR that matches the pattern only once.* *In case the text extracted by OCR is empty, it returns an exception.* *In case the pattern is not empty and the text extracted by OCR doesn't match the pattern only once, it returns an exception.*
Number	*Optional.*	*Returns 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 cannot 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, it returns an exception.* *In case the text extracted by OCR is empty, it returns an exception.* *In case the text extracted cannot be converted to a Calendar, it returns an exception.*

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

Barcode	*Optional.*	*Returns barcode text.* *In case the pattern is empty, it returns the 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 the text extracted by a Barcode engine doesn't match the pattern, it returns an exception.*

Languages

The application can work with several languages. Here are the language term definitions.

Languages are displayed in a table.

ID	Flag	Name	Translation status

The "en-GB" translation is used by the application as the default translation.

Sometimes translations are not fully complete and some terms are missing. The "en-GB" translation is used to identify which terms are missing and how many.

When a term is missing in another language, the "en-GB" term is used in its place.

The en-GB translation cannot be removed.

Create a new language

- *Click on*  ***the new language icon***.
- *Fill in the fields*
  - *The ID must comply with ISO 639 (2 levels).*
  - *Language name*
  - *The icon should be 16x11 px.*
- *Click on* ***the Create button***.

More information about ISO 639.

Edit a language

- *Click on*  ***the Edit language icon***.
- ***Change the fields***.
- *Click on* ***the Edit button***.

Delete a language

- *Click on*  ***the Delete language icon***.
- ***Change the fields***.

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

- *Click on* ***the Delete button***.

Edit language translations

On the top right, the translations can be filtered by key.

- *Click on*  ***the Edit language translation icon***.

Language translations are shown in a table.

Module	Key property	English translation	Translation

By default there are ***three modules*** *that indicate where translations are used:*

- *frontend (used on the frontend)*
- *mobile (used in the mobile view)*
- *extension (used on the frontend by extension modules)*

Two key properties cannot be equal.

- ***Change the fields***.
- *Click on* ***the Edit button***.

Export language translations

- *Click on*  ***the Export language translations icon***.
- *The application* ***retrieves a SQL file*** *with the language translations.*

Import language translations

Download your language package from Language translations.

- *At the bottom right* ***choose a language package file***.
- *Click on* ***the 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 (Simplified)	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	742
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*
Greek	***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***	*725*
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 whether it is available in Language translations section.

A sample of a new language based on Oromo from Ethiopia

*The first step is identifying 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 is to get the country flag; in this case, Ethiopia. Download the flag icons. The flag icon names are based on ISO 369 (one level), which means we are looking for a file named "**et.gif**".*

The third step is to create the new language. Go to Administration > Languages and create a new language with these values:

- *Click on*  ***the new language icon***.
- *Fill in the fields*
  - *The "id" might be "**om-ET**".*
  - *The "name" might be "Oromo".*
  - *The "icon" might be the file named "**et.gif**".*
- *Click on the **Create button**.*

Once the new language has been created, you must set the translations:

- *Click on*  ***the Edit language icon***.
- ***Change the fields***.
- *Click on the **Edit button**.*

Finally, consider sharing your translation with the community.

- *Click on*  ***the Export language translations icon***.
- *The application **retrieves a SQL file** with the language translations.*
- *Contact us via the website contact form, and we will tell you how to share the translation with us.*

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

Utilities

The utilities section in Administration is a collection of useful features designed for use by IT staff.

To take advantage of most of these features, some technical skills are needed.

These features can be categorized as:

Category	*Description*
Security	• *Admin permissions* • *Download tokens* • *Purge data* • *Purge trash* • *Password encryption*
Performance	• *Cache stats* • *Clean conversion cache* • *Hibernate stats* • *Profiling stats*
Low-level access	• *Database query* • *Execute JAR* • *Scripting*
Check specific configurations	• *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*
Other	• *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 administrators set administration access levels for users who have access to the application.

For example, it provides some users with access only to manage users and roles.

On this view, a table displays all users who have administrative access based on their admin role.

More information about the default admin role is available at Security configuration parameters.

Change user-level access

- *Click on*  ***Edit user-level access icon***.
- ***Change access level***.
- *Click on the* ***Edit button***.

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

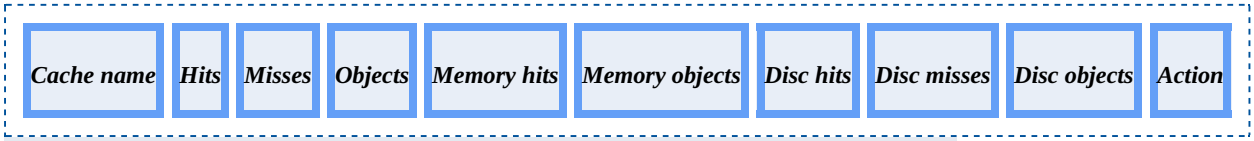
Cache stats

OpenKM takes advantage of cached objects for fast data retrieval.

> *For example, the users and roles lists, when the application is configured to work with LDAP, are cached to reduce traffic between the application and the LDAP server.*
>
> *The associated cached names for LDAP are:*
>
> - *com.openkm.cache.ldapPrincipalAdapter.general*
> - *com.openkm.cache.ldapPrincipalAdapter.name*

The cache can also be used to collect data useful for identifying configuration parameter changes needed for better application performance. 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 misses	Number of failed queries in cache memory.	true
Memory objects	Number of objects in cache memory.	false
Disc hits	Number of successful queries on the cache disk.	true

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

Disc misses	*Number of failed queries on the cache disk.*	*true*
Disc objects	*Number of objects on the cache disk.*	*false*
Action	*Clean cached objects.*	*Not applicable.*

Clear all caches

- *At the top right, click on* ***Reset all button***.

Refresh cache stats

- *At the top right, click on* ***Refresh icon***.

Enable cache stats audit trail

- *At the top right, click on* ***Enable cache audit trail button***.

Enables the field stats:

- *Hits.*
- *Misses.*
- *Memory Hits.*
- *Memory misses.*
- *Memory objects.*
- *Disc hits.*
- *Disc misses.*

Disable cache stats audit trail

- *At the top right, click on* ***Disable cache audit trail button***.

Disable the field stats:

- *Hits.*
- *Misses.*
- *Memory Hits.*
- *Memory misses.*
- *Memory objects.*
- *Disc hits.*

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

- *Disc misses.*

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

Check email

The check email feature helps the administrator check the application's mail configuration.

Check email

- *Fill in the fields*
  - *From*
  - *To*
  - *Subject*
  - *Content*
- *Click the **Check button***

If the application mail configuration is correctly set, it will return an "OK"; otherwise the cause of the error will be shown.

*If the **send.mail.from.user** configuration property is set to **false**, the "From" field will be ignored and the **mail.from** JavaMail configuration property will be used. Note that, depending on the SMTP server configuration, only the address configured in **mail.smtp.user** may be able to send messages. This means 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 the document is selected in the general application user interface).*
  - *Document path.*
  - *Document outside application (choose a document from the file system to upload).*
- *List of available text extractors.*

Check text extraction

- ***Choose a document** to be processed*
  - *OpenKM document:*
    - *Fill in the document UUID field or the document path.*
  - *External document:*
    - *Choose a document from the file system to upload.*
- ***Choose one text extractor** from the list of available text extractors.*
- *Click on the **Check button**.*

After the process is completed, it will show the text extracted by the selected text extractor.

The text extraction process can take a few 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 an integrated file conversion cache.

The file conversion cache is stored in the file system, by default in the $TOMCAT/repository/cache folder.

A typical example of document conversion is generating a thumbnail of each document shown in the default application user interface. For example, to get a thumbnail of a Microsoft Word document, it is necessary to convert the first page into an image. This process consumes some CPU.

To prevent consuming more CPU than necessary, the documents are converted only when necessary â€" the first time they are accessed or after a document is updated. In other cases, the file is retrieved from the cache.

The administrator can take control of these cached files and clean them by type or clean them all.

Cached files:

Type	*Description*
DXF	*Caused by AutoCAD previsualization conversion.*
PDF	*Several causes:* • *Document previsualization.* • *User conversion on demand.*
SWF	*Document previsualization.*
Thumbnail properties	*Document thumbnail in the application desktop view.*
Thumbnail Search	*Document thumbnail in the application search view.*
Thumbnail Lightbox	*Document thumbnail in the lightbox.*

Clean conversion cache

- ***Choose the cached file to be cleaned***
- *Click on the **Clean** button.*

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



The operation can take a few seconds or minutes depending on 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 allows the user to download a zip file with important information about the 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 the application log for the current day. Old logs will not be downloaded.

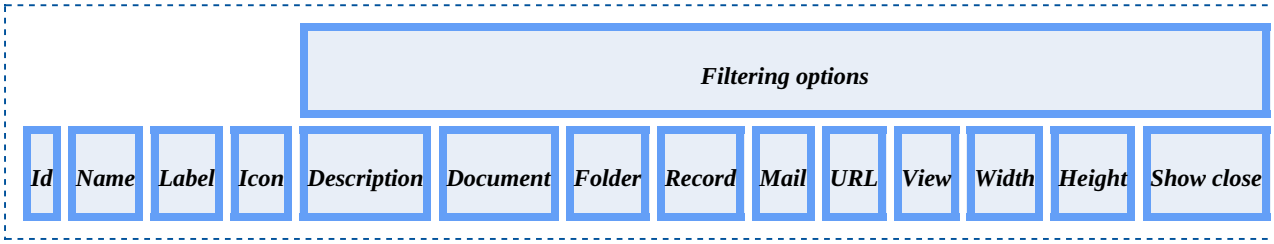
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Custom actions

The Custom actions feature helps add custom action options in the KCenter user interface.

> *Sample: Custom actions in KCenter*

Users are shown in a table.



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, you 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 entire user interface)* • *Run-javascript (the script will be executed in the UI without showing any 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, a close button is shown.*	**
Script	*The JavaScript code that will be executed by the UI.*	****
	Available actions:	

Action	•  *Edit a custom action.* •  *Delete a custom action*	

** Mandatory when Popup-wizard is selected.*

*** Option is shown when Popup-wizard is selected.*

**** Mandatory when Popup-wizard or Full-screen is 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 the administrator import data from a database source to a destination folder within the 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 in the source OpenKM.*
Destination folder	*UUID of the destination folder within the 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 whether you want to import this data.*
Concurrent folders	*Check whether you want to import this data.*
Concurrent records	*Check whether you want to import this data.*
Concurrent mails	*Check whether you want to import this data.*
Nodes	*Check whether you want to import this data.*
Users and roles	*Check whether you want to import this data.*
Node relations	*Check whether you want to import this data.*
Bookmarks	*Check whether 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 **the Import button**.*

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

Database query

This feature helps administrators access database data easily and quickly without a third-party database 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 chosen, a list of available tables is shown.*	*Yes.*
Hibernate	*Neutral database queries based on Hibernate query format.*		*No.*

- *Writing more than one query is allowed.*
- *Each query must be written on a single line; line breaks are not allowed.*
- *Execute will process all written queries in the same order they were written, top to bottom. Queries are executed as FIFO (first in, first out).*
- *If a query fails, it will not stop the execution of the others.*

Execute JDBC query

- *On the bottom right **choose type JDBC**.*
- *Write some valid SQL.*
- *Click the **Execute button**.*
- *Optionally, 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 the bottom right **choose type Hibernate**.*
- *Write some valid HQL.*
- *Optionally you can check the "Show SQL" option.*
- *Click the **Execute button**.*



When the "Show SQL" option is checked, the real SQL executed is shown in the results.

Example:

```
select count(*) from NodeBase;
select count(*) from NodeFolder;
```

Allowed sentences

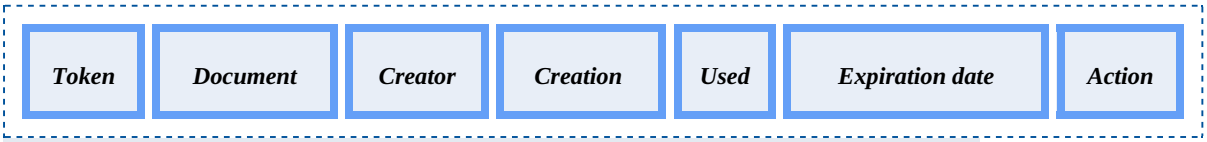
- *On the bottom right **choose a valid SQL file**.*
- *Click the **Import SQL script button**.*

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

Download tokens

The download token feature helps application users to share documents with external users without needing to be logged into the application. With the download tokens log feature, the administrator can take control over the documents that have been shared, the users who shared them, and when they were downloaded.

The download tokens log is shown in a table structure:



Field	Description
Token	*The unique token UUID.*
Document	*The shared document.*
Creator	*ID of the user who shared the document.*
Creation	*Date when the document was shared.*
Used	*Indicates whether the document has been downloaded.*
Expiration time	*Indicates the date after which the document can no longer be downloaded.*
Action	*Deletes a token.*

Delete token

- *Click on*  ***the Delete token button***.

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

Execute JAR

A JAR file is a compressed file that uses 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 written in Java.

This feature helps administrators create their own classes, package them into a JAR file, and execute them from OpenKM.

Examples of usage:

- *When an administrator wants to perform an operation across the entire repository that affects many nodes, based on some logic or specific conditions, they can take advantage of building minimal code with the OpenKM API. Some examples include changing security, removing or moving files, or renaming files based on their metadata values, among others.*
- *Check a crontab task.*

Requirements:

- *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 valuable information about database queries, finding out performance problems and the possible causes.

By default, the Hibernate stats feature is disabled because log analysis consumes hardware resources.

This feature should only be enabled by an administrator in order to investigate possible performance problems with the application.

Each time the application is restarted all the previous analysis is removed. It's important that when the administrator enables this feature, he or she takes into consideration that the application should not be stopped.

Information provided by Hibernate stats

Information provided by Hibernate stats is shown in several tables:

- *General data.*
- *Query statistics.*
- *Entity statistics.*
- *Collection statistics.*
- *2nd level cache statistics.*

Time columns end with the '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 milliseconds.

When the application experiences a performance problem, in most cases it will be caused by a query that needs a long time to complete its execution.

*On Query statistics, all the database queries executed by the application are shown. 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.

- *At the top right click on*  *Enable button.*

Clear Hibernate stats

- *At the top right click on*  *Clear button.*

Disable Hibernate stats

> *This option is only available when Hibernate stats is running.*

- *At the top right click on*  *Disable button.*

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

LDAP Search

The LDAP Search feature helps administrators check 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's distinguished name.*
Password	*The LDAP user's password.*
Search base	*The LDAP search base.*
Search filter	*The filter used in the LDAP search.*
Attribute	*The attribute returned by the 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 take control of how the objects (documents, folders, emails and records) are indexed by the Lucene search engine.

On the main screen, a table shows 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's 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 author.*
created	*Indicates when the object was created.*
name	*Object name.*
userPermission	*User permissions.*
rolePermission	*Role permissions.*

*Depending on the type of object (**_hibernate_class**), there can be more indexed fields.*

Search indexes

- *On the top right, click on* ***Search indexes button***.
- *In the input box, paste an object's UUID (from the desktop view you can get any object's UUID on the Properties tab; select the UUID and press CTRL+C to copy).*
- *Click the* ***Search button***.
- *From the table list, click on* ***List icon***.

*The filtering input box uses the syntax **FIELD : value**. Here are 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 with this search:

```
name:paran*
```

But be careful, because using quotation marks you won't get any results:

```
name:"paran*"
```

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

If you want to locate all PDF documents whose names start with "Paran", the correct query is:

```
name:paran*pdf
```

This query would also work (note the dot before the extension):

```
name:paran*.pdf
```

*But this query will fail 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 look up documents with a given extension, I recommend using the MIME type property this way:

```
mimeType:application/pdf
```

In this query, we are looking for all documents whose names start with "parana" and that have a keyword called "dog":

```
name:parana* AND keyword:song
```

You can learn more about query syntax at Lucene Query Syntax.

Enable show terms

When show terms is enabled, extra fields named terms are shown on document objects with the terms used by the Lucene search engine to index that document.

- *On the top right, click on*  ***Show terms button***.

Disable show terms

- *On the 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 administrators access Tomcat log files directly from the application.

Logs are shown in a table structure:

File	Action

*The main Tomcat log file is named **catalina.log**.*

This file rotates each night, creating a new catalina.log file and saving older ones with the name catalina.log.year-month-day.

Purge all logs

The purge option will clear all Tomcat logs.

- *On the top right, click the* ***Purge button***.

Download file as ZIP

- *Choose a log file.*
- *Click the* ***Download as ZIP icon***.

Browse log file

- *Choose a log file.*
- *Click the* ***Browse icon***.
- *Use filter options.*

Filter options:

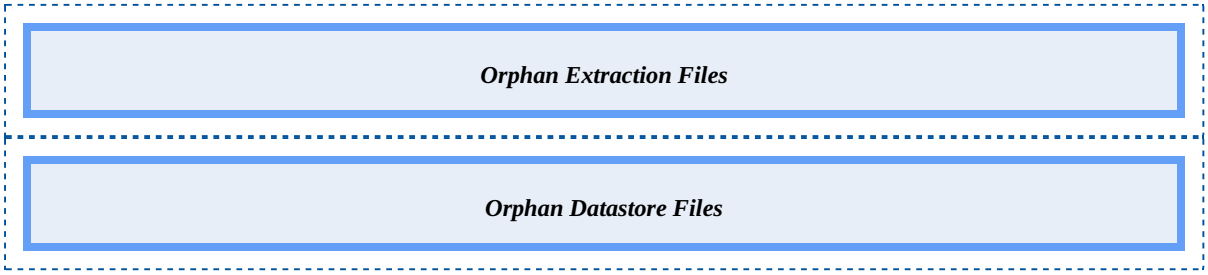
Option	Description
Begin	*Indicates the row position.* *By default, 0 indicates the start of the document.*
End	*Indicates the row position.*

	By default, -1 indicates the end of the document.
String	*Filter by text.*

Orphan Files

The Orphan Files feature displays all orphan extraction files and orphan datastore files.

Both are shown in a table structure:



Filter action

- *To search files, click on*  ***the Filter button***.

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

Password encryption

The password encryption feature helps administrators encrypt a password that will be used by OpenKM, for example for LDAP integration.

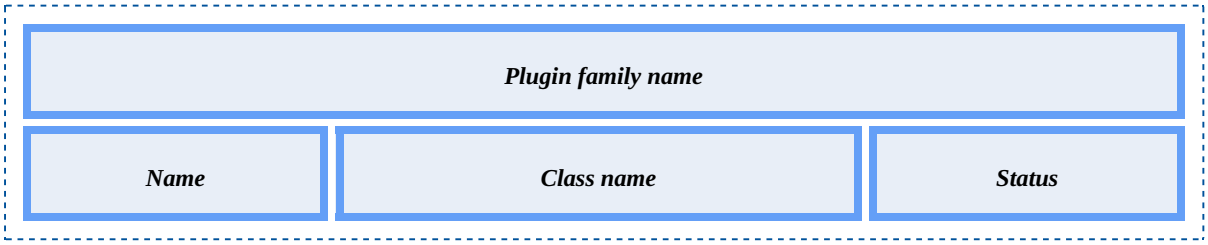
Field	*Description*
Password	*The password to be encrypted.*
Algorithm	*Allowed algorithms:* • *DES* • *AES*
Method	*Allowed methods:* • *Server* • *Cluster*

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

Plugins

The application includes a plugins feature system that offers an easy way to customize and extend OpenKM functionalities. By default, the application includes a general-purpose plugins implementation; these can be disabled, changed, or increased in number depending on the type of plugin. For example, for an antivirus plugin it makes sense to disable it or replace it with another; in the case of a text extractor it makes sense to disable it 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 the automation feature.* *More information at:* • *Automation* • *Creating your own Automation Validation*
Automation actions	*Automation*	*Validations are used in the automation feature.* *More information at:* • *Automation* • *Creating your own Automation Action*
Template parsers	*OCR Zone*	*A data extraction parser used in the 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*	*A document identification checkpoint used in the OCR Zone.* *More information at:* • *OCR templates* • *Creating your own OCR Validation Control plugin*
Option select values	*Metadata*	*Retrieves select values from server-side logic. For example, it helps retrieve data from a 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 implement complex validation rules that are not covered by the 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 user and role information. For example, to connect to an LDAP server or an 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*	*The conversion service is used to convert documents to other formats.* *More information at:* • *Creating your own Document Converter*
Metadata adapters	*UI*	*Visualization logic for the 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 integrate 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 help find application performance problems.

By default, the Profiling stats feature is disabled because log analysis consumes hardware resources.

This feature should only be enabled by the administrator in order to investigate possible performance problems with the application.

Each time the application is restarted, all previous analyses are removed. It's important that when the administrator enables this feature, they take into consideration that the application should not be stopped.

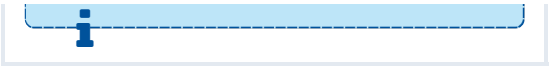
Profiling stats are shown in a table structure:



Field	*Description*
Class	*The Java class name.*
Method	*The Java method name in the class.*
Calls	*Number of calls.*
Max dur.	*Maximum duration of the calls.*
Min dur.	*Minimum duration of the calls.*
Avg dur.	*Average duration of the calls.*
Total dur.	*Accumulated duration time.*
Action	*List of all calls.*

The duration time is in milliseconds.

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



Enable profiling stats

- *On the top right, click on*  ***Enable button***.

Disable profiling stats

- *On the top right, click on*  ***Disable button***.

Clear profiling stats

- *On the top right, click on*  ***Clear button***.

Refresh profiling stats

- *On the top right, click on*  ***Refresh button***.

List of detailed calls

- *Click on the*  ***View icon***.

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

Purge data

The purge data feature helps administrators delete users and role grants assigned to objects (documents, folders, emails, 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 cannot be reverted. Use with care.

- *In any list, click the*  ***Delete icon***.

Filter

- *To search in any list, click the*  ***Filter button***.

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

Purge trash

The purge trash feature helps the administrator clean all users' trash.

When a user deletes objects (folders, documents, emails, or records), they are placed in the user's trash to prevent accidental deletions and to allow recovery of lost data.

Although users have a feature to clean their own trash, they usually do not use it. That can result in a lot of garbage in the trash.

The administrator should periodically check the repository growth, and especially the size of the trash.

Purge

This operation cannot be reverted. Use with care.

- ***Type Yes** into the input box.*
- *Click the **Purge button**.*

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

Rebuild indexes

The Rebuild Indexes feature helps the administrator rebuild Lucene indexes.

Indexes	*Description*
Text extractor	*Extracts all document contents and rebuilds the indexes.*
Lucene indexes	*Rebuild indexes with actual content.* *When you choose this option, you can rebuild all node types or only the selected ones. 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 content that no longer exists in the repository.*

Rebuild

- *Choose the index to rebuild.*
- *Click the **Execute button**.*

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

Repository checker

The repository checker feature helps the administrator verify the repository's integrity.

When the administrator thinks the system could be compromised, for example a hard disk failure or similar, he can use the Repository checker feature to verify the application's integrity.

Option	Description
Path	*The path from which the verification process will start.* *Common values:* • */okm:root* • */okm:trash* • */okm:personal* • */okm:mail* • */okm:templates*
Fast	*The fast verification will only check the existence of the latest document version.*
Versions	*Verifies the existence of all document versions. This option has priority over the Fast option.*
Checksum	*Calculates the checksum of the repository document and compares it with the database value.*

Check repository

The process can take several minutes or hours depending on the repository's size and the options checked.

The repository checker process will show progress as a percentage. Before this is shown, the application calculates the total number of nodes that should be checked. In a huge repository, this calculation can take several minutes; please be patient.

- *Set the path.*
- *Choose repository checker options (Fast, Versions, Checksum).*
- *Click the **Check** button.*

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

Repository export

The repository export feature helps the administrator export the whole or part of the OpenKM repository to the file system. The output will be the same folder and file structure as set in 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 document versions are exported.*

Export

- *Set the repository path.*
- *Check the repository export options.*
- *Click the Export button.*

Understanding output format

When metadata and history are not checked

The result will be a folder structure containing documents and mails. Although a document may have several versions, only the

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

current version will be exported.

When only metadata is checked

The result will be a folder structure containing documents and mails. Although a document may have several versions, only the current version will be exported.

For each folder, document, or mail, an ***.okm*** *file will be created. For example, for a file named "Administration.html", a file named "Administration.html.okm" will be created in the same path. The* ***.okm*** *files contain all the metadata information associated with the node in JSON format.*

Example of an .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}
}
```


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



Keep in mind that the JSON output format can change if the main objects (folder, document, mail, or record) have fields added, removed, or modified.

When 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)*

When 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, an ***.okm*** *file will be created. For example, for a file named "Administration.html", a file named "Administration.html.okm" will be created in the same path. The* ***.okm*** *file contains all the metadata information associated with the node in 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 administrators import data from the file system.

Initial import (import standard folders and documents from your OpenKM server).

Import from a 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) is present.*
History	*When enabled, the importer assumes all versions of the document are present.* *More information about the document versions format is available at Repository export.*
Restore UUIDs	*When enabled, the importer will keep the same node UUID.* *When an application is integrated with third-party software, the third-party software usually stores the node's UUID in its own database as the main reference. These references should not be lost; that's why in these scenarios it is necessary to preserve the UUIDs.*

Especially when importing from a previous OpenKM export, you must use the same import options that were chosen during export; otherwise, the import will fail.

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 the **Import button**.*

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

Scripting

Scripting is an advanced feature that enables administrators to execute 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 do the same things with a few mouse clicks. For compatibility reasons there is also an Automation task where you can put 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 a script as part of the application starting process

A script can be executed as part of the 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 workflows) each time the OpenKM demo is restarted.

Execute a script as part of the application stopping process

A script can be executed 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 are several variables that are directly injected into the scripting environment and can be used.

List of variables that can be used directly:

Variable	*Description*
systemToken	*The system token.*
node	*Involved node (NodeDocument, NodeFolder, etc.)*

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

uuid	*Node's unique identifier.*
file	*Only available when uploading or updating a document's content.*
userId	*The user ID.*

Example:

```
import com.openkm.util.*;
import com.openkm.api.*;

OKMProperty okmProperty = ContextWrapper.getContext().getBean(OKMProperty.class);

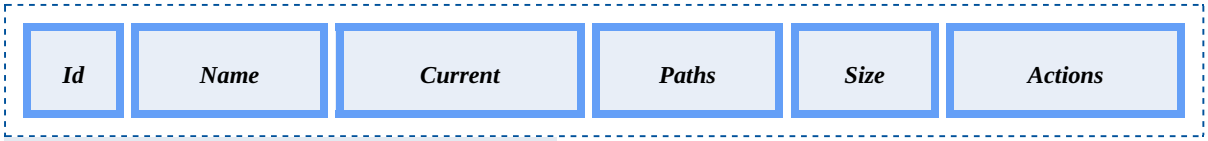
// Variable uuid is injected by automation to the scripting environment
okmProperty.addKeyword(null, uuid, "test");
```

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

Shards

The shards feature manages disk sharding.

Shards are shown in a table structure:



Field	Description
Id	*The unique id of the shard.*
Name	*The name of the shard.*
Current	*Boolean indicating whether it is active.*
Paths	*Paths to the datastores.*
Size	*Total size.*
Actions	*New Shard.* *Refresh.* *Refresh size.* *Edit a shard.*

Shard actions

- *To create a new shard, click on* ***New shard button****.*
- *To refresh, click on* ***Refresh button****.*
- *To refresh the size, click on* ***Refresh size button****.*
- *To edit a 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 shows 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

The Workflow tenant view shows the registered workflows in the current tenant. In this view, you can enable workflows for the current tenant.

Workflows are shown in a table structure.



Description:

Property	*Description*
Workflow name	*You must register the workflow name with the exact name used in the workflow definition.*
Actions	*Available actions:* • *New.* • *Edit.* • *Delete.*

Register workflow

- *Click the top-right to add a Workflow tenant.*
- ***Set the workflow name***.
- *Click on the* ***Create button***.

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

Workflow forms definition

Workflow forms are 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 the latest DTD on our website http://www.openkm.com/dtd/workflow-forms-**{major}.{minor}**.dtd where **{major}** and **{minor}** 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 the 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 a DTD. In the previous example, the DTD is **workflow-forms-2.6.dtd**.*
- *The XML begins with the **<workflow-forms> tag** and ends with **</workflow-forms>**. In this hierarchy, the metadata groups are defined.*

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 identifies the **beginning and end** of a new **workflow form task definition**.*
- *Each task has a **unique name**, **"evaluate price"**.*
- *Do not use special characters in the name.*

Attribute	*Description*	*Required*
task	*Unique field identifier.*	*true*

	Two metadata groups can't have the same name. ***The name must be unique****.* *The name must start with "**okg:**".* *Do not use special characters in the task name. We suggest (0-9a-zA-Z_).*	
defaultValueClassName	*Set custom JAVA class for setting initial default values.*	*false*

Two form tasks can't have the same name.

Do not use special characters in the 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 is 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 a file system DTD

If the server has no access to the Internet, then the DTD file must be accessible in some way; for example, you can store a copy on your application server.



The XML definition must be changed to refer to the new DTD location; otherwise you will get an application error while trying to process a DTD file.

It can be a file system path (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.*

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

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 child elements, validator elements, 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*
    - *Download*
    - *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*
- *Download*
- *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 in the user interface for decision-making.

For example, two buttons: accept and deny. One to accept something and the other 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 same 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 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, the UI must use width values based on percentages, 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 the user interface.*	*true*
name	*Unique name identifier* *Two fields can't have the same name in the same task definition.* ***Name must be unique***.	*true*
width	*The width of the HTML element.* *In the case of KCenter UI, you 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.* *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 a node element.*
- *The* ***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 a node element.*
- *The **uuid** is a valid document uuid.*
- ***data** is a valid mapping variable. This **variable** is **used to dynamically set the uuid attribute** on the 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 Validator 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 that is not included in these predefined ones, you can use a regular expression. For example, imagine you want to restrict an input to lowercase 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 valid document 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 you want to assign a uuid dynamically, because it is used as a 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 cannot have the same name in the same task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the 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 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 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, and optionally documentName if you want to force the document name.

To update an existing document, you should set the document UUID and set the type to "update".

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
```

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

```
  </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 that 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 cannot have the same name within the same 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, the 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 field will not be editable 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, this can be used to dynamically load the true or false value of the checkbox.*	

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

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 it 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 same task definition. The name **must be unique**.*	*true*
type	*Value type. 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 the OpenKM API.*	*false*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the 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 field cannot be modified in the user interface. By default, the value is "**false**".*	*false*
timeFormat	*When the input type is a date, we can specify whether 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 same 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, the 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*

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

optionsData	*When present, it is an identifier used to load options.*	*false*
readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*

Child elements:

- *Option. Pairs 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**"/> when a hierarchical relationship is set.*
- *Validator. For more information, see the Workflow Validator element.*

***** There are 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 used; 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" >
      <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 statement should be placed in 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 retrieve a list from external applications or for other purposes, you can build a class to obtain the option values from those systems.

```
<?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>
```

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

For more information, read: Creating your own Option Select Values plugin.

Hierarchical relationship

- *The field option value of "**childrenSelect**" has a parent relationship with "**parentSelect.**" It's defined by **parentElement="parentSelect."***
- *The option value "**benefits**" of the field "**childrenSelect**" has a parent value of "**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)
);
```

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

```
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>
```

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

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 same task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* *In the case of KCenter, the UI must use width values based on percentages; 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>
```

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 same task definition. The name **must be unique**.*	*true*
value	*Specifies a default value for this component.*	*false*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the 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 number of characters in the input required to begin filtering results.*	*true*
dialogTitle	*The title of the HTML dialog.*	*true*

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

data	*When present, it is an identifier used to load data.*	*false*
readonly	*The 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 are three options to get suggest box values (using **one of them is mandatory**):*

- *filterQuery + valueQuery*
- *className*

No other combination is allowed.

Based on filterQuery and valueQuery (without table attribute)

*Now **filterQuery** and **valueQuery** can be regular SQL queries.*

You should only use {0} as the parameter representing the user input 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}'" />
```

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

```
  </property-group>
</property-groups>
```

Based on className

To obtain a list from external applications or for other purposes, your class can be built to retrieve 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

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

Workflow Text field

Print text that can contain HTML tags.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.* *Starting with version 7.1.41, the label can render HTML. In the case of an HTML value, the string must be encoded, for example:* `<font color="red">red text</font>` *It 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 the data value is set from the workflow transition, then the value must not be converted.*	*true*
name	*Unique field identifier* ⚠ *Two fields can't have the same name in the same task definition. The name* ***must be unique****.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the UI must use width values based on percentages; for example:* • *width="50%" will work* • *width="100px" will be ignored by the KCenter UI.*	*false*
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.

```
<?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="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

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 fields can't have the same name in the same 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, the 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.*
description	*General description.*	*false.*
data	*When present, it is an identifier used to load data.*	*false.*
readonly	*The 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 choose from.

Attribute	*Description*	*Required*
label	*The text shown as the label in the user interface.*	*true*
value	*A unique value identifier.* *Two options cannot have the same value. **The value must be unique**.*	*true*
parentValue	*When present, it specifies the option's parent.* *The parent must be present in another select.*	*false*
selected	*When present, it specifies that an option should be pre-selected when the form is rendered.*	*false*
cssClass	*CSS class.* *For more information about creating your own styles, see: CSS.*	*false*

Parent:

- *Select. For more information, please see: Workflow Select field.*

Basic select with selected option

- *The attribute selected="true", when present, 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 relationships

- *The field option value of "**children**" has a parent relation with "**parent**". It is defined by **parentElement="parent"**.*
- *The option value "**benefits**" of the field "**children**" has a parent value "**directives**" in the 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 same task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* *In the case of KCenter, the UI must use width values based on percentages, 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>
```

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 same task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* *In the case of KCenter, the UI must use width values based on percentages (%), 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 information 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 provides access to all features, organized by type.*
- *The **Quick search box** is at the top right.*
- *The **Toolbar** contains the most commonly used features.*
- *The **Desktop** tab enables you to navigate the repository.*
- *The **Search** tab enables querying 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 lifecycle, saving time and money. It also allows the social activities around the content to connect people to other people, information to other information, and people to information, helping to manage the collective intelligence of the company's human resources more efficiently.

The OpenKM Profiles feature helps administrators set what is displayed for each user of the application.

OpenKM provides the solution to:

- *Control the growing volume of information to be managed.*
- *End the diversification of information storage sources.*
- *Prevent the duplication of data.*
- *Obtain valid information through the control of document versions.*
- *Reduce the time and resources used in locating information.*
- *Avoid knowledge leaks or wrongful access by users, thereby complying with the obligations set in the LOPD (Spanish Data Protection Law).*
- *Restrict unauthorized users' access to the information.*

Through the OpenKM system, all relevant and valuable information can be accessed and found more quickly to resolve

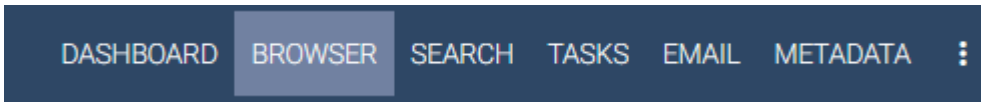
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

problems and make decisions.

Menu bar

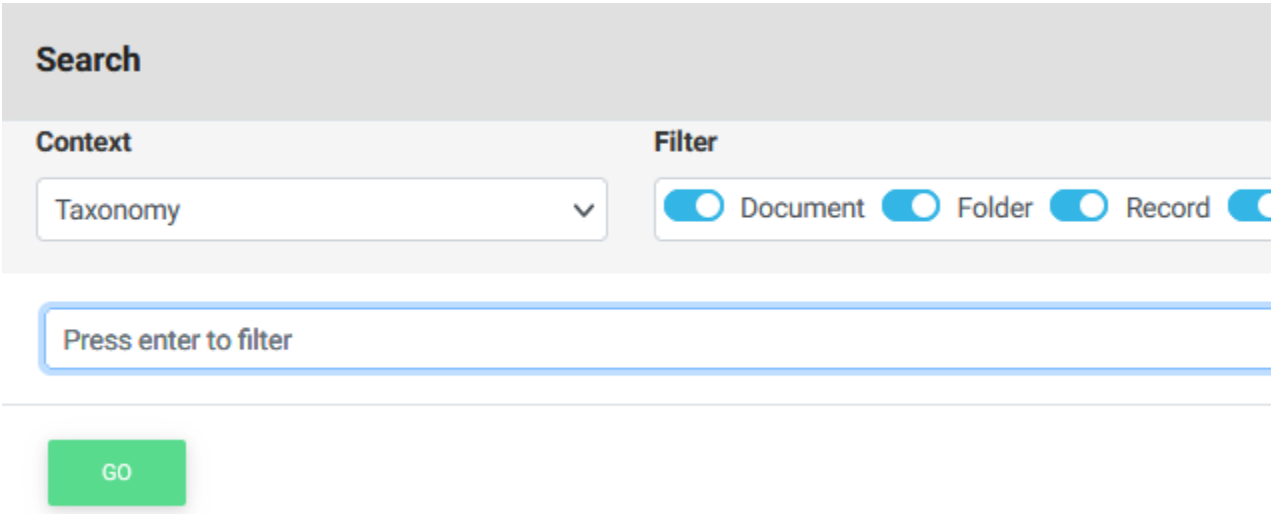
The menu bar at the top has all features organized by type:

- *File.*
- *Edit.*
- *Tools.*
- *Bookmarks.*
- *Templates.*
- *Help.*



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 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 the 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.

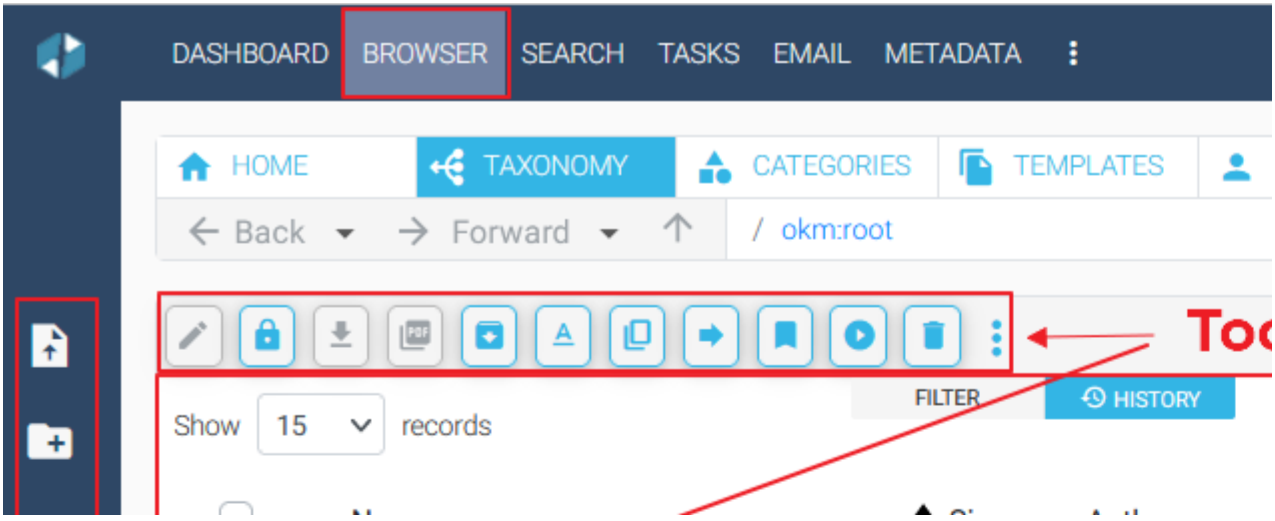
Browser

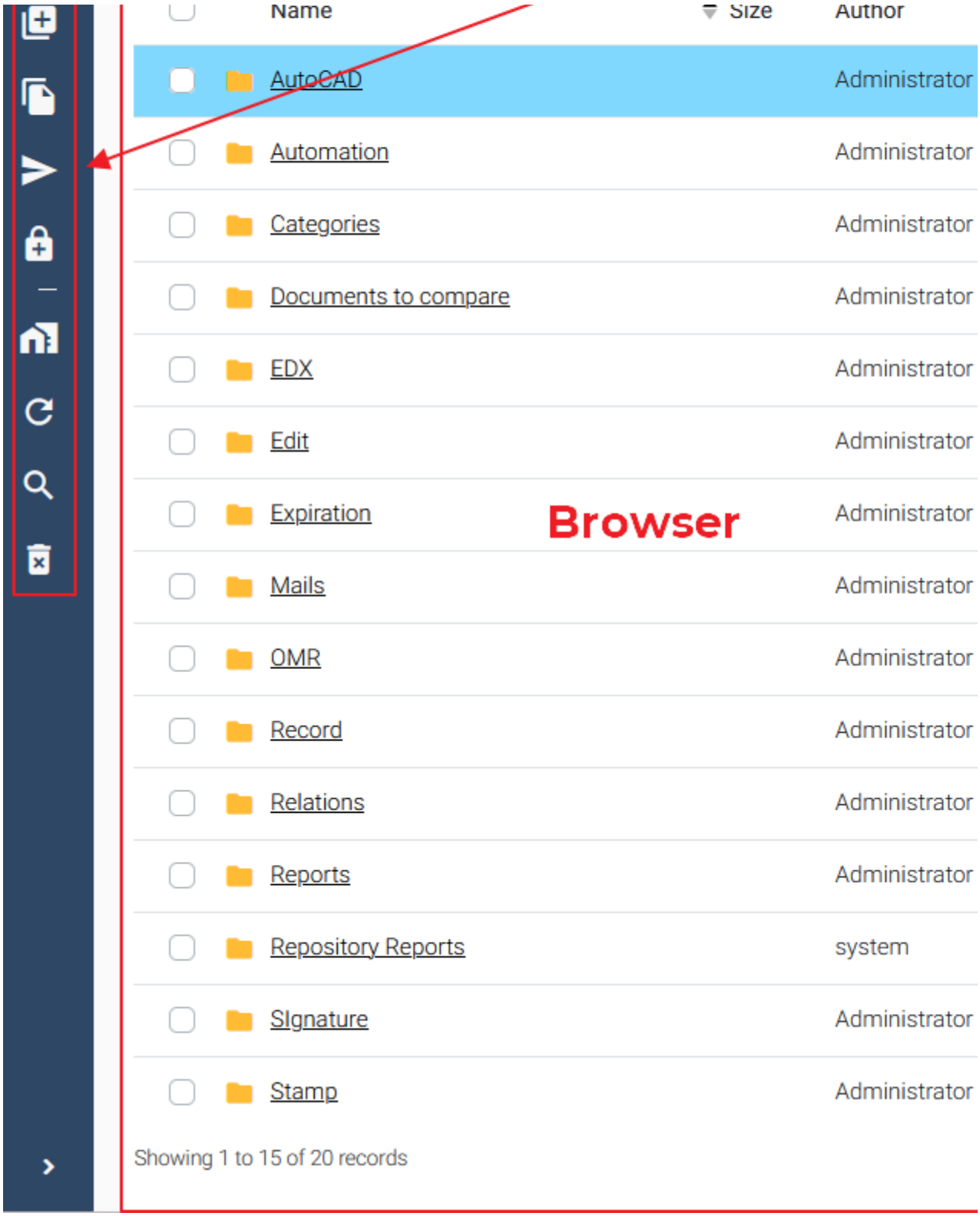
The browser panel at the top center shows a table of nodes (documents, folders, emails, and records) for the selected navigation node.

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 navigation node or a browser node.

> *The selected node in the navigation panel determines the content shown in the browser panel. If the last node chosen by the user is in the navigation or browser panel, the properties panel will show the properties of the active node in the navigation or browser.*





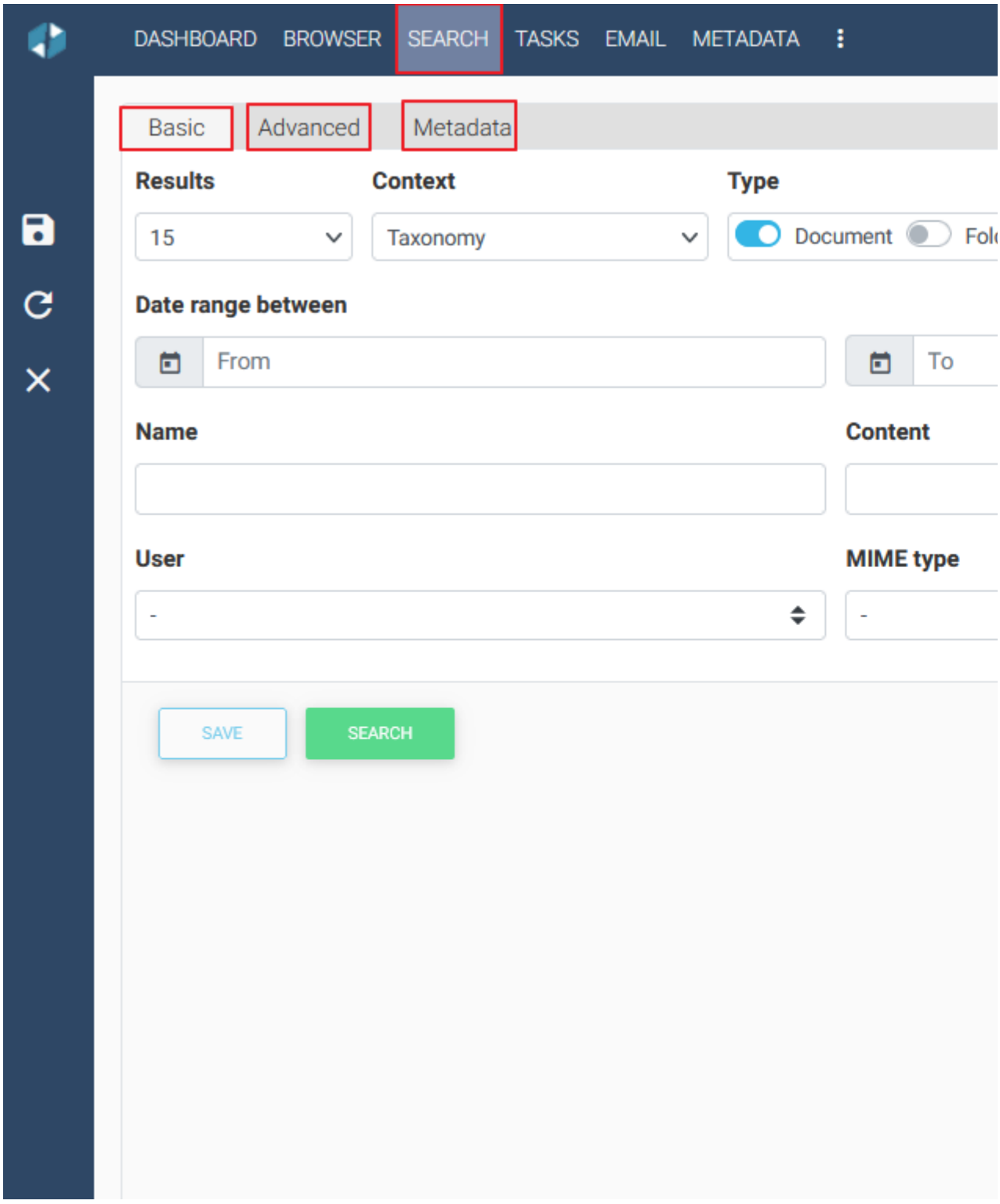
Search

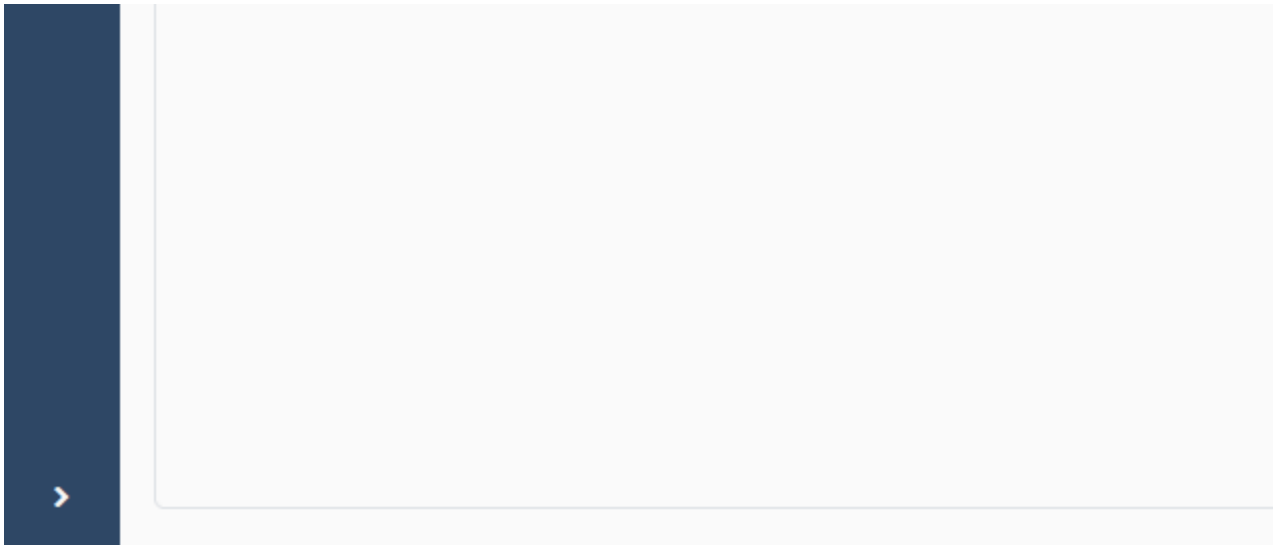
The search tab enables querying the repository. This tab gives access to different searches to query the content of the

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 for keywords, metadata, and additional criteria, such as location in the application.*



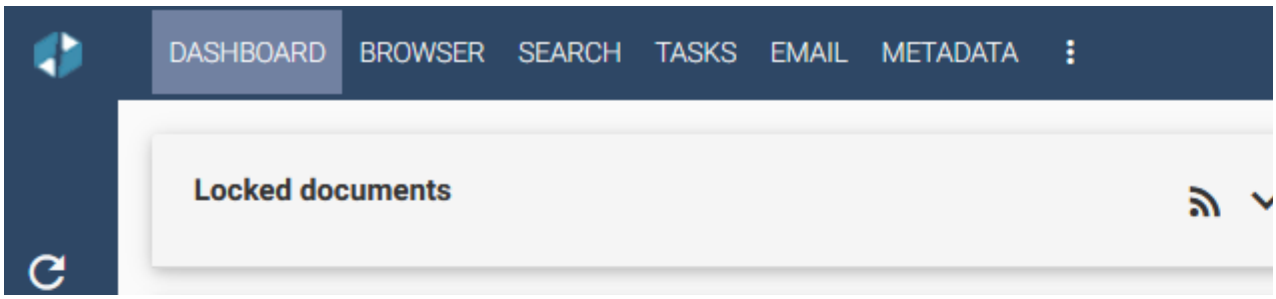


Dashboard

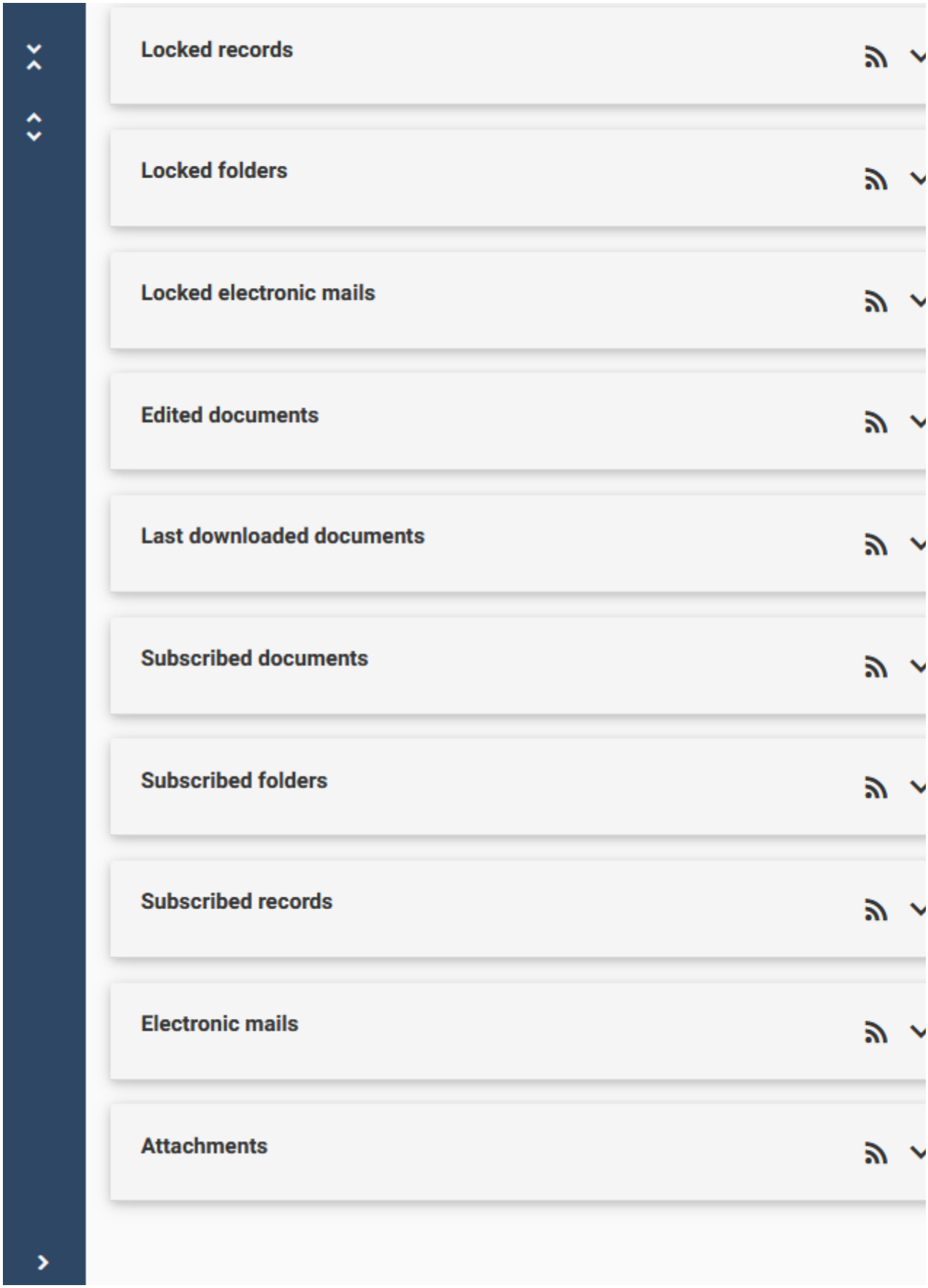
The dashboard panel provides a set of dashboards; each user has a set of personal dashboards that display the information relevant to them and help provide a global view of the application's activity.

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**. Provides virtual navigation across all mail repositories.*
- ***Mail account**. Helps users 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 manage the application.

> *More information in the Administration guide.*

Status

The status panel provides general information:

- *OpenKM news alert.*
- *The connected user.*
- *User repository size.*
- *Chat.*
- *Alerts.*
- *HTML documents in the editor.*
- *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 are simple.

This quick start guide gives you an introduction to some of OpenKM's features:

- *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, it focuses on concepts; in practice, a good taxonomy should be simple, easy to remember, and easy to use.*

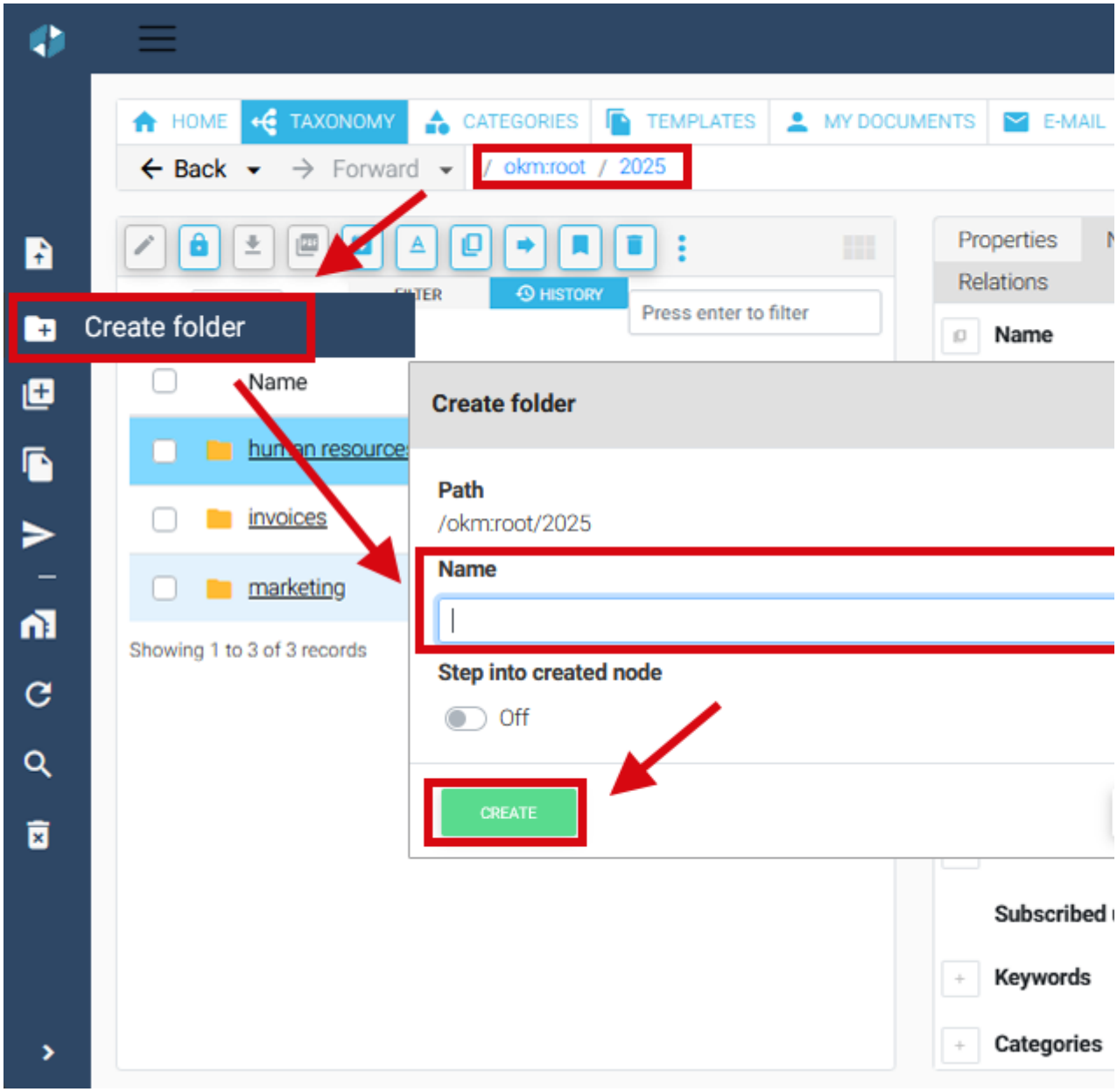
The first step should be analyzing the company's organization and defining a folder hierarchy to store the organized documents.

Take this one as a sample:

/okm:root/	***2015/***	***invoices/***
		human resources/
		marketing/
	2016/	***invoices/***
		human resources/
		marketing/

To create a folder:

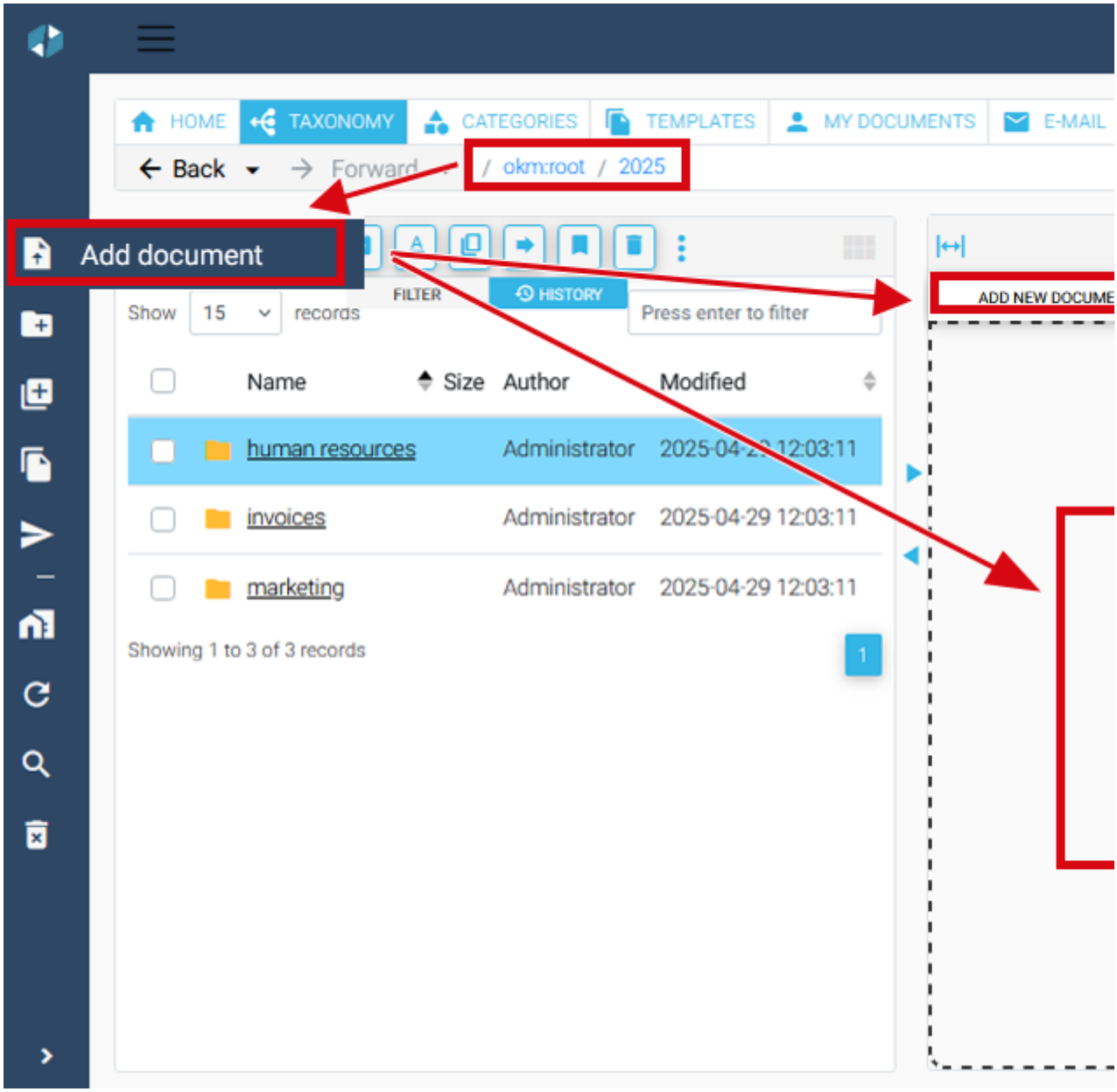
- ***Select a node** from the taxonomy where you want to create the folder.*
- *Click on the **folder icon** in the toolbar.*
- *Set the **name**.*
- *Click the **Create** button.*



We suggest spending some time thinking about and designing your taxonomy. We suggest keeping fewer than 200 elements per folder for the best UI performance.

Add documents

- ***Select a node*** *from the taxonomy where you want to add the document.*
- *Click the* ***add document*** *icon in the toolbar.*
- ***Select the document*** *from your computer or* ***drag & drop****.*



Edit documents

*Document editing 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 simultaneously want to edit the same document in OpenKM.*

*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.*

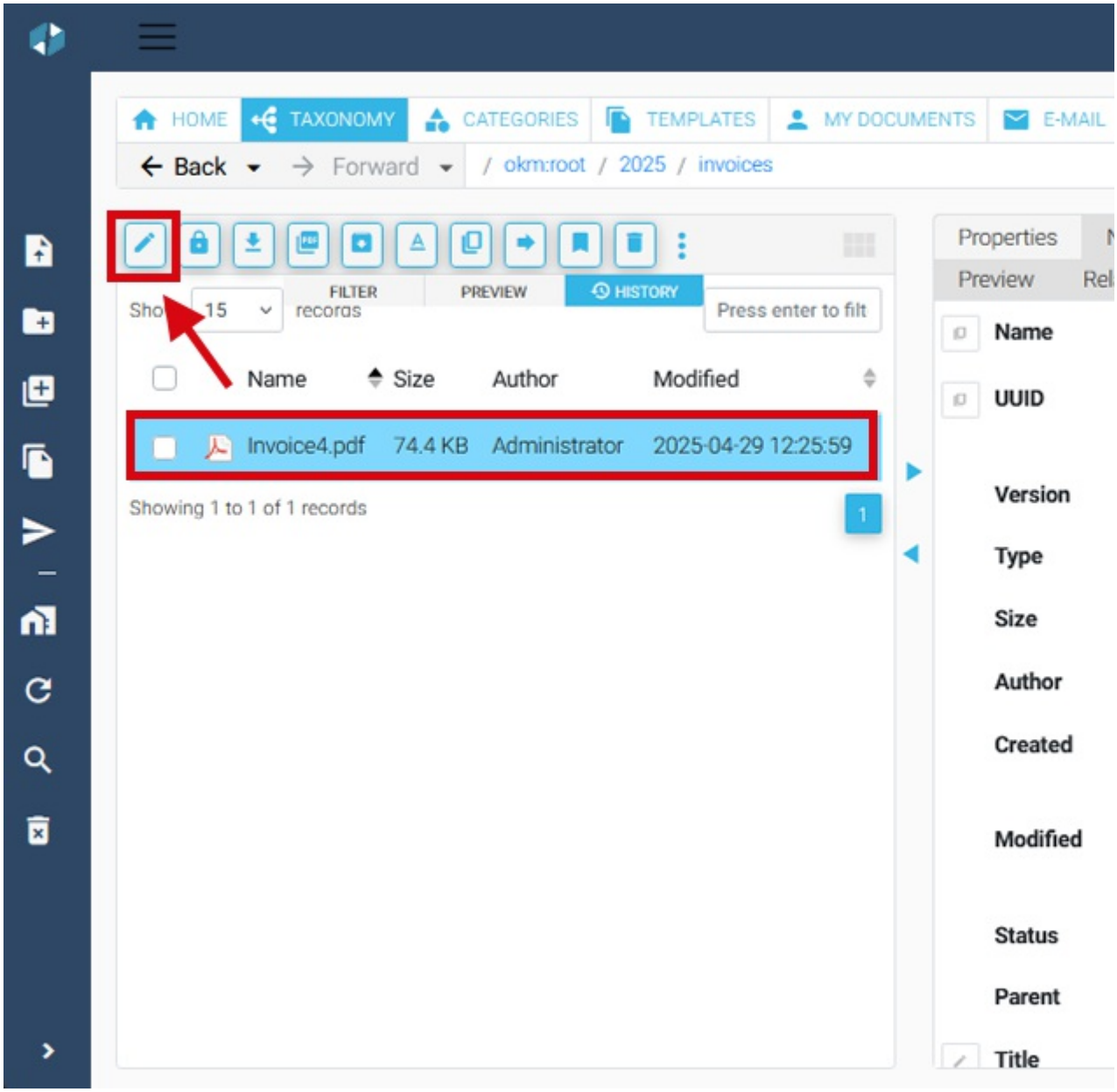
*The term "**check-in**" describes updating a document to replace the previous version. By checking in the item, the user can allow others to edit the document without worrying about overriding changes that others have made.*

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 in 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 checked out.

First, check out the document:

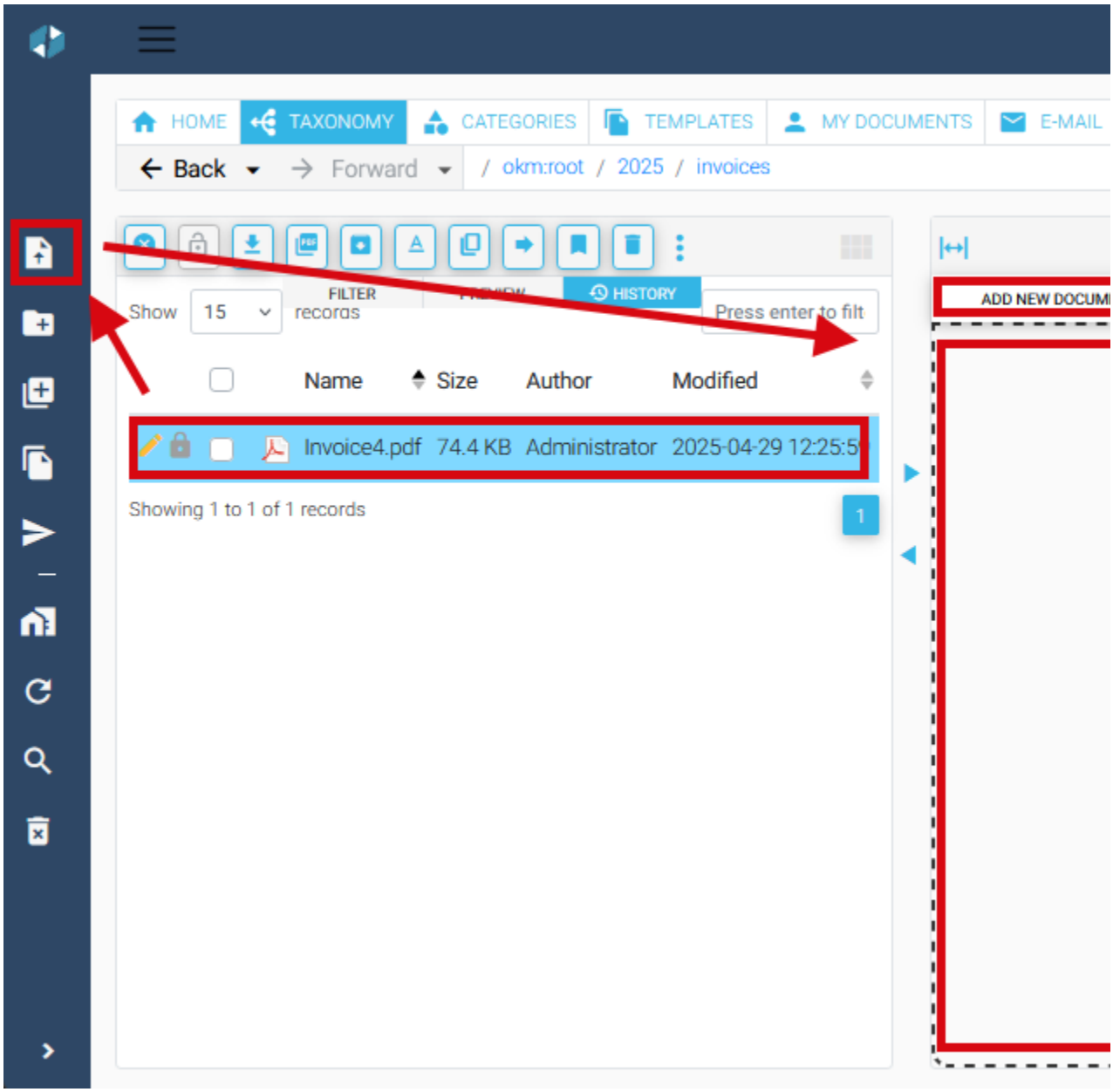
- ***Select a document***.
- *Click the **Edit** icon.*



Finally, check in the document to upload the changes:

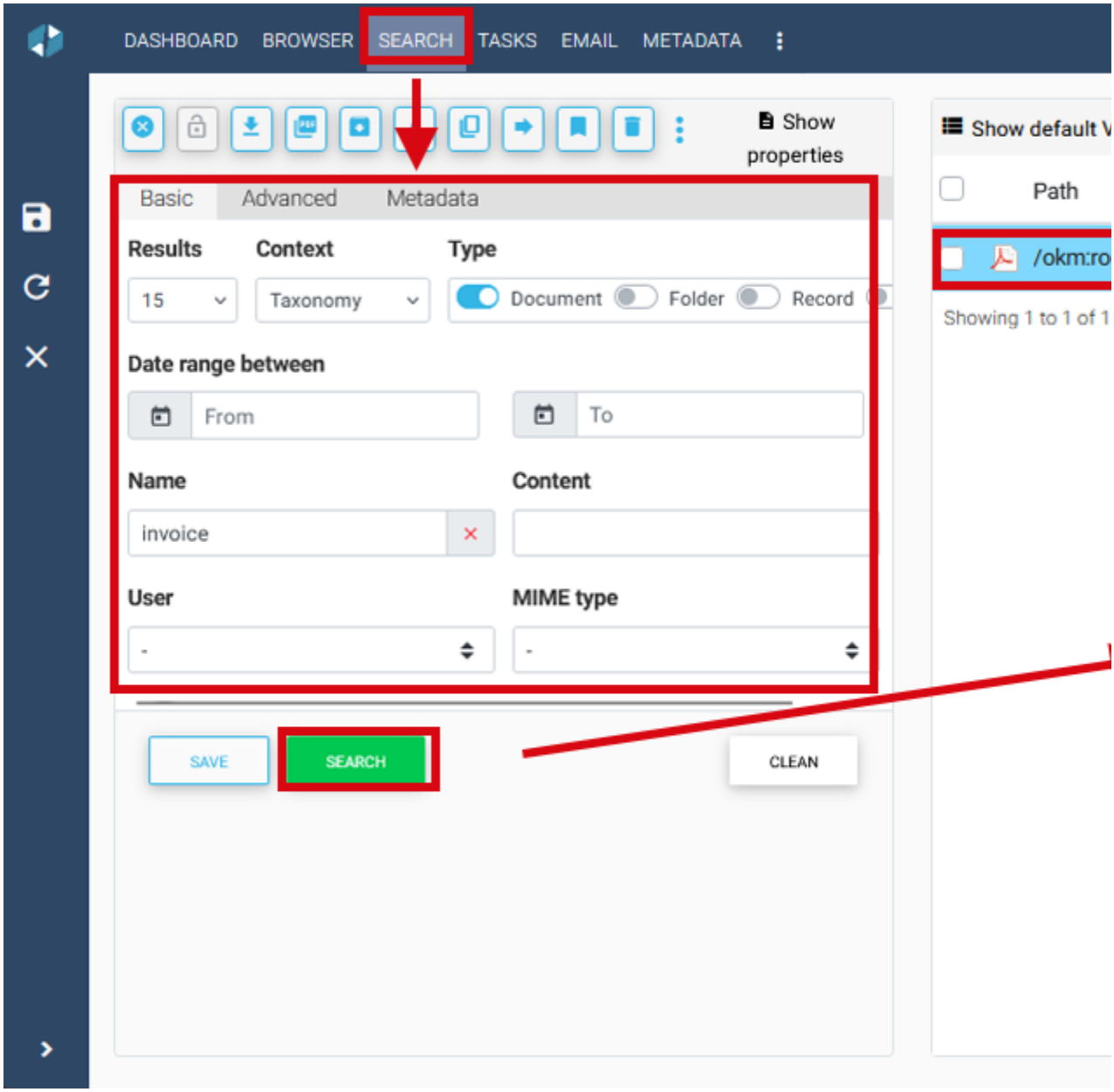
- *Select the **edited document**.*

- *Click the **Add document** icon.*
- ***Select the document** from your computer or drag & drop.*



Search documents

- *Go to the **search** view.*
- *Enter the search term for the query.*
- *Click on the **Search** button.*



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

Essential fundamentals

Basic concepts

OpenKM has several built-in types of nodes:

- *Folder.*
- *Document.*
- *Mail.*
- *Record.*

*A node type **folder** is a **storage unit or a container**. Like a document, it also has a name but is not a file; it can contain documents, records, and other folders.*

*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 (which in the mail context are called attachments).*

*In essence, there are 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 contain documents, mails, folders, and other records.*

> ***Non-digital record sample:***
>
> *When OpenKM is used to store the location of papers in the real world, each record is only a set of metadata fields that helps users locate papers, and there is no binary information associated (like an 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 and mails associated with that process.*

Understanding taxonomy, keywords, categories and metadata groups.

Spending time on node categorization is good practice. A good categorization will help users quickly find the desired information, use less time, save money, and be more efficient in your business processes.

OpenKM helps you with this by providing basically four ways to recategorize 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, in practice, a good taxonomy should be simple, easy to remember, and easy to use.*

For best performance in the UI, we suggestâ€"whenpossibleâ€"a taxonomy where each parent has 200 children or fewer.

Each folder in a taxonomy represents a "concept" and lists a number of nodes (folders, documents, mails, or records) associated with it.

Keywords

A keyword, in the context of document management, is a particular word or phrase that describes the content. Setting keywords is the simplest and most basic system for categorizing the nodes in your repository, but at the same time it is the least powerful. For large repositories, we encourage using metadata groups.

Limitations of keyword usage:

- *When you have thousands of keywords, searching by keywords can be inefficient (you are not able to refine the search and you get a lot of results with the same keywords; users do not follow a restricted cataloging method). A tag cloud diagram in these cases is usually an unusable graphical representation of keywords.*
- *It's difficult for users to set keywords because they do not know the company's complete keyword dictionary.*
- *Users write keywords in similar ways but not always the same (for example, users use the words "building", "construction", and "edifice" indistinctly).*

You can get the best results when:

- *You limit the number of users who are able to set keywords.*
- *You use a controlled keyword dictionary (more information at http://www.w3.org/2004/02/skos/).*
- *You share a complete company keyword dictionary with your users. That helps them understand how knowledge is mapped in your organization and helps users find it.*

Categories

In OpenKM, the categories are a virtual taxonomy that helps navigate taxonomy nodes across other concepts. Each category node represents a "concept" and lists a number of nodes (folders, documents, mails, or records) associated with it.

We suggest using categories rather than keywords.

Metadata

Metadata groups are data that describe other data, providing information about certain nodes' content. Metadata groups help create controlled classifications and are a powerful way to locate nodes and refine search queries.

With metadata groups you can simulate the keywords and categories features. We encourage you to spend some time investigating the metadata feature, and we suggest using it rather than keywords and categories.

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

Plan your data organization

Before 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 intensive use of your application.*

A good analysis of your organization and an understanding of the application can help you find the best way to solve your problems. OpenKM is not a closed application; it is more than a couple of features focused on document management. OpenKM is a flexible application and can be used in several ways. It's a good idea to spend some time understanding your organization, your business logic and the processes involved and how the OpenKM features can work together in your day-to-day, helping your organization become more efficient and save money.

Some questions you should ask yourself about your organization:

- *Where are your information and documents?*
- *What kind of documents do you have?*
- *Are your documents in electronic format or not?*
- *What format are your documents? PDF, MS Office, LibreOffice, etc.*
- *How are you currently finding the documents? How would you like to find them, and which terms do you consider relevant?*
- *Who can access the information and in which way?*
- *Do you need a workflow process?*

What are the objectives of the previous questions?

- *List in 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-digitized information that should be processed (scanned). The digitization process can be a basic process.*
- *Identify for each type of information (document type) what information is relevant (metadata). For example, invoices usually have an invoice number, a date range, and customer information, among others. The metadata will help users retrieve information quickly and efficiently.*
- *Identify the security that must be applied to each document type. Determine who can read, write, or delete information. Also, your security needs will help you set the level of application logging you require. OpenKM provides a full audit of any user action that can be configured in a more or less restrictive way.*
- *Identify workflows and the people involved in them.*

> *The goal is to create a plan with a calendar and milestones.*

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

What are the goals you are looking for in the implementation of the application? Usually they could be:

- *Set some order in your organization. The application "forces" users to work in a certain way, which benefits your company because it helps you "set rules" that all users must follow.*
- *Consolidate your intellectual capital. By consolidating all your company's knowledge in a single place, that knowledge becomes available to be shared among all your users, and you can set rules for it.*
- *Help comply with legal regulations.*

Try to set a specification in a single document answering these questions and others. It will help you plan how to organize the information.

Good practices

- *Consider OpenKM a project that affects all your departments and usually your customers.*
- *Choose the leaders.*
- *Implement the project gradually. You can start with a department, a group of users, or a specific business process. Think of the implementation as a virus in your company. A good leader should demonstrate the benefits of the new system and show them to other users. Users usually perceive any change in the company as negative; if they perceive the application as something that helps their work (efficient search, saving time, and solving real problems), they will be the first to be interested in using it. One of the major goals of OpenKM is an intuitive UI, which helps in this process (after two weeks the user feels comfortable using the application).*
- *Keep it simple; the best solution is usually the easiest way to solve problems.*
- *Listen to your users and try to get them involved in the project.*
- *Periodically review your plan and try to refine it and identify possible problems.*

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

Working with the menu bar

On the top menu bar, all available features are organized by type. The menu options are 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. Other options not described here may be shown in your menu bar if you have OpenKM extensions enabled.

The 'Massive' column in the tables indicates you can select several nodes at the same time in the browser panel to apply the same action.

File

Feature	*Description*	*Massive*
Find folder	*Finds a folder 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 can be executed only in the context of the selected stack.*	*No.*
Find mail	*Finds an email in the repository.* *The search can be executed only in 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 in the context of the selected stack.*	*No.*
Find similar document	*Finds similar documents in the repository. The feature uses the search engine heuristic based on the contents of the selected document.* *The search can be executed only in the context of the selected stack.* *This feature is only enabled when a document is selected in Browser view.*	*No.*
Create folder	*Creates a new folder in the selected node of the navigator panel.*	*No.*
Add document	*Uploads a new document to the selected node of the navigator panel.* *When the option "Import documents from ZIP" is checked, you import the contents of a ZIP file while respecting the folder tree hierarchy. 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 an email notification to other application users.* *Optionally, based on **Administration** > Profiles configuration or **Administration** > Automation, a wizard is shown. The wizard may require you to set metadata groups, categories, or keywords.*	*No.*
Add record	*Creates a new record in the selected node of the navigator panel.*	*No.*
Download document	*Downloads the selected document from the browser panel.*	*No.*
Download document	*Downloads the selected document from the browser panel in PDF format.* *The feature is enabled when it is possible to convert the document to PDF format.*	*No.*

as PDF		
Send link	*Sends a URL to the selected document in the browser panel by email.*	*Yes.*
Send document attachment	*Sends the selected document from the browser panel as an email attachment.*	*Yes.*
Generate download link	*Generates a download token for the selected document in the browser panel and sends the URL.* *For more information take a look at Download tokens.*	*No.*
Forward e-mail	*Forwards the selected mail in the browser panel.*	*No.*
Write mail	*Creates a new email directly from the user interface.*	*No.*
Create from template	*Creates a new document based on an existing one in the templates space.*	*No.*
Export to ZIP	*Downloads the selected nodes compressed into a ZIP file. If you have problems with filename encodings, please take a look at Client configuration.*	*Yes.*
Start workflow	*Starts a workflow on the selected node in the browser panel.* *You can start a workflow on any node type (document, folder, mail, or record).* *Optionally, some workflows require you to fill in a form as part of starting them.* *The available workflows are set in **Administration** > Profiles.*	*No.*

Refresh	*Refreshes the user interface.*	*No.*
File uploader	*Bulk upload feature.* *This feature requires Java to be enabled in your browser. More information at Bulk Uploader.* *If you're using a recent version of Google Chrome, keep in 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	*Empties the user's trash bin.*	*No.*
Exit	*Exits the application.* *Users should always exit the application using this menu option and not just by closing the web browser. In any case, after 30 minutes the user's web session on the server will be automatically terminated.*	*No.*

Examples of allowed queries for 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 a question mark "**?**" as a placeholder for any unknown or wildcard character; for example, the query "**?015**" will return occurrences where the first character doesn't matter.*
- *The query "**2015**" is a synonym of "**2015***". In this search view, the application always adds "*****" at the end of your query.*

Edit

Feature	*Description*	*Massive*
Lock	*Locks the selected document in a browser panel.*	*Yes.*
Unlock	*Unlocks the selected document in the browser panel.* *Only the user who locked the document or a 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 document changes. The application sends the notification by email.*	*No.*
Cancel edit	*Cancels a document edit.* *Only the user who edited the document or a user with administration credentials (force cancel edit) can cancel the edit of the node.*	*Yes.*
Delete	*Deletes the 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.*

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

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 with the navigation of taxonomy nodes across other concepts.*	*Yes.*
Add keyword	*Adds a keyword to the selected node.* *When the thesaurus is enabled, you can choose terms from it (SKOS).*	*Yes.*
Merge PDF	*Merges two or more PDF files. The* ***result of the process is a new PDF document****.* *You must choose 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 resulting from merging the selected PDF files.*	*No.*
Add metadata group	*Adds a metadata group to the selected node.*	*Yes.*
Update metadata group	*Updates the field values of a metadata group of the selected node.*	*Yes.*
Remove metadata group	*Deletes the metadata group of the selected node.*	*Yes.*
Add subscription	*Subscribes to the selected node.*	*No.*

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

	You can subscribe on any node type (document, folder, mail or record). In 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	*Unsubscribes from the selected node.*	*No.*

*The document editing 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 to administrators.*	*No.*
User configuration	*Shows the user configuration. You can change your password and email address.*	*No.*

OCR data capture	*Applies Optical Character Recognition to the selected node.* *The data captured is stored in a metadata group.* *This option is only available if you have enabled an OCR template. For more information about it, take a look at **Administration** > OCR templates.*	*No.*
Web page import	*Captures a web page and imports it as a 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 into another format.* *Available options:* • *When the **Save** checkbox is checked, a new document is created in the repository.* • *When the **Download** checkbox is checked, the converted document is automatically downloaded.*	*Yes.*
DXF preview	*When the AutoCAD extension feature is enabled, AutoCAD files are previewed as DXF.* *This option is only available if you have enabled the **Autocad preview** feature.*	*No.*
PDF preview	*When the AutoCAD extension feature is enabled, AutoCAD files are previewed as PDF.* *This option is only available if you have enabled the **Autocad preview** feature.*	*No.*
Add new document encrypted	*Uploads a new document and encrypts it.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*

Encrypt document	*Encrypts the selected document in the repository.* *When a document is encrypted, all previous version history will be removed (because previous version history is not encrypted).* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
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 the decrypted version of the selected encrypted document.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
Decrypt document	*Decrypts the selected document in the repository.* *This option is only available if you have enabled the **Cryptography** feature.*	*No.*
Send new message	*Sends a new internal message using OpenKM internal messaging.* *This option is only available if you have enabled the **Messaging** feature.*	*No.*
Sign document	*Applies a digital signature to the selected document.* *This option is only available if you have enabled the **Digital signature***	*No.*

	feature.	
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 Google Drive.* *This option is only available if you have enabled the **Google Drive** feature.*	*Yes.*
Import from Google Drive	*Imports data from Google Drive to the selected folder in the Navigator.* *This option is only available if you have enabled the **Google Drive** feature.*	*Yes.*
Create new document	*Creates a new document and starts editing with Google Drive integration view.* *Google Drive integration allows collaborative editing with several users at the same time.* *This option is only available if you have enabled the **Google Drive** feature.*	*No.*
Edit	*Edits the selected document with Google Drive integration view.* *Google Drive integration allows collaborative editing with several users at the same time.* *This option is only available if you have enabled the **Google Drive** feature.*	*No.*

HTML Editor	*You can enable or disable the integrated HTML editor.* *This option is only available if you have enabled the **HTML Editor 4** feature.*	*No.*
Live Editor	*You can enable or disable the Live Edit feature. When enabled, you edit the selected document in its native application (for example MS Office, OpenOffice, 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 Google Chrome, keep in mind that Java applets won't work. Please take a look at Browser configuration for more info.*	*No.*

Bookmark

Feature	*Description*	*Massive*
Home	*Goes to the home node.*	*No.*
Set default home	*Changes the default home to the selected node.* *When a user logs in, the application opens the user's home node in desktop view.* *By default, the 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	*Bookmarks the selected node.* *You can select nodes in navigation and browser panels from desktop.* *More information at Working with desktop.*	*No.*

Edit Bookmark	*Shows the bookmark manager.*	*No.*

From the bookmark view, you can see the list of nodes added to bookmarks by the user.

When you click on a bookmarked node, the application opens the node in desktop view.

Templates

*The Templates menu dynamically builds a menu hierarchy based on "**/okm:templates**" nodes. When you click on any templates menu option, the process of creating a new node automatically starts (usually a new document based on a template document).*

Help

Feature	*Description*
Documentation	*Opens a new browser tab with the official OpenKM documentation URL.*
Support request	*Opens a new browser tab with the official community support URL.*
Version notes	*Opens a new browser tab with the application's changelog URL.*
About OpenKM	*Shows a pop-up with information about the application.*

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 folder in the repository.* *The search can be executed only in the context of the selected stack.*	*No.*
Find document	*Find 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	*Find 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	*Create a new folder in the selected node of the navigator panel.*	*No.*
Download document	*Download the selected document in a browser panel.*	*No.*
Download document as PDF	*Download 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	*Lock the selected document in a browser panel.*	*Yes.*
Unlock	*Unlock the selected document in a browser panel.*	*Yes.*
Add document	*Upload a new document to 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. The wizard can require you to set metadata groups, categories, or keywords.*	*No.*
Print	*Print the selected document in a browser panel.*	*No.*

Edit	*Edit a document (check-out).*	*Yes.*
Update	*Upload a new document version (check-in).*	*No.*
Delete	*Delete the selected node.*	*Yes.*
Cancel edit	*Cancel document editing.*	*Yes.*
Add metadata group	*Add a metadata group to the selected node.*	*Yes.*
Delete metadata group	*Delete the metadata group of the selected node.*	*Yes.*
Add subscription	*Subscribe to the selected node.* *You can subscribe to any node type (document, folder, mail, or record). If you subscribe to a particular node of type folder or record, you'll be notified by email when any change in that node or its children's hierarchy occurs. Keep in mind that, in order to receive these email notifications, you have to configure which events you want to be notified about in the user profile. See Profile Notification tab for more info.*	*No.*
Remove subscription	*Unsubscribe from the selected node.*	*No.*
Start workflow	*Start 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	*Refresh the user interface.*	*No.*
Home	*Go to the home node.*	*No.*
Previous node	*Go to the last node visited.*	*No.*
History	*Manage the history of nodes visited.*	*No.*
Next node	*Go to the next node visited.*	*No.*
Stamp	*Apply a stamp to a document.* *This option is only available if you have enabled the **Stamp** feature.*	*No.*
Menu options	*Display an extended menu with custom options.* *This option is only available if you have enabled the **Menu options** feature.*	*No.*

Working with the desktop

Almost all 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 the 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 the root that enables easy access to documents and emails.

The navigation panel is divided into several stacks:

- *Taxonomy.*
- *Categories.*
- *Metadata.*
- *Templates.*
- *My documents.*
- *E-mail.*
- *Trash.*

Each stack contains a tree (folder and records hierarchy) to help organize the nodes in the repository and help the user to locate the data quickly and efficiently.

*The **taxonomy** stack is the space for collaborative work; that means it is the space where several users modify and share company data.*

*The **categories** stack is a virtual taxonomy space that helps with the navigation of taxonomy nodes across other concepts.*

> *Categories are a hierarchical tree structure of category and subcategory folders. The idea is these folders and subfolders represent conceptual terms in a tree structure.*

*The **metadata** stack is a virtual taxonomy space that helps with the navigation of taxonomy nodes across metadata field values.*

> *Metadata view is a hierarchical tree structure of metadata groups. Each possible metadata value is represented as a tree node structure.*
>
> *Only fields of type Select, SuggestBox and Checkbox are displayed (free-text fields like inputs or text areas are not shown).*

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

You can create your own Stack Metadata Adapter visualization. For more information, see: Creating your own Stack Metadata Adapter (deprecated).

*The **templates** stack is the space to set templates for the company. That means it is where company templates are set to be used by several users to create new company documents.*

*The **My documents** stack is a private space for each user; that means that only the user can access this section, and it is the default location where personal data is stored.*

*The **E-mail** stack is a private space for each user; that means only the user can access their data in this section and it is the default place where the user's emails are stored.*

*The **Trash** stack is a private space for each user; that means that only the user can access this section. When a user deletes a node it is automatically moved to their trash. When a user empties their trash the document is permanently removed from the application.*

Administrators have no restrictions on accessing all repository areas, including private user sections like "My Documents", "E-mail", or "Trash".

The selected node from the navigator panel determines the content shown in the browser panel. All the nodes shown in the browser panel have the same parent (selected node on the tree).

In the case of virtual stacks (categories, metadata, and thesaurus) the nodes shown in the browser panel do not have the same parent (selected node on the tree); rather, they are shown because they have a relation (category, metadata, or thesaurus value) with the selected node on the tree.

*When you right-click a node in the tree with the mouse button, a **contextual menu** is shown. The available menu options in the contextual menu depend on the selected stack.*

A complete description of all menu options is available at: Working with the menu bar.

*Available menu options in the **Taxonomy** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Add document.*
- *Add record.*
- *Add bookmark.*

- *Set default home.*
- *Export to ZIP.*

*Available menu options in the **Categories** stack are:*

- *Create folder.*
- *Delete.*
- *Move.*

*Available menu options in the **Templates** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Export to ZIP.*

*Available menu options in the **My documents** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Export to ZIP.*

*Available menu options in the **E-mail** stack are:*

- *Create folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy*

*Available menu options in the **Trash** stack are:*

- *Restore.*
- *Purge.*
- *Purge trash.*

The Metadata and Thesaurus stack views have no menu options.

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

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**.*
- *You are subscribed to a folder **with subfolders**.*
- *You are subscribed to a **read only** folder **without subfolders**.*
- *You are subscribed to a **read only** folder **with subfolders**.*

Browser

The browser panel at the top center shows a table of nodes (documents, folders, emails and records) that are contained in the selected node of the navigation panel.

*At the top of the browser panel there's a **path** that indicates the current location in the navigation panel (for example "**/okm:root/**"). Below, there is a list of the nodes related to the selected node in the navigator panel.*

Usually the nodes that have the selected node as parent are listed, except in the Categories, Metadata, and Thesaurus views where the nodes listed are related to the selected node and do not have the same parent.

By default the browser table shows the columns:

- *Status (icons display node status; for example, if the node is locked the **Lock** **icon** is shown).*
- *Checkbox (to apply actions to several nodes at the same time).*
- *Icon (indicates node type).*
- *Name (the name of the node).*
- *Title.*
- *Language (only for documents).*
- *Size.*
- *Update date.*
- *Author.*
- *Version (only for documents).*

*Also, extra columns based on metadata field values can be shown in the browser table. For more information about how to enable it, take a look at: **Administration** > Profiles.*

*When you right-click a node with the mouse button, a **contextual menu** is shown. You will have different menu options depending on the selected node type and the visible stack panel combination.*

A complete description of all menu options is available at: Working with the menu bar.

*Available menu options when the **Taxonomy** or **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 the **Categories**, **Metadata**, and **Thesaurus** stacks are visible are:*

- *Download.*
- *Edit.*
- *Update.*
- *Cancel edit.*
- *Lock.*
- *Unlock.*

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

- *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 the **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.*
- *Send link.*
- *Send document attachment.*

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

- *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 the ***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 the ***Trash*** *stack is visible are:*

- *Restore.*
- *Purge.*

The ***checkbox*** *helps you* ***select several nodes*** *at the same time and* ***apply an action to all of them****.*

> ***You can perform actions on several nodes at the same time*** *(you have a checkbox in each row to select a node). The bulk 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 a node an **icon** will appear to the side of the node indicating that **you are subscribed**. Keep in mind that in order to receive these email notifications you have to configure which events you want to be notified about in the user profile. See Profile Notification tab for more info.*

When no checkbox is selected, the action will be applied to the selected node (the blue row marks the selected node in the browser view).

The Massive icon displays a contextual menu for performing quick actions:

Feature	*Description*
Select all	*Selects all the nodes.*
Select all folders	*Selects all the nodes of type folder.*
Select all documents	*Selects all the nodes of type document.*
Select all emails	*Selects all the nodes of type mail.*
Select all records	*Selects all the nodes of type record.*
Remove selection	*Deselect all nodes.*

Pagination

*When the **pagination feature** is enabled, an extra toolbar is shown:*



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 the destination node in the Navigator panel.*

Properties

The properties panel at the bottom center shows a collection of tabs that contain grouped properties or features, like preview, associated with the selected node. The selected node can be a navigator node or a browser node.

Basically, the selected node in the navigator panel determines the content shown in the browser panel, and

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

depending on whether the last node selected by the user is in the navigator or browser panel the properties panel will show the properties of the active node in the navigator or browser.

The features described below are the default features. In your properties tab other features not described here may be shown 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***. *This tab shows a* ***list of the users subscribed to the node***. *For quick node sharing, there are two special icons named* ***URL*** *and* ***WebDAV****; when clicking on them, they will copy a browser or WebDAV URL to your clipboard.*
Notes	*Displays a list of notes related to 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 of the document's versions.* *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 document versions except the active one. This action freezes repository size but must be taken into consideration that it cannot be reversed.*

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

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 the "Recursive permission changes" checkbox is checked, the changes will be applied recursively.* *You may also be interested in reading:* • *Security configuration.* • *Security configuration parameters.* • *"**Advanced filter**" option on Profile General tab.*
Preview	*Displays a preview of the document.*
Relations	*Displays the relations of the document to other nodes (documents, folders, emails or records). From this view you can manage the relations of the document, adding or removing relations.* *To set a new relation:* • *First you 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 are three types of 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 the View checkbox is not checked is*

	that these events usually create a large list of actions and are often 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** to the topic.* • ***Quote a reply** in the topic.* • ***Edit a reply** in the topic.* • ***Delete a topic**.* *To delete or edit a post in the topic you must be the owner or have administrator privileges.* *This option is only available if you have enabled the **Forum** feature.*
Workflow	*Displays the workflows associated with the document.* *From this view you can:* • *Show the **details** of the workflow variables.* • ***Show a graph** where the current node is marked.* *This option is only available if you have enabled the **Workflow** feature.*
Text extracted	*Displays the text extracted from the document.* *You will only see the text when the document has been processed by the text extractor worker. For more information: **Administration** > Statistics.* *This option is only available if you have enabled the **Text extracted** feature.*

Tabs on nodes of type folder

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

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 of the users subscribed to the node***. *For quick node sharing, there are two special icons named* ***URL*** *and* ***WebDAV****; when clicking on them, they will copy a browser or WebDAV URL to your clipboard.* *More information: Folder style.*
Notes	*Displays a list of notes related to the folder. From this view you can add, edit, or remove your own notes.* *Administrators can manage other users' 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 recursively.* *You may be interested in 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 are three types of 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 the View checkbox is not checked is that these events usually create a large list of actions and are often 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 folder.

From this view you can:

- ***Add a new topic**.*
- ***Post a reply** to the topic.*
- ***Quote a reply** to the topic.*
- ***Edit a reply** of the topic.*
- ***Delete a topic**.*

To delete or edit a post in the topic you must be the owner or have administrator privileges.

*This option is only available if you have enabled the **Forum** feature.*

Workflow	*Displays the workflows associated with the folder.* *From this view you can:* • *Show the **detail** of the workflow variables.* • ***Show a graph** where the current node is marked.* *This option is only available if you have enabled the **Workflow** feature.*

Tabs on nodes of type mail

Tab	*Description*
Properties	*Displays 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, they will copy a browser or WebDAV URL to your clipboard.*
Notes	*Displays a list of notes related to 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 the "Recursive permission changes" checkbox is checked, the changes will be applied recursively.* *You might be interested in reading:* • *Security configuration.*

- *Security configuration parameters.*
- *"**Advanced filter**" option on Profile General tab.*

Preview

Displays a preview 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, emails or records). From this view you can manage the relations of the mail, adding or removing relations.

To set a new relation:

- *First you 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 are three types of 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 the View checkbox is not checked is that these events usually create a large list of actions and are often 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	*Displays the workflows associated with the mail.* *From this view you can:* • *Show the* ***detail*** *of the workflow variables.* • ***Show a graph*** *where the current node is marked.* *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 record and categories***. *This tab shows a* ***list of the users subscribed to the node***. *For a quick node sharing, there are two special icons named* ***URL*** *and* ***WebDAV****; when clicking on them, they will copy a browser or WebDAV URL to your clipboard.*

Notes	*Displays a list of notes related to 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 the "Recursive permission changes" checkbox is checked, the changes will be applied recursively.* *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, emails or records). From this view you can manage the relations of the record, adding or removing relations.* *To set a new relation:* • *First you 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 are three types of 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 the View checkbox is not checked is that these events usually create a large list of actions and are often 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	*Displays the workflows associated with the record.* *From this view you can:* • *Show the* ***detail*** *of the workflow variables.* • ***Show a graph*** *where the current node is marked.*  *This option is only available if you have enabled the* ***Workflow*** *feature.*

Shortcuts



Shortcuts can be used from the desktop view.

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	*Download.*
CTRL+G	*Show UI testing popup.*
CTRL+Z	*Show the UI log.*
CTRL+N	*Create a folder.*
F5	*Refresh the desktop view.*
INSERT	*Add a new document.*
ALT+B	*Bookmark a node.*
F1	*Show help.*
CTRL+ALT+Q	*Exit the application.*
ALT+DELETE	*Delete a node.*

Shortcuts are configurable from Profiles.

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

Working with search

The search tab enables querying the repository. The search view is divided into four sections:

- *Searches panel.*
- *Search panel.*
- *Control panel.*
- *Results.*

Search panels

The Search panel on the left-hand side is a 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 has two stacks:

- *Stored searches.*
- *User news.*

***Stored searches** shows a list of the user's searches that have been saved for later use. By double-clicking a stored search, you can execute it again. Right-clicking a stored search will show a contextual menu with these options:*

Feature	*Description*
Run	*Run stored search.*
Delete	*Delete an stored search.*

The stored searches and user news are only visible to the user who created them.

***User news** shows a list of user searches that are used as a push service (usually executed every 30 minutes). These searches are periodically executed and the results are shown in the **Dashboards** > **News** panel.*

User news helps users stay periodically updated about repository news based on their own queries.

Only new results that have appeared since the last time the query was executed will be shown.

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 for querying the application's content:

- ***Default search** is a full-text search. Default search is also available from the top-right corner of 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 and advanced** search by **checking or unchecking** 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 stem, base, or root form, generally a written word form. The stem does not need 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 natural language processing tools, and indeed not all tools even use such a list. The stopwords are removed from search. For example, "the", "is", "at", "which", and "on" are stopwords.*

Operators and wildcards

*Internally, the **search engine uses the "AND" operator** by default.*

You can use wildcards:

- *Add an asterisk "*****" as a placeholder for any unknown or wildcard terms, for example, "***2015**", "**2015***", "**20*15**" or "***2015***".*
- *Use a question mark "**?**" as a placeholder for any unknown character; for example, the query "**?015**" will return occurrences where the first character doesn't matter.*
- *The query "**2015**" is a synonym of "**2015***". In this search view, the application always adds "*****" for you at the end.*

Default search

Default search is a full-text search which searches only by document content.

Default search shows result nodes only in the "/okm:root" context; results from other contexts are automatically filtered.

You always have quick access to the Quick search box feature at the top-right corner.

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 three tabs:

- *Basic (most common fields).*
- *Advanced.*
- *Metadata.*

The available criteria from the basic view:

Field	*Description*	*Wildcard allowed*
Context	*Sets the context in which the query is executed.* *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 nodes of typed documents have a language value.*	*Yes*

		
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 000 milliseconds. That means if you want to query a node modified on 2015-08-15 you 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 the range ("**from**" or "**to**"); you can set only one date.*	*No*

The available criteria from the advanced view:

Field	*Description*	*Wildcard allowed*
Folder	*Restrict results to folders.*  *When the Recursive checkbox is checked, it also includes subfolders.*	*No*
Category	*Restricts results to categories. The nodes must be associated to a specific category.*	*No*
Type	*Restrict the types of nodes:* • *Document* • *Folder* • *Mail* • *Record*  *When the Mail checkbox is checked, extra fields (from, to, subject) are shown.*	*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 mail subject field.* *This field is only available if you have **checked the Mail checkbox**.*	*Yes*

From the 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.*

In most input and text area fields you can use wildcards.

Control panel

The control panel at the top-right corner has several features:

Feature	*Description*
Switch to Advanced mode checkbox	*When checked, the advanced mode search view is shown.*
Results compact view checkbox	*When checked, the results are shown in a compact table with minimal details.*

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

View metadata groups checkbox	*When checked, metadata groups associated with each node are shown in the results view.* *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, it will be set as a 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 on whether the "Save as user news" checkbox is checked, the search will be saved as a "stored search" or as "user news".*
Page results lists	*Set the limit on the number of nodes shown in the results list.*
 ***Clean button***	*Resets the search.*
 ***Search***	*Executes the search.*
Results	*Indicates the total number of results.*
 ***Go to previous***	*Goes to the previous list.*
 ***Go to next***	*Goes to the next list.*
 ***Export to CSV***	*Downloads a CSV file with the results.*
 ***Export to ZIP***	*Downloads a ZIP file with the results.*

Results

The Results panel at the bottom center shows a table of nodes (documents, folders, mails, and records) which are the result of querying the repository. The Results panel is shown in:

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

- *Compact view (a table with a single row for each node).*
- *Full view (a detailed description for each node).*

Stars indicate 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 right-click in the compact view, a contextual menu with the options is shown:

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 the folder location.* *Only shown in folder nodes.*
Go to e-mail	*Goes to the 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 the full view you can see detailed information for each node and some features are also available:

Feature	*Description*
Download	*Download the node.* *Only shown in document or mail nodes.*
Notes	*Shows notes. From this view you can manage the notes of the node.*
Preview	*Preview the node.* *Only shown in document or mail nodes.*
Options	*Shows additional mail fields (cc, bcc, reply, extra fields).* *Only shown in document or mail nodes.*

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

Working with Lucene queries

*Keep in 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 enable it at **Components** > **Search** > **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 in mind this can be difficult unless 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 search, you can specify a field or use the default field. In OpenKM, the default field contains the extracted document text. You can search a field by writing the field name, followed by a colon ":" and a term. For example, this query will search for all nodes in taxonomy whose name is "animals":*

  ```
  context:okm_root AND name:animals
  ```

You can search for a phrase using quotes:

```
name:"big animals"
```

Term modifiers

Lucene supports 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 wildcard search, use the "?" symbol. For example, to search for "text" or "test":*

  ```
  te?t
  ```

- ***Multiple**: To perform a multiple-character wildcard search, use the "*" symbol. For example, to search for "test", "tests" or "tester":*

  ```
  test*
  ```

You can't use a "" or "?" at the beginning of a query.*

Proximity searches

Lucene also supports finding words that are within a specific distance. To perform a proximity search use the tilde "~" symbol at 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 whose field "name" is between "cat" and "horse" but will not 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 in mind this can be difficult unless 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 search, you can specify a field or use the default field. In OpenKM, the default field contains the extracted document text. You can search a field by writing the field name, followed by a colon ":" and a term. For example, this query will search for all nodes in taxonomy whose name is "animals":*

  ```
  context:okm_root AND name:animals
  ```

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

You can search for a phrase using quotes:

```
name:"big animals"
```

Term modifiers

Lucene supports 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 wildcard search, use the "?" symbol. For example, to search for "text" or "test":*

```
te?t
```

- ***Multiple**: To perform a multiple-character wildcard search, use the "*" symbol. For example, to search for "test", "tests" or "tester":*

```
test*
```

You can't use a "" or "?" at the beginning of a query.*

Proximity searches

Lucene also supports finding words that are within a specific distance. To perform a proximity search use the tilde "~" symbol at 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 whose field "name" is between "cat" and "horse" but will not 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 conjunction operator: this means that if there is no boolean operator between two terms, the "OR" operator is used. These two queries are equivalent:

```
cat animal
```

And this one:

```
cat OR animal
```

AND

The AND operator matches documents where both terms exist. The symbol "&&" can also be 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 forces 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 contain the term after the "NOT". The symbol "!" can be used to replace the word "NOT".

For example, to search for documents that contain "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 search documents which contain "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:

```
(cat 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 conjunction operator: this means that if there is no boolean operator between two terms, the "OR" operator is used. These two queries are equivalent:

```
cat animal
```

And this one:

```
cat OR animal
```

AND

The AND operator matches documents where both terms exist. The symbol "&&" can also be 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 forces 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 contain the term after the "NOT". The symbol "!" can be used to replace the word "NOT".

For example, to search for documents that contain "animal" but not "cat":

```
animal NOT cat
```



The NOT operator cannot be used with just one term.

-

This operator excludes documents which contain the term given after the "-" character:

For example, to search documents which contain "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:

```
(cat 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 uses 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 characters ":" and "." with "_".*

```
okp_consulting_text:value
```

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



When working with metadata fields, you need to enable the ***sanitize.lucene.fields*** *configuration property; 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 help provide a global view of the application's 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 access level of each user.

The information is organized in several boxes:

Box	*Description*
Locked documents	*Displays a list of the documents currently locked by the user.*
Locked records	*Displays a list of the records currently locked by the user.*
Edited documents	*Displays a list of the documents currently checked out by the user.*
Last downloaded documents	*Displays a list of the last 20 documents downloaded by the user.*
Subscribed documents	*Lists document subscriptions by the user.*
Subscribed folders	*Lists folder subscriptions by the user.*
Subscribed emails	*Lists email subscriptions by the user.*
Emails	*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 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" can be found at: Working with search.

Workflow

In the Workflow panel, the workflows being executed are shown. The Workflow panel is organized in two sections:

- *Workflow lists.*
- *Workflow detail.*

Workflow lists

In the Workflow lists panel, the workflows in execution are shown, organized in two boxes:

- *Pending tasks.*
- *Unassigned pending tasks.*



The workflows in the "Pending task" box have yet to be assigned to the logged-in user and are waiting for them.

The workflows in 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 themselves (one of whom is the logged-in user). The workflow is waiting for this automatic assignment.

Workflow detail

In the Workflow detail panel the workflow details are shown. From this view, you can see low-level information such as "process instance", "process definition", or "data", and also ***go to the next workflow task by completing the workflow form*** *.*

To complete a workflow task:

- ***Click on a workflow in the Pending task*** *box. The workflow form will be shown.*
- ***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 to emails created by OpenKM.

*Just place an 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 valid HTML code.

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

Activity log extension

Digital Signature viewer helps you view the activity log of a node.

*This option is only available if you have enabled the **Activity log** extension.*

More information is available 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 and out, rotate, show or hide layers and print.

AutoCAD preview extension allows viewing 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 requires third-party software to be installed. Ask your OpenKM technical staff about it.*

Toolbar



Feature		*Description*
	***Change Icon Page***	*Allows you to change to the previous or next icon page.*
	***Layer List***	*List of the layers in the drawing.*
	***Print***	*Print the document.*
	***Zoom In***	*Performs a zoom out.*
	***Zoom Out***	*Performs a zoom in.*

	***Zoom Extents***	*Allows you to move the zoom-in zone.*
	***Zoom Window***	*Allows you to draw a window to zoom in.*
	***Toggle Black/White***	*Allows you to toggle between black and white 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 for an individual text string in the drawing.*
	***About***	*Information about the CADViewer JS version.*
	***Layer Off***	*Turn the layer off.*
	***Swap Layers***	*Allows you to swap between layers.*
	***All Layers On***	*Turn all layers on.*
	***Measurement***	*Allows you to select two points in the drawing to measure the distance between them. Measurement must be calibrated first.*
	***Calibrate Measurement***	*Allows you to calibrate measurements to define units of measure in the drawing.*
	***Toggle Birds-Eye Window***	*Open the bird's-eye window.*
	***Adjust Line Thickness***	*Adjust the line thickness in the drawing.*
	***Save Redline***	*Save current red lines.*

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

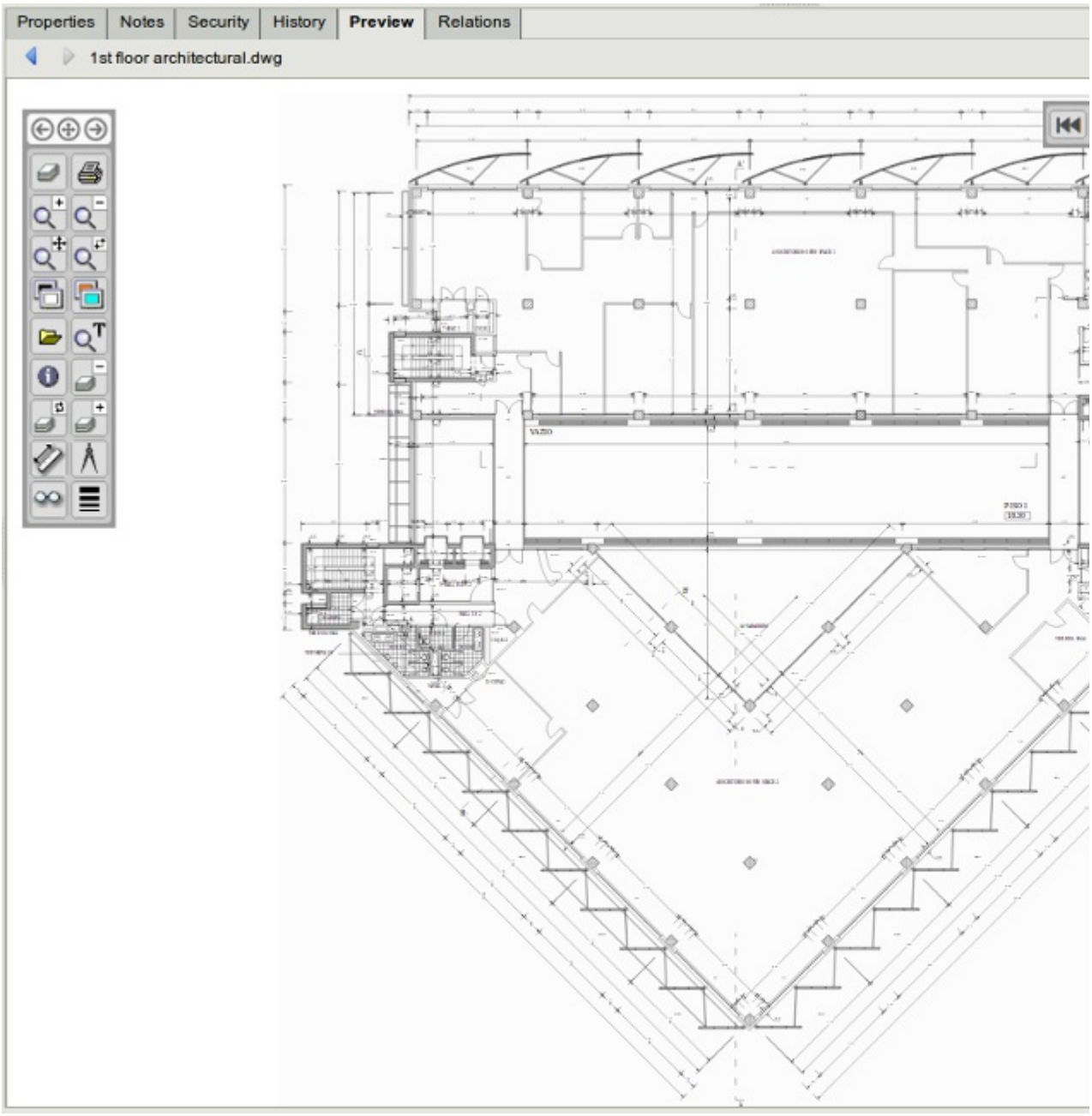
	***Load Redline***	*Allows to load red lines previously saved.*
	***Delete Single Redline***	*Allows to delete one red line by selecting it in the drawing.*
	***Delete Last Redline***	*Delete last red line.*
	***Redline Color***	*Allows to change the color of the red line.*
	***Redline Thickness***	*Allows to change the thickness of the red line.*
	***Make Sticky Note***	*Allows you to insert a sticky note.*
	***Clear Visible Redlines***	*Clear all visible red lines.*
	***Freehand Redline***	*Allows you to insert a freehand red line.*
	***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 you to insert a red line with a filled rectangle shape.*
	***Redline Rectangle***	*Allows you to insert a rectangular red line.*
	***Redline Arrow***	*Allows to insert a red line with an arrow.*
	Redline Text	*Allows you to write text and select where to place it in the drawing.*

		
	***Zoom Extents***	*Allows you to move the zoom-in zone.*
	***Zoom Window***	*Allows you to draw a window to zoom in.*



	Feature	*Description*
	***First Page***	*Go to the first page of the drawing.*
	***Previous Page***	*Go to the previous page of the drawing.*
	***Next Page***	*Go to the next page of the drawing.*
	***Last Page***	*Go to the last page of the drawing.*

Default preview



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

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	*Displays the result of checking the digital signature.*
Trust	*Displays the result of checking whether the signer's certificate is trusted.*
Revocation	*Displays the result of checking the certificate revocation list.*
Validity	*Displays the result of checking certificate validation or the signing time.*
Download the signed version	*You can download the signed document.*
Certificate information	• *You can download the certificate.* • *Displays the subject.* • *Displays the issuer.*
Validity information	*Displays the signing time.*
Signed properties	• *Displays the signing reason.*

	• *Displays the 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 that can be used to advise users about imminent and expired documents.



*This option is only available if you have enabled the **Document expiration** feature.*



More information is available at Enable extensions.

Add document expiration

- ***Choose a document**.*
- ***Add** the metadata group "**Expiration**".*
- ***Fill in the form**.*
- *Click the **Add button**.*

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

Extra tab workspace extension

The ExtraTab allows you to easily add new tabs with a name and a specific URL.

*This option is only available if you have enabled the **ExtraTab** feature.*

Keep in mind that additional tabs appear in the three-dot menu rather than as visible items in the main bar.

The URL request includes several parameters:

- *The authentication **token** (encrypted)*
- *The **lang** (language ID)*
- *The node **uuid***
- *The **userId***

Configuration

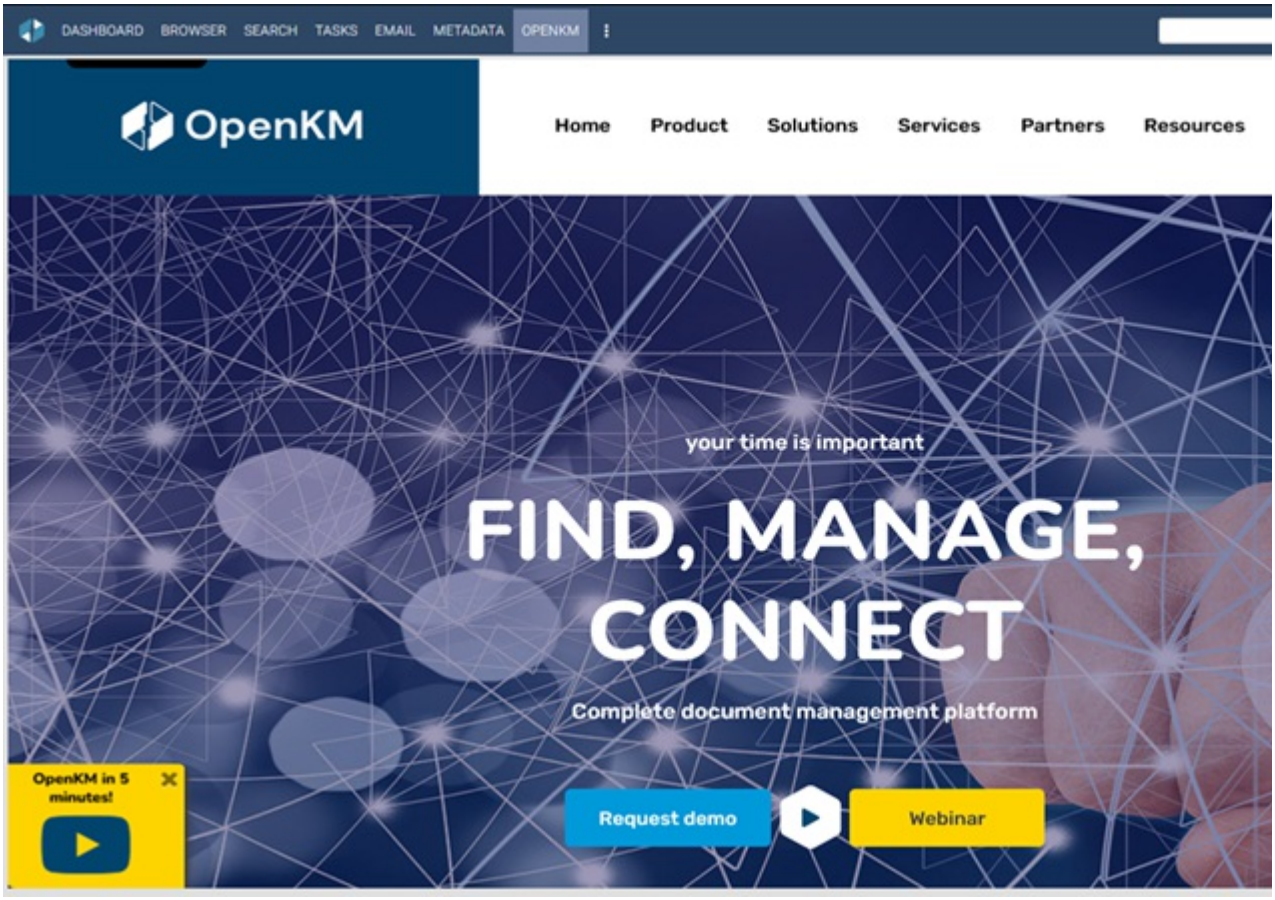
Field / Property	*Type*	*Description*
extension.extra.tab.workspace	***List***	*A title-and-URL pair. You can add a single line (where the character ';' separates the title and URL) or several lines.* `OpenKM;https://www.openkm.com` *If **the URL does not start with http**, it is considered an **OpenKM relative path** and will be merged with your OpenKM server URL as **YOUR_OPENKM_URL/url**.* *When you embed a URL in an iframe that belongs to a subdomain that is different from the application's domain, you must include that domain in the configuration; otherwise, the browser will block it for security reasons. You can find more information about the **okm.content.security.policy** configuration parameter in the 'Experimental configuration parameters' section.*
webservices.visible.properties	***List***	*To make the values of the parameter "**extension.extra.tab.workspace**" accessible to all users, that parameter must be added to the current list.*

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

If you forget to add the property name, only administrators will be able to retrieve the value, so some extensions may produce errors for regular users.

More information is available at Configuration parameters.

Sample



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

External preview extension

The External Preview feature helps you preview documents based on an external URL request.

*This option is only available if you have enabled the **External Preview** extension.*

Two parameters are passed in the URL request:

- *The document **uuid**.*
- *The **downloadUrl**.*

Configuration

Field / Property	Type	Description
extension.external.preview	List	*Each row is a pair of values that declare preview content; 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* *When you embed a URL in an iframe that belongs to a different subdomain from the application's domain, you must include that domain in the configuration; otherwise, the browser will block it for security reasons. You can find more information about the **okm.content.security.policy** configuration parameter in the section Experimental configuration parameters*

More information is available at Configuration parameters.

Sample

Configuration

Create a file named test.jsp and copy it to the $TOMCAT_HOME/webapps/openkm folder.

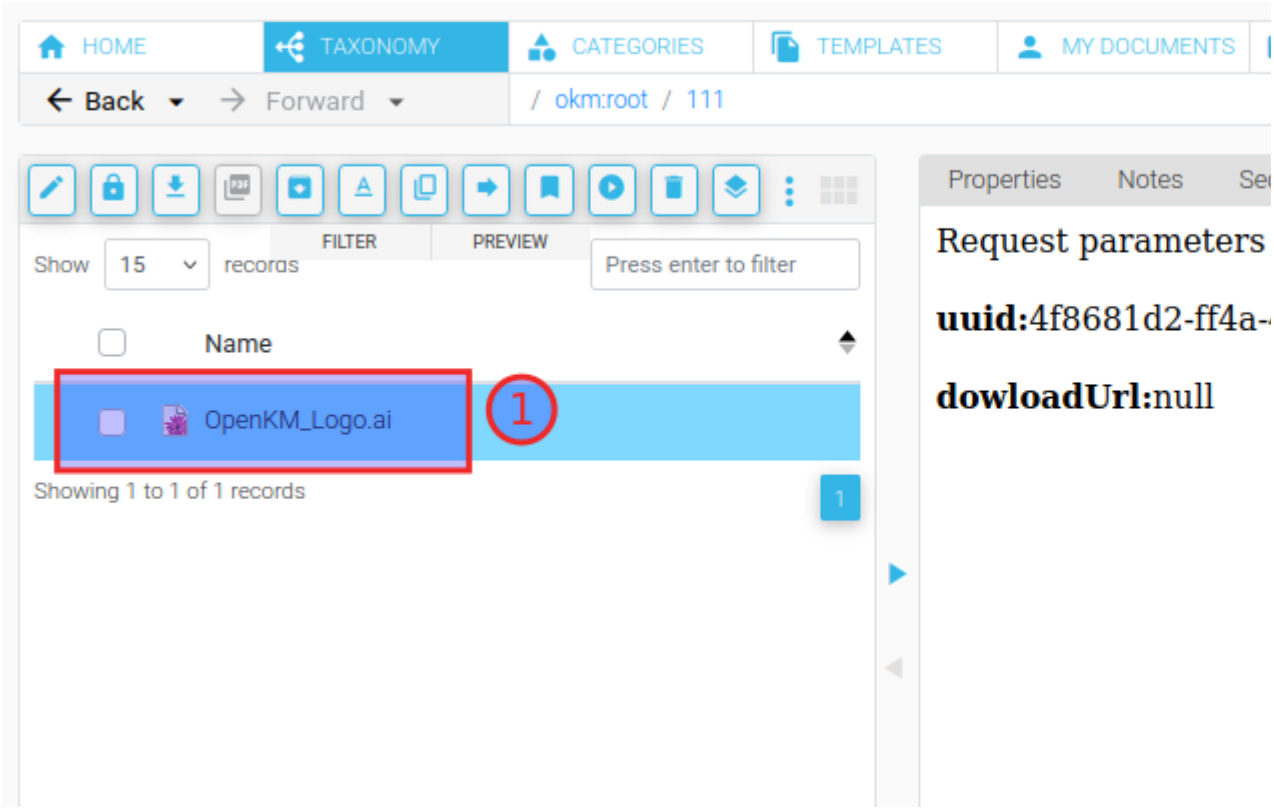
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

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

```
<?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 />
 <br />
 <b>dowloadUrl:</b><%=request.getParameter("downloadUrl")%><br />
</body>
</html>
```

Usage

- *Choose a document with one of the MIME types set in the configuration parameters.*
- *Open the Preview tab. The external URL based on the document's MIME type will be shown.*



External tab document extension

The External Tab Document feature helps you show tab content in a properties tab based on an 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 in the 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 pair of values with a tab content declaration; the format is:* ***name;url*** *name1;http://localhost:8180/openkm/test.jsp* *name2;http://www.openkm.com* *When you embed a URL in an iframe that belongs to a subdomain different from the application's domain, you must include that domain in the configuration; otherwise, the browser will block it for security reasons. You can find more information about the **okm.content.security.policy** configuration parameter in the section Experimental configuration parameters*
webservices.visible.properties	***List***	*To make the values of the parameter "**extension.external.tab.document**" accessible to all users, the parameter "**extension.external.tab.document**" must be added to the current list.* *If you forget to add the property name, only administrators will be able to retrieve the value, so some extensions may produce errors for regular users.*

More information at Configuration parameters.

Sample

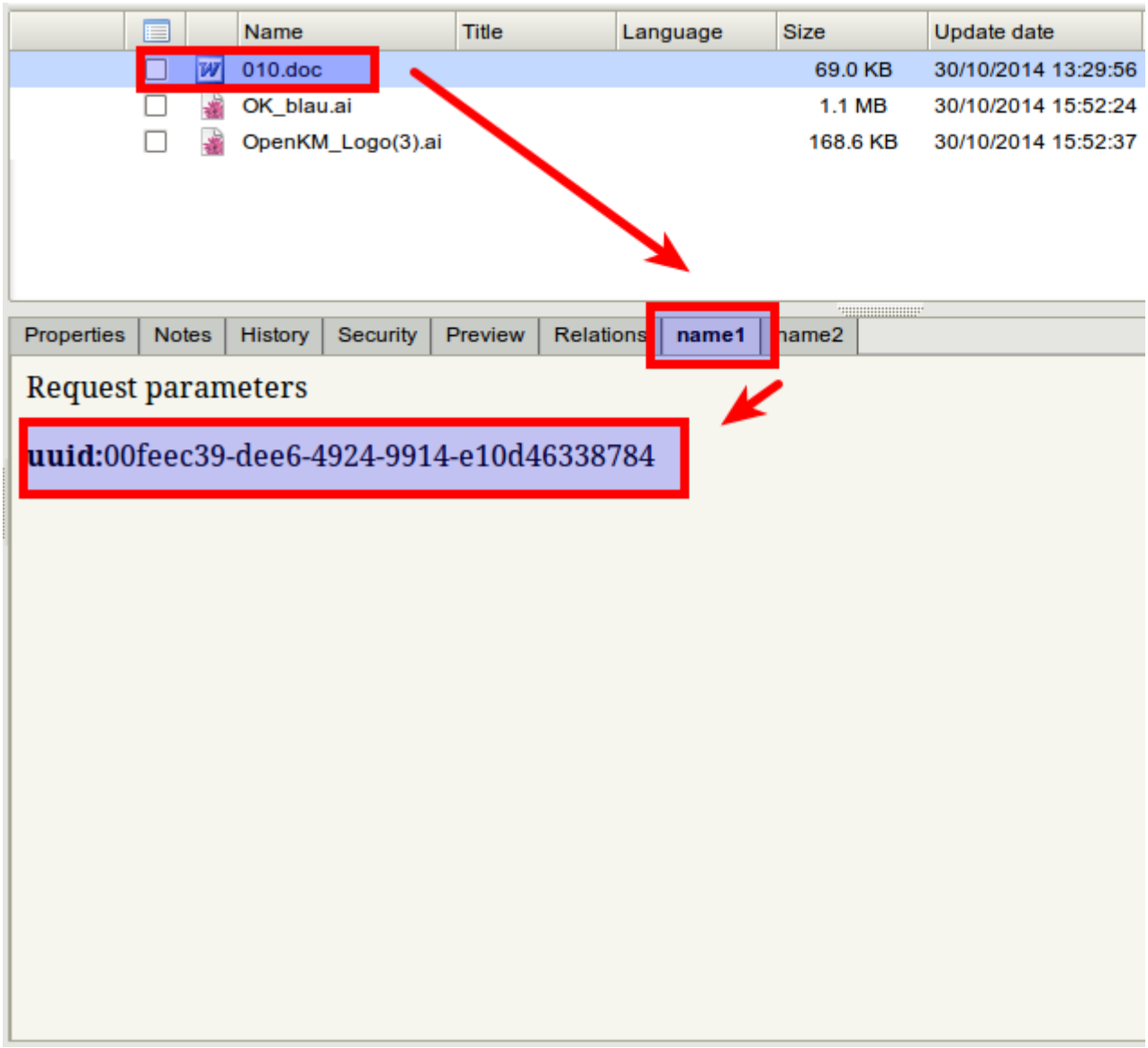
Configuration

Create a file named test.jsp and copy it into the $TOMCAT_HOME/webapps/openkm folder.

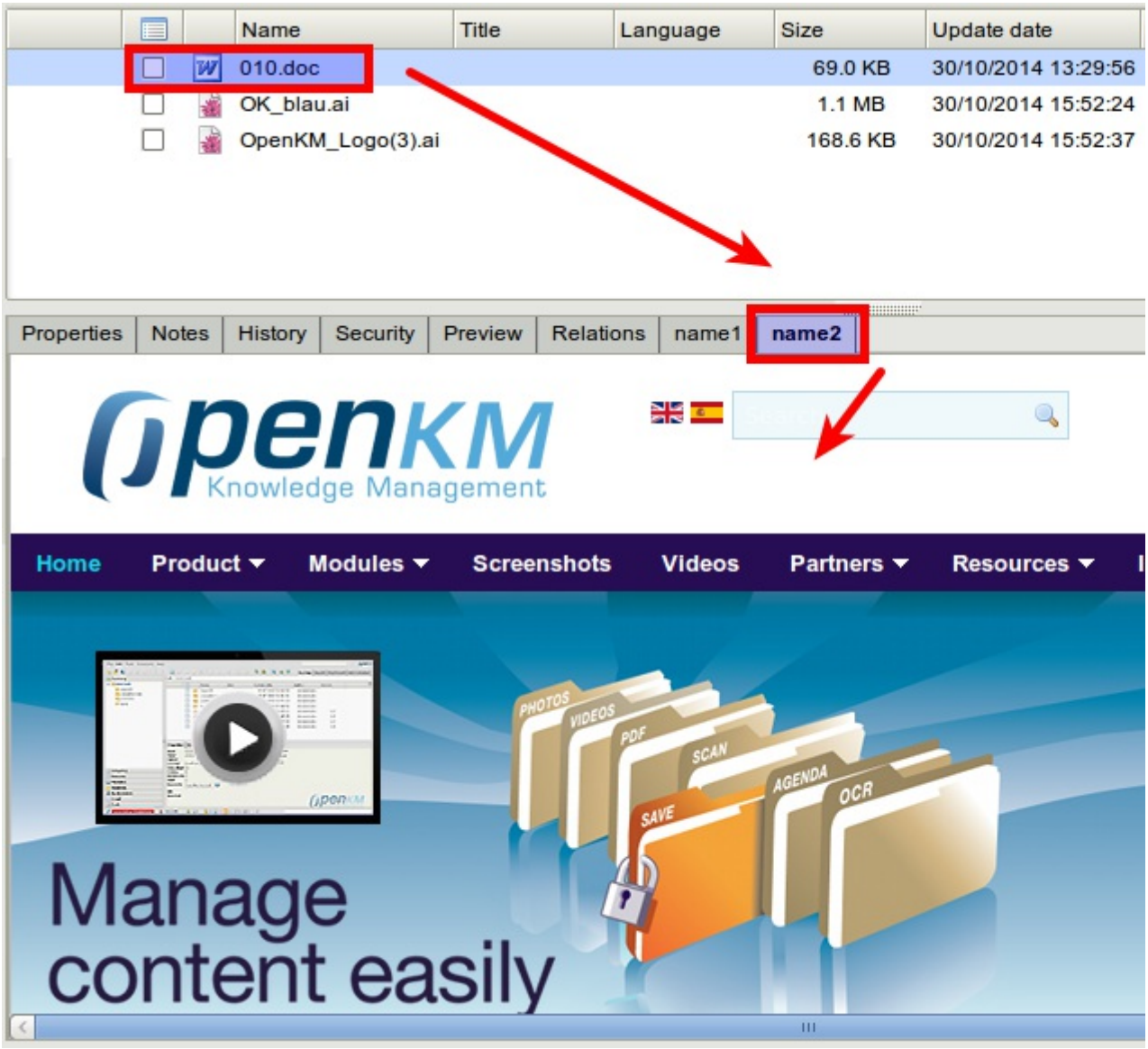
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***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 display tab content in the properties tab based on an external URL request when a folder node is selected.

*This option is only available if you have enabled the **External Tab Folder** feature.*

There are several parameters passed in the URL request:

- *The authentication **token** (encrypted)*
- *The folder **uuid***
- *The **userId***

Configuration

Field / Property	Type	Description
extension.external.tab.folder	**List**	*Each row contains a pair of values that declare tab content; the format is:* ***name;url*** *name1;http://localhost:8180/openkm/test.jsp* *name2;http://www.openkm.com* *When you embed a URL in an iframe that is hosted on a subdomain different from the application's domain, you must include that domain in the configuration; otherwise, the browser will block it for security reasons. You can find more information about the **okm.content.security.policy** configuration parameter in the section Experimental configuration parameters*
webservices.visible.properties	**List**	*To make the values of the parameter "**extension.external.tab.folder**" accessible to all users, the value "**extension.external.tab.folder**" must be added to the current list.* *If you forget to add the property name, only administrators will be able to retrieve the value; therefore, some extensions may produce errors for regular users.*

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



More information is available at Configuration parameters.

Sample

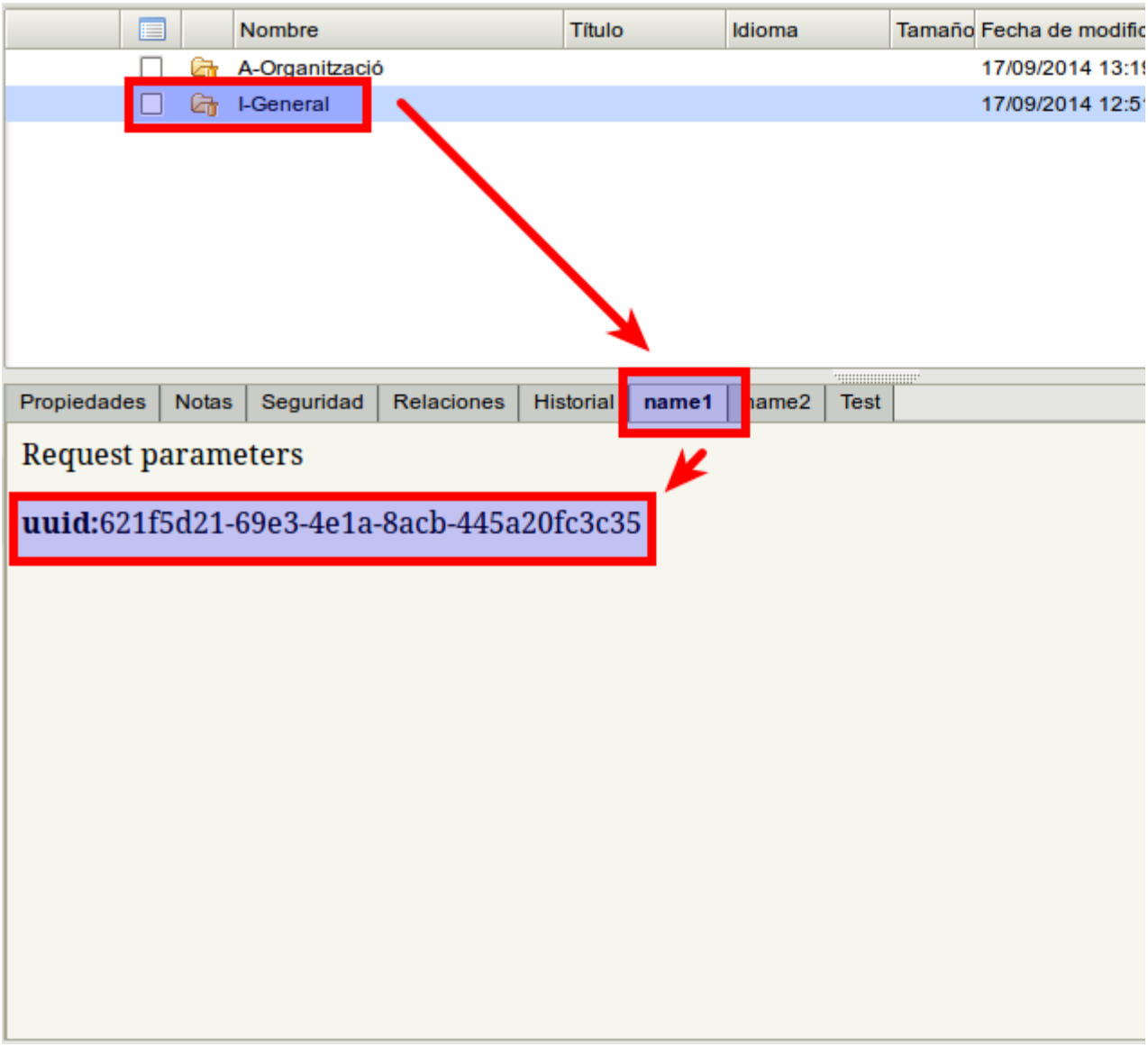
Configuration

Create a file named test.jsp and copy it into the $TOMCAT_HOME/webapps/openkm folder.

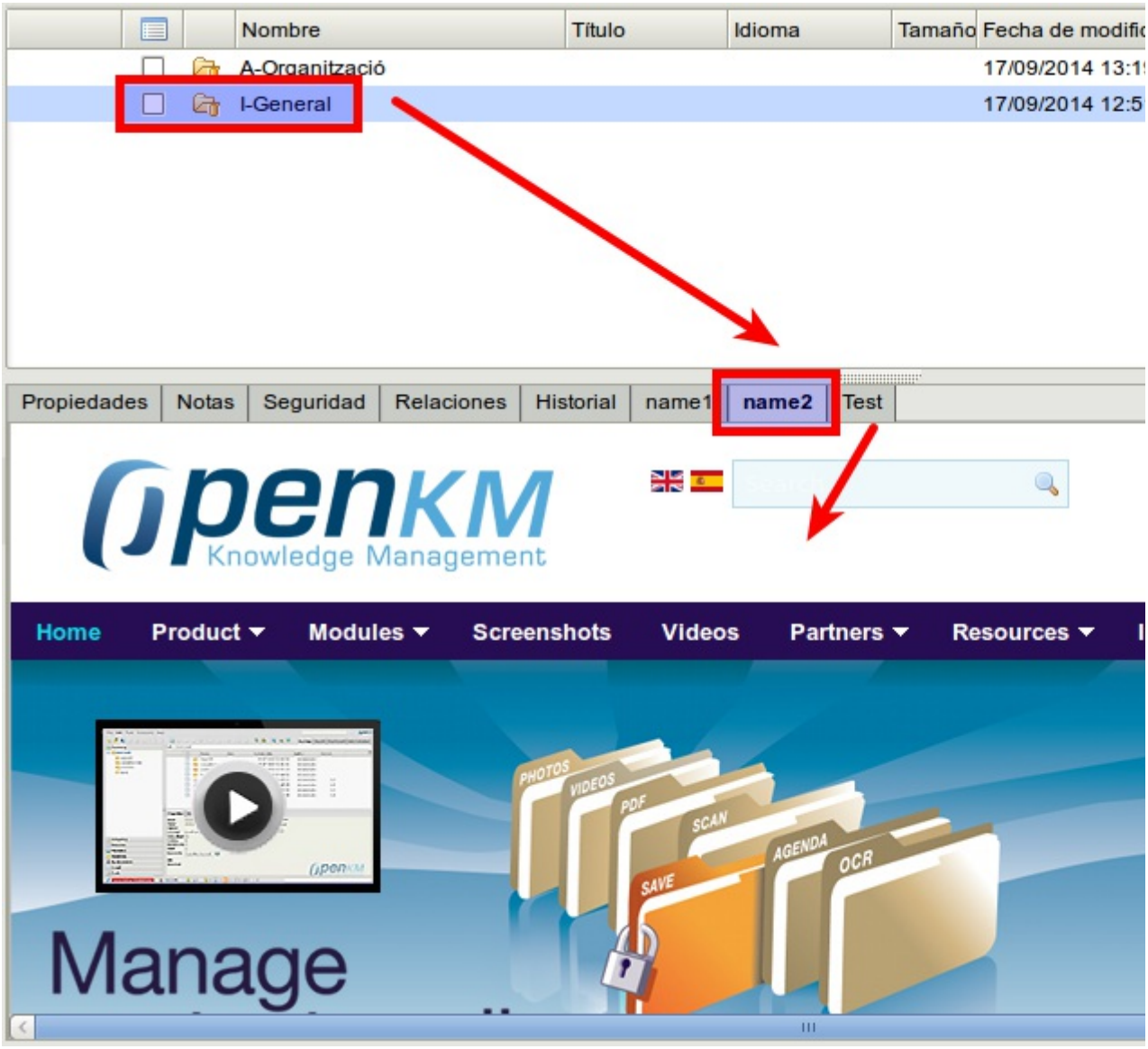
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a folder***.
- ***Click the "name1" tab***.



- ***Click the "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 display tab content in the properties tab based on an external URL request when a mail node is selected.

*This option is only available if you have enabled the **External tab mail** feature.*

Several request parameters are passed in the URL:

- *The authentication **token** (encrypted)*
- *The mail **uuid***
- *The **userId***

Configuration

Field / Property	Type	Description
extension.external.tab.mail	**List**	*Each row is a pair of values that declare a tab's content; the format is:* ***name;url*** *name1;http://localhost:8180/openkm/test.jsp* *name2;http://www.openkm.com* *When you embed a URL in an iframe that is on a subdomain different from the application's domain, you must include that domain in the configuration; otherwise, the browser will block it for security reasons. You can find more information about the **okm.content.security.policy** configuration parameter in the section Experimental configuration parameters*
webservices.visible.properties	**List**	*To make the values of the parameter "**extension.external.tab.mail**" accessible to all users, add "**extension.external.tab.mail**" to the current list.* *If you forget to add the property name, only administrators will be able to retrieve the value; as a result, some extensions may produce errors for regular users.*

More information is available at Configuration parameters.

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

Sample

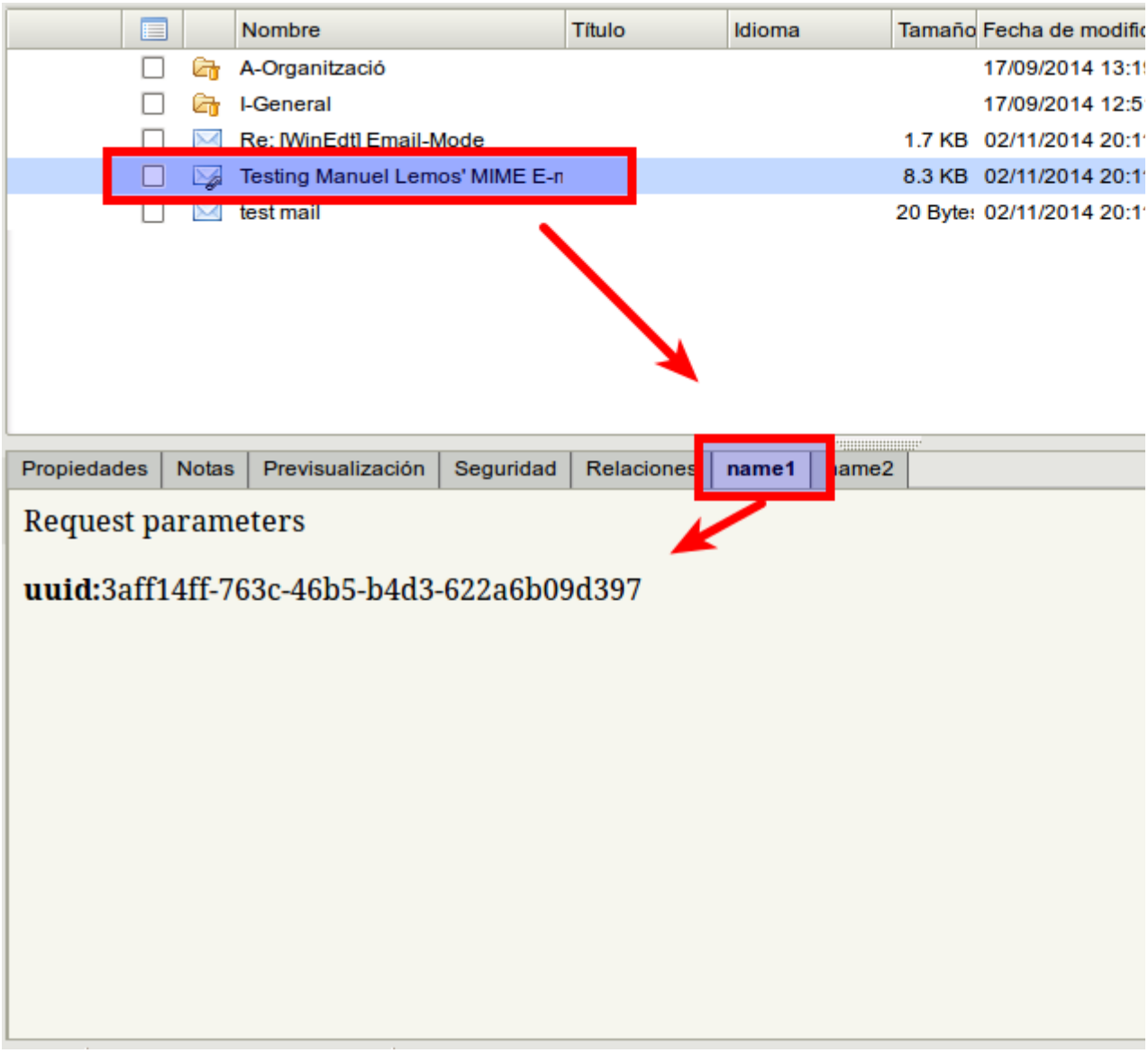
Configuration

Create a file named test.jsp and copy it into the $TOMCAT_HOME/webapps/openkm folder.

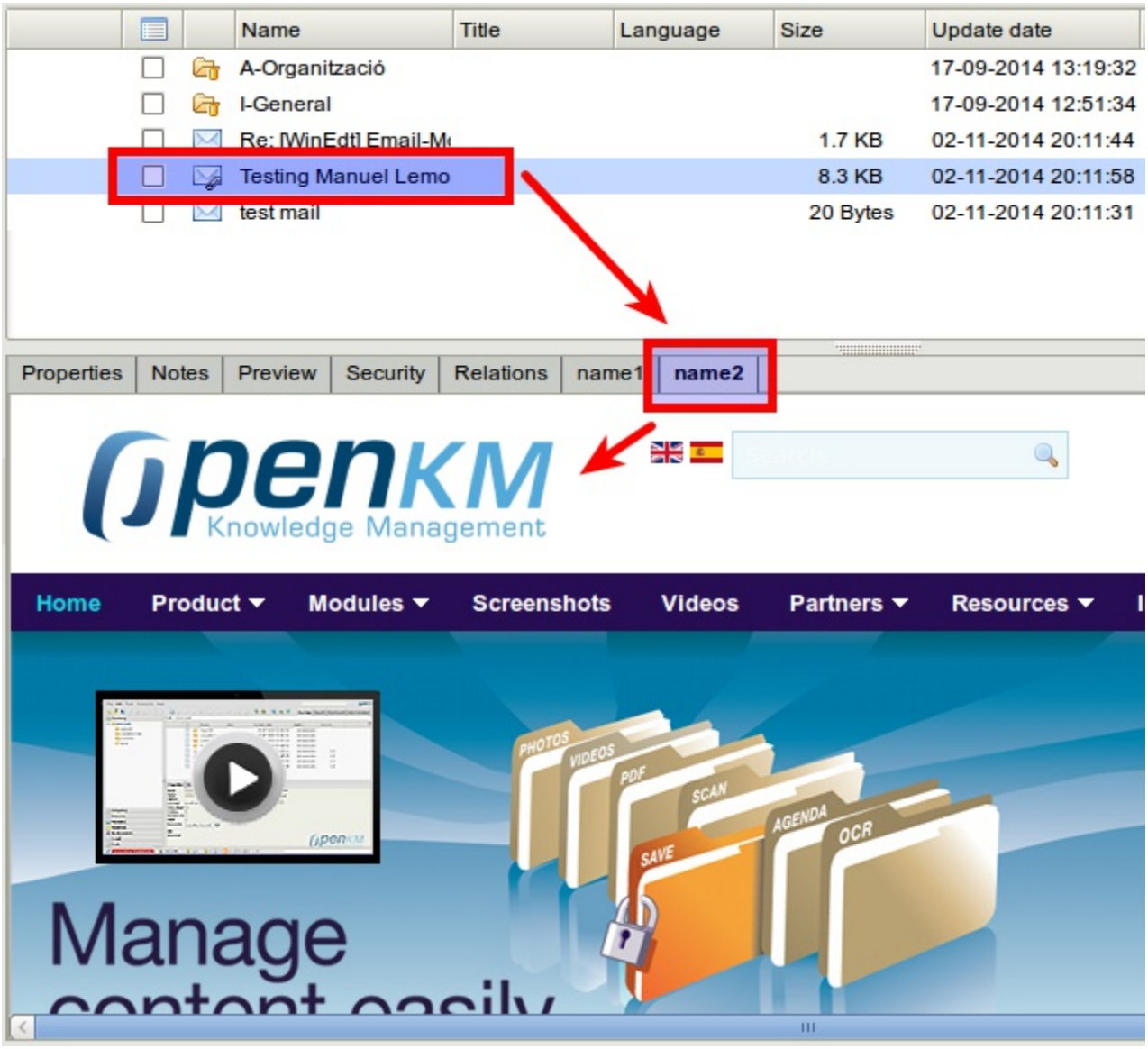
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose an email***.
- ***Click the "name1" tab***.



- ***Click the "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 display tab content in the properties tab based on an 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 in the 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 pair of values declaring a tab's content; the format is: **name;url*** *name1;http://localhost:8180/openkm/test.jsp* *name2;http://www.openkm.com* *When you embed a URL in an iframe that belongs to a subdomain different from the application's domain, you must include that domain in the configuration; otherwise, the browser will block it for security reasons. You can find more information about the **okm.content.security.policy** configuration parameter in the section Experimental configuration parameters*
webservices.visible.properties	***List***	*To make the values of the parameter "**extension.external.tab.record**" accessible to all users, the value "**extension.external.tab.record**" must be added to the current list.* *If you forget to add the property name, only administrators will be able to retrieve the value, so some extensions may produce errors for regular users.*

More information is available at Configuration parameters.

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

Sample

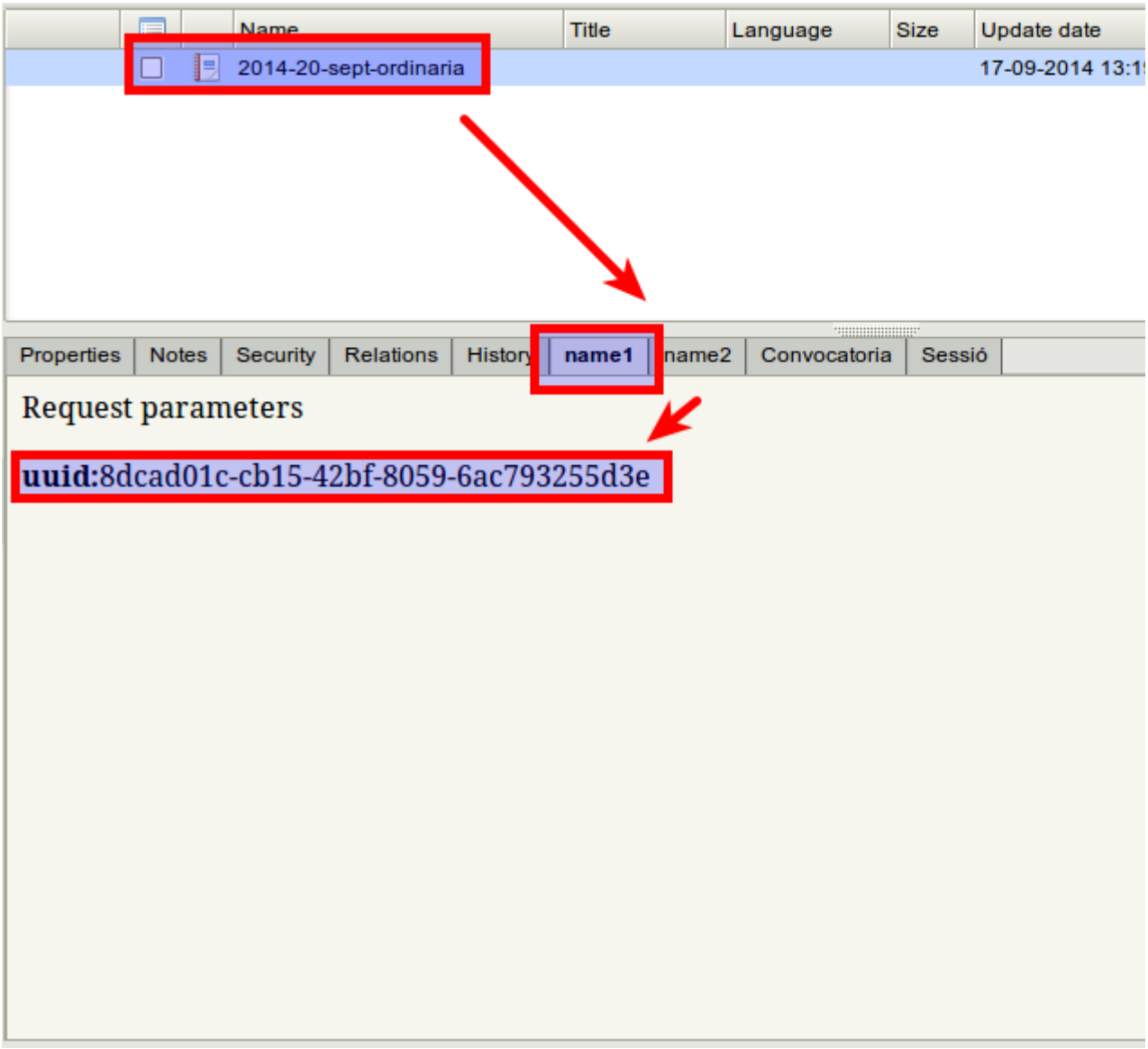
Configuration

Create a file named test.jsp and copy it into the $TOMCAT_HOME/webapps/openkm folder.

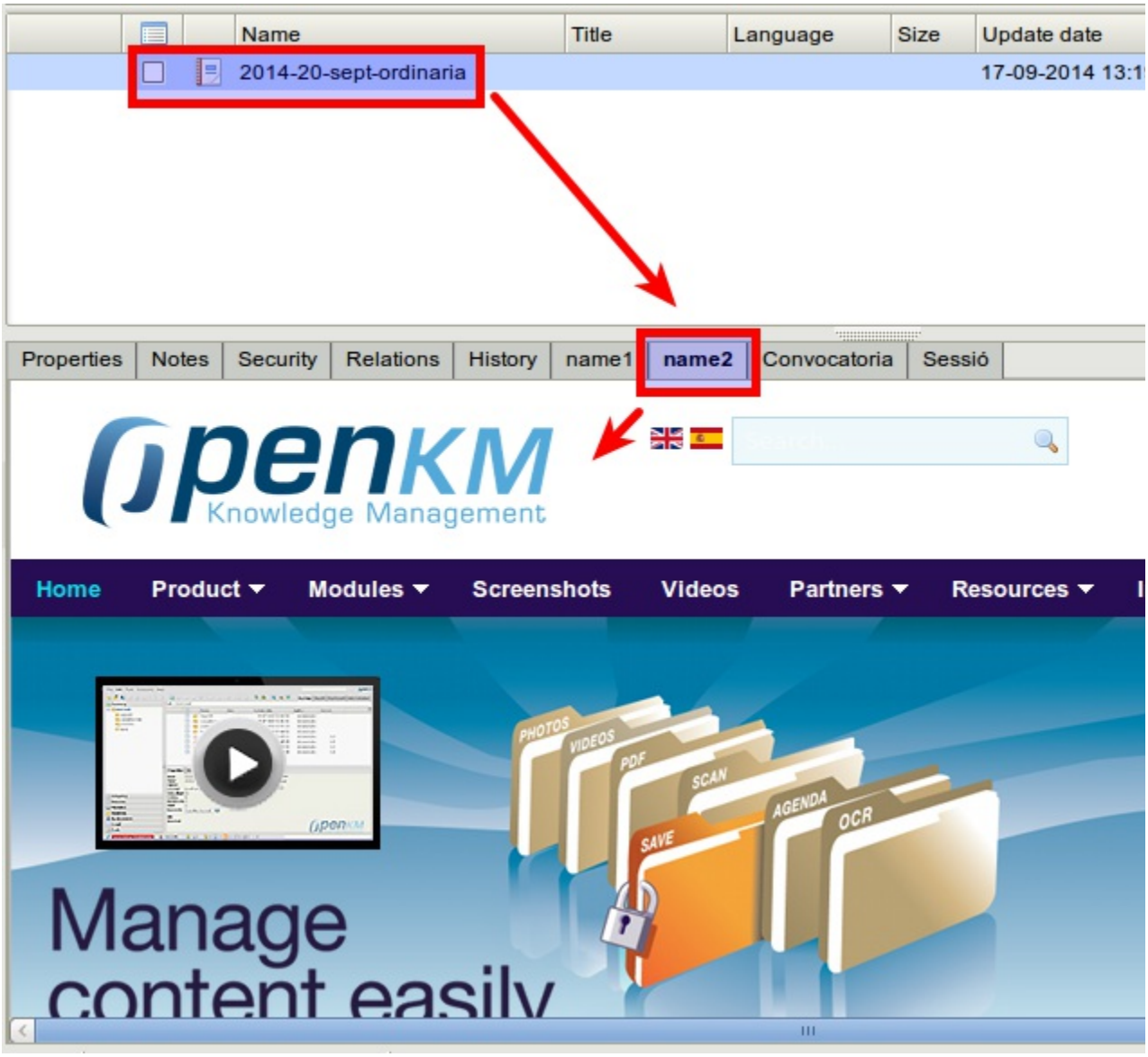
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a record***.
- ***Click on*** *the "****name1****" tab*.



- ***Click on** the "**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
OpenKM
Knowledge Management
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 display card content in the dashboard panel based on an external URL request.

This feature is available in OpenKM version 8.1.10.

*This option is only available if you have enabled the **External dashboard widget** feature.*

There are several request parameters in the URL:

- *The authentication **token** (encrypted)*
- *The record **uuid***
- *The **userId***

Configuration

Field / Property	*Type*	*Description*
extension.external.widget.dashboard	***List***	*Each row is a set 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.* *When you embed a URL in an iframe that is on a subdomain different from the application's domain, you must include that domain in the configuration; otherwise, the browser will block it for security reasons. You can find more information about the* ***okm.content.security.policy*** *configuration parameter in the section Experimental configuration parameters*
webservices.visible.properties	***List***	*To make the values of the parameter "**extension.external.widget.dashboard**" accessible to all users, the parameter name "**extension.external.widget.dashboard**" must be added*

		to the current list. *If you forget to add the property name, only administrators will be able to retrieve the value, so some extensions may produce errors for regular users.*

More information is available at Configuration parameters.

Sample

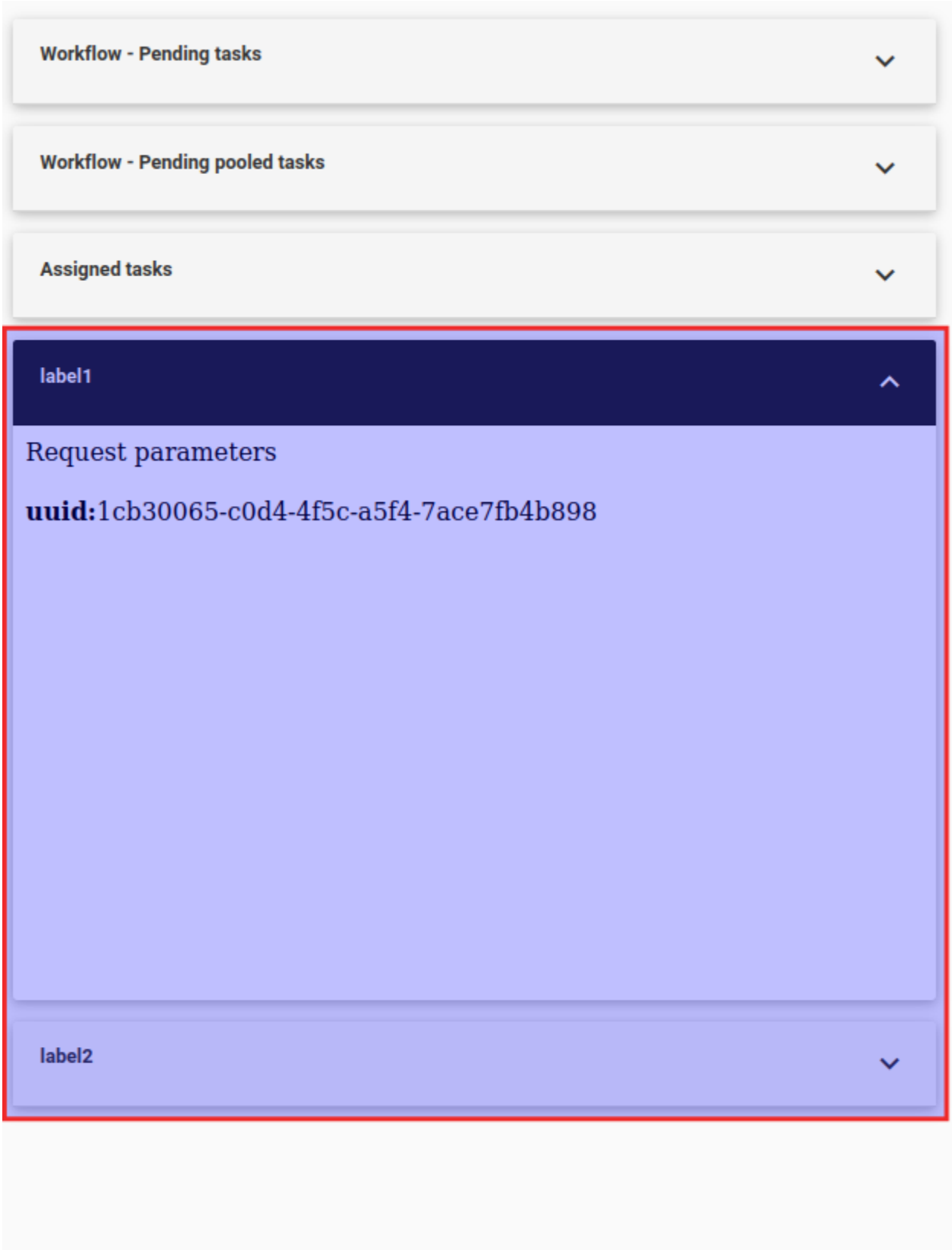
Configuration

Create a file named test.jsp and copy it into the $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 editing.*
Link document	*Add a link to a document.*
Link folder	*Add a link to a folder.*
Add image	*Add an image.*
Download link	*Add a download link.*
Code	*Add a code section.*

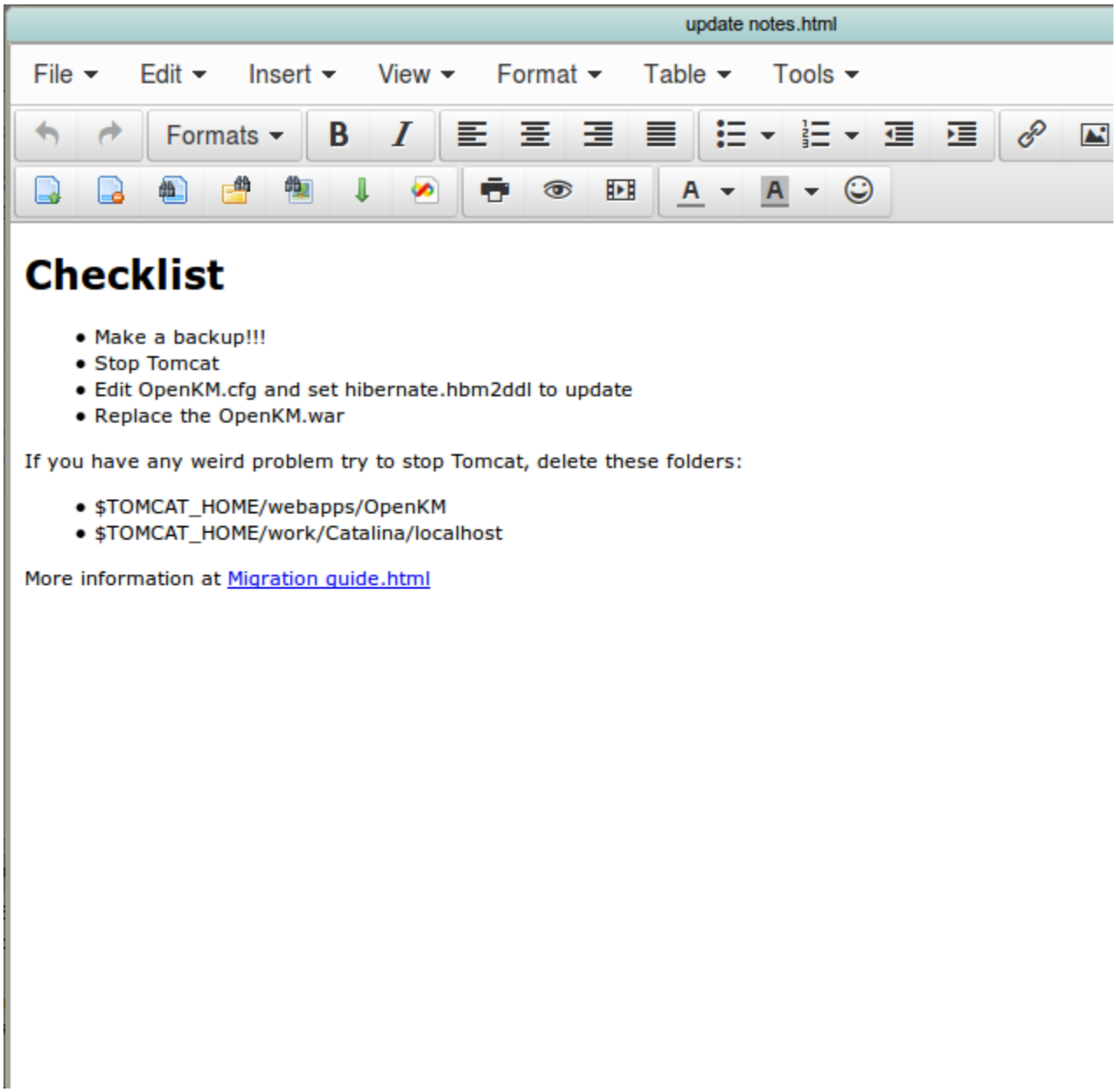
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

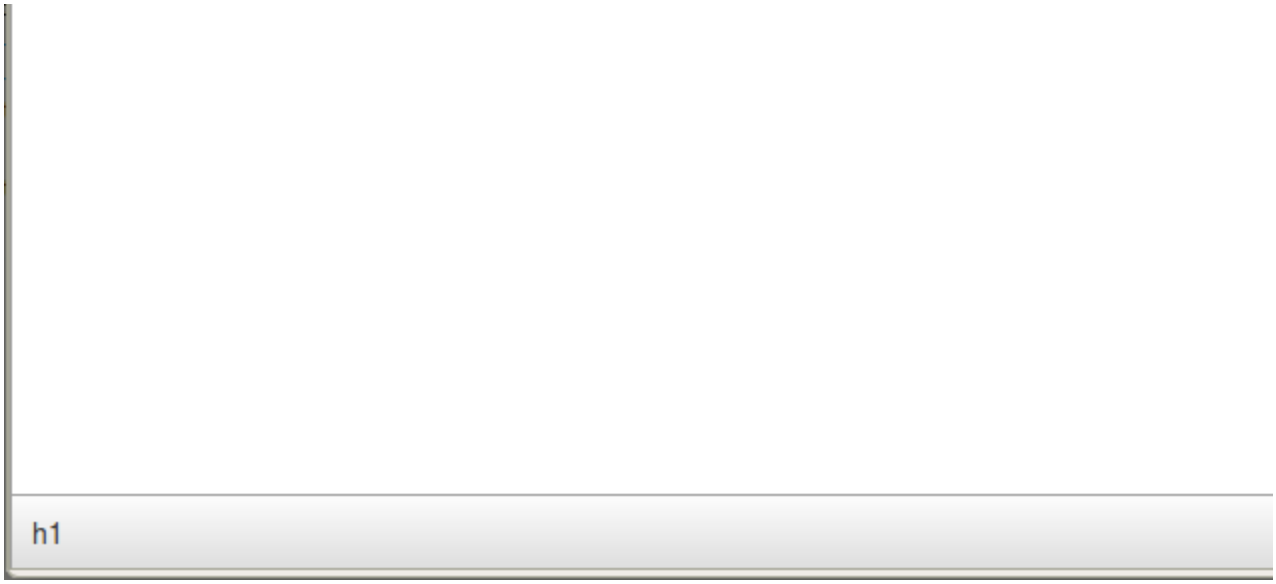
You can navigate through linked documents and folders or download documents from the Preview view.

Edit HTML

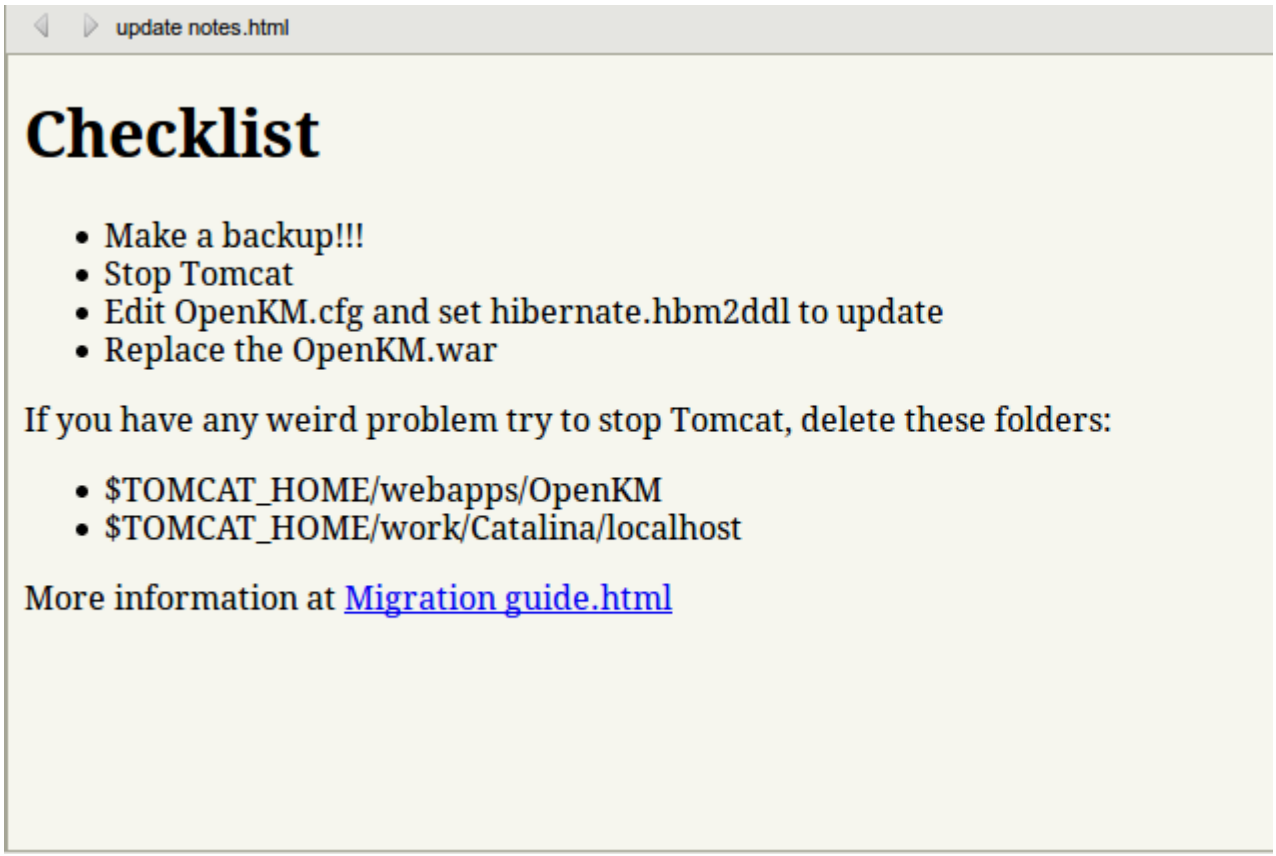
- ***Choose an HTML document***.
- *Click the* ***Edit icon***.
- *Modify the content.*
- *Click the* ***Update icon***.

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 (document, folder, email, or record). 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 the Dashboard view. This forum contains all the node topics.

Nobody can delete the "document discussions" forum in the Dashboard view.

Create new topic

- ***Choose a node** (document, folder, email, or record).*
- *Click on the **New topic button**.*
- ***Fill in the form**.*
- *Click on the **Create button**.*

Create reply

- ***Choose a topic on a node**.*
- *Click on the **Post reply button**.*
- ***Fill in the form**.*
- *Click on the **Create button**.*

Update post

- ***Choose a topic on a node**.*
- *Click on  the **Edit icon**.*
- ***Modify the form**.*
- *Click on the **Update button**.*

Delete post

- ***Choose a topic on a node**.*
- *Click on  the **Delete icon**.*
- *Click on the **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 an image or text.

Document stamping

- ***Choose a document** (PDF or convertible to PDF).*
- *Click on*  ***the Stamp icon**.*
- ***Choose the stamp type** from the list.*
- *Click on **the Stamp button**.*

When the selected document is not a PDF, a new PDF document with the stamp will be created.

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

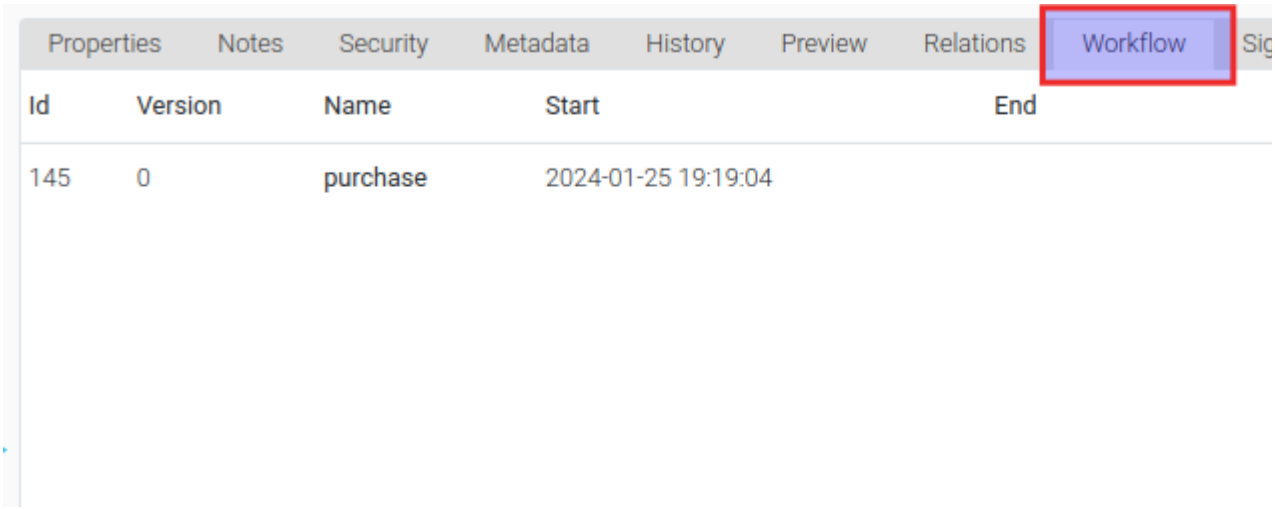
Workflow extension

The workflow extension helps show, in chronological order, the tasks performed and the workflow's status graph.

Usage

Show workflows assigned to a node

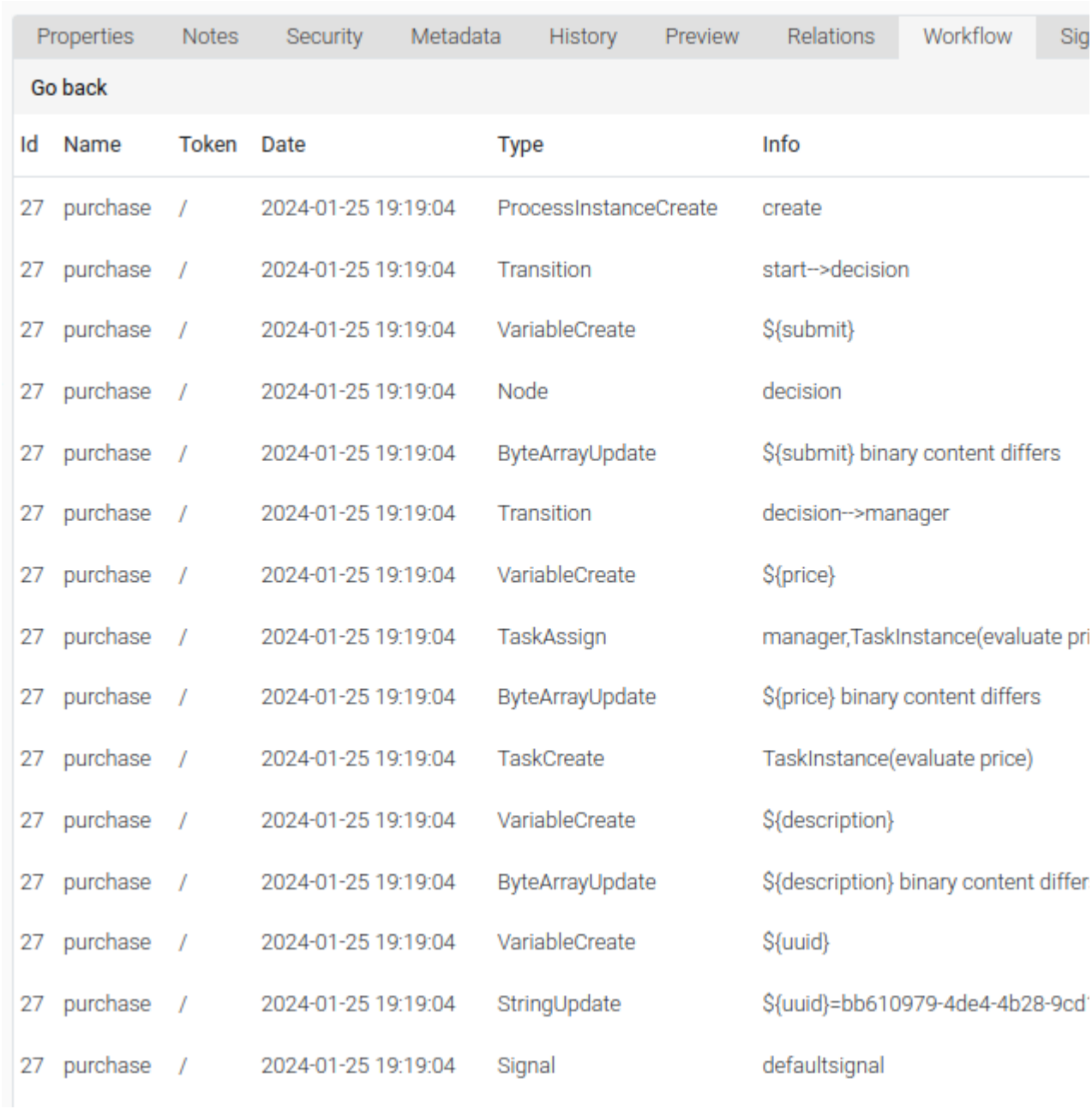
- *Choose a node.*
- *Select the Workflow tab.*



Log

- *Click on **Log** to show detailed information about the workflow.*
- *Click on **Go back** to return to the list of assigned workflows.*

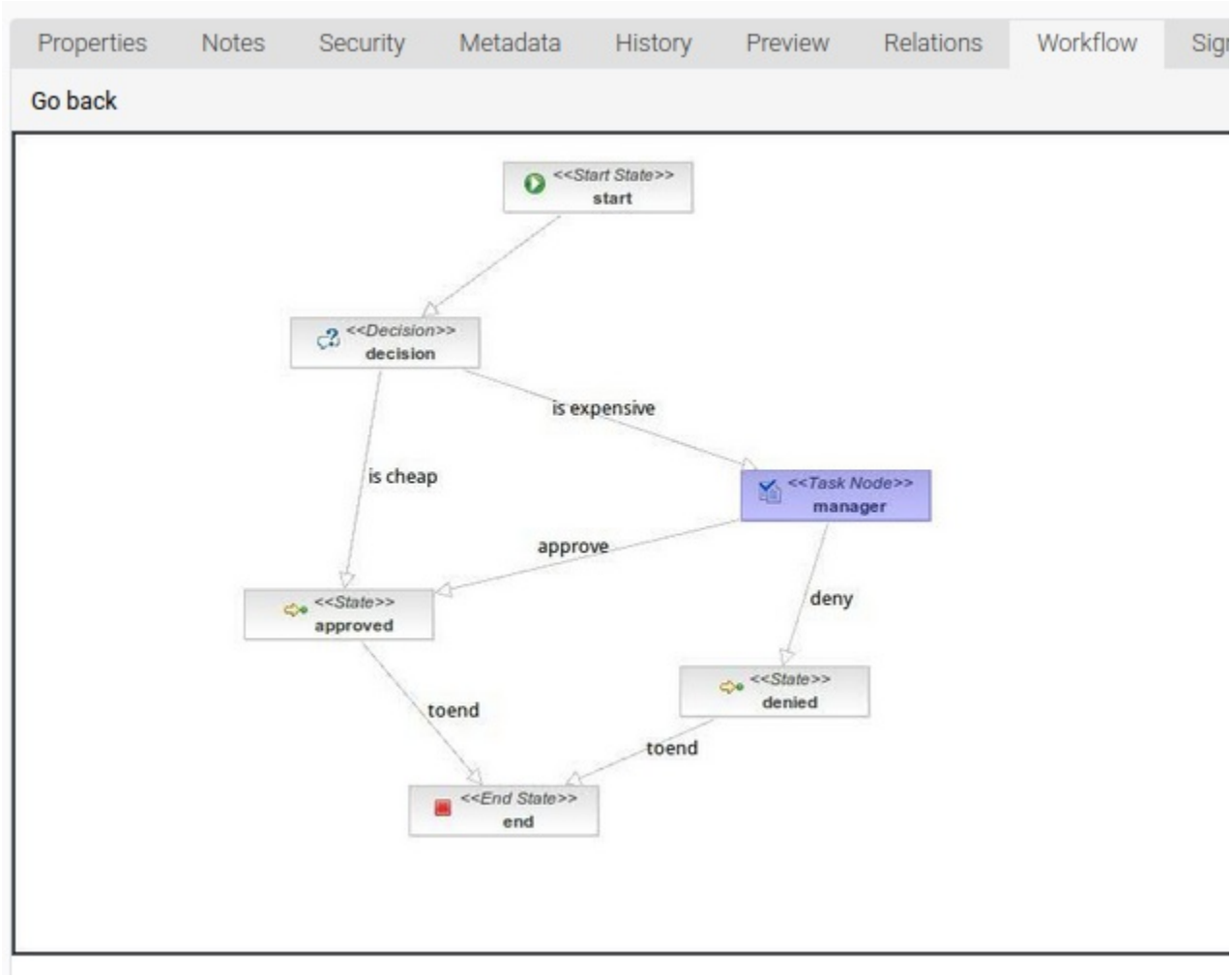
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



Graph

- *Click on **Graph** to 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 PowerPoint and Microsoft Outlook.

With the OpenKM Microsoft Outlook add-in, you can save emails to the OpenKM repository from within Microsoft Outlook.



More information at Microsoft Add-in.

Digital Signature Client

Digital Signature Clients help you quickly sign documents with an electronic digital signature in 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.

Update guide

*Download the latest "**Updater utility**" named OKMUpdater.jar*

- *Please make a backup before performing any update.*
- *In case you have customizations like workflows, automations, crontabs, or other OpenKM modules, keep in mind these customizations might also need to be migrated.*

Once downloaded, put it in the $TOMCAT_HOME directory and execute:

```
$ java -jar OKMUpdater.jar -l
```

Follow the wizard steps.

More information is available 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 the browser cache (this prevents "The response could not be deserialized" errors).

If you have a Cluster configuration, be careful because all OpenKM instances share the same database. 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 on slave nodes.

```
$ java -jar OKMUpdater.jar -l -m
```

Troubleshooting

If you have problems after configuring HTTPS in Tomcat, follow these steps:

- *Change the Tomcat configuration back to the HTTP port 8080.*
- *Execute the OKMUpdater utility.*
- *Change the Tomcat configuration to the HTTPS port 8443 again (change only server.xml and web.xml).*

In early OpenKM installations, the upgrade process may require some preliminary steps. If you see an error like this one, you have this problem:

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



To be able to use the updater utility, you must 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 and attach the OKMUpdater.log and $TOMCAT_HOME/logs/catalina.log files.

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

Changelog

v8.2.3

- *Release: 2026-01-27*
- *Initial release.*

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

Updater utility

OpenKM "Updater utility" is a Java administrative tool that helps perform upgrades automatically. You can download it from https://download.openkm.com/okm/OKMUpdater.jar.



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

- *Check whether it's located in the $TOMCAT_HOME folder; otherwise, it will raise an error message.*
- *Ask the administrator for some information.*

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

- *OpenKM URL (usually something accessible from the server like http://localhost:8080/openkm).*
- *The application is configured as a service [Y/N].*
- *The valid OpenKM ROLE_ADMIN user (okmAdmin).*
- *The administrator user password.*

- *Check the service configuration, the user connection, and the OpenKM version.*
- *If the utility is successful in these checks, it will require confirmation from 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 with the newly downloaded one.*
- *Modify the parameter **hibernate.hbm2ddl** to **update** in the file $TOMCAT_HOME/openkm.properties.*
- *Start OpenKM again.*
- *Check whether OpenKM is available, 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 in the right order; otherwise, it may cause information loss.

Types of actions:

- ***SQL**: Go to **Administration > Utilities > Database query** (Choose SQL from the selection)*
- ***HQL**: Go to **Administration > Utilities > Database query** (Choose HQL from the selection)*
- ***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***



It is important to know which actions are executed in PRE and in POST.

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



Sometimes there may be 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 later*

By default, OpenKM is installed with a specific Tomcat version. However, every few weeks a new version is released, and you may want your installation to be up to date. In this case, you should open a ticket on the OpenKM support website to ensure this new version has been packaged by the OpenKM Team. Once you have verified that the new version is available and have downloaded it, you must follow the steps described in this section of the documentation.



Before updating Tomcat, we recommend upgrading to the latest OpenKM version and ensuring you have a recent working backup.



Although you can use a pristine Tomcat, we always recommend using those provided by the OpenKM team because they have additional files and default configurations that work best with OpenKM.

Quick method

The easiest method to update your Tomcat instance is to follow these steps:

- *Stop the current Tomcat / OpenKM process.*
- *Download the latest Tomcat provided by the support team.*
- *Remove the TOMCAT_HOME/lib folder contents and replace them with the downloaded ones.*
- *Start Tomcat / OpenKM again.*

Once the process has finished you can verify the new version:

```
$ cd TOMCAT_HOME
$ ./bin/catalina.sh version
```

If you have any doubts or questions, please open a support ticket.

Linux

Preliminaries

- *Check in the **openkm.properties** file whether 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 on the destination. Sometimes when you move data, the applied security might change and 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, it might damage the repository. Please check that the file has been correctly copied to the destination.*
tomcat-XXX/openkm.xml	*File*	*OpenKM security configuration file.* *This file might not exist. You can find this file only in some special installations.*
tomcat-XXX/logback.xml	*File*	*OpenKM log configuration file. Might not exist.*
tomcat-XXX/repository	*Folder*	*The folder where documents are stored.* *If there's a mistake copying this folder, it might damage the repository. Please check that the folder has been correctly copied to the destination.*
tomcat-XXX/plugins	*Folder*	*Directory with OpenKM plugins.* *It 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 exist. This file exists only if the 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 **with the new path**.*
- ***Restart OpenKM service.***
- ***Do a quick test.***

Suggested quick test:

- *Browse the repository.*
- *Download an existing file.*
- *Upload a new file and try to download it.*

Remember to update the backup script because the location of the Tomcat directory may change. On Linux you can check it by running (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 in mind that the backup script usually backs up the entire /home/openkm directory, so move it to another place.

If you want a more recent branch you can try Tomcat 9.0, but keep in mind that this new branch has not been fully tested and is not supported by the OpenKM team.

Windows

Preliminaries

- *From Services, check the user who's running OpenKM. Usually, the user should be "Local Services", but sometimes a*

standard user is used. Remember when registering the OpenKM service again to choose the right user.

- *Check in the openkm.properties file whether 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 as an Administrator user.*
> - *Go to the tomcat-XXX/bin folder.*
> - *Execute the next command:*
>
> ```
> $:> service.bat remove
> ```
>
> *Second option:*
>
> - *Open a terminal as an Administrator user.*
> - *Execute the next command:*
>
> ```
> $:> sc delete OpenKM
> ```

- ***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 on the destination. Sometimes when you move data, the applied security might change and 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, it might damage the repository. Please check that the file has been correctly copied to the destination.*

tomcat-XXX\openkm.xml	*File*	*OpenKM security configuration file.* *This file might not exist. You can find this file only in some special installations.*
tomcat-XXX\logback.xml	*File*	*OpenKM log configuration file. Might not exist.*
tomcat-XXX\repository	*Folder*	*The folder where documents are stored.* *If there's a mistake copying this folder, it might damage the repository. Please check that the folder has been correctly copied to the destination.*
tomcat-XXX\plugins	*Folder*	*Directory with OpenKM plugins.* *It 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 exist. This file exists only if the jBPM engine has been installed.*

- ***Configure OpenKM as service***

- *Open a terminal as an Administrator user.*
- *Go to the 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* ***with the new path***.
- ***Restart OpenKM service.***
- ***Do a quick test.***

Suggested quick test:

- *Browse the repository.*
- *Download an existing file.*
- *Upload a new file and try to download it.*

Manual update



Nowhere in the public documentation will you find any explanation about how to do a manual upgrade.

The whole migration section and the OpenKM Updater tool have been created to help users with the OpenKM upgrade process. Unfortunately, the automatic upgrade tools

Unfortunately, the automatic upgrade tools are not compliant with all companies' policies and, in some cases, it is mandatory to do a manual upgrade.

We have decided not to explain them here because, although most migration steps between OpenKM versions are easy, some require your attention. For better service and to keep your system safe, we think working together on it is the best option.



If this is your case, please create a ticket on our OpenKM customer support website and we will help you with it.

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

Oracle Java to Temurin

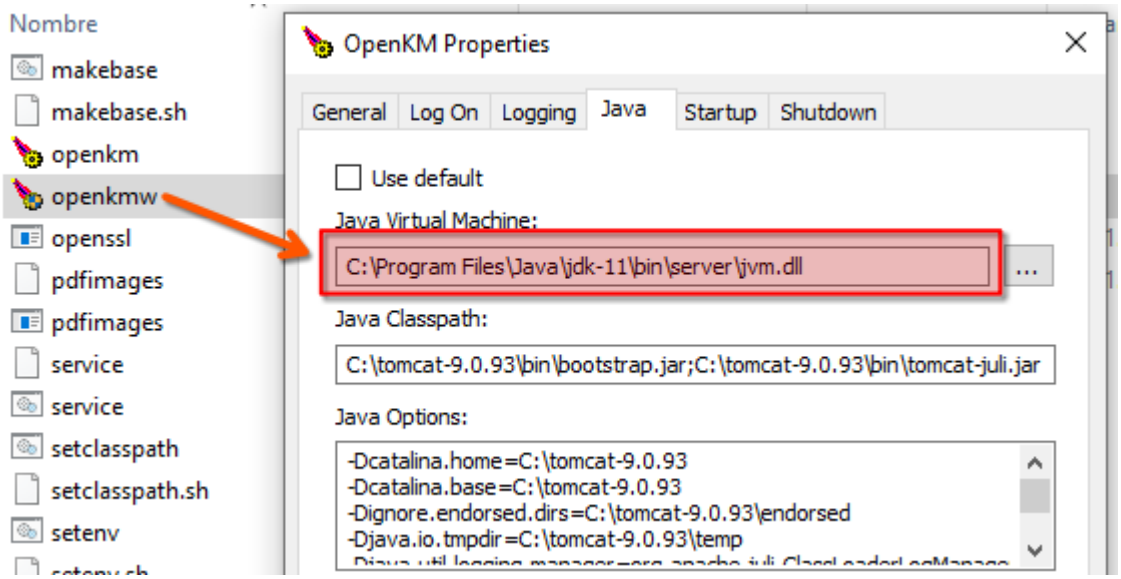
To switch from Oracle Java to Temurin Java in Windows, follow these steps:

- *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 the OpenKM installation directory and execute the "openkmw.exe" program.*
- *Go to the "Java" tab and change the Java Virtual Machine path to the Temurin installation directory.*
    - *It should be something 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 8.1

Change

- *Java 17 support.*
- *Tomcat 10 support.*
- *Major library updates (added, updated, and removed libraries to improve security).*

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 in a few steps.*
>
> *More information about the OpenKM professional portable development environment is available 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 CPUs*
- *4GB RAM*
- *64-bit 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

The OpenKM personal development environment is distributed as a virtual machine image (OVA).

The personal development environment comes with everything you need to start working quickly:

- *Linux Mint OS.*
- *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 stopping almost all applications while running the virtual machine.

We suggest installing it on a desktop computer rather than a laptop. It can work on a laptop with good hardware.

We suggest using an SSD to get extra performance in the virtual machine.

With some users, we have observed that running the virtual machine from a secondary disk provides extra performance; it seems some motherboards do not handle the OS and virtualization on the same hard disk well.

How to get the personal development environment

You can download the virtual machine file from the OpenKM download center.

- *Go to https://download.openkm.com*
- *Click the "OpenKM Professional Download Center" button.*
- *Log in with your customer credentials.*
- *In the top menu, choose Tools.*
- *In Tools, ensure you have chosen OpenKM version 8.1.x.*
- *Click on the Open KM-PorDev.ova file.*



The download link is sent by e-mail.

The link can be used only once.

You have 24 hours from when the e-mail is sent to use the link; after this time the link expires.

How to install

The OpenKM personal development environment can be installed on most virtual machine software like VirtualBox, VMware, and Hyper-V.

Because OVA (Open Virtual Appliance) is not 100% compatible with virtualization software, users of VMware usually need to make a few modifications to the file or extract the mkv to import the virtual machine.

If you are looking for a quick deployment, we encourage using VirtualBox.

If you decide on VirtualBox, follow the steps described in the next video https://www.youtube.com/watch?v=9ugBYf5ggu8&feature=emb_logo

License activation

The personal development environment must have an Internet connection to register the license.

Finally, the license for your personal development environment must be activated; to do so, follow these steps:

1- Open a terminal (the password for the user ***openkm*** *is* ***openkm****).*

2- Execute the following command

```
$> pdl-reset
```

3- Start the OpenKM service.

```
$> sudo service openkm start
```

4- Open an issue on the OpenKM customer support website. Share the application ID available on the administration home page.

How to upgrade the development environment

Because the virtual machine comes with OpenKM, jbpm-console, and compilation libraries, the upgrade process has at least three 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 the OpenKM tool as you would for any standard OpenKM installation. 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 comes with the update.sh script.

```
$> cd /home/openkm/Development/tomcat
$> ./update.sh
```

Step 2 - OpenKM compilation libraries upgrade

- *Close Eclipse.*
- *Edit the download.sh script and set the latest OpenKM and jbpm-console versions in the header.*

```
$> cd /home/openkm/Development/portable/lib
$> nano download.sh
```

Sample script for version 7.1.22 (remember to set the correct JBPM version based on the compatibility table in the 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 and set the latest OpenKM versions in the header.*

```
$> nano register.sh
```

Sample script for version 7.1.22 (remember to set the correct JBPM version based on the compatibility table in the 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 correct jbpm-console version based on the compatibility table in the JBPM Console documentation section.*
- *Follow the steps described in the Migration guide.*

Troubleshooting

Importing the OVA File into a Non–VirtualBox Virtual Machine

*If you encounter issues importing the provided **OVA** file into a virtual machine platform other than **VirtualBox**, it is likely due to differences in virtualization formats. Since most VM vendors do not follow a standardized format, direct OVA imports between different platforms are often not supported.*

To resolve this issue, follow these steps:

- ***Rename** the OVA file extension from .ova to .tar.*
- ***Extract** the contents of the .tar archive using your preferred extraction tool.*
- *Inside the extracted files, locate the **virtual disk image**, typically with the extension .vmdk.*
- ***Create a new virtual machine** manually in your chosen virtualization platform.*
- *When prompted for a disk image, **attach the extracted** .vmdk **file** as the virtual disk.*

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

This process effectively recreates the original virtual machine using the extracted disk image, making it compatible with non–VirtualBox environments.

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 Java development tools like NetBeans.

Requirements

Software:

- *Eclipse*
- *OpenJDK 11*



Your Eclipse must have support for:

- *Maven*
- *Subversion (recommended)*

Hardware:

Suggested minimum hardware configuration:

- *2 CPUs*
- *4 GB of RAM*
- *64-bit 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-bit).*



More information at https://eclipse.org/downloads/.

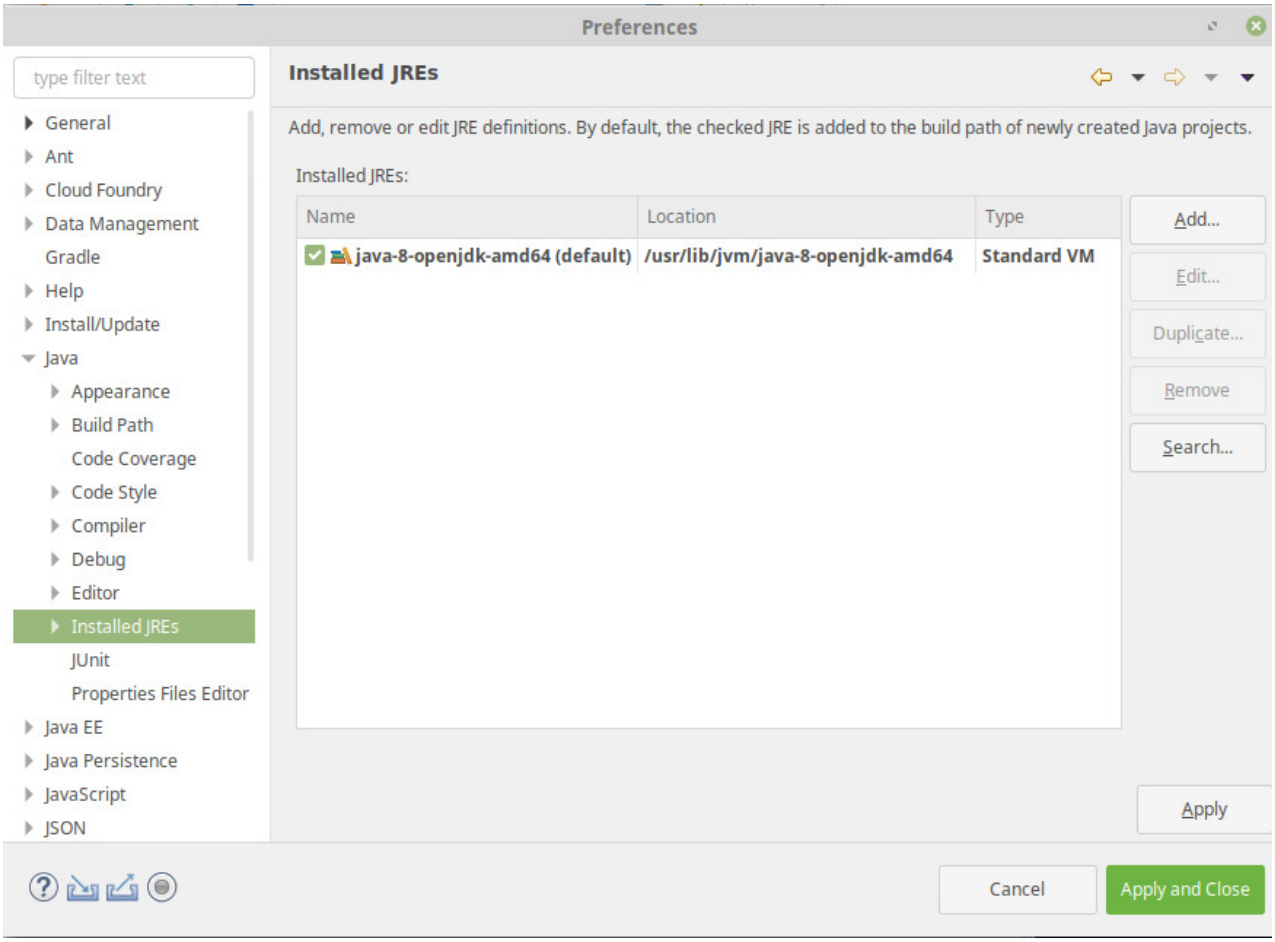
Configure Java

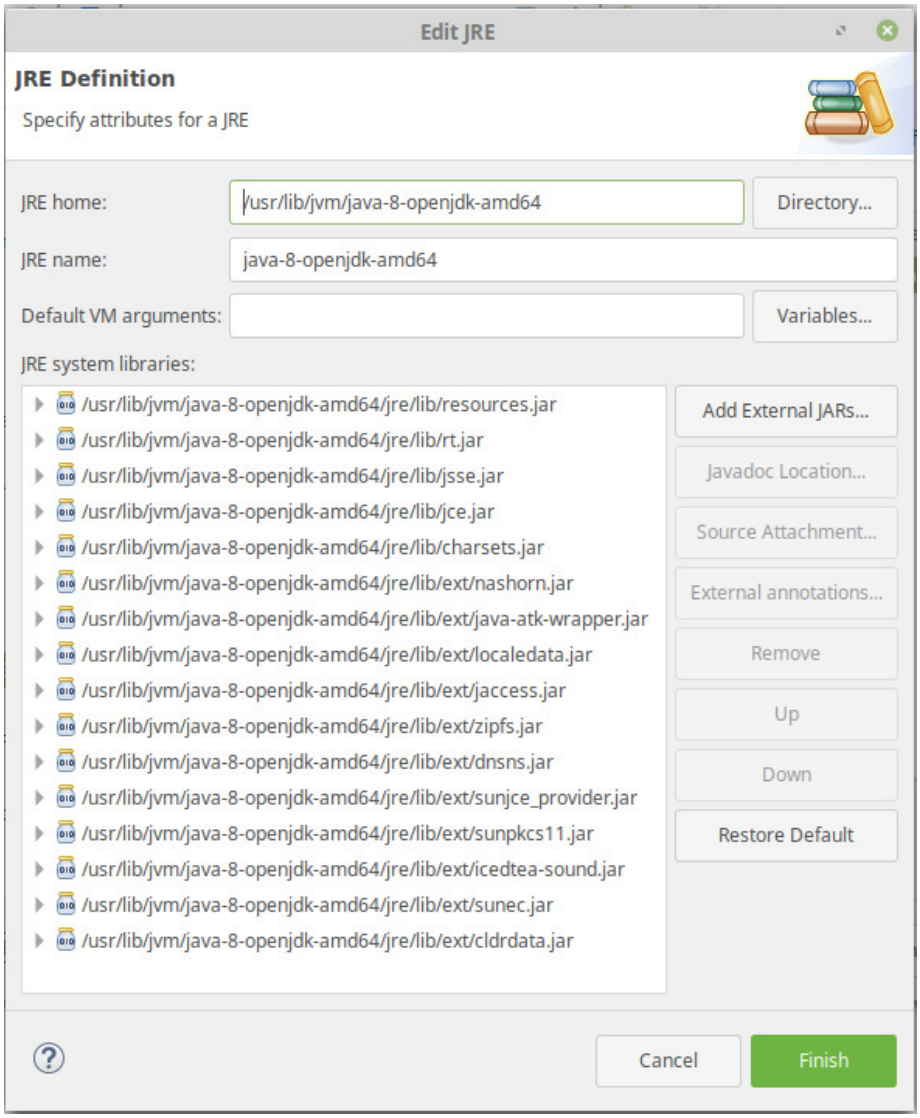


You must have Java installed on your server; otherwise, go to https://learn.microsoft.com/en-us/java/openjdk/download and download the latest OpenJDK 11 version according to your OS and hardware architecture. Execute the Java installer and follow the installation wizard.

*Go to **Window** / **Preferences** / **Java** / **Installed JRE**. You must have OpenJDK 11 defined here:*

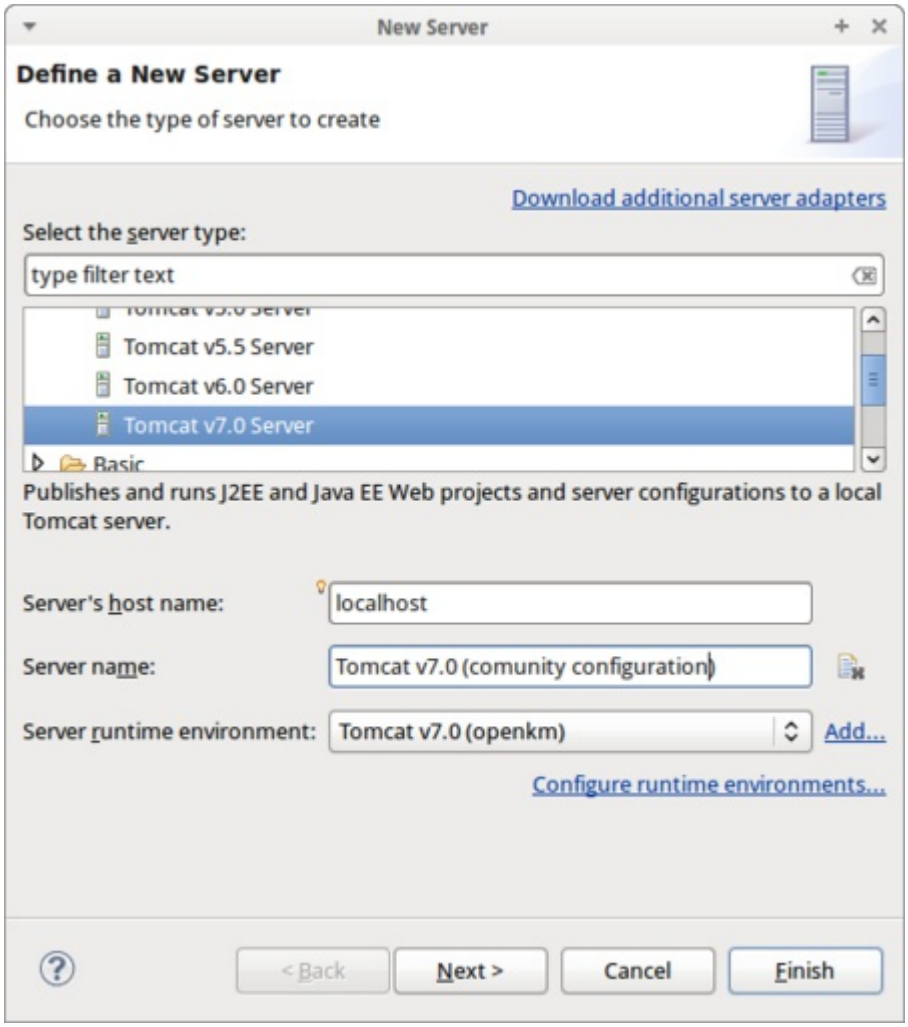
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


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



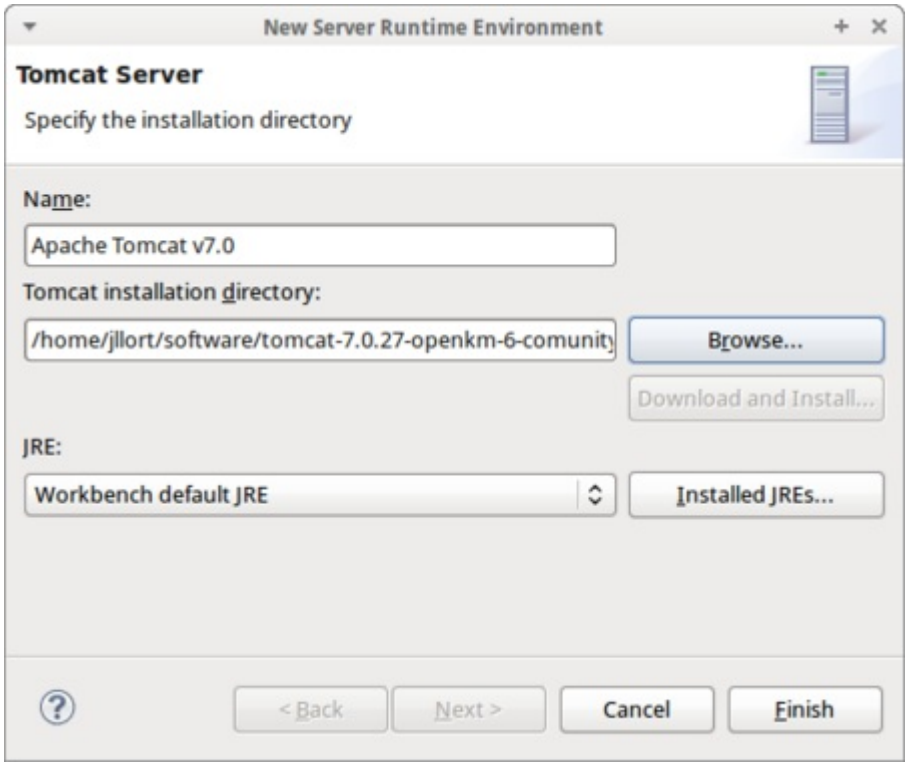
Configure Tomcat server

*Go to the "**Servers**" tab. Right-click and, from the contextual menu, choose "**Add / server**". From the list choose "**Tomcat 7 Server**":*



*Configure a new **runtime server environment**:*

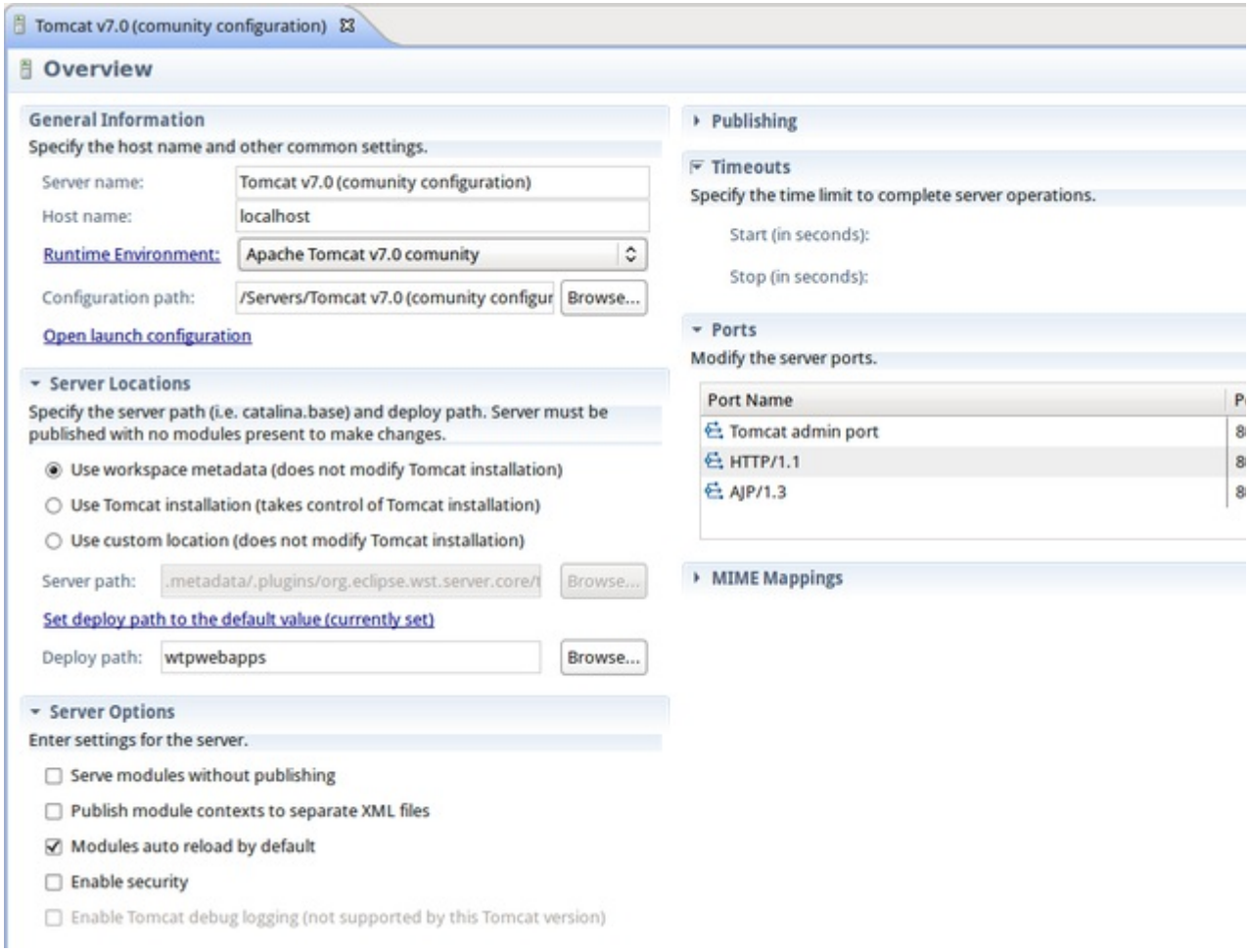
> *Choose the "Workbench default JRE" if your default JRE is "OpenJDK 11", or choose "OpenJDK 11" from the list of installed JREs.*



*Check the box **Use Tomcat installation** (takes control of Tomcat installation).*

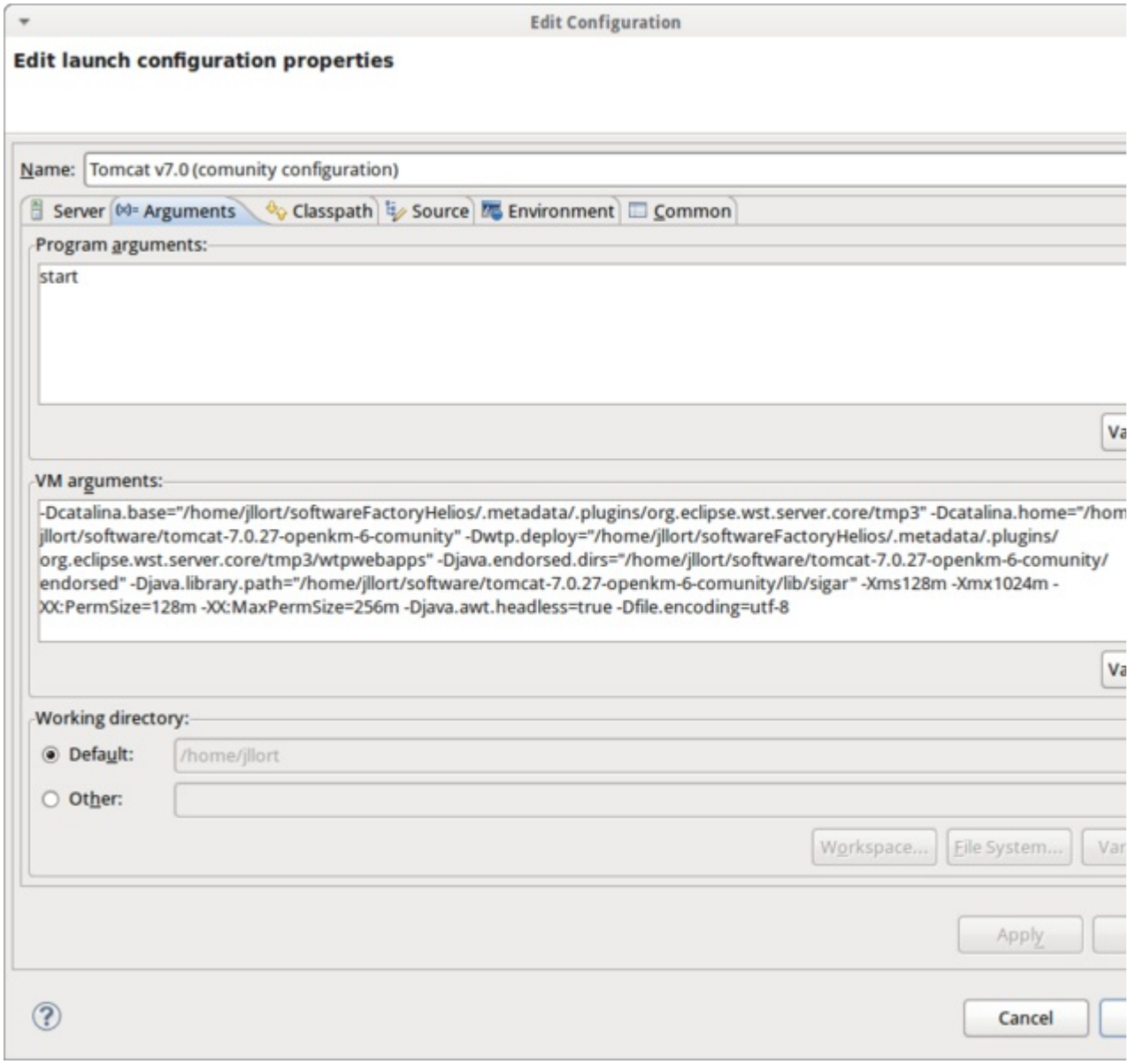
*Increase the start and stop **timeouts** (for example, to 180 and 120 seconds, respectively).*

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

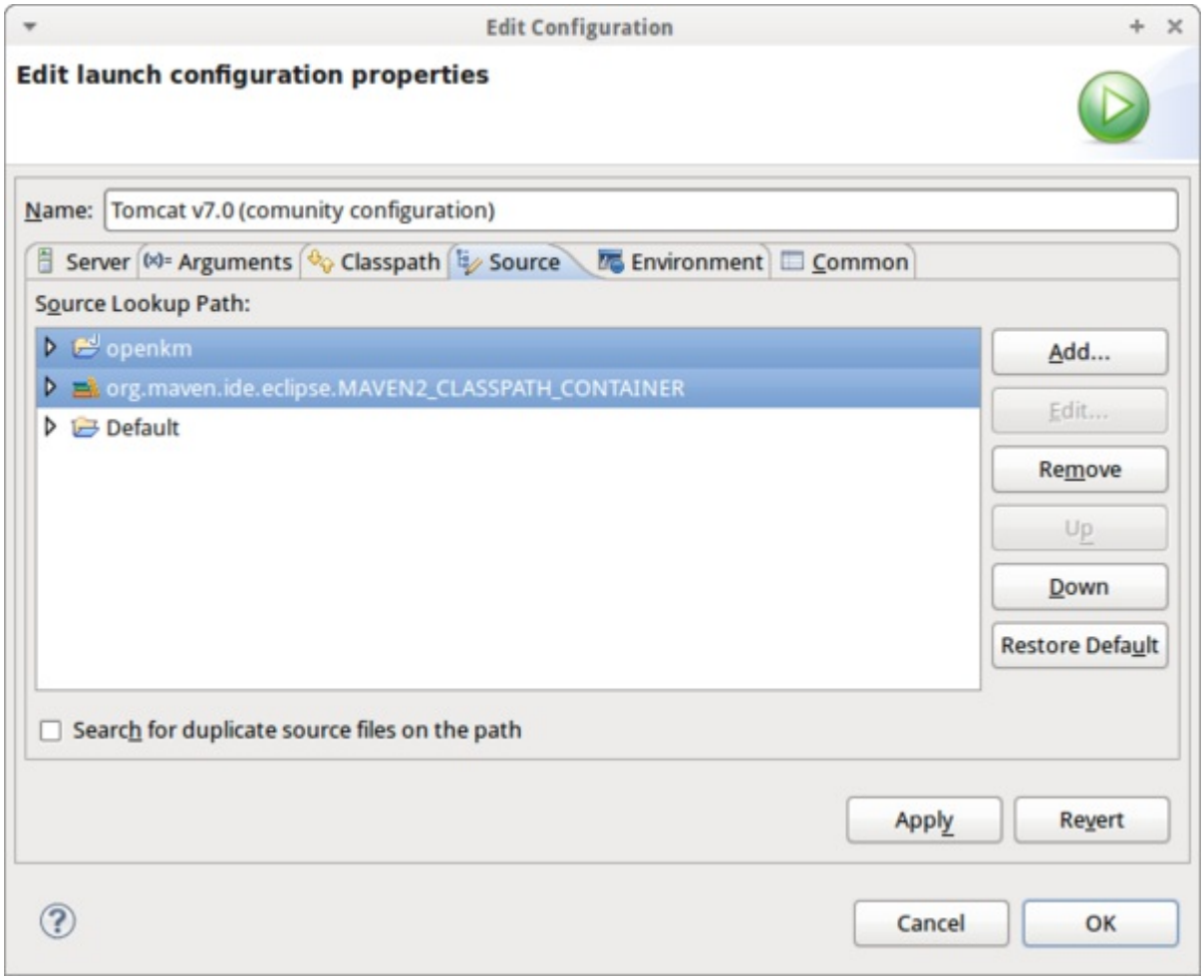


*Edit the **launch configuration**: add **memory parameters** (we suggest 4 GB) and **UTF-8** file encoding in the VM arguments field:*

```
-Xms1024m -Xmx4096m -Djava.awt.headless=true -Dfile.encoding=utf-8
```



*Add your **OpenKM project** in the Source tab:*



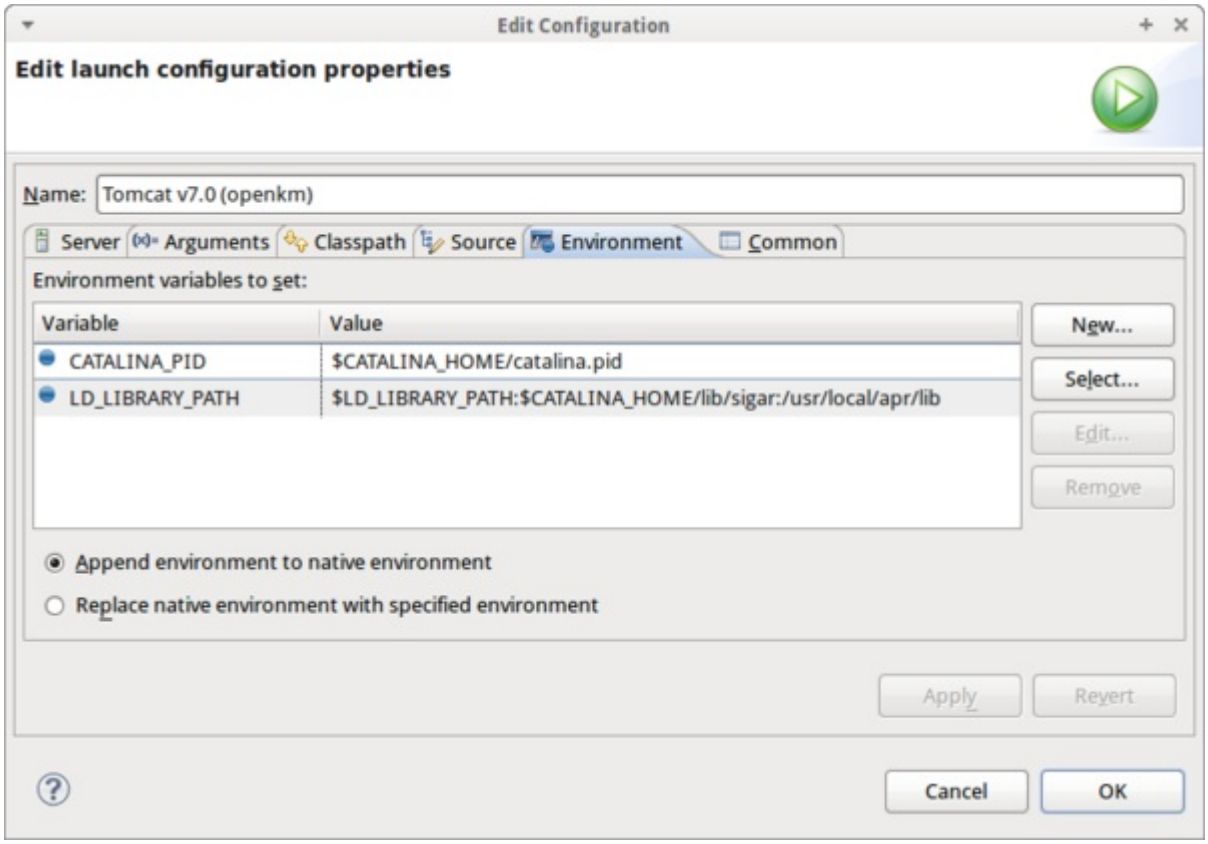
*Add **CATALINA_PID** to the environment variables:*

```
CATALINA_PID=$CATALINA_HOME/catalina.pid
```

or

```
CATALINA_PID=$CATALINA_HOME/catalina.pid
```

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



Maven support for Eclipse

The Eclipse version comes with Maven support plugins installed by default.

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 for library dependencies 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, execute 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 by 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 it from the 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 the **Add** button to set your local Maven.*



We do not suggest that you apply this change; the default embedded Maven configuration usually works in most 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 useful when you have problems with your configuration. The default OpenKM installation logs important events such as errors and warnings. It is possible to change this configuration by editing the file $TOMCAT_HOME/conf/logging.properties.

The default Tomcat log configuration can generate many messages. These files are stored in $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 connect to port 8000.



More information at:

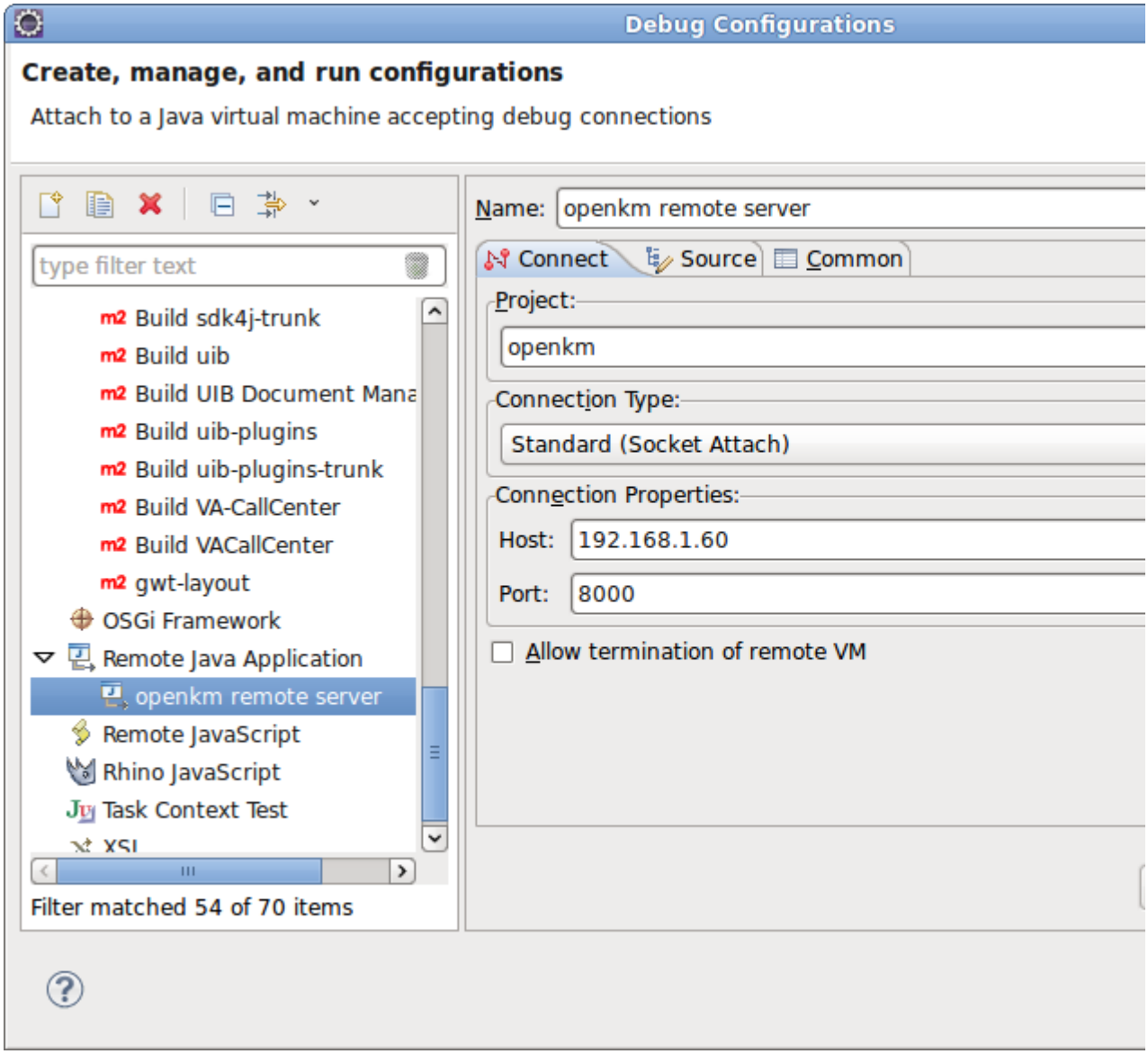
- *Tomcat development: How do I configure Tomcat to support remote debugging?*
- *Como hacer debug remoto en Tomcat desde Eclipse .*

Eclipse configuration

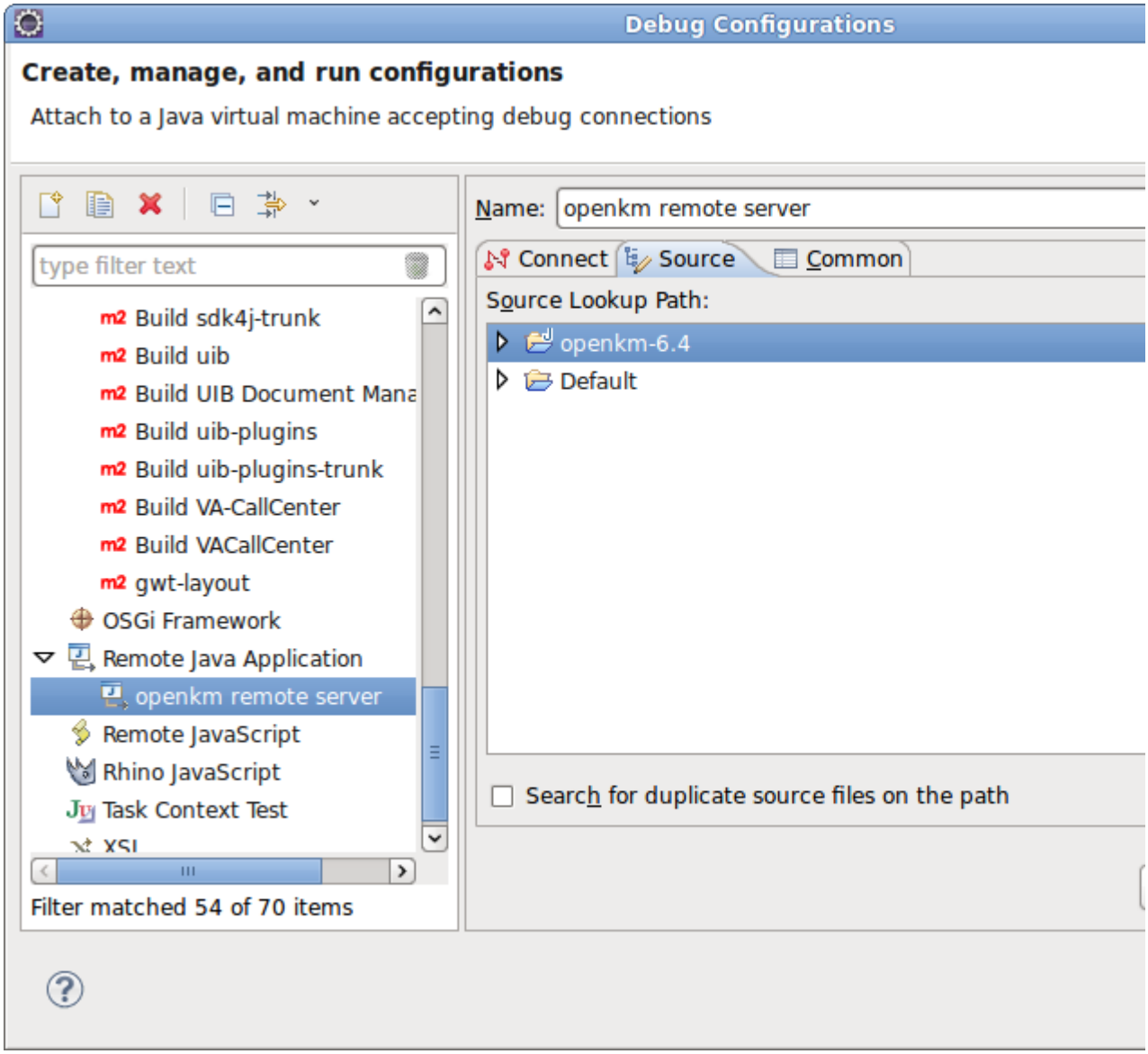
*Open Eclipse "**Debug Configurations**", then choose "**Remote Java Application**", right-click and choose "**New**".*

*Set the "**host**" and "**port**" of the remote server:*

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



Include your OpenKM Java project in the Source tab:



You can debug normally by setting breakpoints, etc.

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

Eclipse known issues and troubleshooting

Maven raises 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 by a Windows OS limitation. To solve it:*

- *Move the* ***.m2*** *repository to* ***c:***
- *Go to* ***Eclipse*** *>* ***Windows/Preferences/Maven/User Settings*** *->* ***Create your own settings.xml*** *with the following content:*

```
<settings>
  <localRepository>c:/.m2/repository</localRepository>
</settings>
```

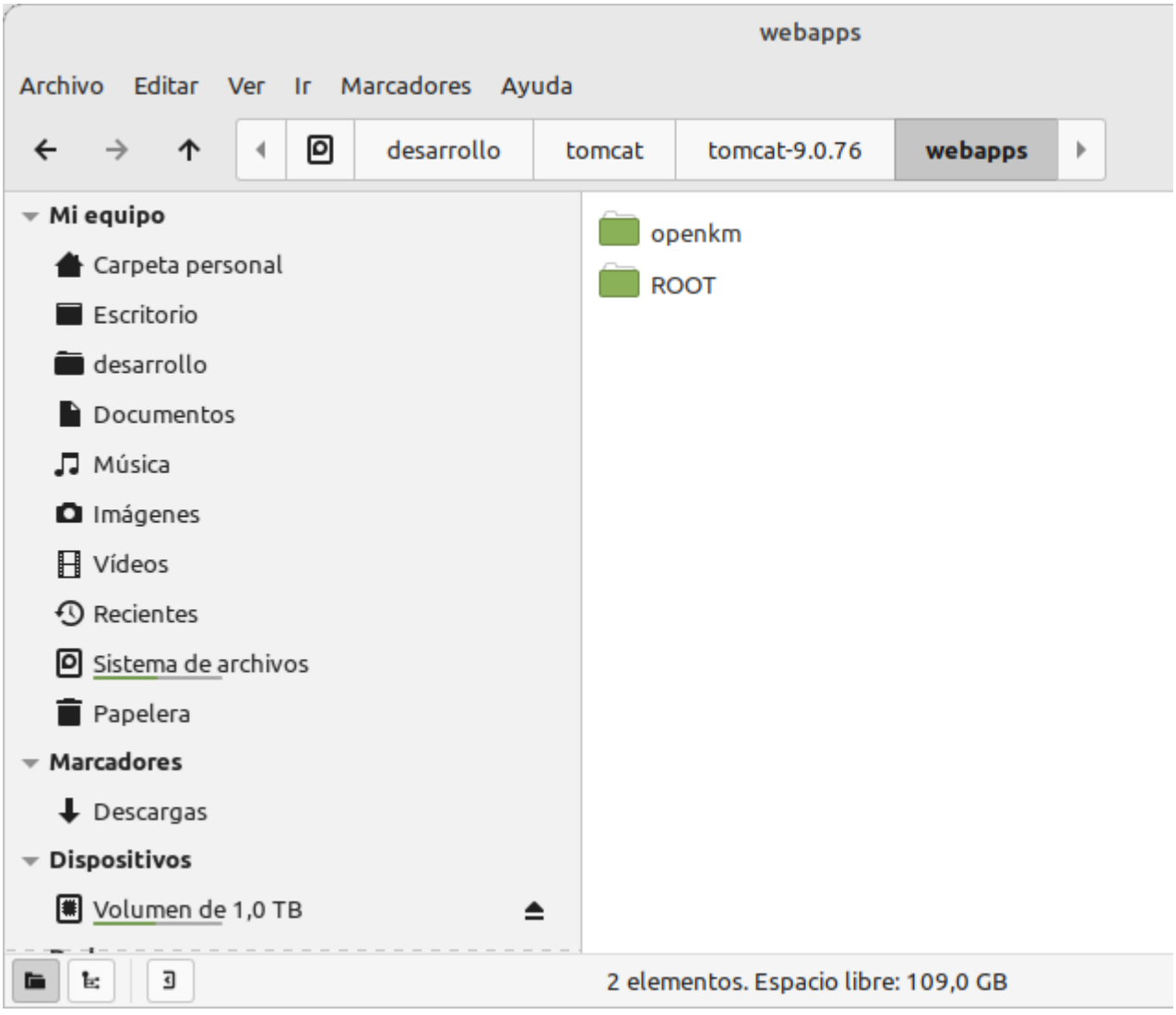
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Development tips

Tomcat application deployment

For faster development, we recommend not deploying openkm.war directly; instead, create in $TOMCAT_HOME/webapps a folder alias of target/openkm:

> *To create symbolic links in Windows use the "**junction.exe**" application.*
>
> *More information at http://technet.microsoft.com/en-us/sysinternals/bb896768.*



Sample deployment script

You can use a deployment 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 applications

You can profile local and remote Java applications using Java VisualVM.



For local applications it is easy because they are detected automatically and shown in the listing. For remote applications you need to do a little work.

Edit Tomcat startup scripts

*Edit the **$TOMCAT_HOME/bin/setenv.*** files (setenv.sh for Linux and setenv.bat for Windows).*



*The **java.rmi.server.hostname** should match the IP of the server, usually 127.0.0.1; replace it with the correct 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.*
- *You can also change the port to 9090 using the **com.sun.management.jmxremote.port** property.*

Authentication

*To enhance security, enable authentication. Edit the **$TOMCAT_HOME/bin/setenv.*** files (setenv.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 control 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:

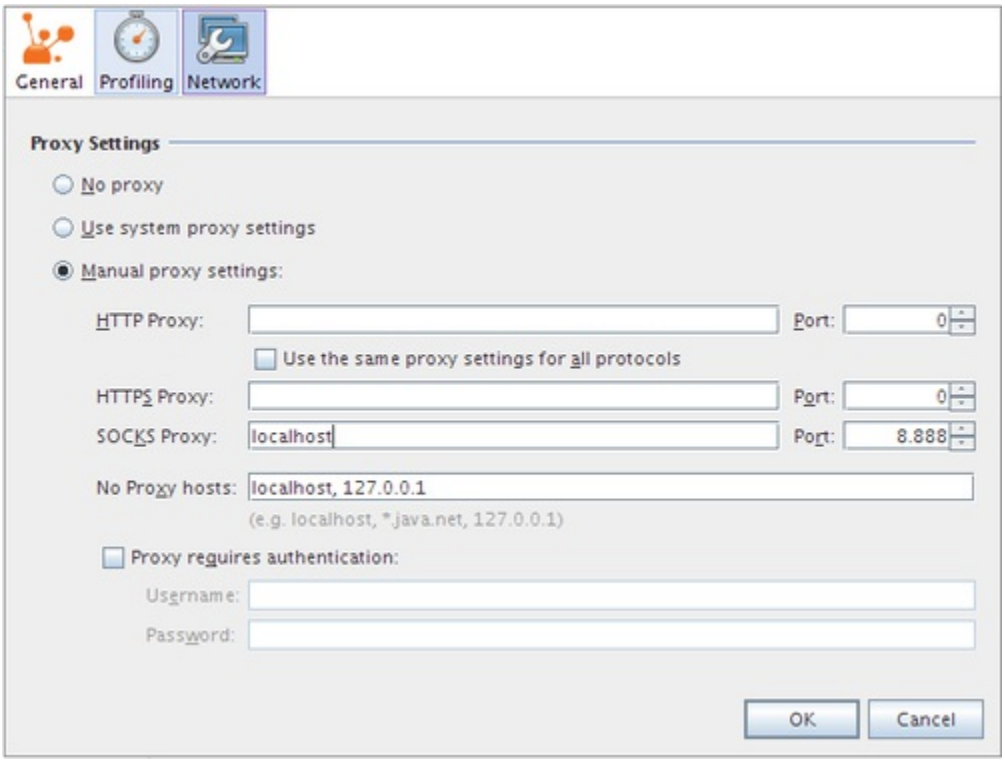
```
$ chmod 600 $CATALINA_HOME/conf/jmxremote.access
$ chmod 600 $CATALINA_HOME/conf/jmxremote.password
```

Monitoring Remote JVM Over SSH

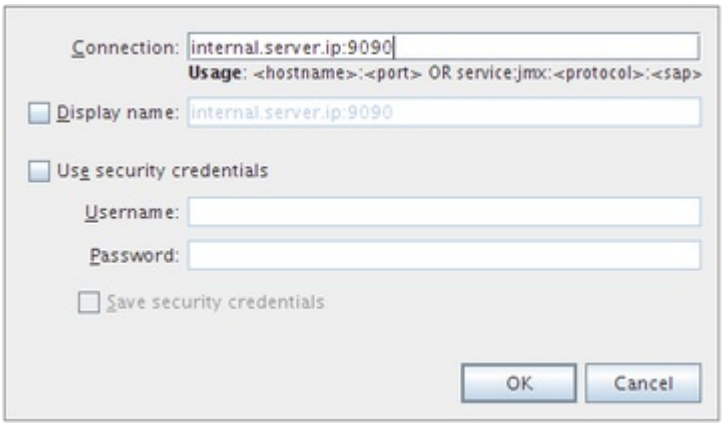
The easiest way to connect to the remote JMX is to use a SOCKS proxy.

```
$ ssh user@external.server.ip -D 8888
```

Configure VisualVM to use this proxy. Go to ***Tools*** *>* ***Options*** *>* ***Network*** *and edit* ***Manual Proxy Settings****. Configure the 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, 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 collects application runtime information. HPROF is capable of presenting CPU usage, heap allocation statistics, and monitor contention profiles.

For example, it can collect CPU usage information by sampling threads.

*Edit the **$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 the Tomcat process is stopped.*

More information at:

- *HPROF: A Heap/CPU Profiling Tool in J2SE 5.0*
- *HPjmeter*

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

Application version compatibility table

Application compatibility changes can be caused by:

- *DTD definition changes. Usually, a newer application version supports older DTD versions.*
- *Library upgrades.*

If your OpenKM server can't access the Internet, you can download this DTD and copy it to a convenient place on 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-2.2.dtd	Since 8.1.19
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 an introduction.

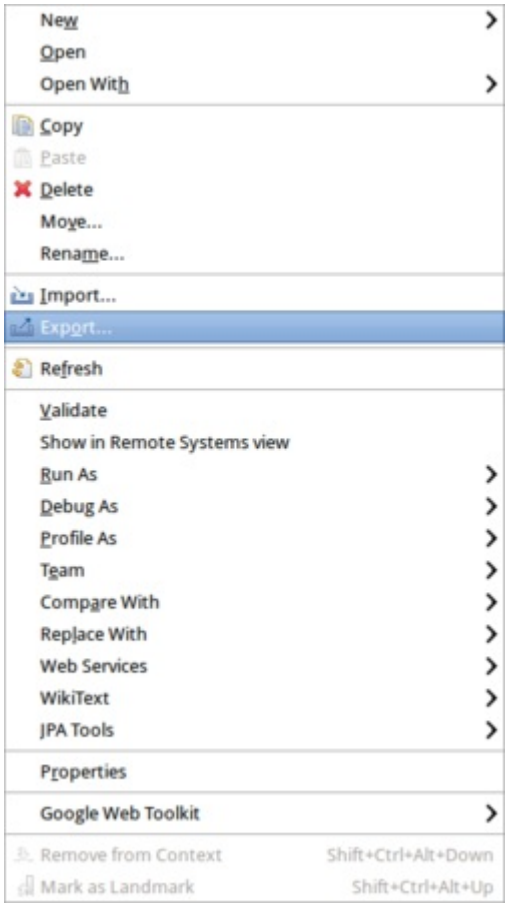
The following steps may vary depending on the Eclipse version.

1. ***Choose the Java file*** *from Project Explorer or Navigator view.*
2. ***Right-click*** *(a contextual menu will be shown) 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 entry point.*
8. ***Click Finish****.*

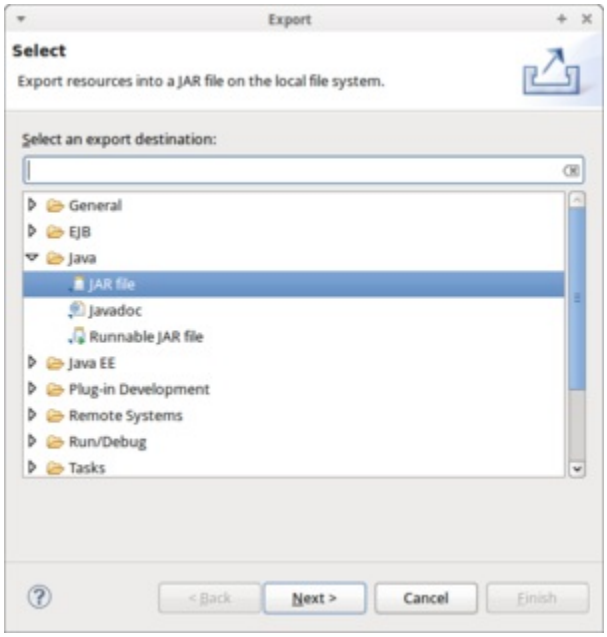
Images

Select the file to be exported as a JAR. Right-click and select the " ***export****" option.*

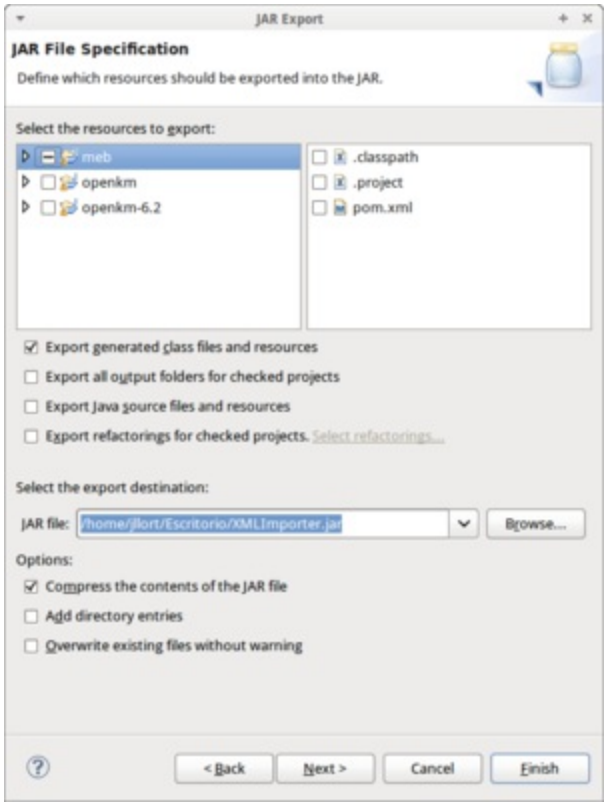
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



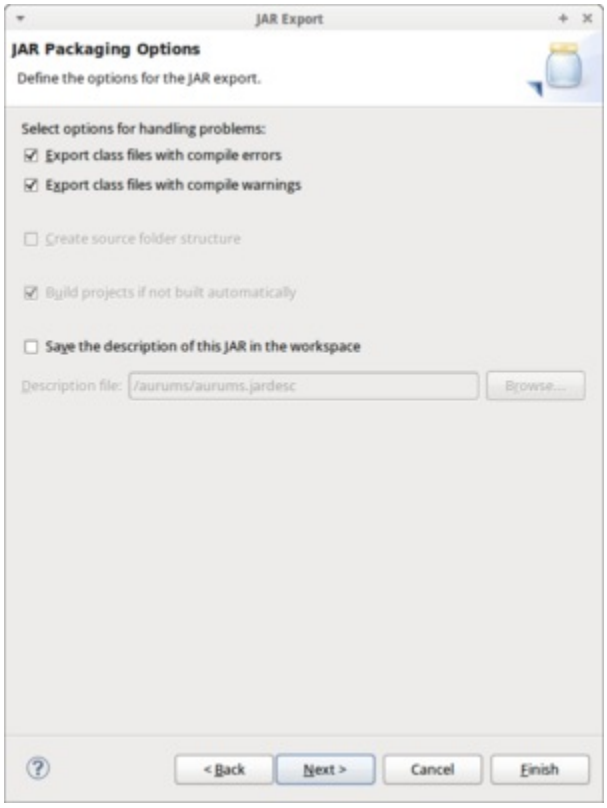
*Select the "**.jar**" file option and click the "**Next**" button.*



*Select the "**Java classes to be included**" for the JAR file and click the "**Next**" button.*

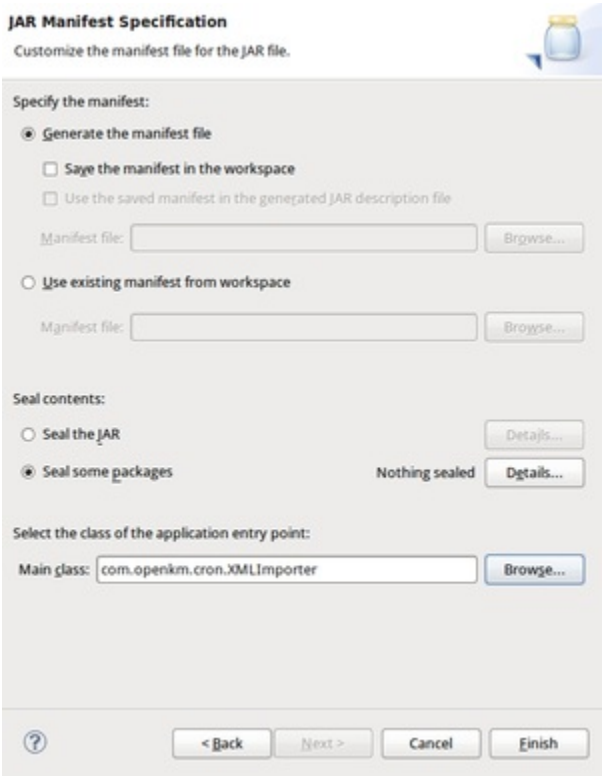


*Click the "**Next**" button.*



*Select the "**Main class**" and click the "**Finish**" button.*

> *If your Java class implements "**public static void main(String[] args)**", then you must choose your Java main class as the entry point, otherwise you can skip this step.*



Sample JAR file

Requisites:

- *Must implement the method "**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 deployment** of JAR packages.*

With 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, by connecting with third-party tools).*
- *Register a new suggestion analyzer (build your own logic to suggest form values).*
- *Register a new text extractor.*
- *Register a 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 a new plugin* ***create a JAR file*** *and* ***copy it into your $TOMCAT/plugins*** *folder. More information at Creating a JAR file with Eclipse (you can also build your JAR files with Maven).*
- *Go to* ***Administration > Utilities > Plugins*** *and click the*  *Reload plugins button at the top right.*
- *In* ***the plugins table****, the new plugin will be shown.*



A plugin configuration may require changes to Configuration parameters or a restart of the application. For example, the 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 "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 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 about Register a new plugin.

Method descriptions

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 piece of text.*
getName()	*String*	*Returns 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 in the package "com.openkm.plugin.antivirus".***
- *The new Antivirus Analyzer class **must be annotated with "@PluginImplementation".***
- *The new Antivirus Analyzer class **must extend "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 plugin system will try to load it from there.*



*Do not forget 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 checks the file for viruses. If the return value is not null, the application will consider that a virus has been found.*

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 in the package "com.openkm.plugin.automation.action".***
- *The new Automation Action class **must be annotated with "@PluginImplementation"**.*
- *The new Automation Action class **must extend "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 the application plugin system will try to load it from there.*

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

*Do not omit the tag **@PluginImplementation**; otherwise, the application plugin system will not be able to retrieve the new class.*

More information about registering a new plugin.

Method descriptions

Method	*Type*	*Description*
executePre(Map<String, Object> env, Object... params)	*void*	*The method executed by the Automation event when the validation conditions succeed, at the "pre" stage.*
executePost(Map<String, Object> env, Object... params)	*void*	*The method executed by the Automation event when the validation conditions succeed, at the "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 the 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>*	*Returns a list of events.* *The method returns a list of valid events for the action in the pre stage.*
getValidEventsAtPost()	*List<EnumEvents>*	*Returns a list of events.* *The method returns a list of valid events for the action in the post stage.*

Understanding the env variable

The env variable is present in the isValid method; it is a Map of values injected by automation. These map values provide information about the node involved in the event and other related information.

> *It takes some time until you become familiar with the AutomationUtils class, because its methods to retrieve data are centralized.*
>
> *Take a look at Variables by automation event to see which variables are available for each event.*

For example, to retrieve the node that caused the event.

```
NodeBase node = automationUtils.getNodeToEvaluate(env);
```

Understanding the params variable

The params variable is present in the isValid method; it is an array of Objects. This array is filled by the Automation event based on the number of parameters set on the Action class (which will not be Automation.PARAM_TYPE_EMPTY).

The array can have several Objects with distinct types; it is necessary to retrieve the parameters as the correct object type.

> *It takes some time until you become familiar with the AutomationUtils class, because its methods to retrieve parameters are centralized.*

For example, to retrieve the first parameter as a 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 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 - optical mark recognition - will be required.*

Description of available Src values:

Property	*Description*

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

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 - optical mark recognition -.*

Detecting recursion

When you create your automation actions, you should consider recursion cases. For example, if your action is linked to the "CREATE_DOCUMENT" event and the action creates a new document, you can enter an infinite loop. For this reason, when you create a new Action you should take special care with these cases.



Basic recursive samples:

- *Automation task linked with "**Create document**" event when an **action** that is set also **creates a new document**.*
- *Automation task linked with "**Set metadata group**" event when an **action** that is set also **changes metadata group values**.*

Complex recursive sample:

- *Automation task linked with "**Create document**" event when an action that **changes metadata group values** is set, and an automation task linked with "**Set metadata group**" event when an action that **creates a new document** is set.*

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 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 the application plugin*

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

system will try to load it from there.

> *Do not omit 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 whether conditions are true or false.*
getName()	*String*	*Sets the name that will be shown in the administrative 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*	*Sets 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>*	*Returns a list of events.* *The method returns a list of valid events for pre-validation.*
getValidEventsAtPost()	*List<EnumEvents>*	*Returns a list of events.* *The method returns a list of valid events for post-validation.*

Understanding env variable

The env variable is present in the isValid method; it is a Map of values injected by automation. These map values provide information about the node involved in the event and other related information.

> *It takes some time to get familiar with the AutomationUtils class because the methods for retrieving data are centralized there.*
>
> *Take a look at Variables by automation event to see which 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 params variable is present in the isValid method; it is an array of Objects. This array is filled by the Automation event based on the number of parameters set on the Validation class (which will not be Automation.PARAM_TYPE_EMPTY).

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

The array can have several objects of distinct types; it is necessary to retrieve the parameters as the correct object type.

> *It takes some time to get familiar with the AutomationUtils class because the methods to retrieve parameters are centralized there.*

For example, to retrieve the first parameter as a 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;
```

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

```
@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 "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 plugin 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	Returns the name of the crontab.
getCronExpression	String	Returns the cron expression.
isRunning	Boolean	Returns true when the cron task is running.
execute	Void	Executes 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());
        }
    }
}
```

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

Understanding the crontab job execution context

Execution context:

- *Crontab job is executed by internal users named "system" (**the job does not have 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 in**, the call to the API uses a **null token value**. That indicates the application must use the logged-in session.*
>
> *When **a user is not logged**, the call to the API must use a **non-null token**; otherwise, it 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");*

It is good practice to use the system token when calling API methods in a 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 OpenKM version 7.1.41 and 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 in 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 "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's plugin system will try to load it from there. See the sample below:*

Do not omit the tag ***@PluginImplementation****, otherwise the application's plugin system will not be able to retrieve the new class.*

For more information: Register a new plugin

Method description

Method	*Type*	*Description*
autocompleteFormValues(String uuid, String grpName, List<FormElement> formElementList)	*void*	*The method autocompletes the group's form elements.*

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 "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's plugin system will try to load it from there. See the sample below:*

*Do not omit the tag **@PluginImplementation**; otherwise, the application's 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 default 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 in the package "com.openkm.plugin.conversion".***
- *The new Document Converter class **must be annotated with "@PluginImplementation"**.*
- *The new Document Converter class **must extend 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 the application plugin system will try to load it from there.*



*Do not omit 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 descriptions

Method	*Type*	*Description*
makeConversion(File input, String mimeType, File output)	*void*	*The method converts the input file to a specific format and writes it to the output file.*

getSourceMimeTypes()	*List<String>*	*Returns a list of valid document MIME types that are allowed to be converted.*
getConversionType()	*int*	*Returns the value of the MIME type to which the input document will be converted.* *Allowed values are:* • *Converter.TO_PDF* • *Converter.TO_IMAGE* *When you choose Converter.TO_IMAGE, it is expected to return 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 validators are used to implement complex logic that must be evaluated on the server side and are not covered by the 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 "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 the application plugin system will try to load it from there.*



*Do not omit 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(String value, List<String> uuids)	*String*	*When the returned string equals "", the application considers everything correct; otherwise, the returned text is the error.* *The value variable contains the value of the metadata field, and the uuids are the list of UUIDs of the affected nodes (documents, folders, mails, records).* *If only one node will be updated (the common case), the list will contain only one uuid. If multiple nodes are selected, it will contain the list of all selected nodes to which the metadata changes should be applied.*

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, two identical comments will not be allowed in the metadata field named okp:technology.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 when checking if a node can be promoted to a record.

The application checks if a node can be promoted to 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 to a record.

Form validators are used to implement complex logic that must be evaluated on the server side and are not covered by the 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 "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 the application plugin system will try to load it from there.*

*Do not omit 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, an exception of type ValidationFormException should be thrown.* *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 to 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), which allows you 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 that will not be kept in the database (requires filtering in add and set stages)*
    4. *Apply a 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 "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 plugin system will try to load it 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 setting metadata.*

Example of the Form Validator implementation

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.14//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.14.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting" interceptorClassName="com.openkm.plugin.form.interc
    <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 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 omit 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 PageInfo object 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 PageInfo object 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 column is 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, orderByFiel
   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 it in **Profile** > **General** > **Pagination** > **Pagination plugins**. After that you will see an icon in the file browser which can be used to choose between the default paginator and the new one.*

Creating your own NodeProperties plugin (customize search result objects)

NodeProperties is usually used in Webservices scenarios. Using it, you can customize the result objects from a search query. Usually, as a result of some search query, you only need to use a few variables of the returned objects rather than the complete ones. With large query results, returning less data gives you better performance (less time to fill data on the server side and less time for marshalling and unmarshalling webservices procedures).

You can create your own NodeProperties.

Conditions:

- *The new NodeProperties class* ***must implement the "NodeProperties"*** *interface.*
- *The new NodeProperties class* ***must be declared in the package "com.openkm.plugin.properties".***
- *The new NodeProperties class* ***must be annotated with "@PluginImplementation".***
- *The new NodeProperties class* ***must extend "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

    /**
```

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

```
    * 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 plugin system will try to load it from there.*



*Do not omit 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***	*Defines which values are loaded in the Document object returned by the search engine.*
getProperties(String user, NodeFolder nFolder)	***Folder***	*Defines which values are loaded in the Folder object returned by the search engine.*
getProperties(String user, NodeMail nMail)	***Mail***	*Defines which values are loaded in the Mail object returned by the search engine.*
getProperties(String user, NodeRecord nRecord)	***Record***	*Defines which values are loaded in the Record object returned by the 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 in 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 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 the application plugin system will try to load classes from there.*



Do not forget 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 descriptions

Method	*Type*	*Description*
parse(OCRTemplateControlField	*boolean*	*When it returns true, it indicates that the field complies with the*

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

otcf, String text)		*conditions.* *More information about OCR Template stages at: OCR templates.*
getName()	*String*	*Sets the name that will be shown in the administrator's user interface selector list.*
isPatternRequired()	*boolean*	*When it returns true, it indicates that the pattern field is required.* *It will show an error - 'missing pattern' - in the administrator user interface if a pattern is required and the user has not set one.*
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 in 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 "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 the application plugin 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 descriptions

Method	Type	Description
parse(OCRTemplateField otf, String text)	*Object*	*Returns the captured data value as an Object.* *Allowed Object classes:*

		• *String* • *Calendar (will be converted to an ISO8601 string)* • *Other (will be converted to a string with String.valueOf(object)).*
getName()	*String*	*Returns the name that will be shown in the administrator user interface selector list.*
isPatternRequired()	*boolean*	*If it returns true, the pattern field is required.*  *An error - "missing pattern" - will be shown on the administrator user interface if the pattern is required and the user has not set it.*

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 in 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 "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 the application plugin system will try to load it from there.*



Do not omit 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 descriptions

Method	Type	Description
getName()	*String*	*Returns the name that will be shown in the plugins table.*
getOptions()	*List<Option>*	*Returns a list of 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.form.Option;
import com.openkm.core.DatabaseException;
import com.openkm.db.bean.DbUser;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.AuthException;
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<DbUser> userList = okmAuth.getUsers(null, true);
      for (DbUser user : userList) {
        Option option = new Option();
        option.setLabel(user.getName());
        option.setValue(user.getId());
        options.add(option);
      }
    } catch (AuthException | DatabaseException 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 as of version 8.1.12.

Principal adapters are used by the authentication module to retrieve user and role information. For example, to retrieve users and roles from an LDAP server or an external database that stores all the company's 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 in the package "com.openkm.plugin.principal".***
- *The new Principal Adapter class* ***must be annotated with "@PluginImplementation".***
- *The new Principal Adapter class* ***must extend "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 the application plugin system will try to load it from there.*



*Do not omit the tag **@PluginImplementation**; otherwise, the application plugin system will not be able to locate the new class.*



More information is available at Register a new plugin.

*To **enable the new Principal adapter**, go to **Administration > Configuration parameters** > find the parameter named **principal.adapter** and set its value to **com.openkm.plugin.adapter.OwnPrincipalAdapter**.*

*For this change to take effect, we need to **restart the application**.*

Method descriptions

Method	*Type*	*Description*
getUsers()	*List<CommonUser>*	*Returns the list of all users.*
getRoles()	*List<String>*	*Returns the list of all roles.*
getUser(String userId)	*CommonUser*	*Returns all user data.*
getUsersByRole(String role)	*List<CommonUser>*	*Returns the list of all users who have been assigned a role.*
getRolesByUser(String user)	*List<String>*	*Returns the list of all roles assigned to a user.*

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

String getPassword(String user)	*String*	*Returns the password associated with 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.*  *The 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*	*Assign a role to a user.*
removeRole(String user, String role)	*void*	*Revoke a role from a user.*
getProfiles(boolean filterByActive)	*List<Profile>*	*Returns the list of all profiles.*
getUserProfile(String userId)	*Profile*	*Returns the profile assigned to a user.*
setUserProfile(String userId, long profileId)	*void*	*Change the profile assigned to a user.*
isManageUsers()	*boolean*	*Returns a boolean that indicates whether the user is a manager.*
		Returns a boolean that indicates whether roles are

isManageRoles()	*boolean*	*managed.*

Example of the Principal adapter implementation

The example retrieves users, roles and emails 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");
}
```

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

```
    @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 plugins 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 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 plugin system will try to load from there.*

*Do not omit 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 descriptions

Method	*Type*	*Description*
getFormElements()	*List<FormElement>*	*The method returns a list of FormElement objects that will be rendered in the UI to request report filter values.*
execute(Map<String, String> params)	*PageInfo*	*Returns a list of elements related to the report.* *The params parameter contains the values of the form elements set by the user in the UI.*
getName()	*String*	*Returns the name of the report.*

Example of the Report plugin implementation

In this example, we return a list of documents filtered by document name (provided as an input value) and limited by a results limit (also provided as an input value). If the limit is empty, the default is 100. The list shows only three fields related to each 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);
```

```
    }

    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 the REST API)

The REST Plugin feature helps you extend 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 "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 in the package com.openkm.plugin.rest** because the application plugin system will try to load it from there.*

*Do not forget 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 and returns 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 that returns a file

> *The sample below works in combination with the REST web service "/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 - marshalled - by OpenKM. Basic types like String, Integer, etc. can be used directly, but for complex objects you need to use annotations on classes.

Marshalling and unmarshalling strategies:

- *The result of the executed method is always a String. Internally, the method converts complex objects to JSON.*
- *The result of the executed method 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 client side you must convert the JSON string to an object:

```
// value contains the returned String value by the REST call
```

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

```
SomeObject someObject = new Gson().fromJson(value, SomeObject.class);
```

Complex Object scenario

You can use existing OpenKM classes like Document, Folder, Record, Mail among others, because these classes already have the annotations.

> *Use the annotation (in the sample below the object will be marshalled with the 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 the low-level API, you should always ask OpenKM IT staff before using it in production

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

to understand the benefits and restrictions of this kind of implementation.

The next sample is a project that uses a low-level API implementation for quick uploads and setting metadata at the same time.

The sample comes with a plugin implementation and a client plugin test.

If you have a large upload process and are looking for better 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 in 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 in the com.openkm.plugin.search package** because the plugin system will try to load it from there.*

*Do not omit the tag **@PluginImplementation**; otherwise, the plugin system will not be able to detect the new class.*

More information about Register a new plugin.

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

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 OpenKM to check a user's security access level for 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 "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 the application plugin system will try to load it from there.*



Do not omit the tag ***@PluginImplementation****; otherwise, the application plugin system will not be able to retrieve the new class.*



More information is available 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 set the value to* ***com.openkm.plugin.access.OwnSecurityManager***

For this change to take effect, we need to ***restart the application****.*

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

Method descriptions

In the following table, the permission values 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***	*Returns true when the user in the session has the permission.*
isGranted(NodeBase node, String user, int permissions)	***Boolean***	*Returns true when the user has the permission.*
isGranted(NodeBase node, long nodeClass, int permissions)	***Boolean***	*Returns true when the user in the session has the permission.* *This method only makes sense when the 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;
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 in version 8.1.11

The StackMetadata class implements the logic that takes control of what is shown in the metadata stack - tree nodes on the left panel - and their objects (documents, nodes, folders, records) on the 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 "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 the application plugin system will try to load it from there.*

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



*Do not omit 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 descriptions

Method	*Type*	*Description*
getName()	*String*	*Returns the name shown in the plugins table.*
isPropertyGroupHidden(String pgName)	*boolean*	*Returns whether metadata fields are hidden.*
getProperties(String fldPath)	*Folder*	*Returns folder properties.*
getChildren(String fldPath)	*List<Folder>*	*Returns a list of folders.* *When a folder node is chosen from the tree (left panel), this method is called to retrieve child folders. These folders are displayed in the tree as child nodes.*
getDocumentMetadataChildren(String fldPath)	*List<Document>*	*Returns a list of documents.* *When the folder node is chosen from the tree (left panel), this method is called to retrieve child documents. These documents are displayed in the browser explorer (right panel).*
getRecordMetadataChildren(String fldPath)	*List<Record>*	*Returns a list of records.* *When a folder node is chosen from the tree (left panel), this method is called to retrieve child records. These records are displayed in the browser explorer (right panel).*
getMailMetadataChildren(String	*List<Mail>*	*Returns a list of mails.*

fldPath)		*When a folder node is chosen from the tree (left panel), this method is called to retrieve child mails. These mails are displayed in the browser explorer (right panel).*
getFolderMetadataChildren(String fldPath)	*List<Folder>*	*Returns a list of folders.* *When a folder node is chosen from the tree (left panel), this method is called to retrieve child folders. These folders are displayed in the 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 in the package "com.openkm.plugin.form.suggestion".***
- *The new Suggestion Analyzer class **must be annotated with "@PluginImplementation".***
- *The new Suggestion Analyzer class **must extend 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 the application plugin system will try to load it from there.*



*Do not omit the tag **@PluginImplementation**; otherwise, the application plugin system will not be able to retrieve the new class.*



More information is available at Register a new plugin.

Method descriptions

Method	*Type*	*Description*
getSuggestions(String nodeUuid, String nodePath, Select sel)	*List<String>*	*Returns a list of suggested words to tag the document.* *The method needs the document UUID (nodeUuid) and the document path (nodePath).* *Select can be used as a collection of possible*

		values (an enclosed dictionary).

Example of a 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 in the package "com.openkm.plugin.form.values".***
- *The new Suggestbox class **must be annotated with "@PluginImplementation".***
- *The new Suggestbox class **must extend "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 the application plugin 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 descriptions

Method	*Type*	*Description*
getName()	*String*	*Returns the name that will be shown in the plugins table.*

getSuggestBoxValuesByFilter(String filterValue)	*List<KeyValue>*	*Returns a list of KeyValue objects.*
getExactKeyValueByClassName(String value)	*KeyValue*	*Returns a KeyValue object.*

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 in the package "com.openkm.plugin.extractor".***
- *The new Text Extractor class **must be annotated with "@PluginImplementation"**.*
- *The new Text Extractor class **must extend "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 the application plugin system will try to load it from there.*

> *Do not omit the tag **@PluginImplementation**; otherwise, the application plugin system will not be able to retrieve the new class.*

> *More information is available at Register a new plugin.*

Method descriptions

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 the document version number each time the document is updated.



Keep in mind that changing the version number adapter when the application already has document versions means you sometimes must migrate older version numbers to the new format, or a new implementation must take older ones into consideration.

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 "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 the application plugin system will try to load it from there.*



*Do not omit 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 to **com.openkm.vernum.OwnNumerationAdapter***

*For this change to take effect, you need to **restart the application**.*

Methods description

Method	*Type*	*Description*
getInitialVersionNumber()	*String*	*Returns the first version of a new document.* *The first version is usually "1.0".*
getVersionNumberIncrement()	*int*	*Returns a number that indicates the available increase options.* *Available values:* • *VersionNumerationAdapter.INCREASE_DEFAULT* • *VersionNumerationAdapter.INCREASE_MINOR* • *VersionNumerationAdapter.INCREASE_MAJOR* • *VersionNumerationAdapter.INCREASE_MAJOR_MINOR* *INCREASE_NONE: Does not show any option; increases by default.* *INCREASE_MINOR: Only shows the option to increase the minor version; if not selected, it increases by default.* *INCREASE_MAJOR: Only shows the option to increase the major version; if not selected, it increases by default.* *INCREASE_MAJOR_MINOR: Shows the options to increase the major and minor versions; if not selected, it increases by default.*
String getNextVersionNumber(Session session, NodeDocument nDoc, NodeDocumentVersion nDocVer, int increment)	*String*	*Calculates 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 three types of template documents:

- *PDF*
- *LibreOffice*
- *Text (this includes other text-based formats 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 using the user input values, so you can use them for future searches.

The template syntax is used by the FreeMarker Template Engine. This engine can be used for everything from simple variable substitution to loops and other control structures.



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 an ODT file.*
- *Less time is required to create the document.*

Disadvantages in comparison with Text templates:

- *They produce larger documents.*
- *More difficulty maintaining the templates over 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 "." with "_". Here you can see the matching between the two 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.

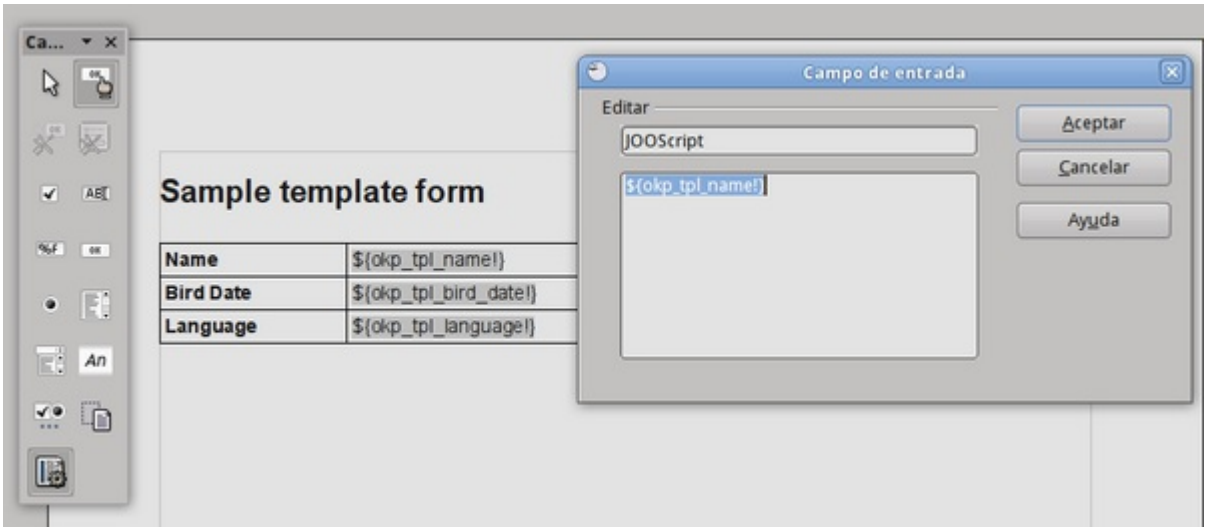
To create a template in LibreOffice go to ***View*** > ***Toolbars*** > ***Form Controls***.

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 Content field, enter the variable name (e.g. "$name"). This field will typically be displayed by Writer as a grayed rectangle with the variable name, and 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 internally, so you can take advantage of FreeMarker's formatting flexibility. For example, you can set date formats in this way:

```
${okp_tpl_birth_date?string("yyyy-MM-dd")}
```

If the date may be null, then you need to check this (in the case of ODT templates you should use #if between "[" and "]"):

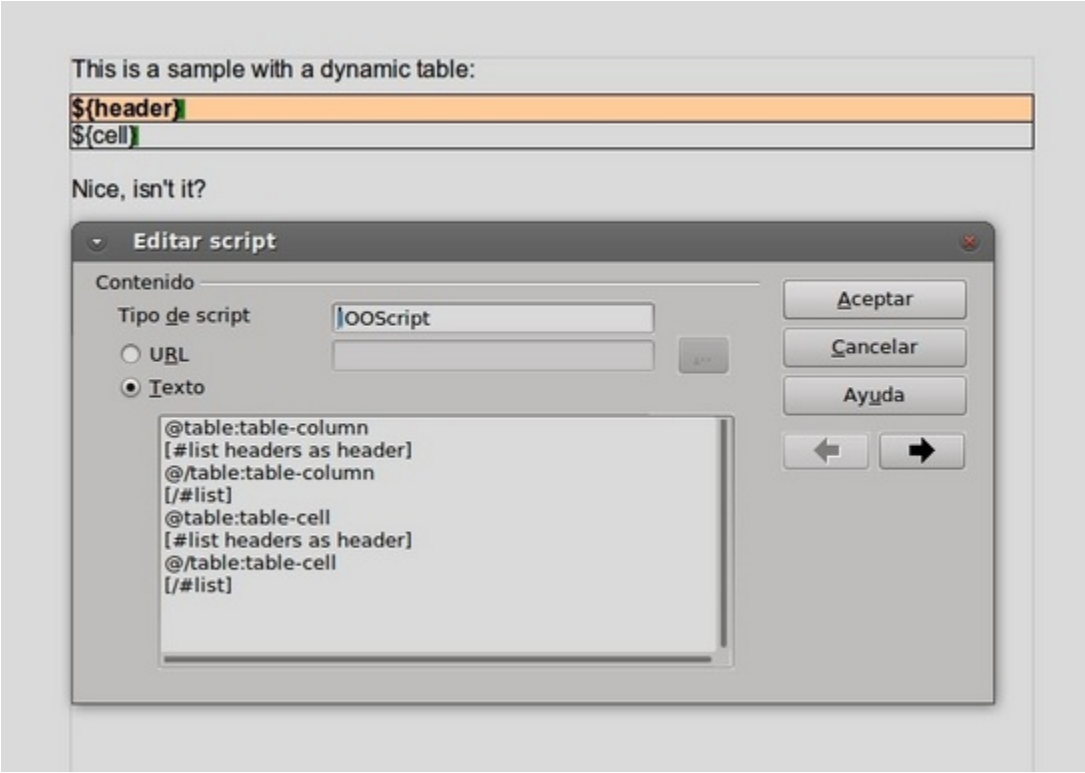
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

```
[#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. Let's see an example.



In the screenshot you can see a document with a table. The header has a colored background and is also 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** the **tpl.odt** file to **/okm:templates/documents***
- ***Register the 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 the tpl.odt file.*

Creating a PDF template

Advantages of using PDF templates:

- *Can produce PDF files.*

Disadvantages compared with text templates:

- *Produce larger documents.*
- *More difficulty maintaining the templates over time.*
- *More time is needed to create the document (especially when the PDF optimization configuration parameter is enabled).*

You can create PDF template forms using several applications:

- *Acrobat Pro*
- *Master PDF Editor*

Every PDF form field name needs to be unique; you cannot set the same property value for 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>
```

To establish a relationship between the template form fields and the property group, name them using a naming convention: replace ":" and "." with "_". 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 recommend 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 internally, so you can take advantage of FreeMarker's formatting features. 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.

Sample configuration

- ***Upload** the **tpl.pdf** file to **/okm:templates/documents***
- ***Register the metadata** definition **okg:tpl** in **Administration** > Metadata*
- ***Enable metadata** in **profiles** at **Administration** > Profiles*
- ***Add the metadata** group okg:tpl to the 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.*
- *Small size of the produced document.*
- *Easily maintain the templates over time.*
- *Less time required to create 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 create a relation between the template form field and the property group, you have to name them following a naming convention: replace ":" and "." with "_". 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 template. This is the template source code:

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

```
<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 missing 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, so you can take advantage of the FreeMarker formatting flexibility. 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.

Sample configuration

- ***Upload tpl.html** file to **/okm:templates/documents** with the contents described above.*
- ***Register metadata** definition **okg:tpl** at **Administration** > Metadata*
- ***Enable metadata** from **profiles** at **Administration** > Profiles*
- ***Add metadata** group okg:tpl to the 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 the 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 the **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 the **Execute button** at the top right.*



*The **type value** - CFG_VALUE column - **must be written in lowercase**!*

Sample

You can access it from source code via the properties.



When the property does not exist, it will be initialized with the default value (in the 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;

}
```

In the password validator class, you can use several configuration parameters:

- *Config.PROPERTY_VALIDATOR_PASSWORD to return the value of the parameter "validator.password".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_LENGTH to return the value of the parameter "validator.password.min.length".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MAX_LENGTH to return the value of the parameter "validator.password.max.length".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_LOWERCASE to return the value of the parameter "validator.password.min.lowercase".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_UPPERCASE to return the value of the parameter "validator.password.min.uppercase".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_DIGITS to return the value of the parameter "validator.password.min.digits".*
- *Config.PROPERTY_VALIDATOR_PASSWORD_MIN_SPECIAL to return the value of the parameter "validator.password.min.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 uses the JasperReports library to generate reports. To build reports, you can use a tool named iReport.

*The **JasperReports Library** is the world's most popular open-source reporting engine. It is entirely written in Java and can use data from any kind of data source to 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. It lets you 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 https://community.jaspersoft.com/.*

In this section you will find an introduction to creating reports with JasperReports and the iReport tool. Consider this a quick introduction, not a complete guide. You should follow tutorials and read books or papers for more detailed information.

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 scripting. For beginners, we suggest starting with SQL-based reports.

- *Introduction to reports based on SQL.*
- *Introduction to reports based on Java.*
- *Reports with parameters.*

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

Introduction to reports based on SQL

- *Install the **iReport** tool.*
- *Execute the **iReport** tool.*



OpenKM comes with a specific JasperReports Library engine that supports older versions but not newer ones. In the Application version compatibility table, you can find a correspondence list between OpenKM and JasperReports Library engine versions.

You can set a specific output version in the iReport tool:

- *Click on **Tools menu** > **Options**.*
- *From **iReport** > **General tab**, click on **Compatibility tab**.*
- *Set the **Jasper version compatibility**.*
- *Click on **OK button**.*

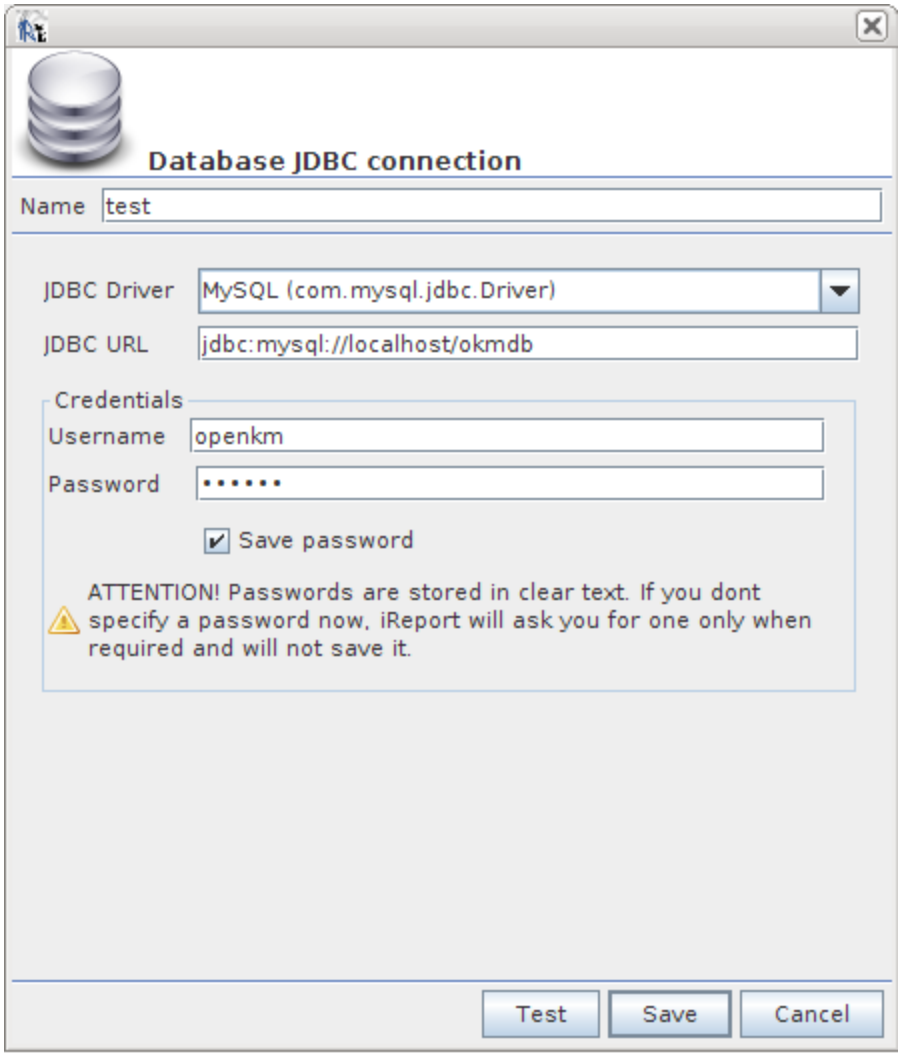
Create a data source

- *Click on the **Report data sources button**.*



- *Click on the **New button**.*
- *Choose the "**Database JDBC connection**" option.*
- *Click on the **Next button**.*
- *Set the **name**.*
- *Choose the **JDBC driver**.*
- *Click on the **Test button** to check the connection parameters.*
- *Click on the **Save button**.*

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

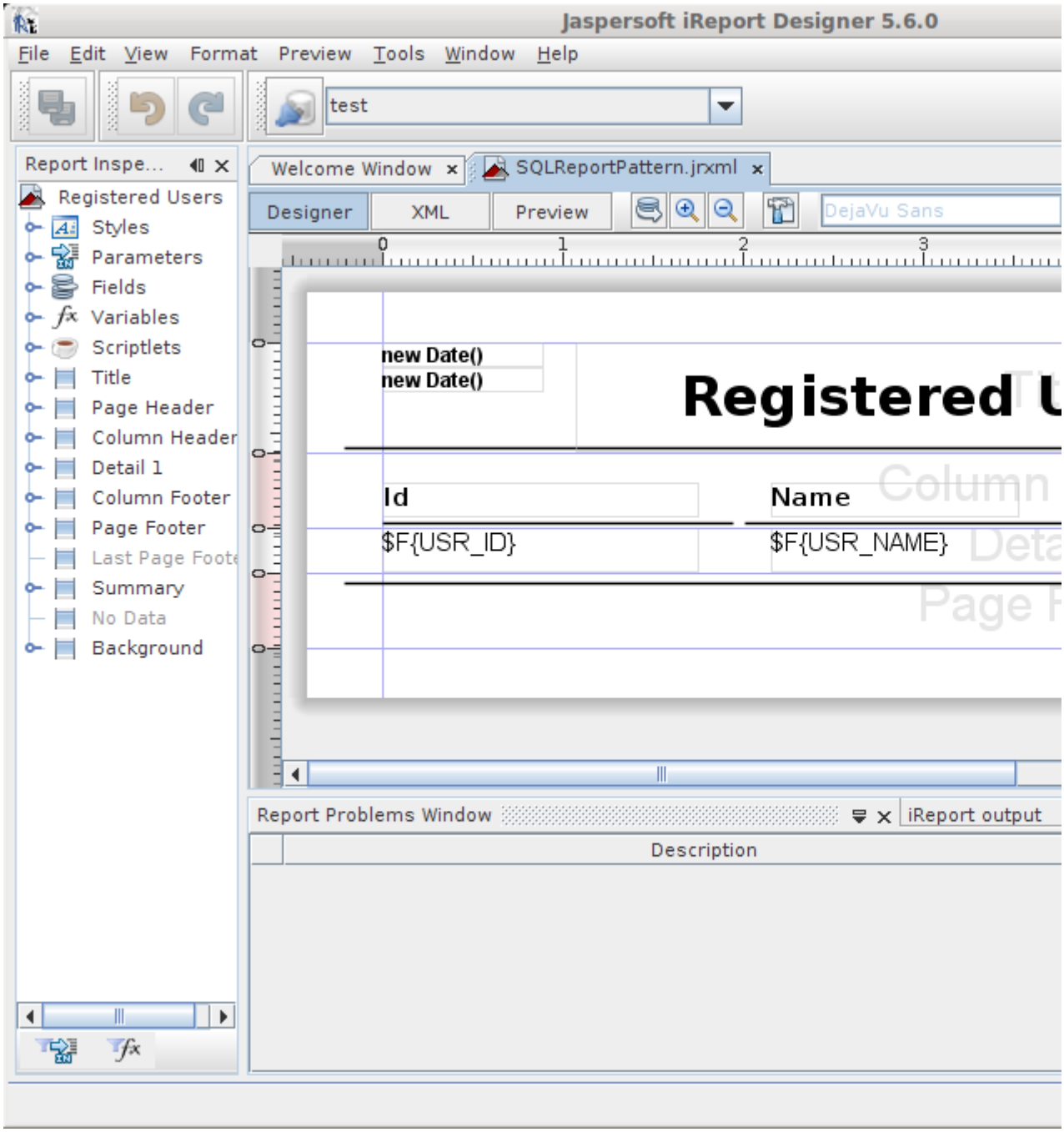


- *Check the **Set as default** checkbox.*
- *Click on the **Close button**.*

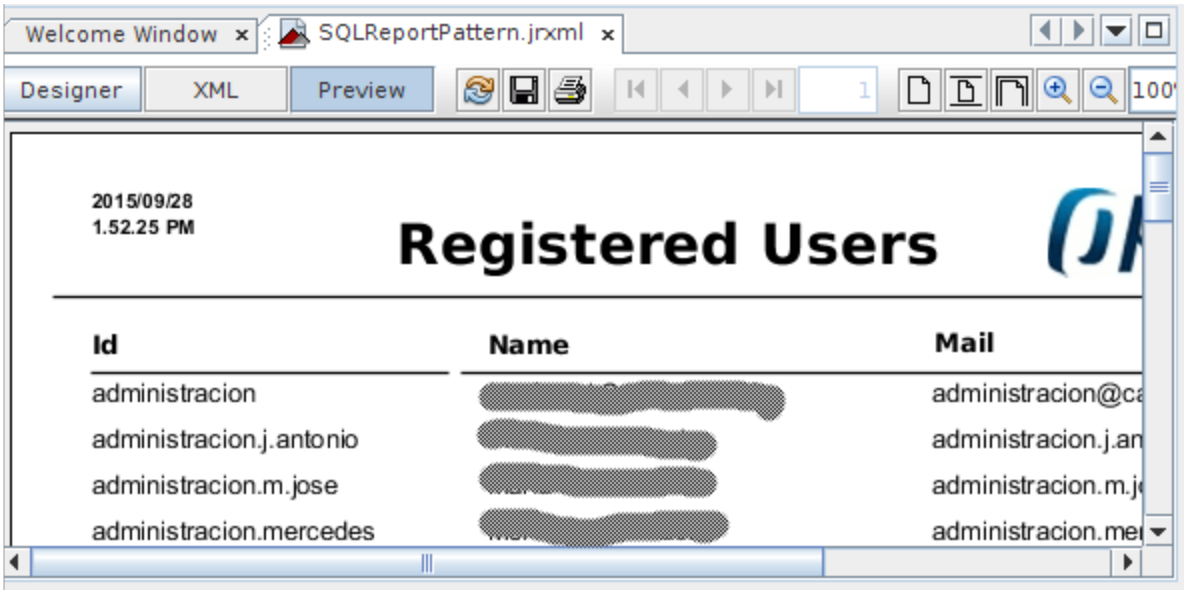
Testing the SQL sample report

> *To test the SQL report sample, you must have OpenKM running.*

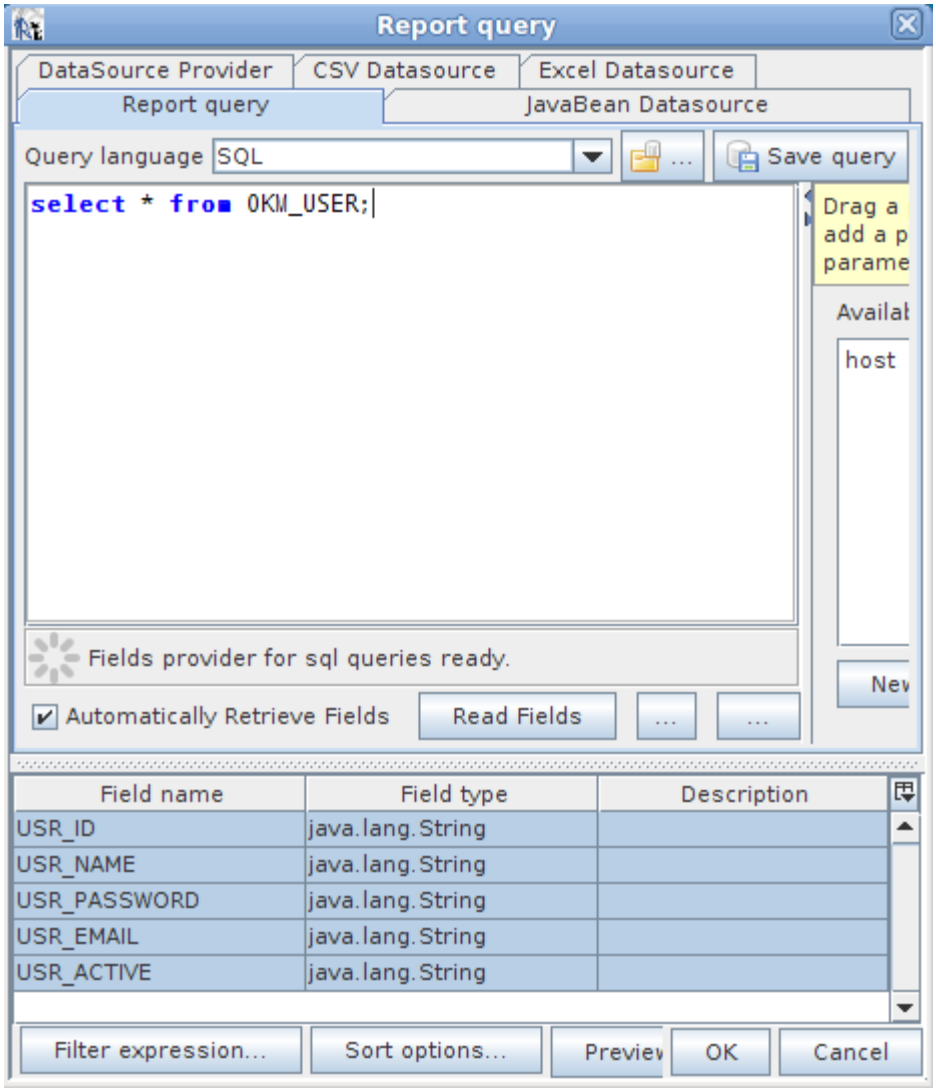
- *Download the SQLReportPattern.jrxml file.*
- *Open the file.*



- *Click on the **Preview tab**.*
- *Set the **host** value (for example http://localhost:8080/openkm).*



- *Click on the **Designer tab**.*
- *Click on*  *the **Report query button**.*



From this point, you can add your jrxml file and register it in OpenKM at ***Administration*** > *Reports*.

A closer look at the JRXML file

The iReport tool creates the Jasper report file in XML format for you.

Although the iReport tool does all the work for you in a friendly user interface, it is a good idea to take a look, at least out of curiosity, at the kind of file being created.

Query section

```
<queryString>
  <![CDATA[select * from OKM_USER;]]>
</queryString>
```

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.

Introduction to reports based on Java

- *Install the **iReport** tool.*
- ***Execute the iReport** tool.*



OpenKM comes with a specific JasperReports Library engine that supports older versions, but a newer one can now be found. At Application version compatibility table you have a correspondence list between OpenKM and JasperReports Library engine versions.

You can set a specific output version from the iReport tool:

- *Click on **Tools menu** > **Options**.*
- *From **iReport** > **General tab**, click on **Compatibility tab**.*
- *Set the **Jasper version compatibility**.*
- *Click on **the OK button**.*

Testing the Java sample report

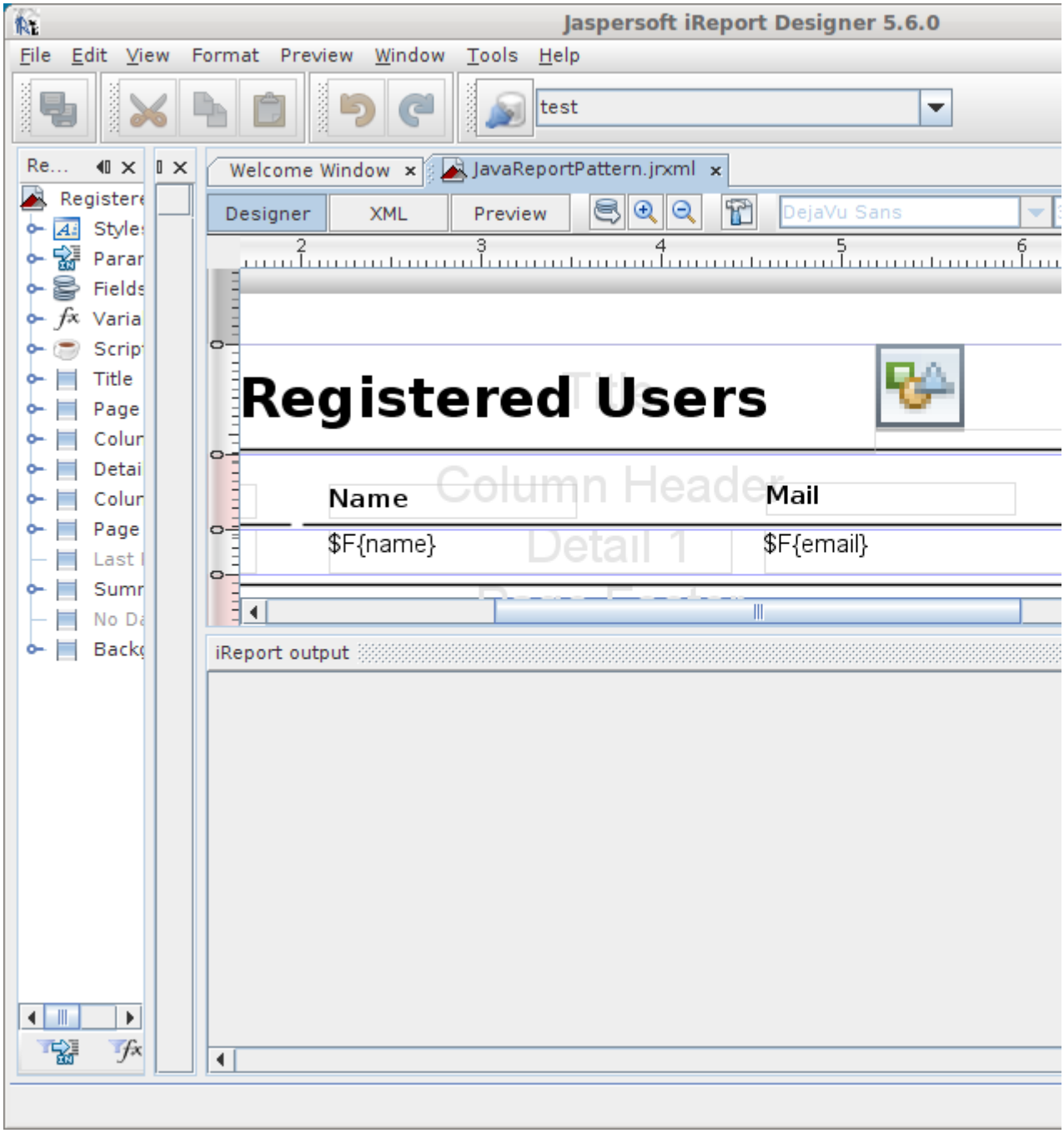


***Reports based on Java cannot be previewed directly from the iReport tool.** They must be checked from OpenKM.*

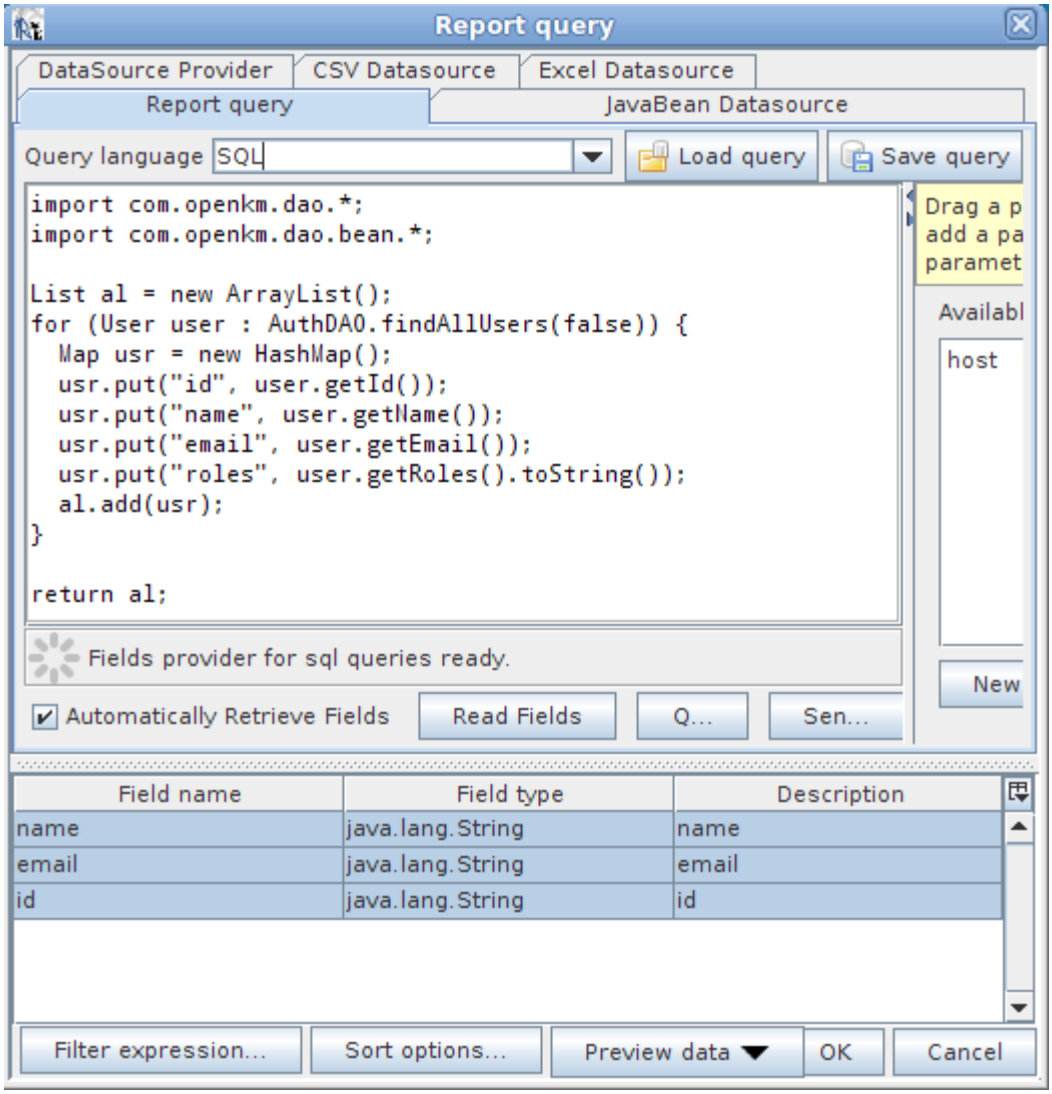
For testing the SQL Report sample, you must have an OpenKM running.

- *Download the JavaReportPattern.jrxml file.*
- *Open the file.*

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



- *Click on* ***the Designer tab***.
- *Click on*  ***the report query button***.



In this case, in place of an SQL query, you can set a Java Scripting (see also Scripting samples).

The script must return a List<Map> object where each value in the map corresponds to a column value.

From this point you can add your jrxml file and register it in your OpenKM at ***Administration*** *>* Reports.

Understanding the code sample

- *Create an empty list.*
- *Iterate through all users by calling the API method AuthDAO.findAllUsers(false).*
- *Create a Map for each User object where each value corresponds to a column.*
- *Add each map to 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);

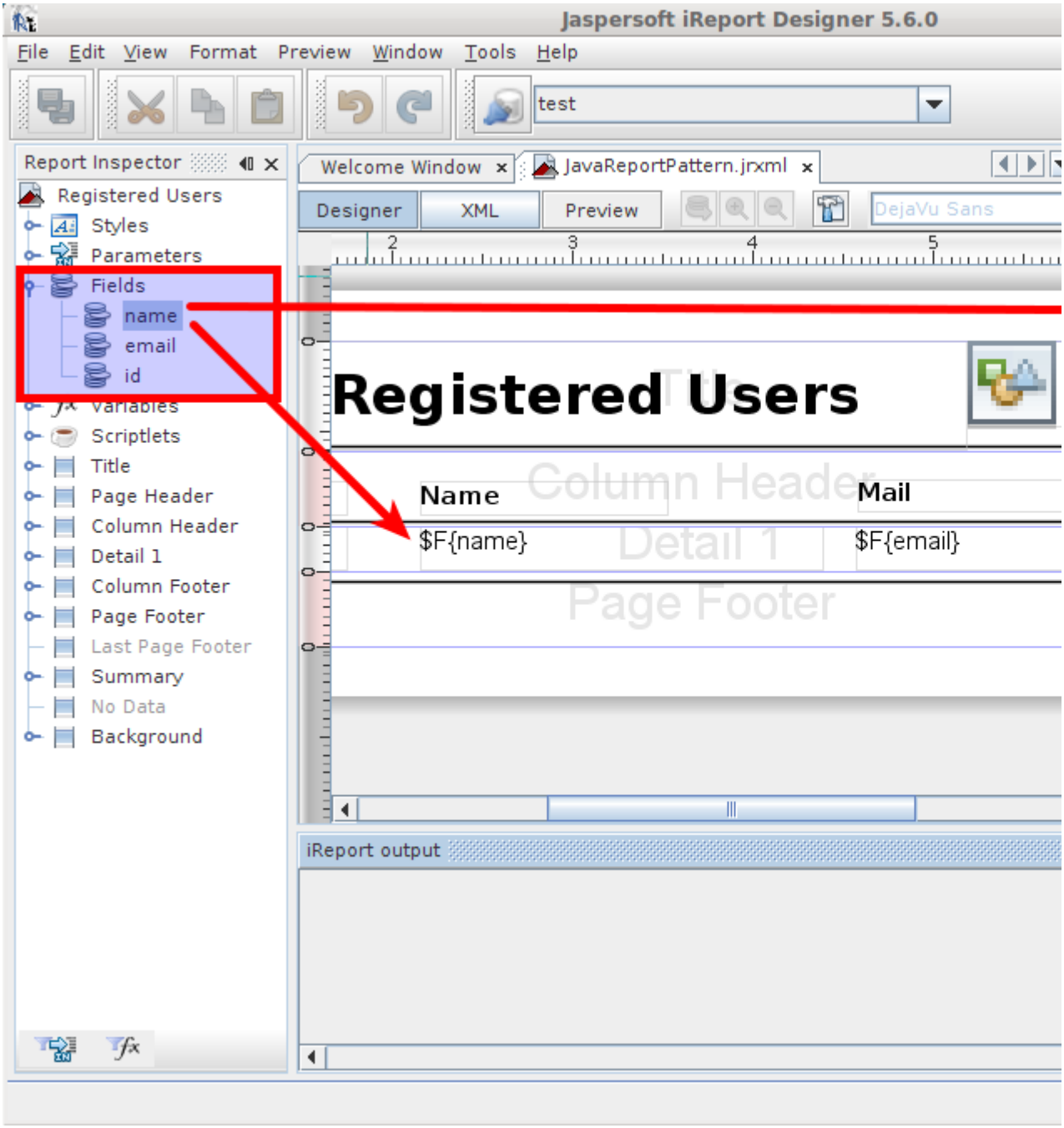
List al = new ArrayList();
for (CommonUser user : authSrv.findAllUsers(false)) {
 Map usr = new HashMap();
 usr.put("id", user.getId());
 usr.put("name", user.getName());
 usr.put("email", user.getEmail());
 usr.put("roles", user.getRoles().toString());
 al.add(usr);
}

return al;
```

Using in the report

- *For each key in the map, create a field.*
- *Use the field in the report.*

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.

Reports with parameters

When you execute a report from OpenKM, a form might be shown requiring the user to enter values for filtering purposes. The most common case is filtering by a range of dates.

*Reports that use form filtering end with the extension "**.rep**". These reports are actually ZIP files renamed to "**.rep**".*

These reports contain two files:

- *JRXML file*
- *A file named params.xml*

Download the UsersLogin.zip and take a look inside (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, two filtering fields have been set:

Field	Type	Description
from_date	**Input**	*Field used to filter the start date.*
to_date	**Input**	*Field used to filter the end date.*

The params.xml file:

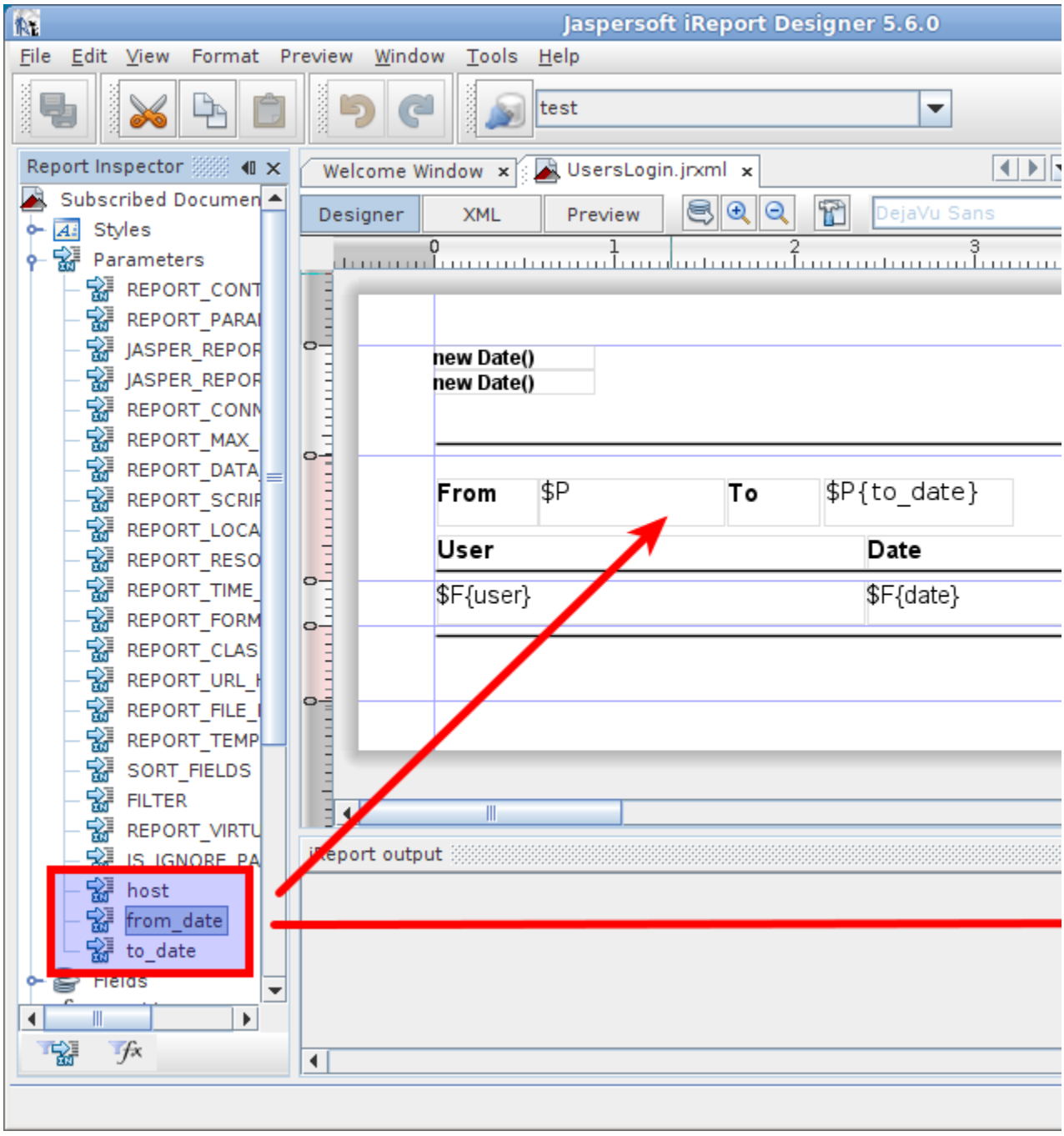
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <input label="From" name="from_date" type="date">
    <validator type="req"/>
  </input>
  <input label="To" name="to_date" type="date">
    <validator type="req"/>
  </input>
</report-parameters>
```

Using fields in the report

Two parameters are created in the report:

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

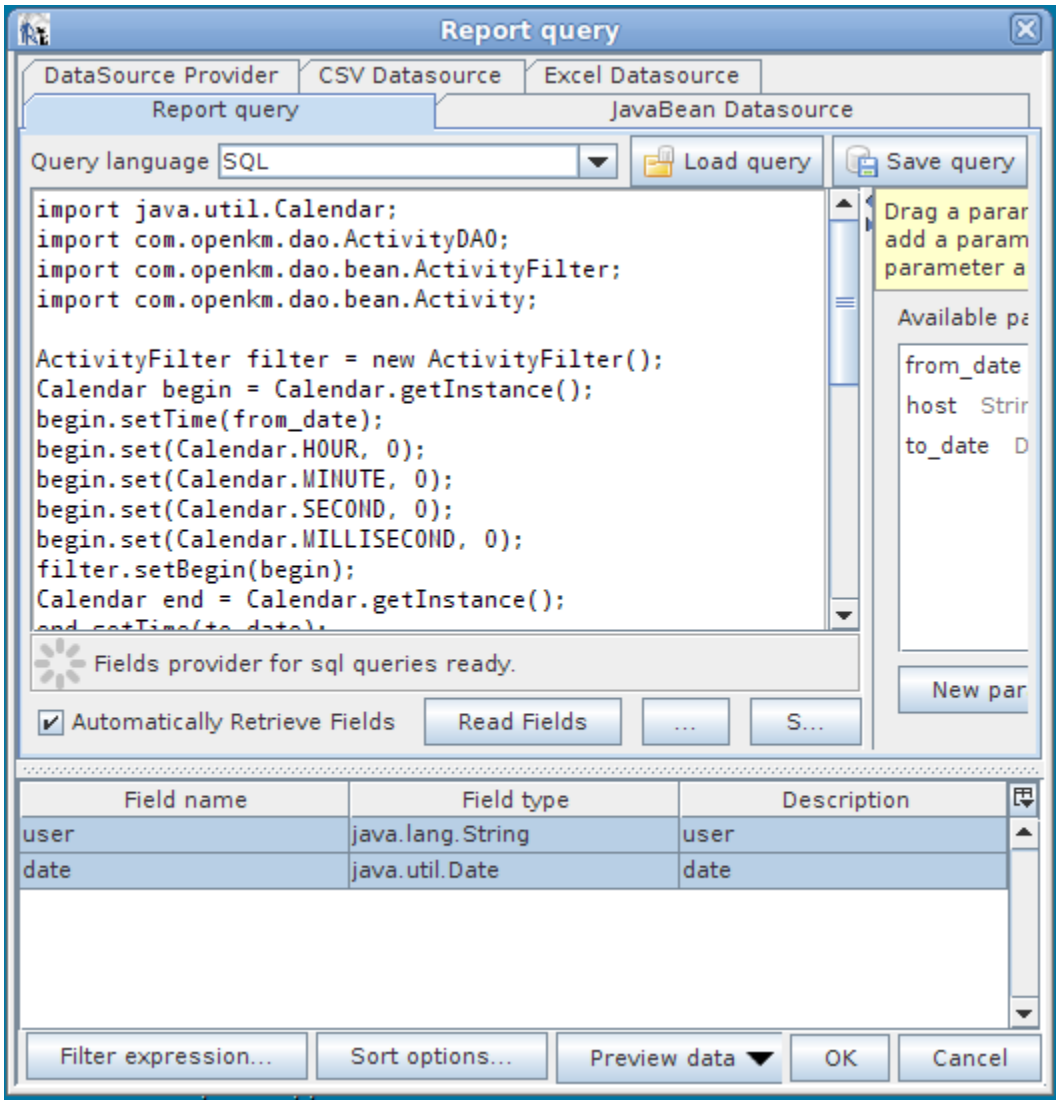
Parameter	*Class*	*Description*
from_date	***java.util.Date***	*Start date.*
to_date	***java.util.Date***	*End date.*



The parameters are also used in SQL or Java reports.

Java sample

- *Click on*  ***the report query button***.



The ***from_date*** *and* ***to_date*** *variables are injected into code execution and can be used directly, as in the sample below.*

```
import org.springframework.web.context.WebApplicationContext;
import com.openkm.db.service.ActivitySrv;
import com.openkm.db.bean.*;
import com.openkm.util.*;
import java.util.*;

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);
```

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

```
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);
}

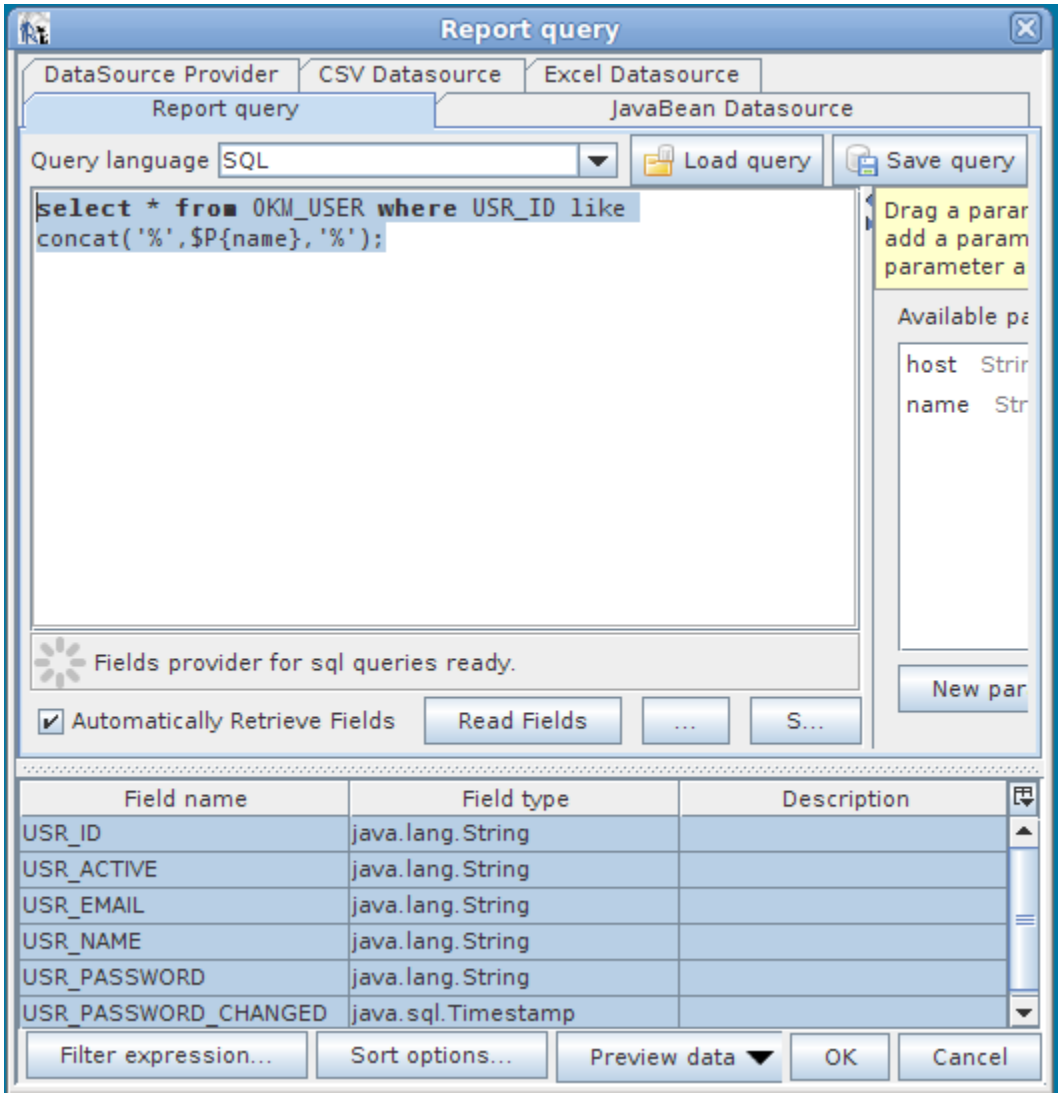
return al;
```

SQL sample



The sample below is not based on the UsersLogin.rep report.

Click on  ***the report query button***.



*In the sample, there is a parameter named "**name**" of class "**java.lang.String**" used for filtering the user list.*

```
select * from OKM_USER where USR_ID like concat('%',$P{name},'%')
```

The engine replaces the "$P{name}" tag with a 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 concat 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 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 the latest DTD on 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 a DTD. In the previous example, the DOCTYPE is **report-parameters-2.1.dtd**.*
- *The XML begins with the **<report-parameters> tag** and ends with **</report-parameters>**. Within this hierarchy the metadata groups are defined.*

Basic example

Available field types:

- *Checkbox*
- *Input*
- *Select*
- *Separator*
- *Suggestbox*
- *Text*
- *TextArea*

More information about field type elements is 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 a file system DTD

If the server has no access to the Internet, then the DTD file must be accessible in some way; for example, you can store a copy on your application server.



The XML definition must be changed to refer to the new DTD location; otherwise you will get an application error while trying to process the DTD file.

This can be a file system path (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.*

Parameter 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 child elements, validator elements, 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 provides the necessary and properly formatted information needed to successfully complete an operation.

The 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 is required "**req**".*
- *Value must be a decimal value "**dec**".*
- *The maximum length 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 that is not included in these predefined ones, you can use a regular expression. 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.

Params Input Field

In most cases, it is used to enter free text, although it can be used to save HTTP links, date values, 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 cannot have the same name in the same task definition. The name **must be unique**.*	*true*
type	*Type of value. Allowed types:* • *text.* • *date.* • *link.* • *folder.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the UI must use width values based on percentages, 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"?>
```

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

```
<!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>
</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, the 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 number of characters in the input to begin filtering 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 the Params Validator element.*

*** There are three options to get suggested box values (using **one of them is mandatory**):*

- *table + filterQuery + valueQuery*
- *filterQuery + valueQuery*
- *className*

No other combination 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 option values used in selects. That's very useful when you have a large number of values.*
- *OKM_DB_METADATA_TYPE stores the metadata table definition (table name, column name, and column type) in the table.*
- *OKM_DB_METADATA_VALUE stores record values.*

Create a 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 the value col00**.*

*The **country_name** is set as **column 1 with the value col01**.*

- *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_
```

- *Click the **Execute button** at the bottom right.*

Insert metadata table values:

- *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_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 the **Execute button** at the bottom right.*



The insert queries, depending on your database, may not match; 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 the 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** can be regular SQL queries.*



You should only use {0} as the parameter the user enters to filter the values from the table.

Create the 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 get a list from external applications or for other purposes, your class can be built to retrieve 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 same task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the UI must use width values based on percentages, 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 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 same task definition.* ***The name must be unique****.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the UI must use width values based on percentages (%), 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	*A unique field identifier.* *Two fields can't have the same name in the same 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, the 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 an SQL query is used to get the select options.*	*

Child elements:

- *Option. Pairs 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"/>** when a hierarchical relationship is set.*

- *Validator. For more information, see Params Validator element.*

** There are 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. You cannot combine several; 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 option values of "**childrenSelect**" have a parent relation with "**parentSelect**". It's defined by*

parentElement="parentSelect".

- *The option value "**benefits**" of the field "**childrenSelect**" has a parent value "**directives**" in the field "**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:

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

```
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 for 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 option values. In 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>
```

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

```
</report-parameters>
```

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

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 same task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the 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 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 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 same task definition. The name **must be unique**.*	*true*
width	*The width of the HTML element.* ⚠ *In the case of KCenter, the 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 in the $TOMCAT_HOME/openkm.properties file.

The parameter hibernate.statistics helps developers enable the Hibernate statistics. By default, this parameter is set to false.

```
hibernate.statistics=true
```

Use these parameters only for development and fine-tuning.

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

Web services

OpenKM exposes an API via Web services:

- *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 username and password to access them.

You can extend the REST API—more information at Creating your own REST Plugin (extending the REST API).

We encourage you to use our SDKs to access OpenKM via the REST API rather than building your own client. More information:

- *The SDK for Java.*
- *The SDK for .NET.*

*In most samples, you'll see parameters named "**docId**", "**fldId**", or "**nodeId**". The value of these parameters must be a valid node **UUID**.*

Example of a nodeId:

- *Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**".*

Almost all samples use the curl command; more information in 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 makes the process easier. Or you can use the **curl** command-line application. For example, you can list a folder's children:*

```
$ 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",
```


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

```
    "permissions": 31,
    "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 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>
```

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/users
```

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 of the OpenKM API, you can convert, for example, a document from Microsoft Word format to PDF, among other available conversion types.

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 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/openkm/rest/conversion/img/txt
```

REST plugin samples

With the REST plugin, you can extend the default REST API.

```
$ curl -u okmAdmin:admin -G \
    -H "Accept: application/json" \
    --data-urlencode "className=com.openkm.plugin.rest.TestRestPlugin" \
    --data-urlencode 'param={"key":"key1","value":"value1"}' \
    http://localhost:8080/openkm/rest/plugins
```

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

Swagger

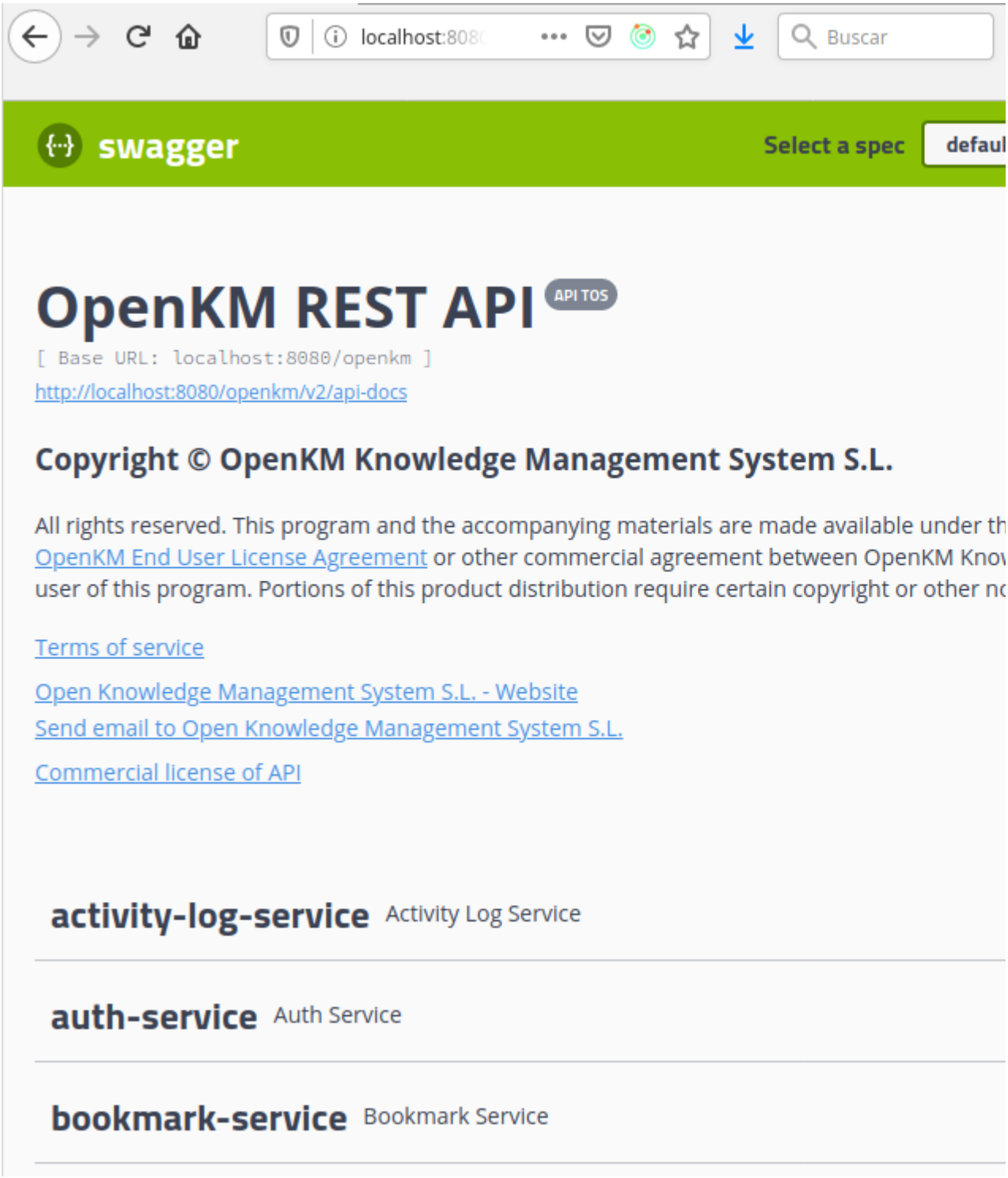
***Swagger** is an open-source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful Web services. While most users identify Swagger by the Swagger UI tool, the Swagger toolset includes support for automated documentation, code generation, and test case generation.*

How to access Swagger

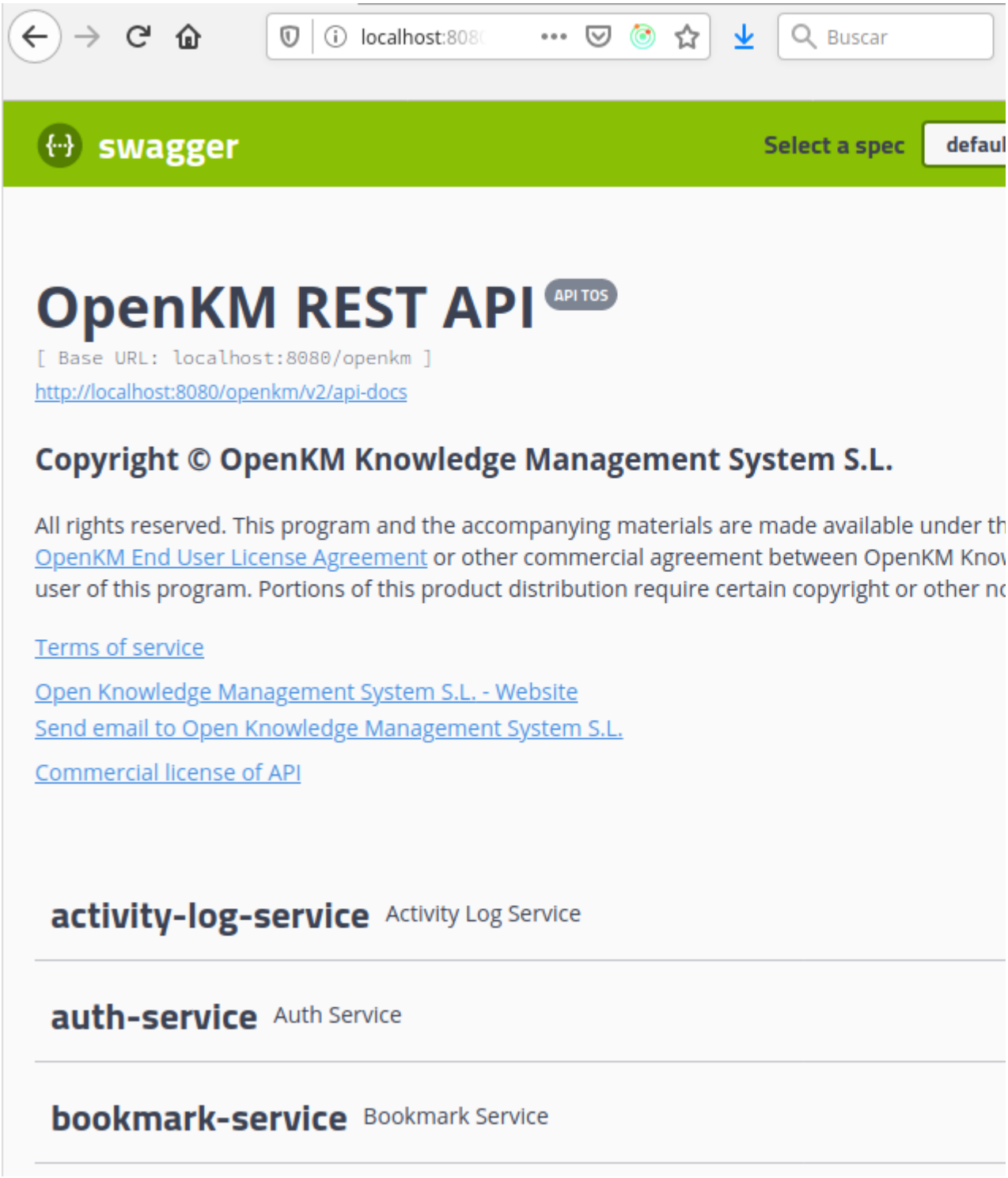
*In the **sample below, we use our OpenKM demo URL**. You **should use your OpenKM URL**.*

To access Swagger documentation, click the URL http://localhost:8080/openkm/swagger-ui.html It will show a screen like the one below asking for authorized user credentials:

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



Then the Swagger tool will be displayed:



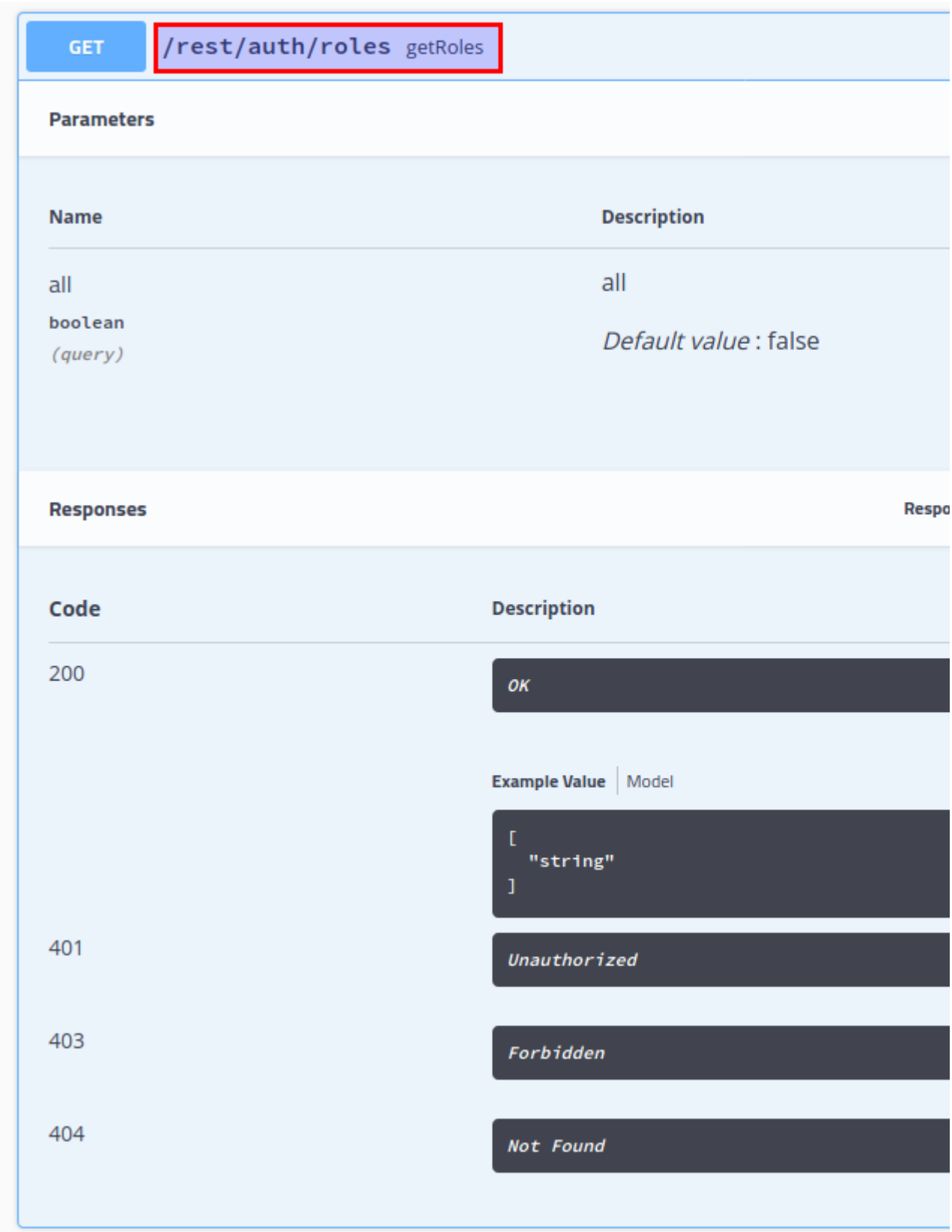
Playing with Swagger

*Let's do a **quick test** of the tool **checking the getRoles** method **from auth-service**.*

- ***Open the auth-service***



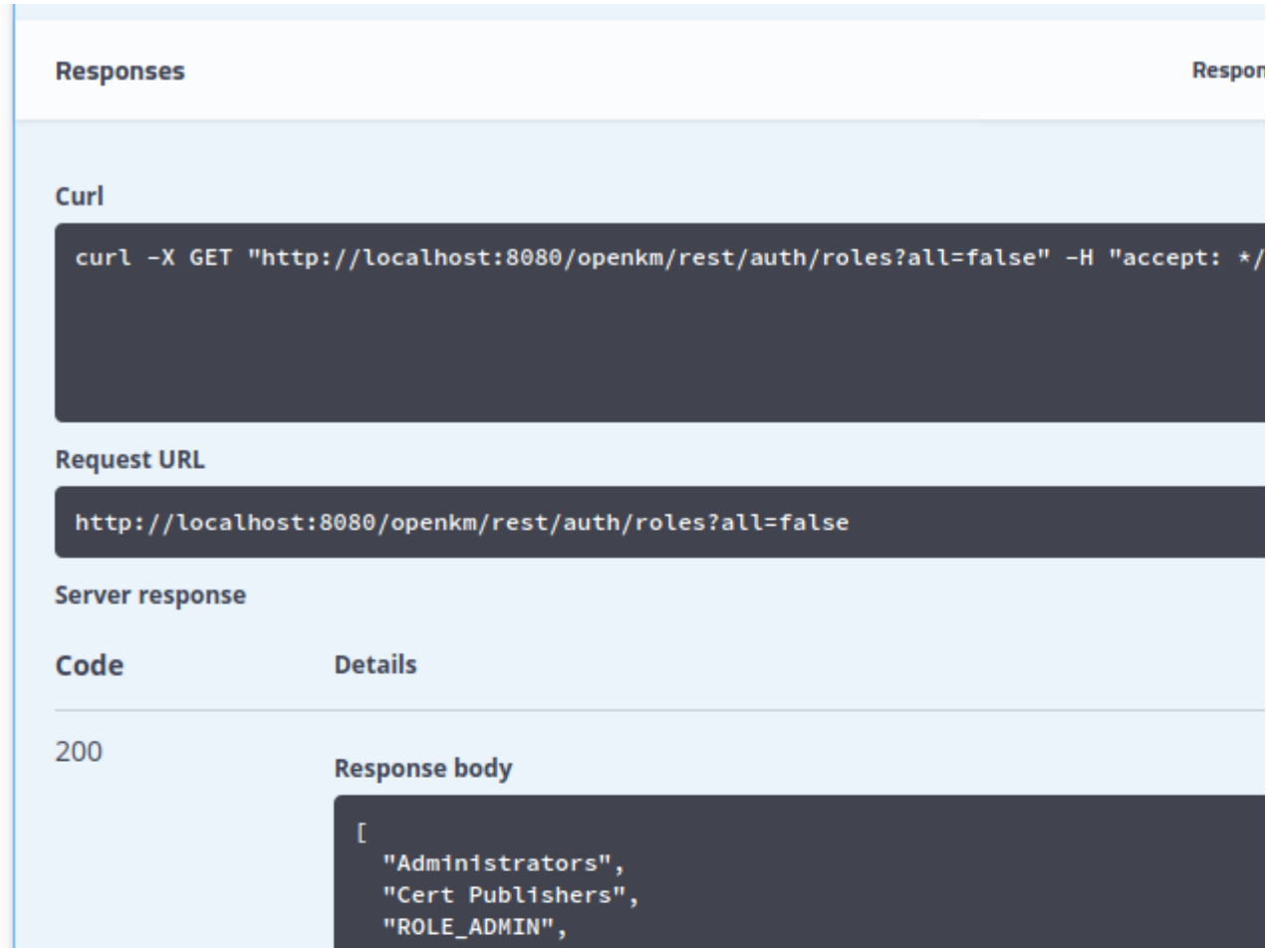
- ***Open the "roles"*** *method and* ***click on the button "Try it out"*** *:*



- ***Click on the "Execute" button.***



- *Finally, the results will be shown.*



```
  "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*

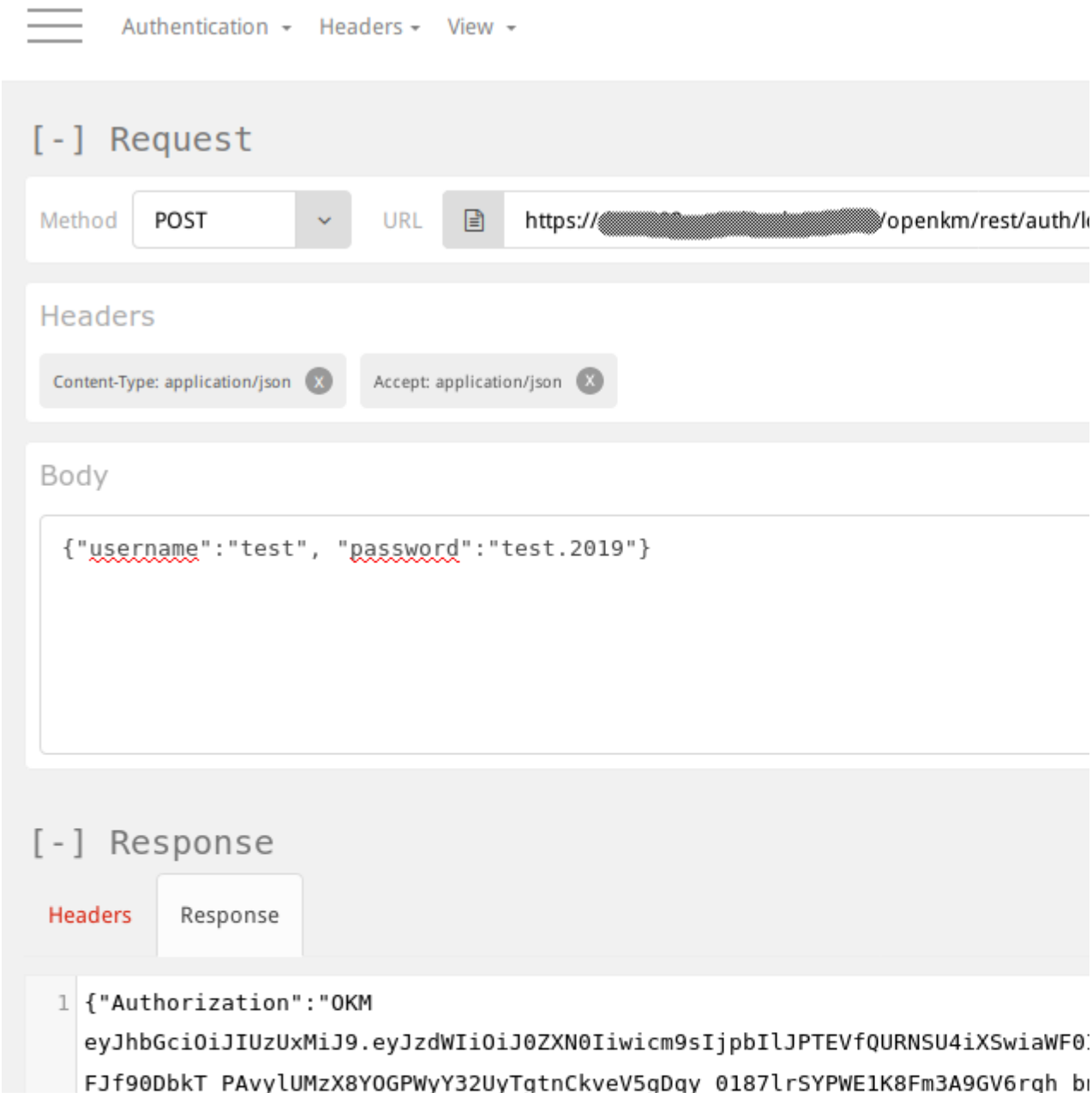
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

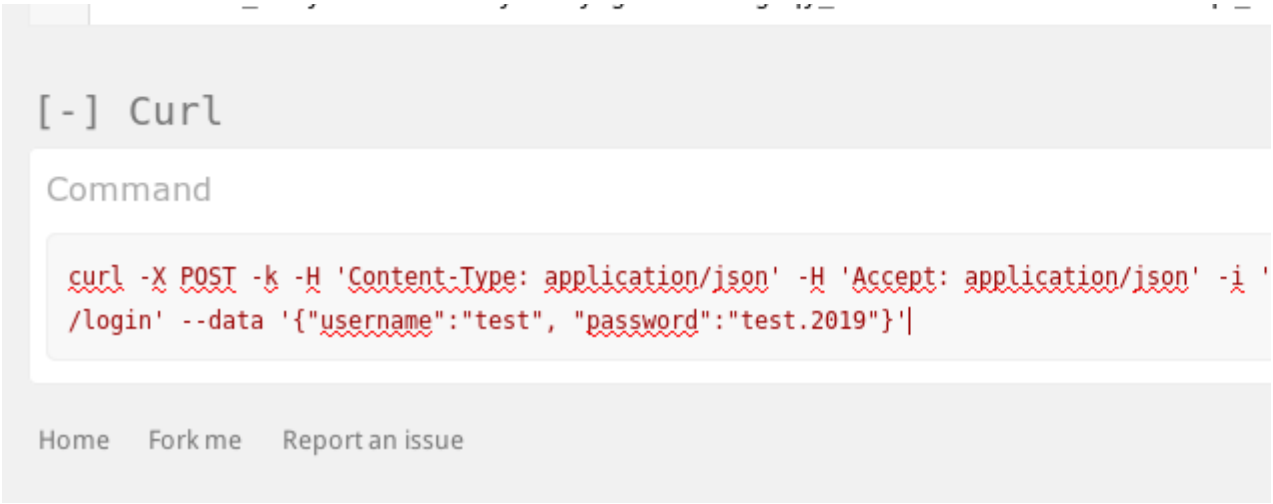
RESTClient sample

RESTClient add-on 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.*

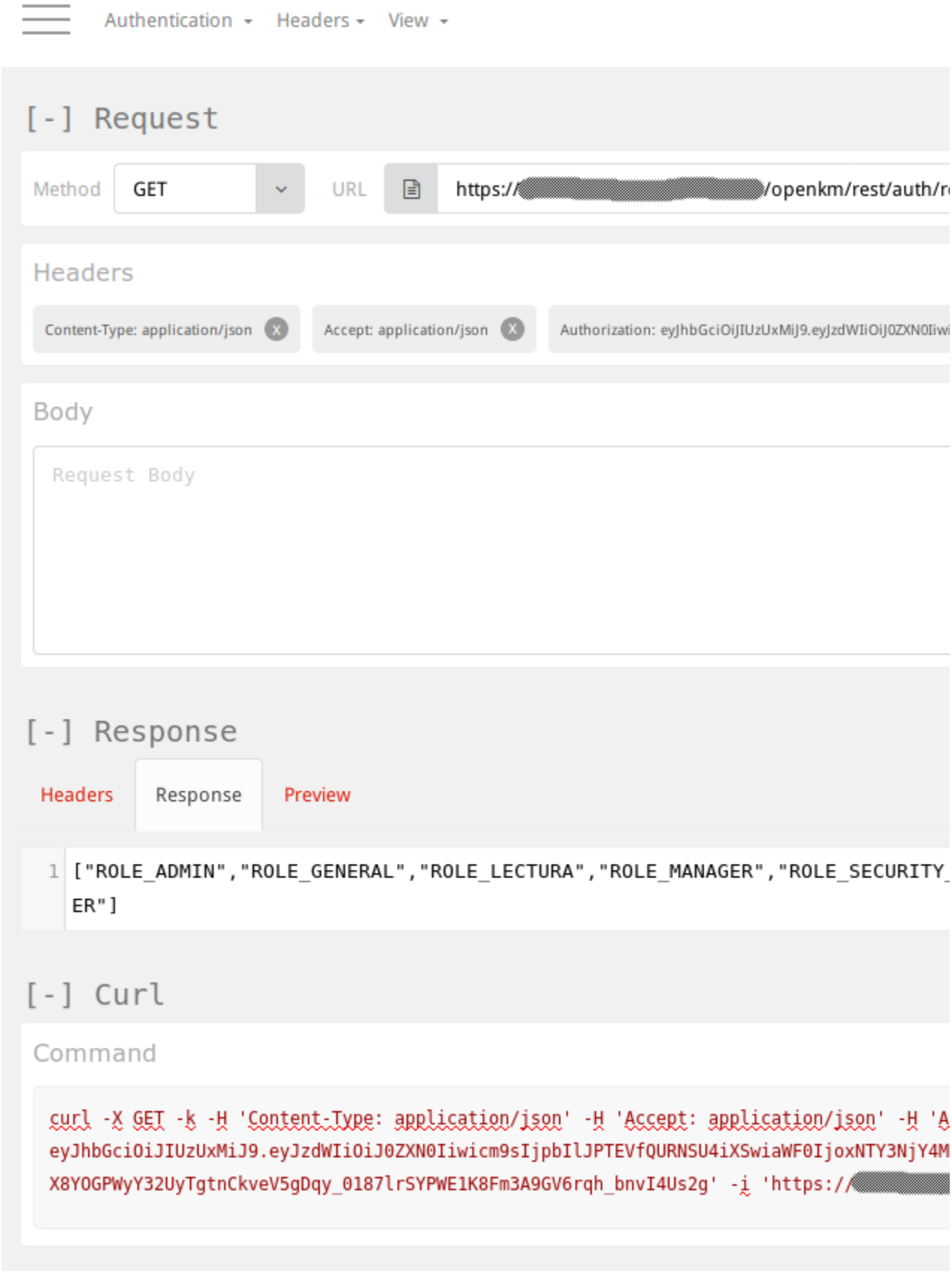




Get roles

- *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 when integrating PHP applications.

> *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, contact OpenKM support.*

Professional	PHP version	Status	Download
Version 8.1.4 and above.	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 above.	7.x	Active	okmPreview-3.0.1.zip

Login

First, you must create the web service object:

```
$this->ws = OKMWebServicesFactory::build(self::HOST);
```

*Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:*

```
$this->ws->login(self::USER, self::PASSWORD);
```

> *Once you are logged in to the web services, the session is kept in the web service object. Then you can use the other API methods.*

*At this point you can use all the Folder methods from the "**folder**" class as shown below:*

```
$folder = $this->ws->folder->getFolderProperties("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5");
```

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

```
<?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 = $this->ws->folder->getFolderProperties("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5");
            var_dump($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 a Document object 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 document name.*

$content *string type; it is recommended to use file_get_contents â€" reads the 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 the /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 the document content (binary data) of the current 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 whose 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***	*Moves a document into 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*

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 the newName parameter value is null, the document will preserve the same name.



Only the binary data and the security grants are copied to the destination; the metadata, keywords, etc. of the document are not copied.

*See the "**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, $name)	***Folder***	*Creates a new folder and returns it as a Folder object.*
Parameters: ***$uuid*** *string type is the destination UUID of the Folder or Record*		

Example:

```
<?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 = $this->ws->folder->createFolder('f93ac8df-e83b-487f-9a14-0af9e910a818', 'test');
            var_dump($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 another 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 whose 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. If the value is null, the folder will preserve 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 another 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 are copied; the metadata, keywords, etc. of the folder are not copied.

Additionally:

- *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 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***	*Returns a ContentInfo object with information about the folder.*
Parameters:		

$uuid *string type is the UUID or path of the folder.*

The ContentInfo object retrieves information about:

- *The number of folders inside it.*
- *The number of documents inside it.*
- *The number of records inside it.*
- *The number of mails inside it.*
- *The size in bytes of all objects in 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***	*The folder is permanently removed from the repository.*

Parameters:

$uuid *string type is the uuid or path of the Folder.*

Usually you will purge folders in /okm:trash/userId - the personal trash user locations - but it is possible to directly purge any folder from the entire repository.



When a folder is purged it can only be restored from a previous repository backup. The purge action removes the folder permanently 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**	*Sets 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**	*Creates 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,	**Folder**	*Creates a new folder from the template and returns*

$language, $categories, $keywords, $propertyGroups, $notes, $properties)		*an object Folder.*

*The $**uuid** parameter is the UUID value of the template file.*

*The $**dstPath** value is the folder destination path.*

*The $**language** parameter is optional.*

When the template uses metadata groups to fill in fields, these values are mandatory and must be set in the 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);
        }
    }

}

$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 in 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 element definitions.*

Parameters:

$grpName *string type: 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 element definitions for 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 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 an error will be raised.*

It is not mandatory to set all field values in the properties parameter; it is enough to include the fields you wish to change.

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 indicates whether the node has 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***	*Returns the XML metadata groups definition.*

> *This method can only be executed by a super user (ROLE_ADMIN member).*

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 suggested metadata field values.*

Parameters:

$uuid *string type is the UUID of the document, folder, mail, or record.*

$grpName *string type. The grpName should be a valid metadata group name.*

$propName *string type. The propName parameter should be a Metadata Select field type.*



The propName parameter should be a Metadata Select field type.



More information is available 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 metadata groups definition into the repository.*

Parameters:

$content *string type. It is recommended to use file_get_contents â€" reads the entire file into a string*

$name *string type. It is the name of the property group.*

This method can only be executed by super users (ROLE_ADMIN member users).

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);
        }
    }

}
```

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

```
$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 their values for a node.*

Parameters:

$uuid *string type is the UUID of the document, folder, mail, or record.*

$grpName *string type. The grpName should be a valid metadata group name.*

The method is usually used to display form elements with their values to be shown or changed by the 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 their values for a node.*

Parameters:

$uuid *string type is the UUID of the document, folder, mail, or record.*

$grpName *string type. The grpName should be a valid metadata group name.*

$versionName *string type. It is the version of the document, folder, mail, or record.*

The method is usually used to display form elements with their values to be shown or changed by the 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 - the marshalling and unmarshalling process - while you are only interested in using a few object variables. If that is your case, you can use NodeProperties classes to retrieve only the object variables 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 parameters "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" tells the query to skip that many results before 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 to 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 - the marshalling and unmarshalling process - while you are only interested in using a few object variables. If that is your case, you can use NodeProperties classes to retrieve only the object variables 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();
?>
```

Integrate preview with other web applications

You can integrate the OpenKM web preview module into other web applications quickly.

Advantages:

- *Quick integration into an iframe in your own web application.*
- *All OpenKM MIME types supported by the preview modules.*
- *Can be used without logging into OpenKM.*
- *Can preview documents that are not in OpenKM.*

*If your OpenKM is available from http://your_domain/openkm, then the OpenKM preview URL should be **http://your_domain/openkm/Preview**?docUrl=http://localhost/test.pdf*

*The **supported MIME types depend on your OpenKM configuration and OpenKM conversion service**.*

These components preview the most common file types:

- ***PDF, Microsoft Office** and **LibreOffice***
- ***Images like JPG, PNG, TIFF, and media like 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 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 code file, for example .java, .php, etc.*
mpgUrl	***String***	*The full URL of a .mp4 or .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 the server. By default it is PDF.js, but with this parameter you can force Flexpaper.* *OpenKM supports 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**, keep in mind to set the "Access-Control-Allow-Origin" header in the response.*

The Java code section below can provide you with 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);

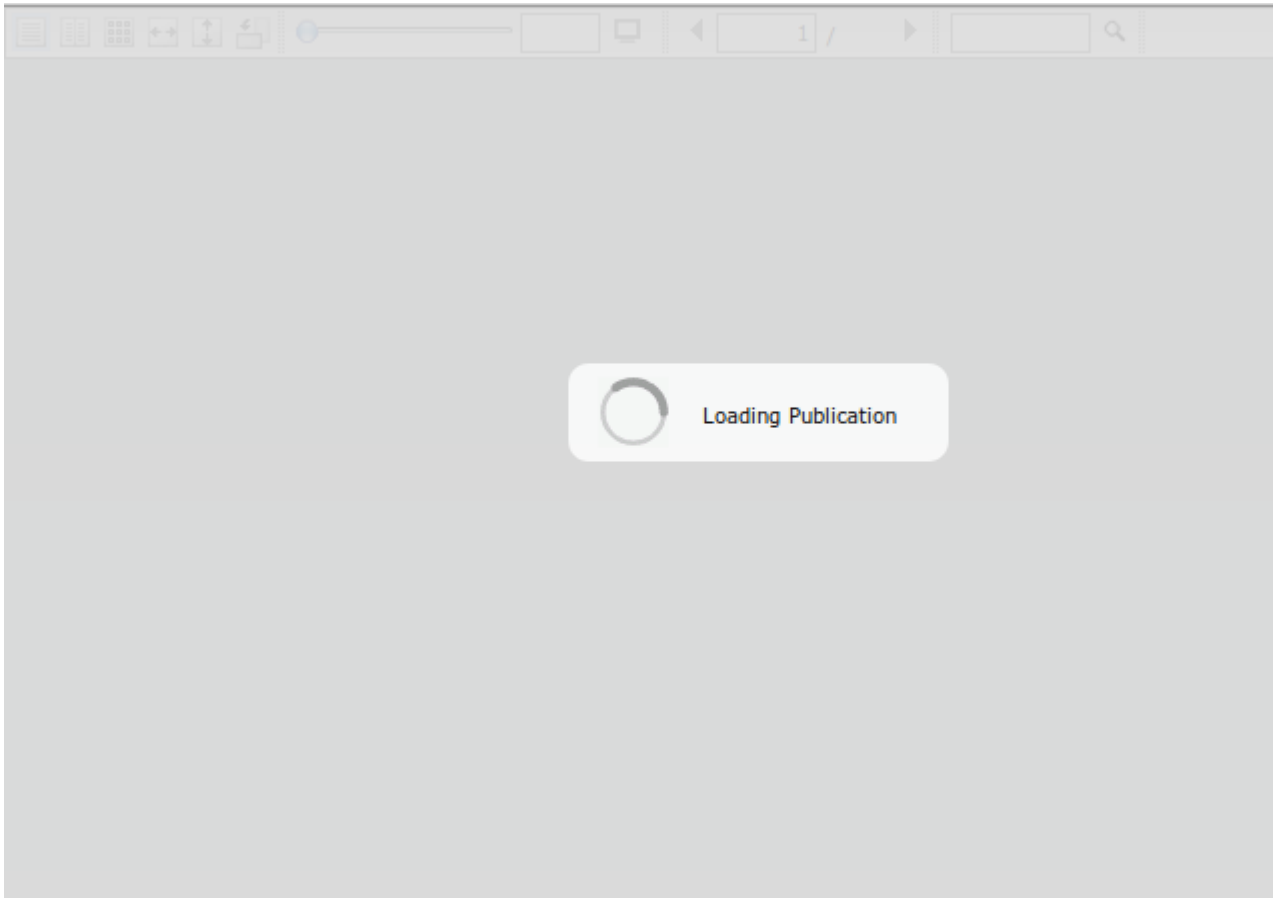
    if (value != null) {
        return value;
    } else {
        return EMPTY_STRING;
    }
}

/**
 * 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 cannot be previewed and you may get a screen like this:



Check if you see any error in your OpenKM log.

*Take a look in the JavaScript console; if it shows an error like "**missing header CORS 'Access-Control-Allow-Origin')**" then you must change your web server configuration to allow it.*

*At http://enable-cors.org/server_apache.html you can find information about how to enable **Access-Control-Allow-Origin** in the Apache web server.*

Samples

Sample	*Description*
Scripting samples	*A collection of samples that use the 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 the property group.* • *Shows the number of documents, folders, and the size for a given path.* • *Gets the path by UUID.* • *Shows all documents used in all running workflows.*
Script - Compact documents with size equal to 0	*Finds all documents with size = 0 and compacts version history to the latest.*
Script - Convert TIF to PDF	*Transforms all the TIF files in the repository to PDF.*
Script - Count all child subfolders	*Counts all child folders and subfolders recursively in the first-level folder (okm:root children folders).*
Script - Exporter	*Executes the repository export.*
Script - Export repository folder to server as a ZIP file	*Exports the folder contents to the server application as a ZIP file.*
Script - Deep folders	*Finds folders with a certain depth.*
Script - Import data from CSV file	*Imports metadata group values from a CSV file.*
Script - Purge all folders into a parent folder	*Purges all folders into a parent without first moving files to the trash.*
Script - Purge all users' trash	*Example of how to clean all users' trash.*

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

Script - Read PDF form fields	*How to read PDF form fields.*
Script - Recursive character renaming	*Renames a character in all folders or documents from a given initial path.*
Script - Recursive repository traversal	*Navigates across all repository nodes recursively.*
Script - Recursive security change	*Changes the security.*
Script - Security change at low level	*Changes the security at the 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 a 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 the number of documents, folders, and the size for 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 by 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 equal to 0

Find all documents with size=0 and compact their version history to the 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 folder.*
- *Because of 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 child subfolders

Count all child folders and subfolders recursively within the 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 the 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 - Folder depth

The script finds all folders at a certain depth.



*The variable **MAX_DEPTH=5** indicates the folder depth.*

```
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);
```

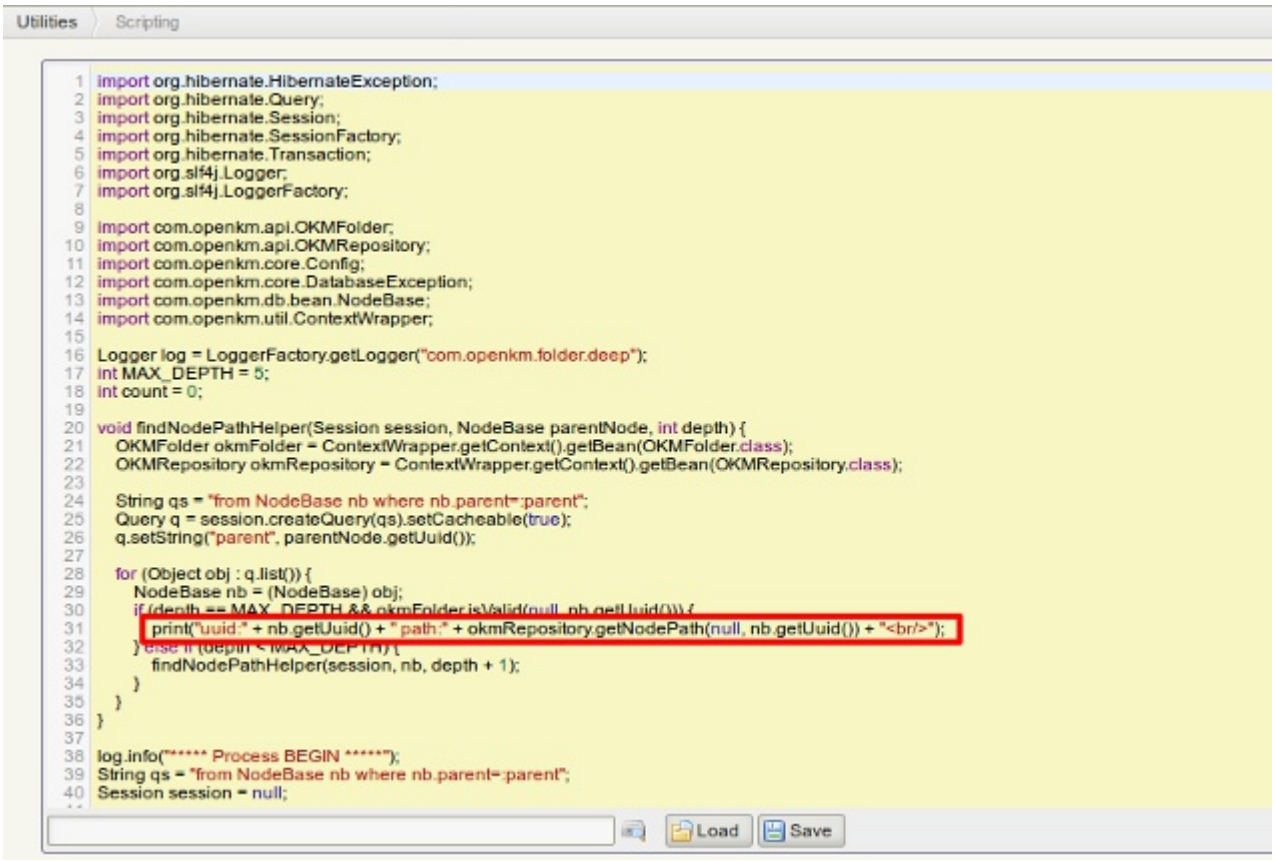
```
    q.setString("parent", Config.ROOT_NODE_UUID);

    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

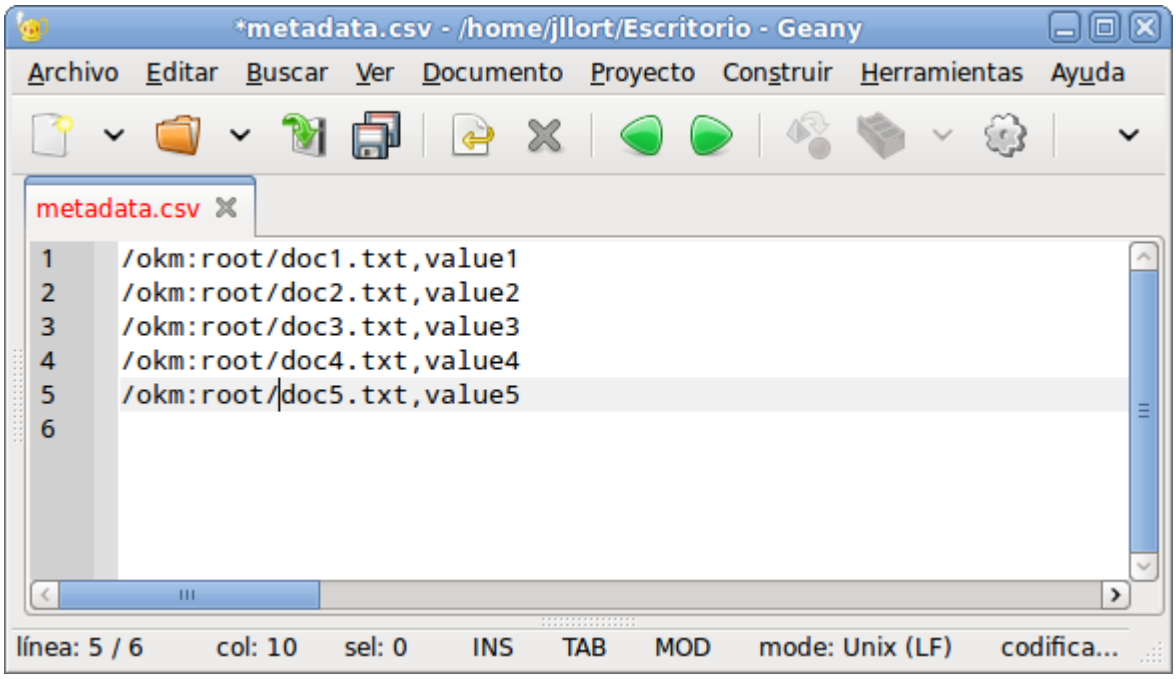
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Script - Import data from CSV file

The script can be executed from the administration scripting (or used as an automation script too) to import metadata values.

- *There's a metadata CSV file with two columns, one for the document path and the other for the metadata value. The script can be easily changed to use more than two columns.*
- *The script searches for documents by document name. In theory there is only one document with the same name in the system; otherwise the script shows an error. Here the document path from the first CSV column could be used, but to provide a more complete sample a search engine has been used to find the document.*
- *For each document found, the script adds a metadata group with the CSV values.*

The CSV file has two columns; the first column contains the document path and the second column has the metadata value.



Download the 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++;
}

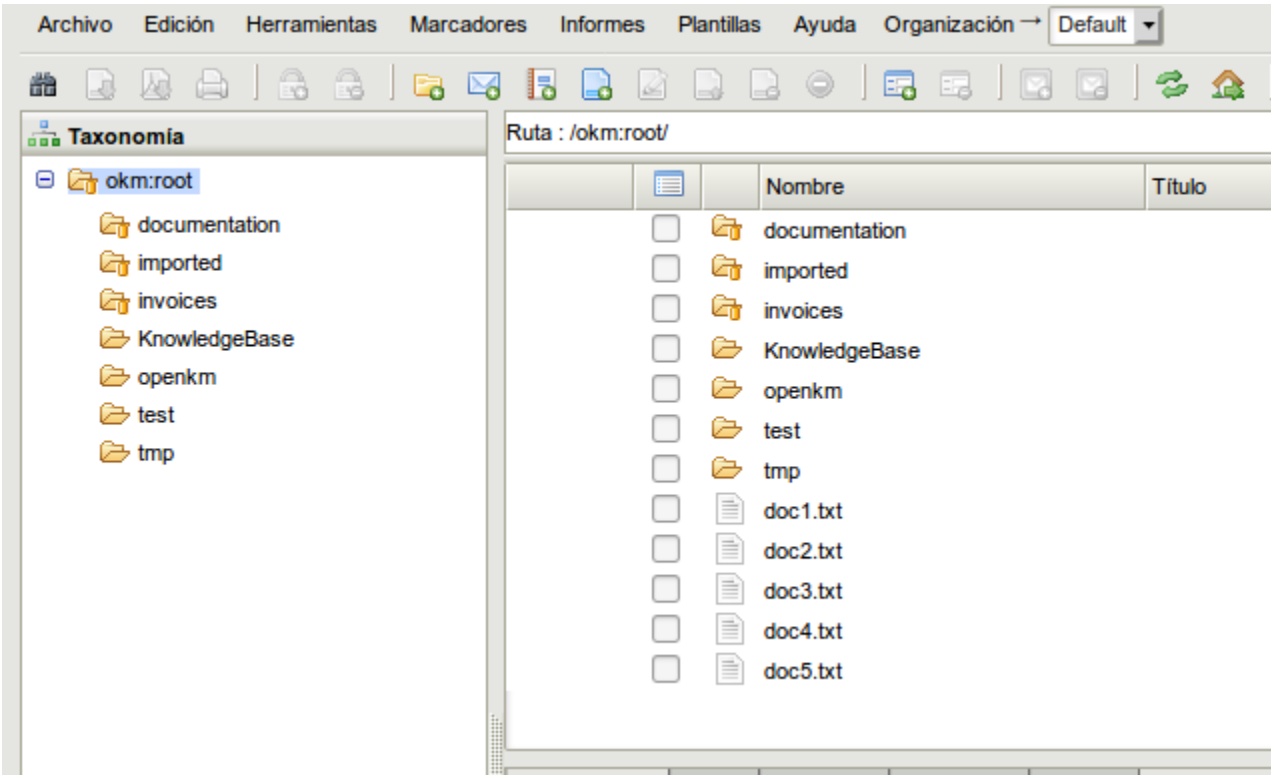
print("Total:" + count + "</br>");
print("Found:" + countFound + "</br>");
print("Error not document:" + countNotDocument + "</br>");
print("Error more then one document found:" + moreThanOneDocumentFound + "</br>");
print("Error not found:" + notFound + "</br>");
print("Error name empty:" + notFound + "</br>");
```

Image

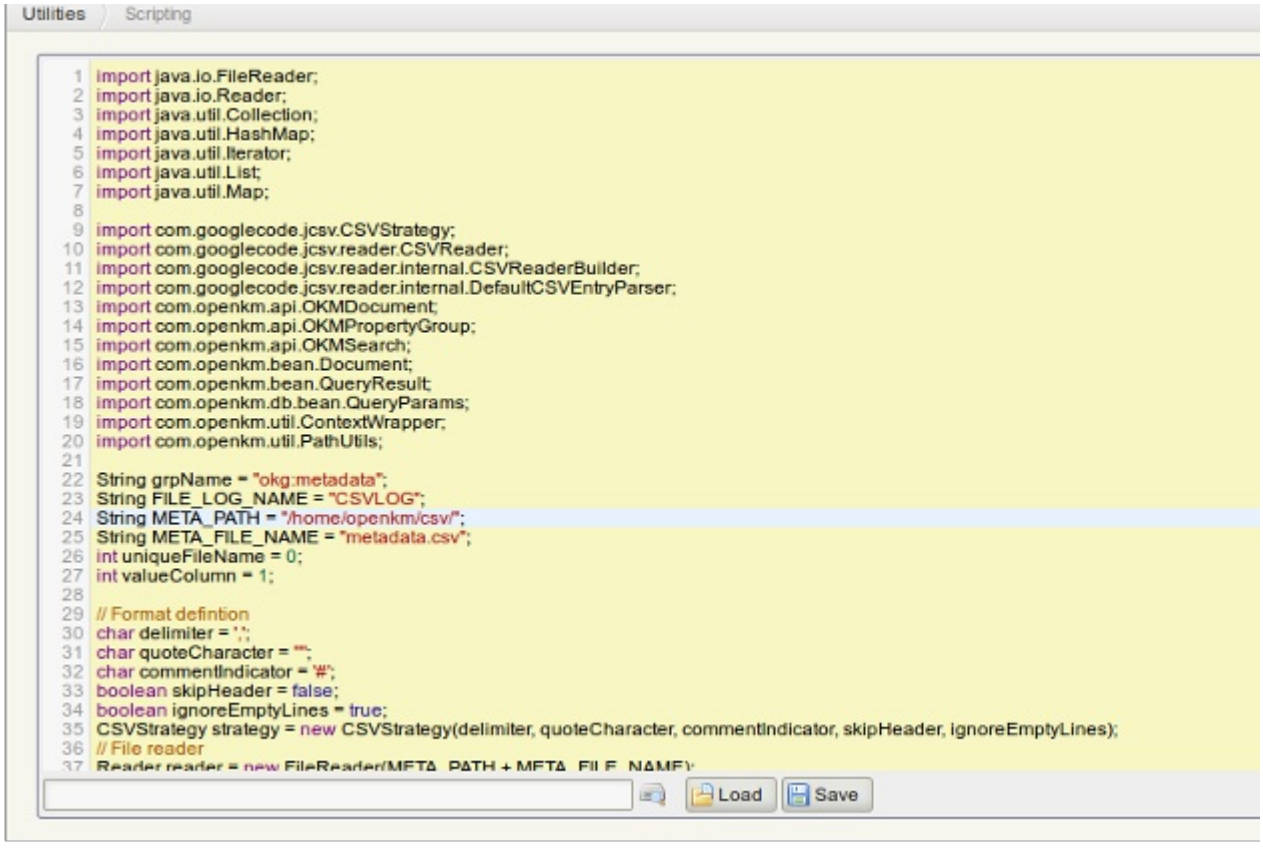
Register the metadata group



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

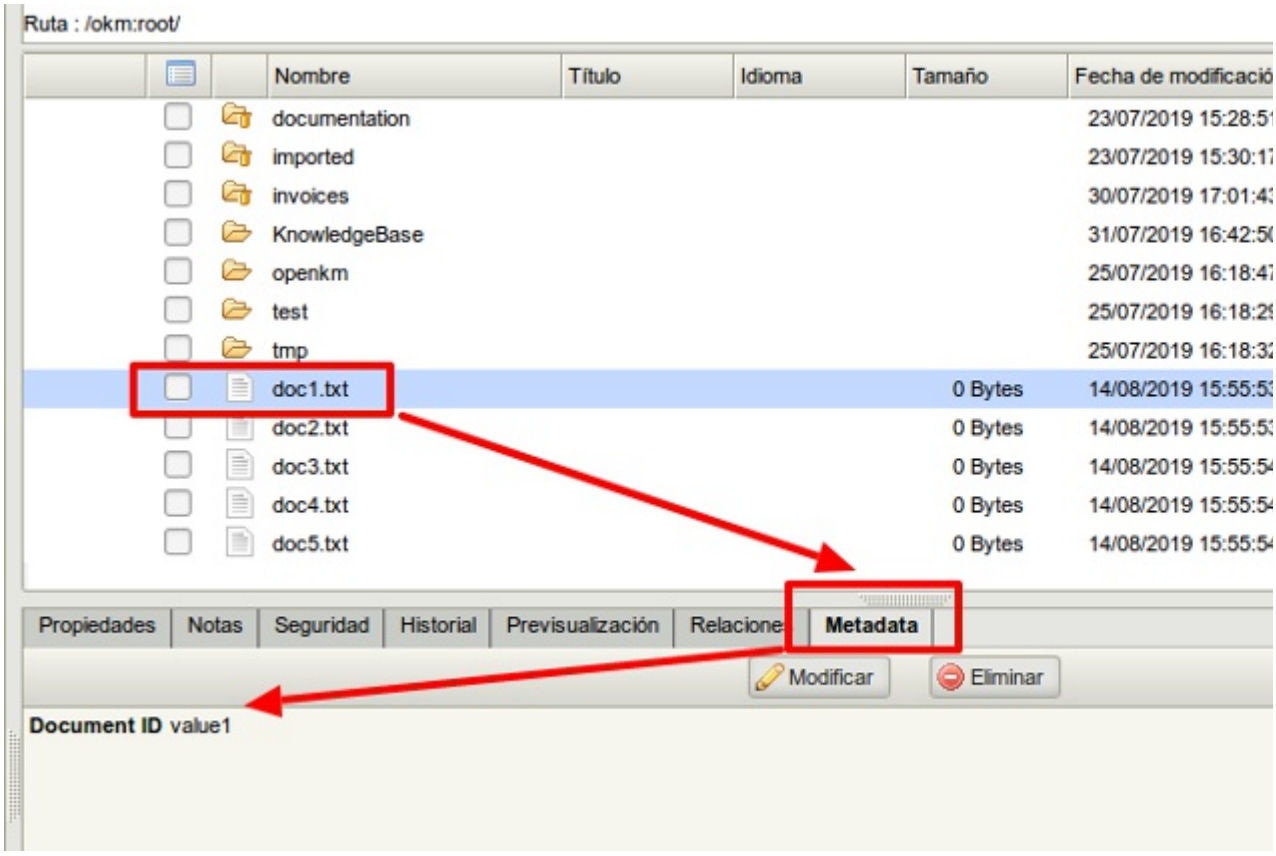
Execution


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 a parent

This script purges all folders into a parent without first moving files to the trash.



The purge action is a non-recoverable action that permanently removes objects from the 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' trash

```
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 a folder and its descendants into the pending text

```
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 that 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 security change

```
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 + SECURITY
    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 it is necessary to propagate security changes under a folder that contains a significant number of nodes. These types of changes made from the UI can take a long time to propagate across the repository, and this script does it much faster.



*Our recommendation is to use it when changes affect more than **100.000 nodes**; otherwise, we recommend doing it from the UI.*



The security change starts with a folder UUID node and propagates across all its children. The logic is to remove existing grants for each node and replace them with new ones based only on roles. Note that you can build other types of logic. Use the example as a basis.

- *The variable **UUID** is the folder Unique Identifier.*
- *The variable **MAX_DEPTH** sets how deep the script should advance into subfolder children (note that 0 indicates the UUID folder).*
- *The arrays **roleName** and **roleGrant** are pairs of role names and role grants to be added.*
- *The variable **LOG_FILE_NAME** is the name of the log file saved in the $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_PER

        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();
        }
    }

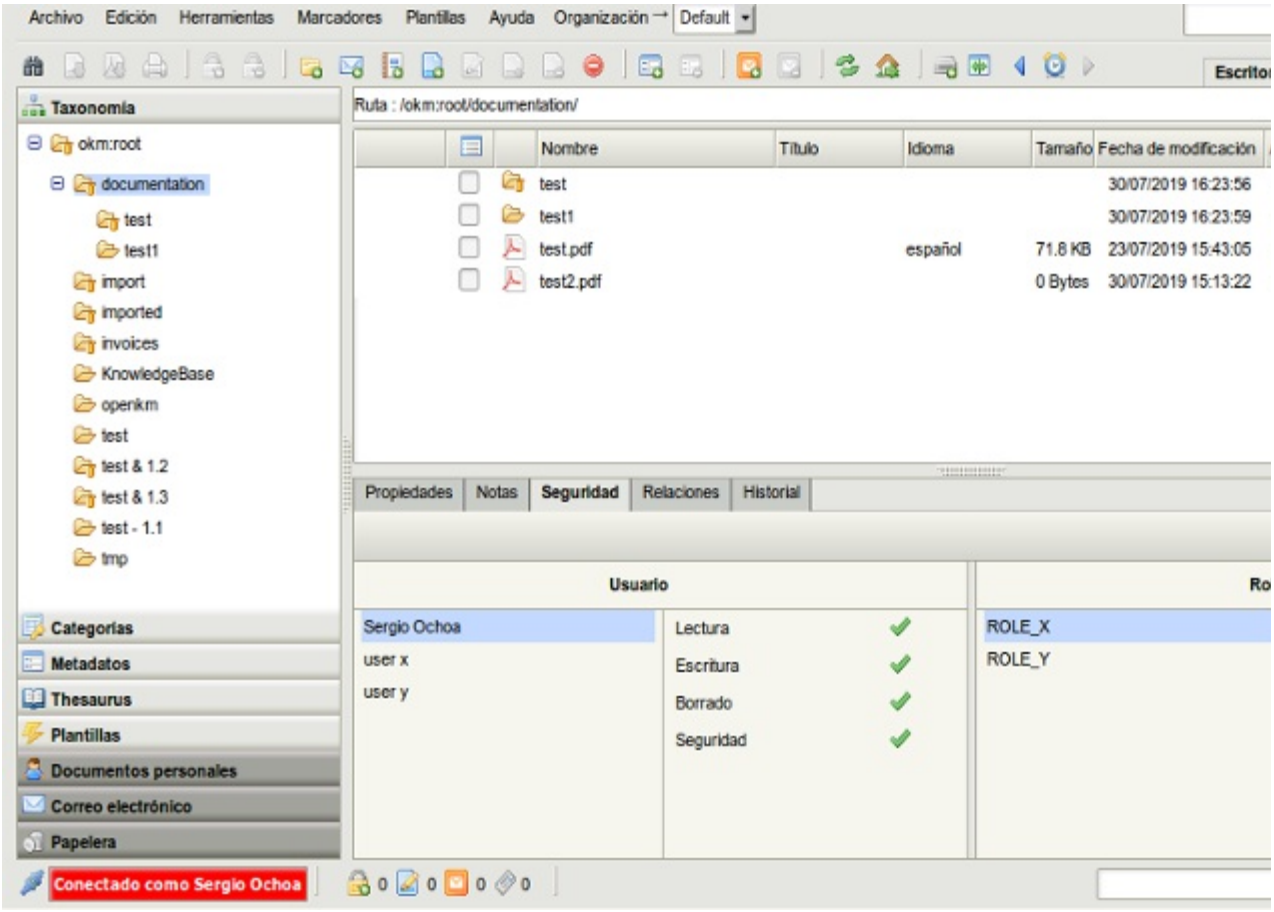

    try {
        FileLogger.info(LOG_FILE_NAME, "**** Security change started ****");
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = null;
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/okmdb", "openkm", "*secret*");
        con.setAutoCommit(false);
        updateSecurity(con, folder_UUID, 0);
        con.close();
        FileLogger.info(LOG_FILE_NAME, "**** Security change ended ****");
    } catch (Exception e) {
        print(e.getMessage());
    }
```



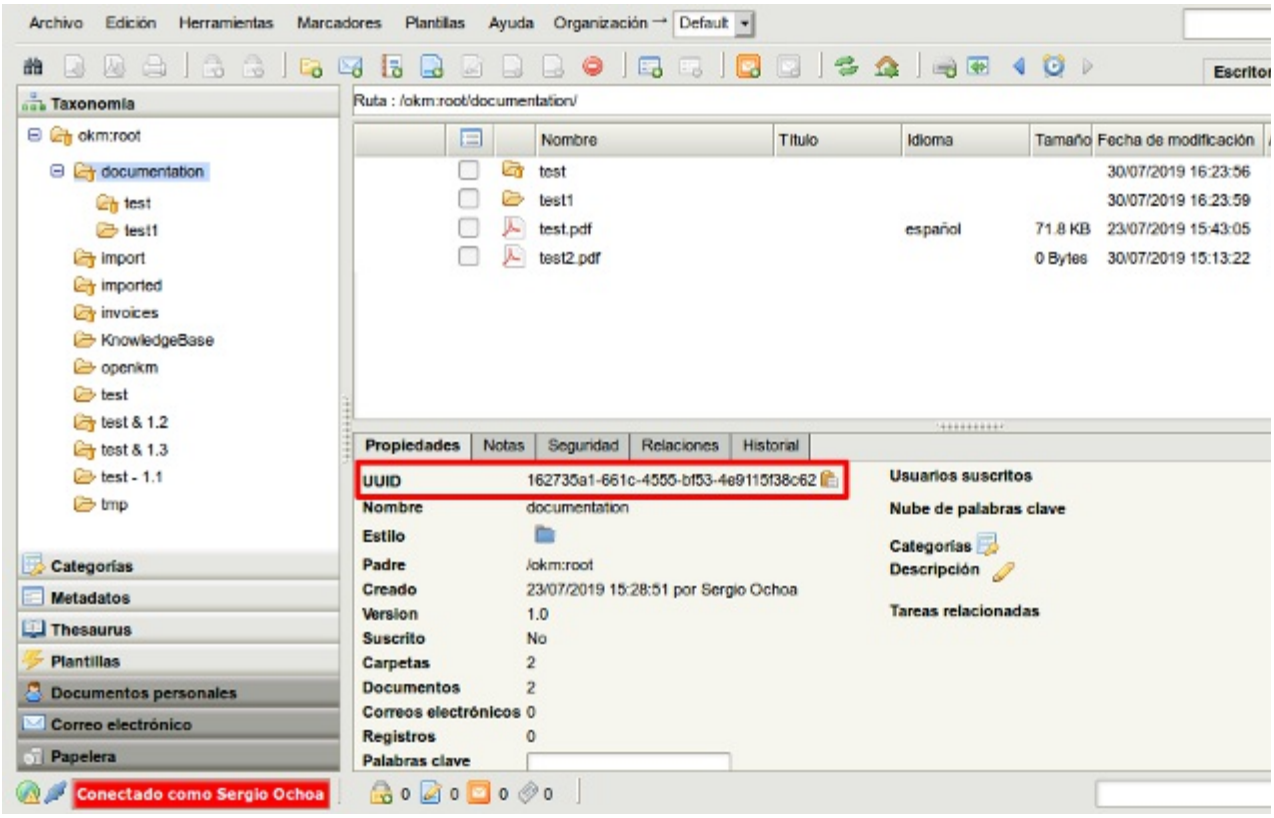
After the script execution, we must reindex Lucene indexes because the security changes were made at the database level, and this is necessary to maintain integrity with the search engine.

Images

Initial security status

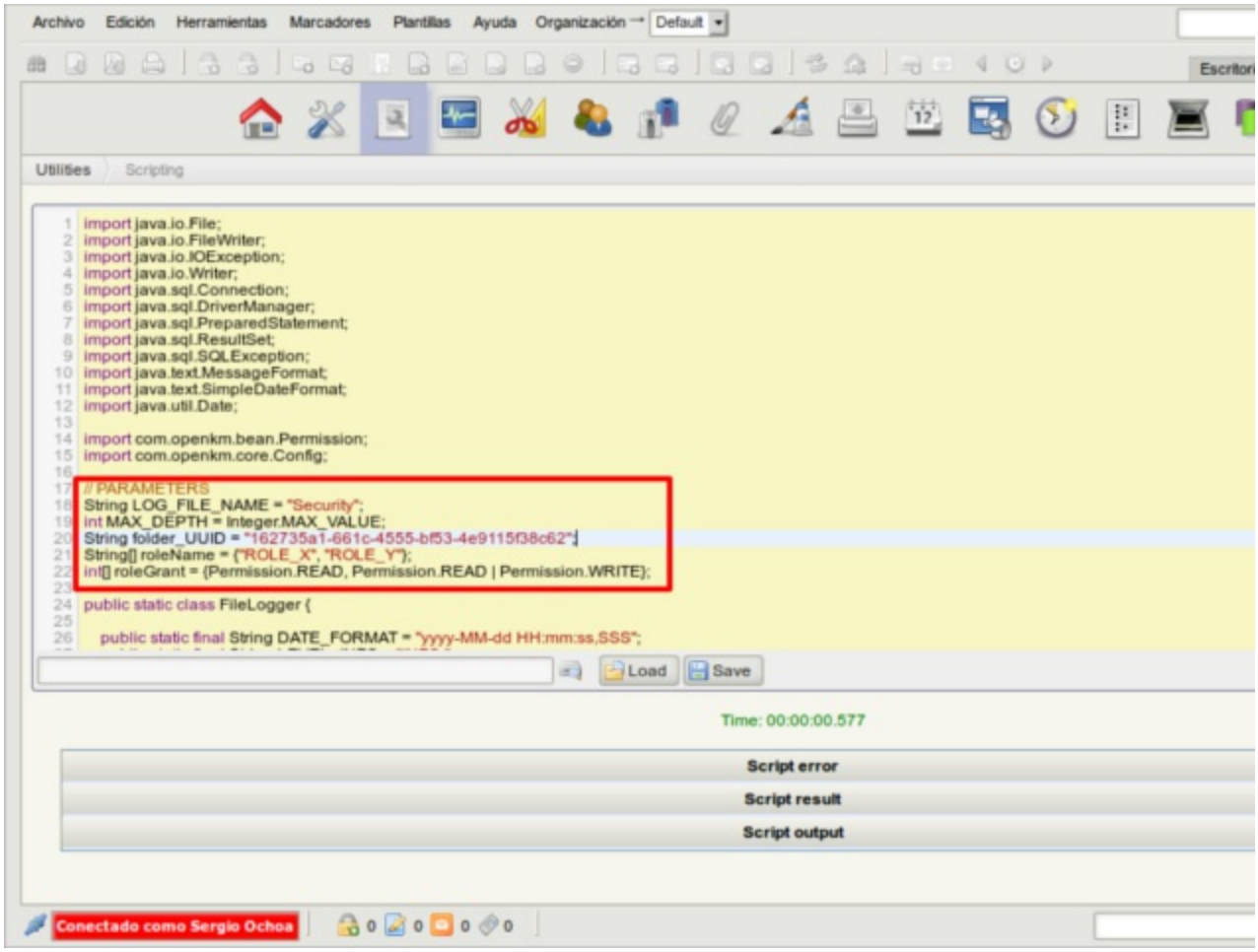


UUID of the folder



Execution

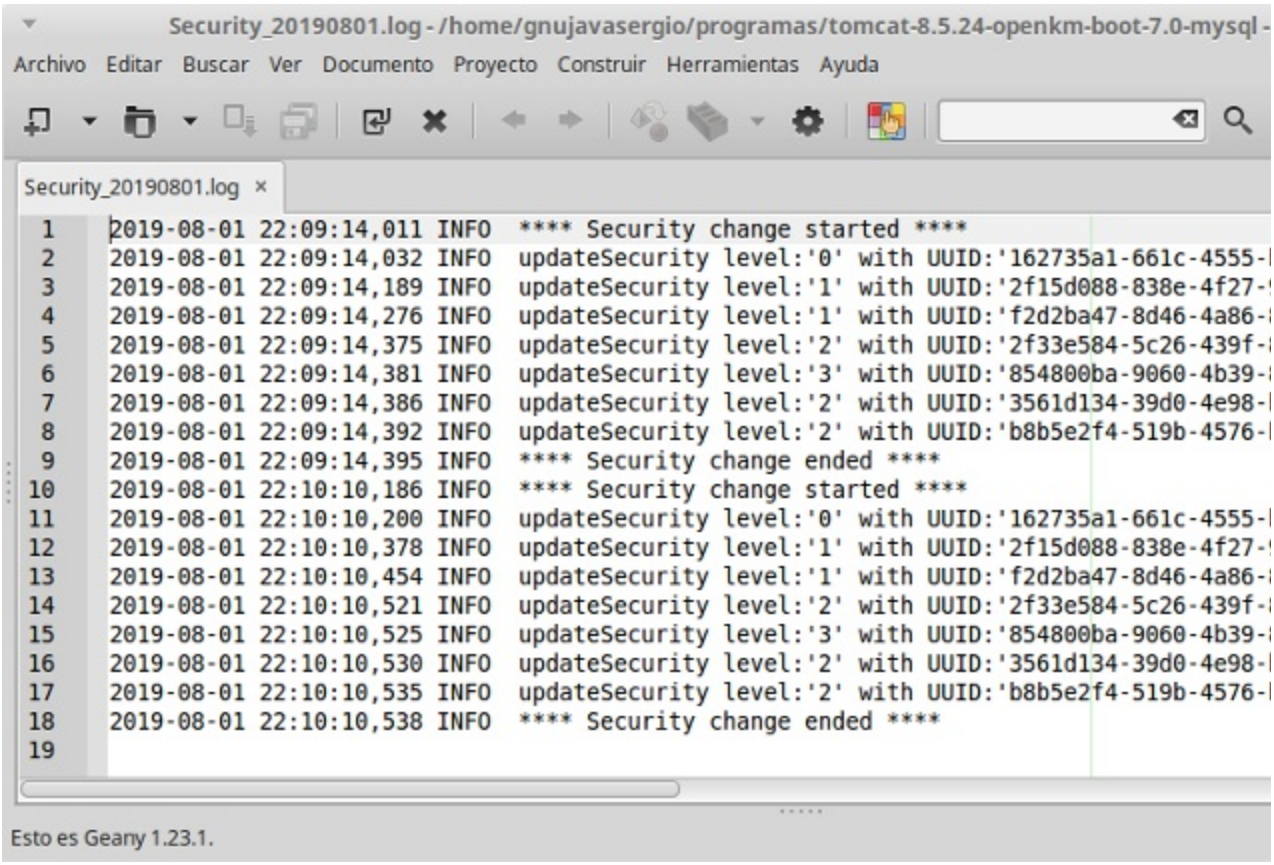
Change the script parameters to your own.



Log

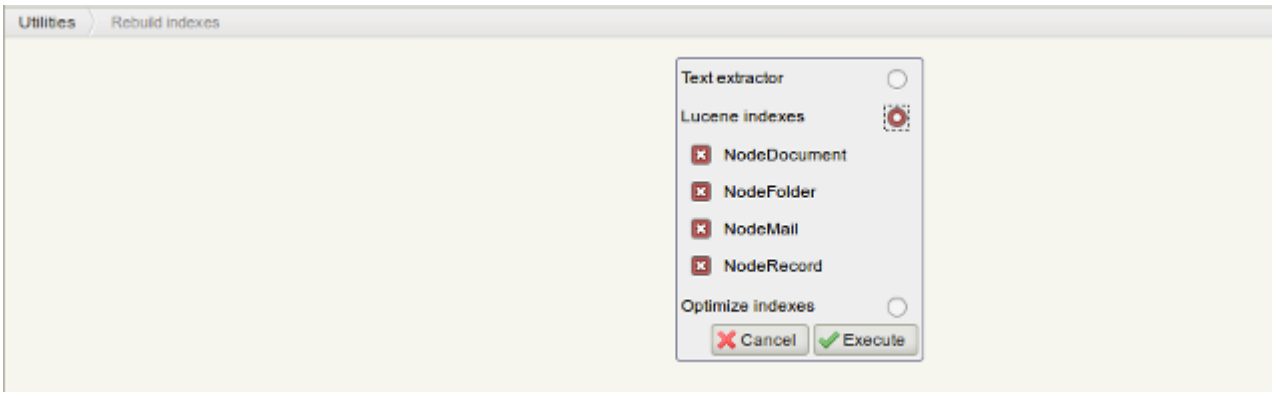
Consider taking a look at the log file to see the progress of the process and to check for errors.

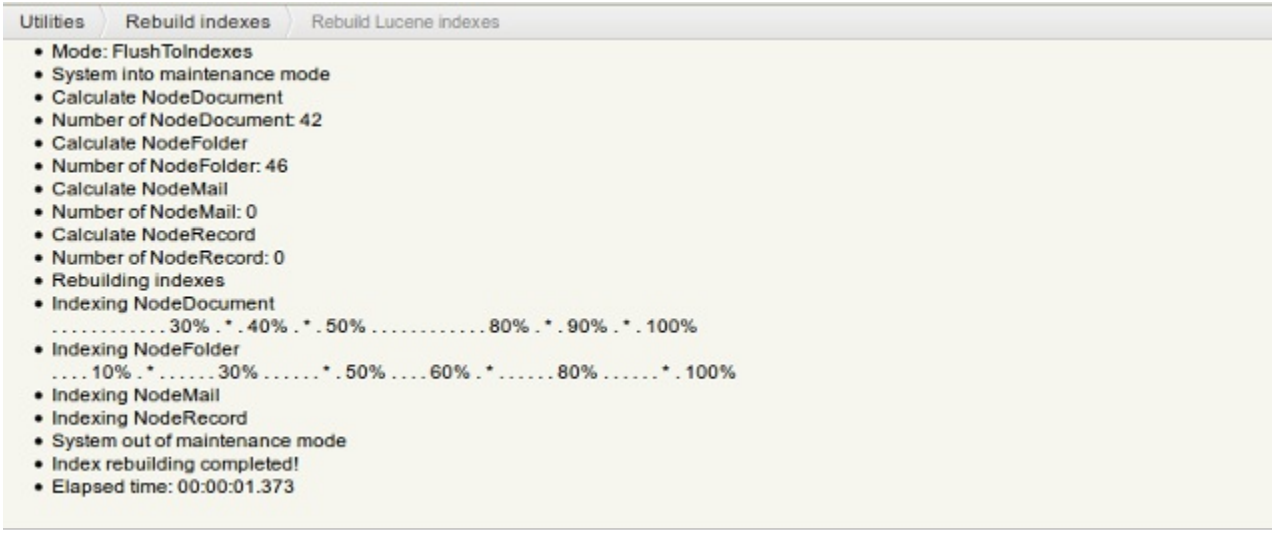
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



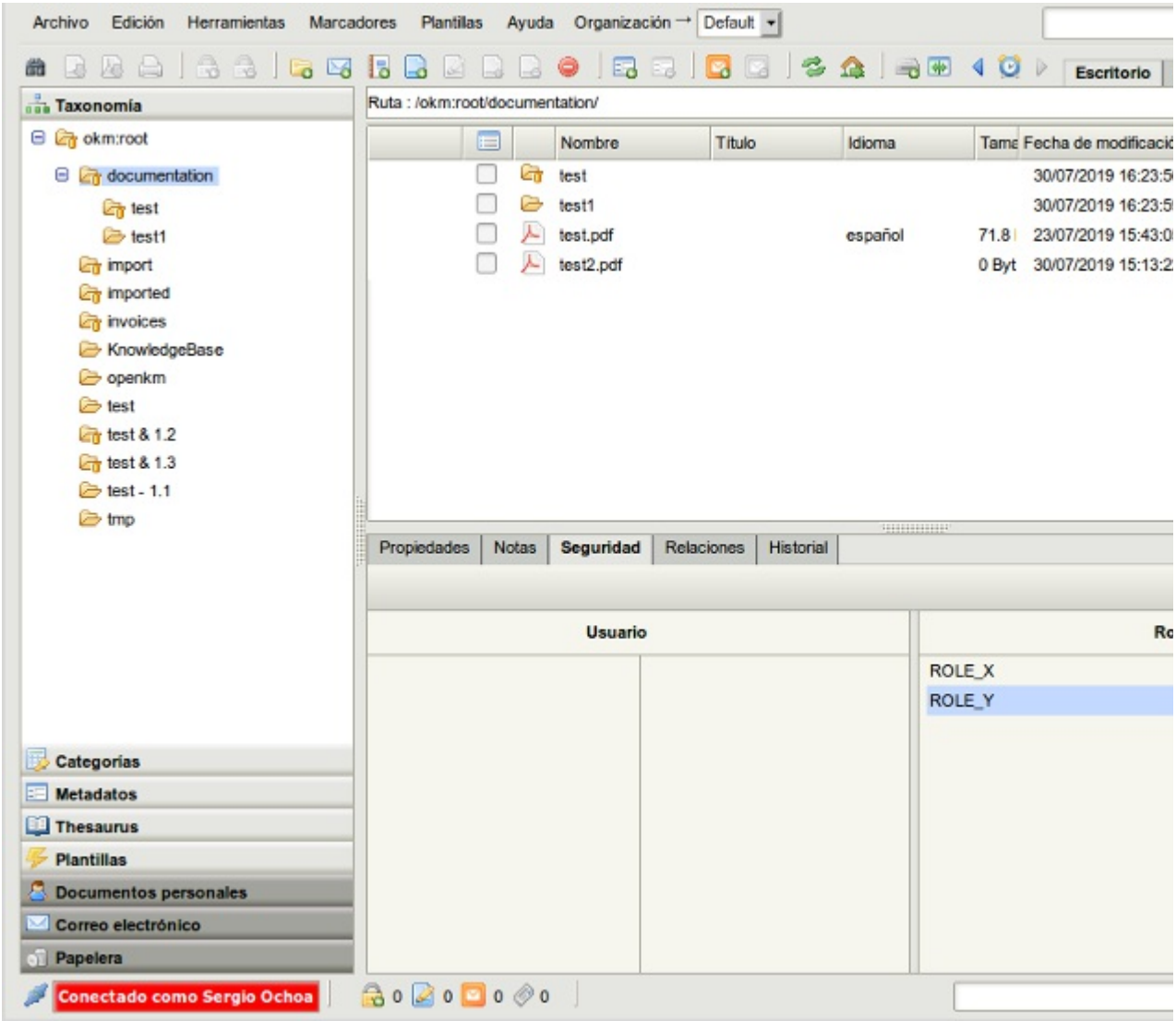
Rebuild Lucene indexes

Because the security changes were made at the database level, it is necessary to reindex the Lucene indexes to maintain integrity with the search engine.





Security changes 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 a 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 metadata group values.*

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

Script for automation - Autonumeric

Creates a unique number for each new document.

- *There is metadata with three fields that store data.*
- *Each time a new document is uploaded, the automation is executed. The automation script creates a unique numeric identifier using the metadata database's sequence feature.*
- *The **document id** is a 6-digit incremental number*
- *The **revision** number starts at 1*
- *The **code** is a unique id + "-" + revision*

To get it running, you need to register a metadata group and then create an automation task based on scripting.

This is only a sample implementation and is 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;
```

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

```
  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 an automation task based on the metadata group creation event:

- ***Name**: Autonumeric*
- ***Event**: Document creation*
- ***At**: Post*

*Use the previous script 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 on metadata values.

- *Property **okp:data.id** stores a unique auto-incrementing 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 the metadata group is changed, automation code is executed which generates **okp:data.id** and renames the document based on mask **projectCode-autonumericId-clientCode-description.documentExtension**.*

To get it running, it is necessary to register a metadata group and then create an automation task based on scripting.

This is only a sample implementation and is 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:

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

```
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 the metadata group creation event:

- ***Name**: AutoRename*
- ***Event**: Add metadata group*
- ***At**: Pre*

*Use the previous script in the **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.*
Crontab sample - Generate HTML report	*Generates an HTML report from the OpenKM activity log.*
Crontab task - XML importer	*Imports 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 imports files from the server's file system.

- *Files are imported from the server file system.*
- *Files are periodically imported by a crontab task to an 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());
  }
 }

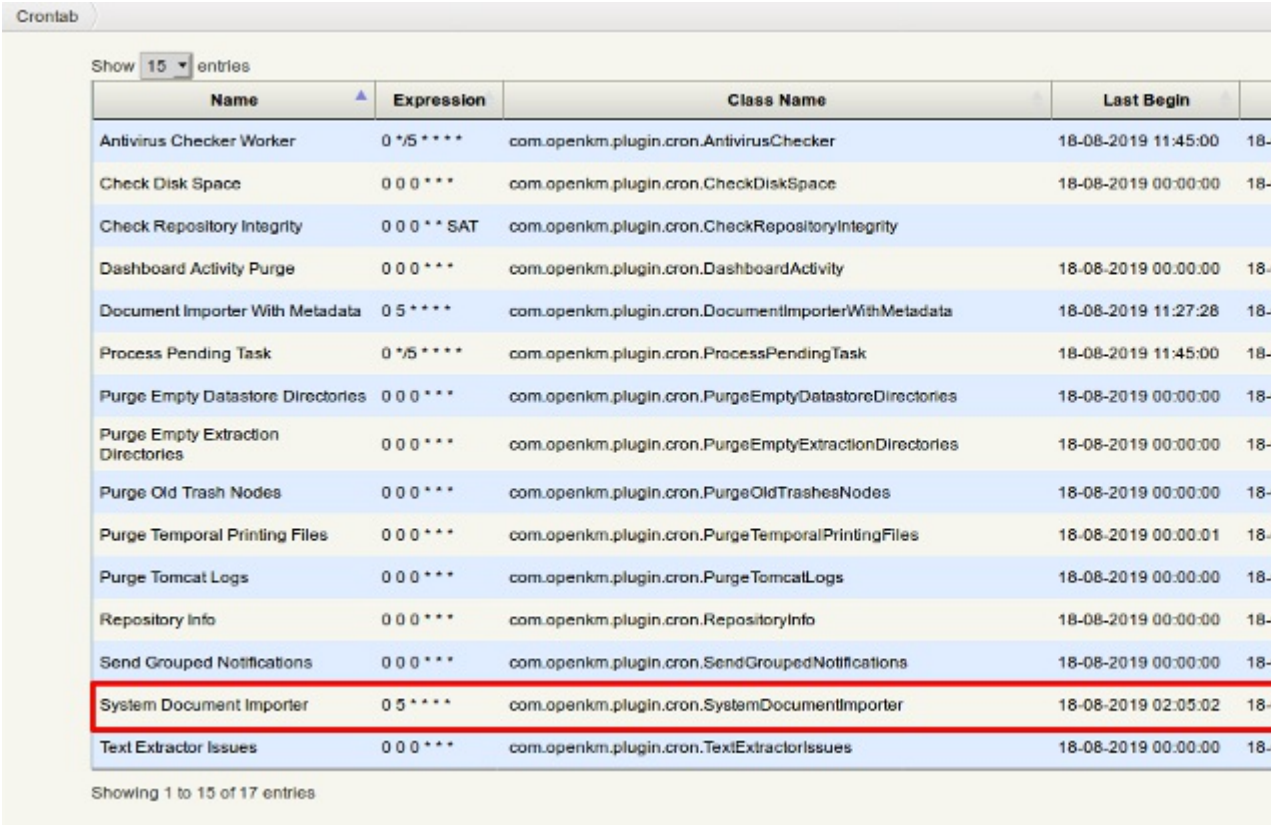
}
```

Register a new plugin

- *To install a new plugin,* ***create a JAR file*** *and* ***copy it into your $TOMCAT/plugins*** *folder. More information is available at Creating a JAR file with Eclipse.*
- *Go to* ***Administration*** *>* ***Utilities*** *>* ***Plugins*** *and click the*  *Reload plugins button at the top right.*
- *In the* ***plugins table****, the new plugin will be shown.*

Images

Crontab task

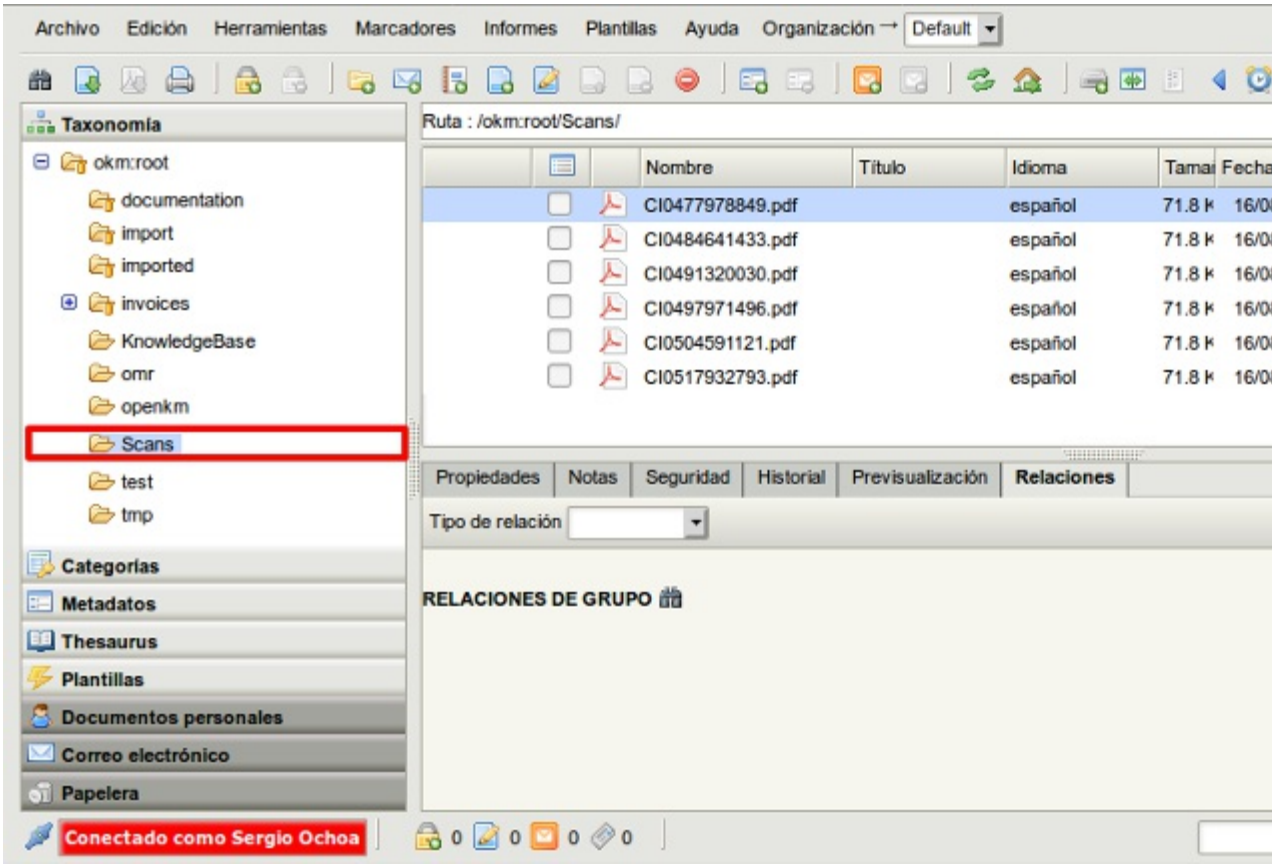


Files in the application server



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 on 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 the text between the characters "**(**" and "**)**" present in the file name.*
- *Files are stored at basePath/year/month (when year or month folders are not present, they are automatically created).*
- *At the end, it sends an email 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;
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 {

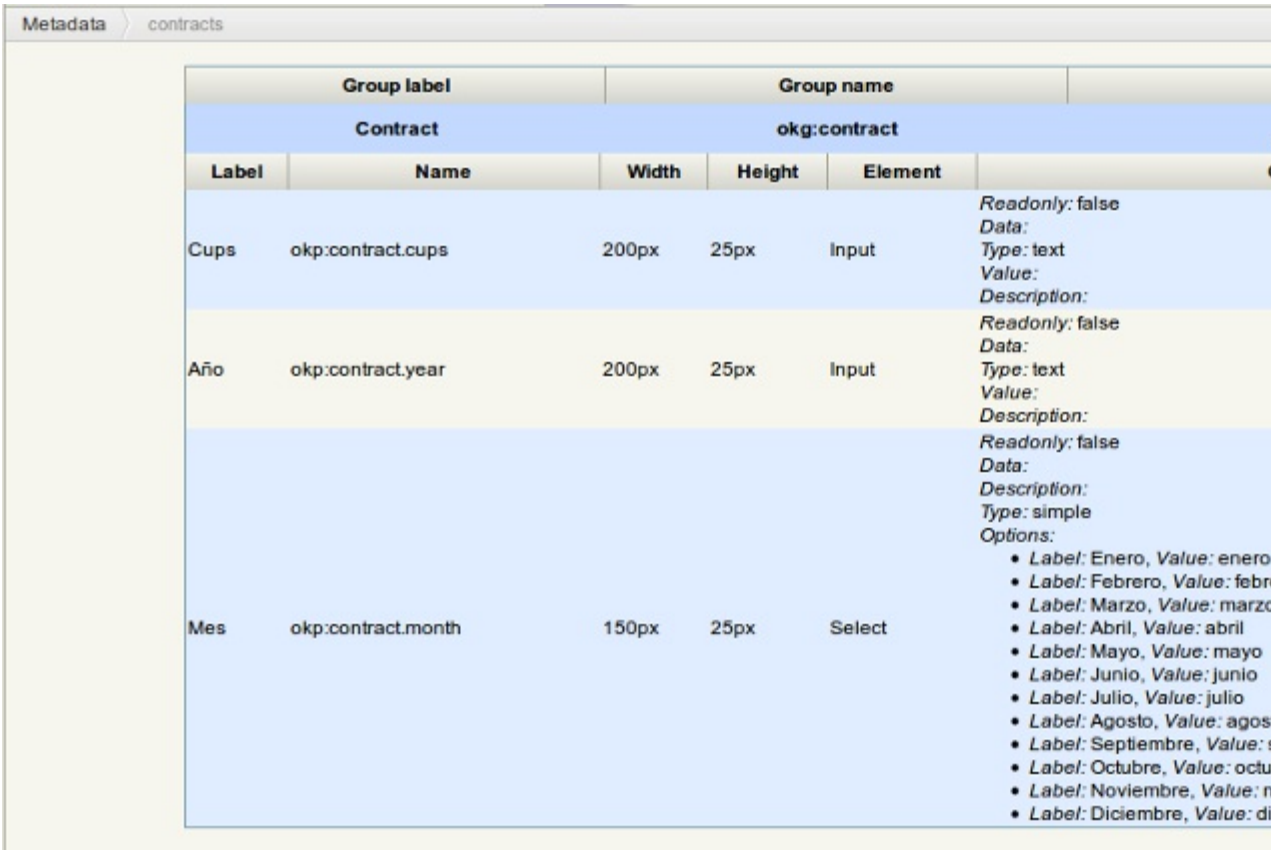
        public String fileName = "";
        public String dstPath = "";
        public String cups = "";
        public String month = "";
        public String year = "";
        public String error = "";
    }
}
```

Register a new plugin

- *To install 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 the*  *Reload plugins button at the top right.*
- *In* ***the plugins table*** *the new plugin will be shown.*

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 on the application server

document 1 - (CUPS001025,REN).pdf	5,9 KB Adobe Acrobat Document
document 2 - (CUPS001026,REN).pdf	5,9 KB Adobe Acrobat Document
document 3 - (CUPS001046,REN).pdf	5,9 KB Adobe Acrobat Document
document 4 - (CUPS001030,REN).pdf	5,9 KB Adobe Acrobat Document
document 5 - (CUPS001045,REN).pdf	5,9 KB Adobe Acrobat Document
document 6 - (CUPS001026).pdf	5,9 KB Adobe Acrobat Document
document 7 - (CUPS001046,REN).pdf	5,9 KB Adobe Acrobat Document
document 8 - CUPS001030 REC.pdf	5,9 KB Adobe Acrobat Document

Result

Mail notification

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

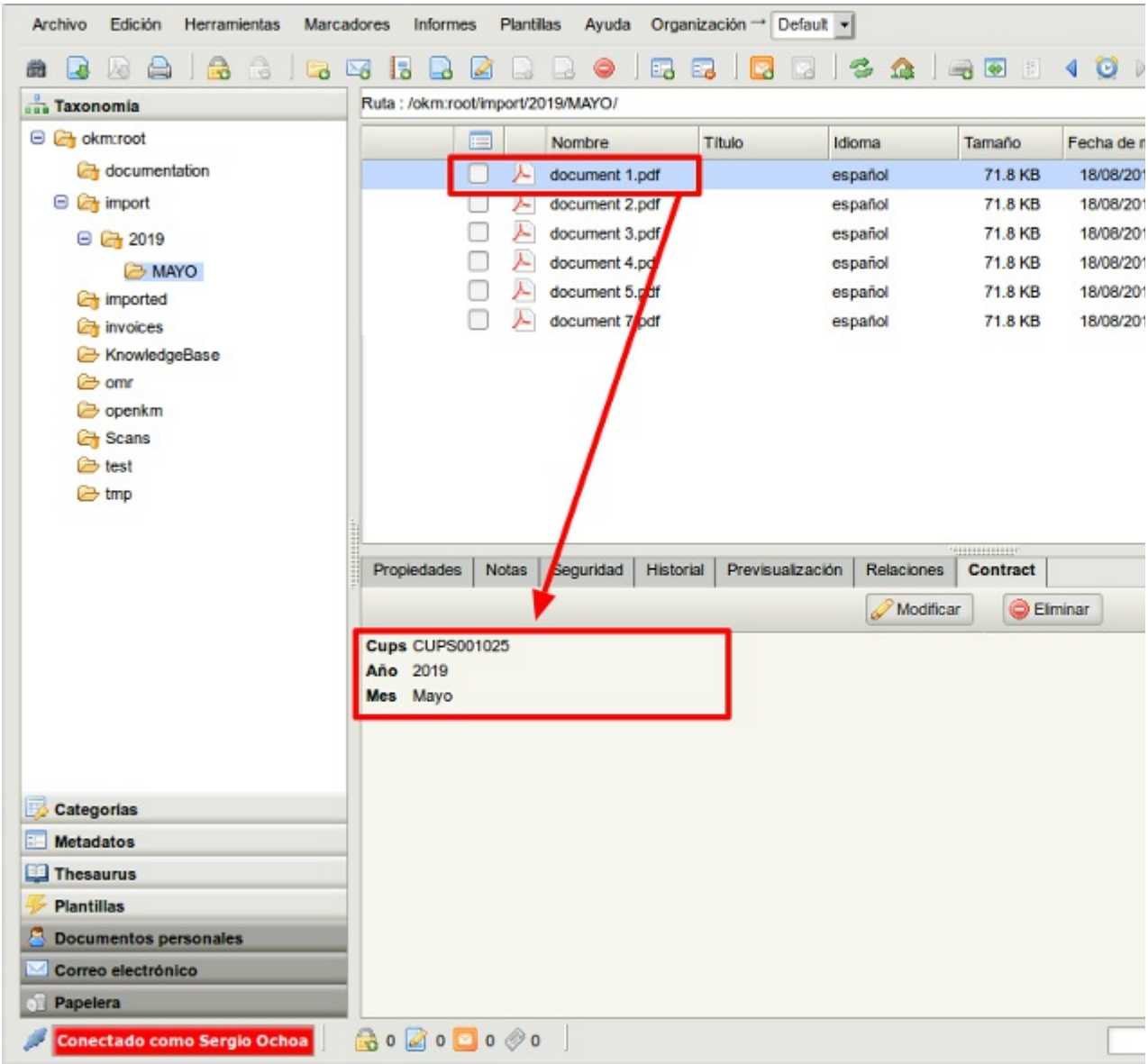
Import report

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

Files imported into the repository

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.

Crontab sample - Generate HTML report

This example 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)) {
```

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

```
    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 a 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 the Reload plugins button at the top right.*
- *The new plugin will be shown in the* ***plugins table****.*

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 based on a Java class packaged in a JAR.

- *The files to be imported are stored at **/home/openkm/import/Output** on the OpenKM server. (Variable **SYSTEM_FOLDER** + Output).*
- *The XML files with metadata values are stored at **/home/openkm/import/logfile** on the OpenKM server. (Variable **SYSTEM_FOLDER** + logfile).*
- *The files are imported into the OpenKM folder "**/okm:root/import**". (Variable **OPENKM_FOLDER**).*

The crontab performs three actions:

- *First, import all files from "**/home/openkm/import/Output**" and delete them from the server after they have been 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();
            }
        }
    }

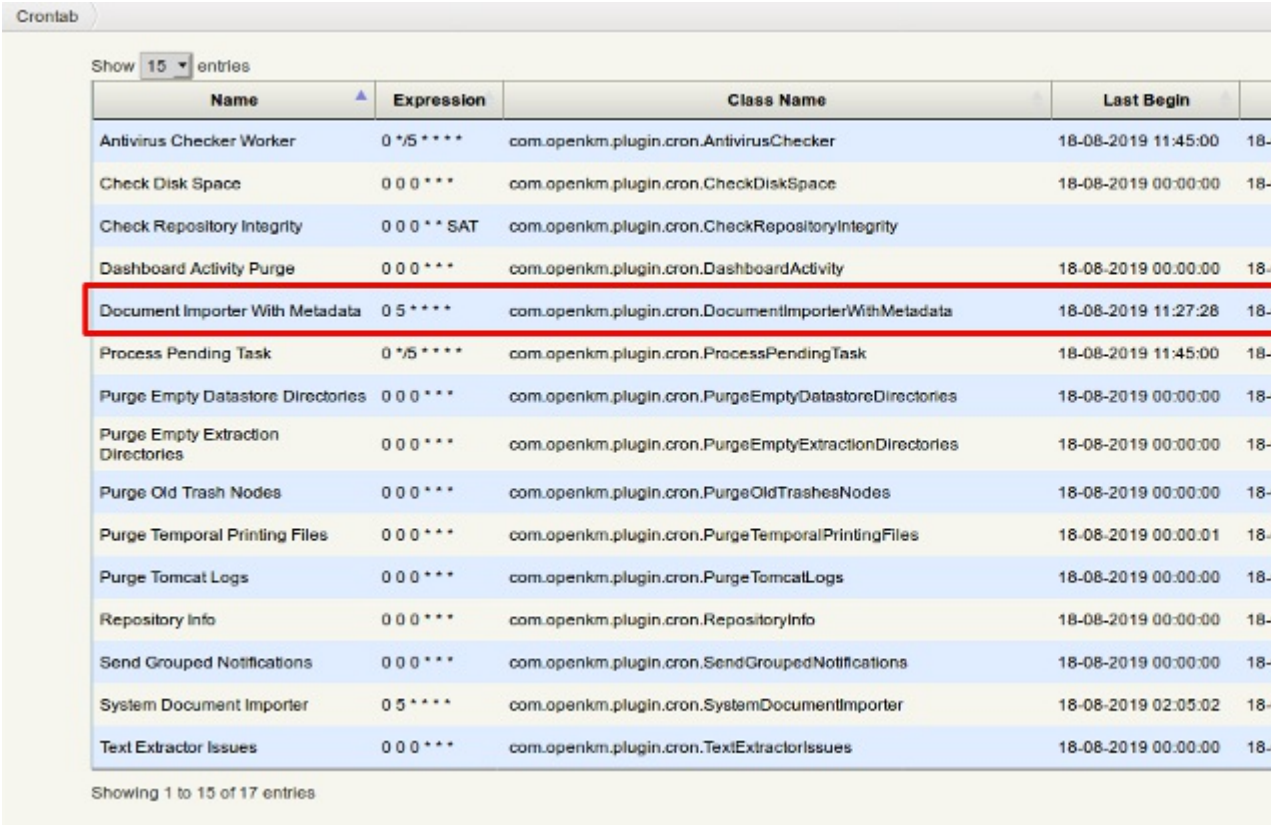
    /**
     * importFiles
     */
    public void importFiles() throws UnsupportedMimeTypeException, VirusDetectedException, ItemExistsException,
            PathNotFoundException, AccessDeniedException, RepositoryException, DatabaseException, AutomationEx
            PromotedAsRecordLockException, PrincipalAdapterException, 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 the*  *Reload plugins button at the top right.*
- *In* ***Plugins****, the new plugin will be shown in the table.*

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-

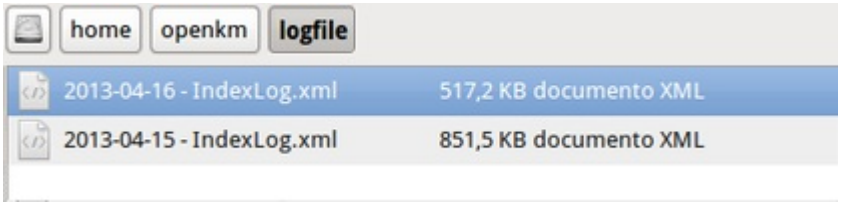
Showing 1 to 15 of 17 entries

Register metadata definition

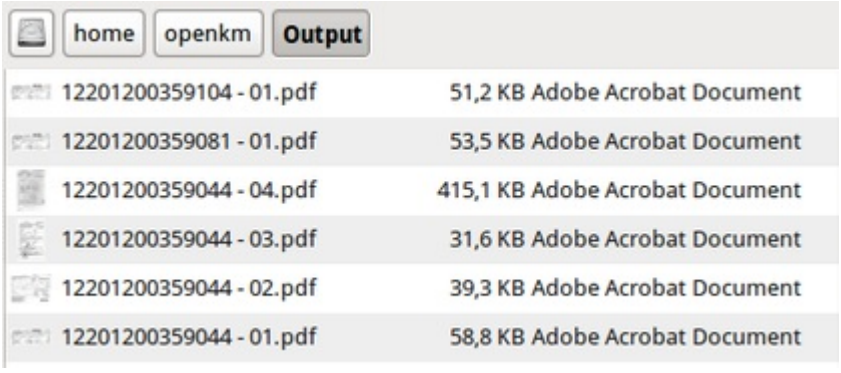
Metadata contracts

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

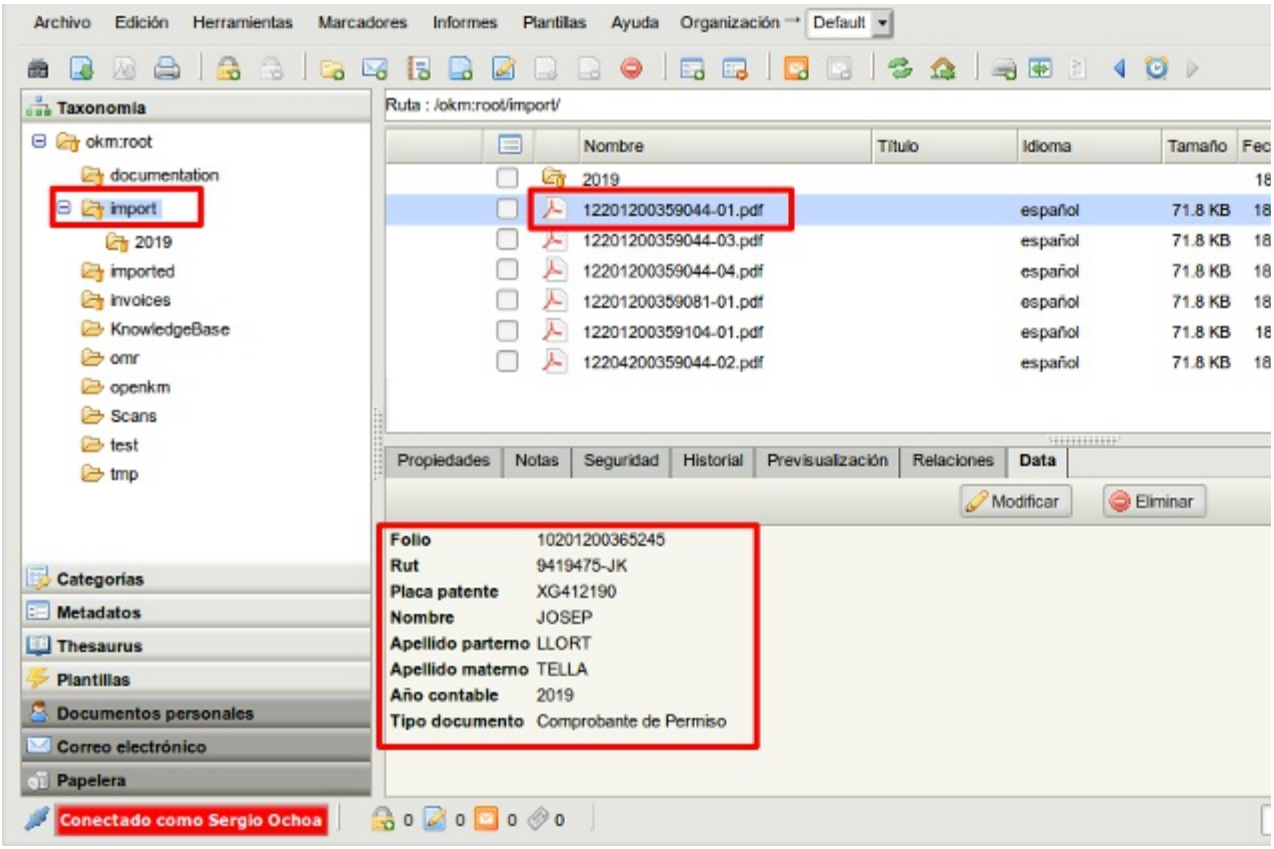


Output folder



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 (before) or post (after). For example, a ***pre*** *document creation event will be executed before document creation, and several event variables can then be changed to modify the parent folder, document name, and other properties. In the case of a* ***post*** *document creation event, it will be executed after document creation. In this case, there are no associated variables for the event; you'll get a NodeDocument object to apply any changes directly via the 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 extend from NodeBase.*	*Node base*
AutomationUtils.NODE_BASE_LIST	*List<NodeBase>* • *NodeDocument* • *NodeFolder* • *NodeMail* • *NodeRecord* *These nodes extend 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 received 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.ALL_PROPERTY_GROUPS	*List<PropertyGroup> propertyGroups*	*List of property groups (metadata groups)*
AutomationUtils.RECORD_NODE	*NodeRecord*	*Record node*
AutomationUtils.TEXT_EXTRACTED	*String*	*Extracted document text*
AutomationUtils.TEXT_EXTRACTOR	*String*	*TextExtractor object*
AutomationUtils.LANGUAGE_DETECTED	*String*	*Language detected*
AutomationUtils.UPLOAD_RESPONSE	*String*	*Upload response*

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

AutomationUtils.TITLE	*String*	*Title*
AutomationUtils.WIKIPAGE	*wikiPage*	*The wiki page*
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*

At Pre	
AutomationUtils.KEYWORDS	*Yes*
AutomationUtils.NOTES	*Yes*
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*

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

At post *At Pre*	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*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*

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

Purge Document

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***

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

AutomationUtils.DOCUMENT_NODE	*No*
At Pre	

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*

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

At Pre
At post

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*

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

AutomationUtils.DOCUMENT_VERSION ***At Pre***	*No*

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*

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

Virus Detected Document

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*

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

AutomationUtils.NODE_CLASS	*No*
At Pre	

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

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

The event is only triggered on the root node. If the node has subnodes, an event will not be generated for them.

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 for them.

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Move Folder

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

The event is only triggered on the root node and the destination node. If the node has subnodes, an event will not be generated for them.

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

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*

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***

At Pre *AutomationUtils.CATEGORIES*	*Yes*
AutomationUtils.CREATION_DATE	*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*

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

At post *At Pre*	
Variable	***For changing***
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 for 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

At Pre ***Variable***	***For changing***
AutomationUtils.MAIL_NODE	*No*

Download Mail from UI for preview

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*Yes*

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*

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

At post *At Pre*	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
AutomationUtils.NODE_CLASS	*No*

Import Mail Attachments

At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Delete Mail



The event is only triggered on the root node. If the node has subnodes, an event will not be generated for them.

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Purge Mail

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

The event is only triggered on the root node. If the node has subnodes, an event will not be generated for them.

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Record

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***

At Pre	
AutomationUtils.RECORD_NODE	*No*

Delete Record

> *The event is only triggered on the root node. If the node has subnodes, an event will not be generated for them.*

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
At post	
Variable	***For changing***
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 for them.*

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
At post	
Variable	***For changing***

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

AutomationUtils.RECORD_NODE ***At Pre***	*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 for them.*

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	
Variable	***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*

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

At Pre
At post

Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
AutomationUtils.NODE_CLASS	*No*

Property Group (metadata)

Add 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*
AutomationUtils.PROPERTY_GROUP_NAME	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*No*

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

Get all property groups

At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.ALL_PROPERTY_GROUPS	*Yes*

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*
AutomationUtils.PROPERTY_GROUP_NAME	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*No*

Remove property group

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

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	
Variable	***For changing***
AutomationUtils.USER	*No*

User Login

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

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***
AutomationUtils.TASK_OBJECT	*No*

Update Task

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

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

Create Note

At Pre	
Variable	***For changing***

At Pre	
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*
AutomationUtils.NOTE_TEXT	*No*
AutomationUtils.NOTE_UUID	*No*

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

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*
AutomationUtils.TEXT_EXTRACTOR	*Yes*

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

At post	
At Pre	
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 email 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 the REST Web Services API.*
Creating users from scripts	*Creation of users from generated script files.*

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 email addresses. Therefore, there's an export file of the global address list from our IT as a .csv containing all the Exchange Legacy DNs and their corresponding email addresses. With this data, the database can be altered so the Exchange Legacy DNs can be changed to normal email 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 is available on 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, it is good practice to configure the user profile to disallow all the features. That will lock UI access for the user.*

Available parameters:

- *The parameter* ***uuid*** *is the document UUID.*
- *The parameter* ***inline*** *is set when the document must be downloaded as an attachment or shown inline. It's a boolean parameter; allowed values are true or false.*

> *When you use the parameter path, it is a good practice to encode the URL to prevent problems with special characters.*

Sample:

```
http://user:password@@localhost:8080/openkm/download?uuid=def34bcc-3291-4e11-9aa3-e86ae32f5a6c
```

This eases 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 a document with a Bash script using REST

Simplified version of the script which makes use of the 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 the file create-users_okm.xlsx

*The string "**newId**" is the **user ID** that will be set.*

*The string "**newPassword**" is the **user password** that will be set.*

*The string "**newMail@mail.com**" is the **user mail** that will be set.*

*The string "**newRealName**" is the **user real name** that 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 activities and actions*
OKMAuth	*Manage security and users. For example add or remove grants on a node, create or modify users or get the profiles.*
OKMBookmark	*Manage the user bookmarks.*
OKMDashboard	*Manage all data shown on the 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 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 a 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 to the repository. For example get the properties of the main root node (/okm:root).*
OKMSearch	*Manage the search feature. For example manage saved queries or perform a new query on the repository.*
OKMStats	*General stats of the repository.*
OKMTask	*Manage tasks. 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 OpenKM's activity log data.

Methods

findByFilter

Description:

Method	Return values	Description
findByFilter(String token, ActivityFilter filter)	**List<Activity>**	*Returns a list of all activity log entries.*

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 entries.*

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 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 change or get security grants.*



To set READ and WRITE access, you should do:

```
int permission = Permission.READ + Permission.WRITE;
```

To check if you have permission access, 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, or 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 the "**null**" value, which indicates that the application must use the "**user session**".*



*In special cases, you might be "**promoted to 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 the user UI login process.*

> *When a user logs in from the UI, some background process creates main user nodes, like /okm:trash. Unfortunately, if the user has never logged in from the UI and logs in from the API, these nodes are still not created and will raise an error, because it is necessary to execute the login method first.*

> *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 tokens by default have an expiration time of 24 hours.*
>
> *After executing the login method, all subsequent calls will automatically use the authentication token.*
>
> *Each time the login method is executed, the login token is replaced with a new one.*

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

When a user logs in to the application for the first time, a method is called internally that creates user-specific folders, like /okm:trash/userId, etc.

From the API point of view, this method should only be executed the first time a user accesses the API, to create the user-specific 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 makes 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 makes 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>***	*Returns 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();
        }
```

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

```
    }
}
```

getGrantedRoles

Description:

Method	*Return values*	*Description*
getGrantedRoles(String token, String nodeId)	***Map<String, Integer>***	*Returns 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>***	*Returns the list of all 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>***	*Returns the list of all 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>***	*Returns the list of all 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>***	*Returns the list of all 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:

```
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 {
```

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);
            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>***	*Returns the list of all profiles.*

When the parameter filterByActive is enabled, the method will return only the active profiles; otherwise, it will return all available profiles.

Each user is 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***	*Returns the profile assigned to a user.*

> *Each user is 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 profile assigned to a user.*

Each user is 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.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*	*Executes 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***	*Returns 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 makes 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 makes 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>***	*Returns 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 tenant assigned to a user.*

A user can have access to several tenants but only has 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*	*Checks if the user's 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 clicks 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 for users. For example, add or remove a bookmark on a node for a user.

Basics

*In most 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 -> "**3887ae75-dd67-4019-b4d1-2f6962815403**";*

*Also, in all methods you'll see a parameter named "**token**". Because Cron tasks are executed in the background without authentication, the methods used in this scenario might use the token parameter. In the default application execution context you must use the "**null**" value, which indicates that the application should use the "**user session**".*

*In special cases you might be "**promoted to 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 for a node.*
The parameter name indicates the name that will be used to store the bookmark. Usually the same name as the nodeId is used.		

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***	*Deletes 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***	*Renames 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-in 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

*In all methods you'll see a parameter named "**token**". Because 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 the value "**null**", which indicates to the application that it must use the "**user session**".*

*In special cases you might be "**promoted to 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 checked out by the user.*

The parameters "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" indicates how many results to skip before 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 documents most recently modified by the user.*

The parameters "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" indicates how many results to skip before 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 documents locked by the user.*

The parameters "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" indicates how many results to skip before 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 records locked by the user.*

The parameters "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" indicates how many results to skip before 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();
        }
    }
}
```

getUserLockedFolders

Description:

Method	*Return values*	*Description*
getUserLockedFolders(int offset, int limit)	***DashboardResultSet***	*Returns a list of the folders locked by the user.*

The parameters "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" indicates how many results to skip before 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 parameters "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" indicates how many results to skip before 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 documents subscribed to by the user.*

The parameters "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" indicates how many results to skip before 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 folders subscribed to by the user.*

The parameters "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" indicates how many results to skip before 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 records subscribed to by the user.*

The parameters "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" indicates how many results to skip before 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 documents most recently created by the user.*

The parameters "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" indicates how many results to skip before 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 documents/notes most recently created by the user.*

The parameters "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" indicates how many results to skip before 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 folders most recently created by the user.*

The parameters "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" indicates how many results to skip before 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 folders/notes most recently created by the user.*

The parameters "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" indicates how many results to skip before 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 records most recently created by the user.*

The parameters "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" indicates how many results to skip before 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 records/notes most recently created by the*

		user.

The parameters "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" indicates how many results to skip before 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, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of documents most recently downloaded by the user.*

The parameters "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" indicates how many results to skip before 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 mails most recently imported by the user.*

The parameters "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" indicates how many results to skip before 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 mails/notes most recently imported by the user.*

The parameters "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" indicates how many results to skip before 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 mail attachments most recently imported by the user.*

The parameters "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" indicates how many results to skip before 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

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

Description:

Method	*Return values*	*Description*
getUserDocumentsSize(String token)	***long***	*Returns the total size in bytes of the user's documents.*

When a user uploads or modifies a document, the entire size of the content â€” including historical versions in case of updates â€” is assigned to the user.

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 to get 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 "user 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 parameters "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" indicates how many results to skip before 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.getLastWeekTopDownloadedL
```

```
                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>***	*Gets the list of the top modified documents from the last month by all users.*

The parameters "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" specifies how many results to skip before starting 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 to 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>***	*Gets the list of the most recently modified documents by all users.*

The parameters "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" specifies how many results to skip before starting 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 to 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>***	*Gets the list of the most recently created documents by all users.*

The parameters "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" specifies how many results to skip before starting 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 to 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>***	*Gets the list of the most recently checked-out documents by all users.*

The parameters "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" specifies how many results to skip before starting 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 to 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;
```

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

```
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>***	*Gets the list of the most recently created folders by all users.*

The parameters "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" specifies how many results to skip before starting 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 to 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>***	*Gets the list of the most recently created records by all users.*

The parameters "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" specifies how many results to skip before starting 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 to 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;
```

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

```
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***	*Marks a node as visited.*

> *From the UI, the nodes are shown in bold until they are visited by the user. This method marks 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*

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

- *UserSubscribedRecords*
- *UserLastCreatedDocuments*
- *UserLastCreatedFolders*
- *UserLastCreatedRecords*
- *UserLastImportedMails*
- *UserLastImportedMailAttachments*
- *In the 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

*In most methods you'll see a parameter named "**docId**". The value of this parameter can be a 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 a 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 a valid folder **UUID or** record node.*

Example of dstId:

- *Using UUID -> "**c4455667-0d6c-45da-1234-d98564323e0**";*

*Also, in all methods you'll see a parameter named "**token**". Because 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 the "**null**" value, which indicates the application must use the "**user session**".*

*In special cases you might be "**promoted to 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 in the repository. Optionally, you can also set keywords		

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

and title values.



The other parameters are not taken into 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 in 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***	*Returns a document's 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***	*Returns the document content.*
When the parameter checkout is true, the document is marked for editing.		

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.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***	*Returns the document content for a specific version.*
The parameter versionId 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 a document is uploaded to OpenKM, it goes into the search engine queue to be processed. Until the process is finished, this method will return an empty string.

From the Database query view, you can execute a query to verify whether a document has already been processed:

```
SELECT NDC_TEXT_EXTRACTED FROM OKM_NODE_DOCUMENT where NBS_UUID='1048eb46-2f32-4011-90e4-e084d838a592'
```

The NBS_UUID value is the document UUID. You can get it from the properties tab in the 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

Description:

Method	*Return values*	*Description*

getChildren(String token, String fldId)	***List<Document>***	*Returns a list of all documents whose 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 non-null and non-empty variables will be taken into 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 639-1 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 editing.*

Only one user can modify a document at a time.

Before starting editing you must perform a checkout action that locks the editing process for other users and allows editing only to the user who 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 document editing.*



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 document editing.*

This method allows cancelling editing 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 superuser (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 whether the document is being edited 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 editing - 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 editing - 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 permanently removed from the repository.*

Usually you will purge documents into /okm:trash/userId - the personal trash user location - but it is possible to directly purge any document from the whole repository.

> *When a document is purged it will only be able to be restored from a previous repository backup. The purge action removes the document permanently 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 a document into a folder or record.*
The value 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 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 the 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 is null, the document will preserve 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 are copied; 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 in the document's 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>***	*Returns a list of available document languages that 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 a template and returns a Document object.*

The dstPath can be a folder or a record UUID.

The parameter language is optional.

When the template uses metadata groups to fill fields, then these values are mandatory and must be set in the 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.*

This method only makes 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***	*Sets annotations for a 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***	*Returns the document annotations for a 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>***	*Returns the list of all checked-out documents 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***	*Stamps a document.*

The parameter type indicates stamp type:

- *"text" with value 0;*
- *"image" with value 1;*

The parameter stId is the unique identifier 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
```

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

```
            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 a document with custom text.*

*The **id** is the id of an image stamp created by 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 enter 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 on 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 a document with a custom image.*

*The **id** is the id of an image stamp created by 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 enter 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 on 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();
        }
    }
```

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

```
}
```

createWizard

Description:

Method	*Return values*	*Description*
createWizard(String token, String docPath, long nodeClass, InputStream is)	***WizardNode***	*Create a new document with a wizard.*

The parameter ***docPath*** *should be a valid folder or record* ***PATH***.

> *The WizardNode contains a list of pending actions that should be done to complete the document creation process. 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 in 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 a specific page as an image encoded as a Base64 string.*

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 whether the node is convertible to PDF.*



*In the case of a node of type **PDF file**, a **false** value will be returned.*

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 whether the document is 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 parameter ***destinationPath*** *should be a 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();
        }
    }
}
```

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

OKMFolder

Basics

*In almost all methods you'll see a parameter named "**fldId**". The value of this parameter can be a valid folder **UUID**.*

Example of fldId:

- *Using UUID -> "**b67fe3ae-bcca-4947-b8ea-0cc7aedd4474**";*

*Also, in all methods you'll see a parameter named "**token**". Because Cron tasks are executed in the background without authentication, methods used in this scenario might use the token parameter. From the default application execution context you must use the "**null**" value, which indicates the application must use the "**user session**".*

*In special cases you might be "**promoted to 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 a Folder object.*

*The variable **path** in the parameter **fld** must be initialized. It indicates the folder path in OpenKM.*

```
Folder fld = new Folder();
fld.setPath("/okm:root/test");
```

The other variables of the Folder (fld) will not have any effect on the folder creation.

We suggest using the method below to create the Folder rather than this one.

Example:

```
package com.openkm;
```

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

```
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 a Folder object.*

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***	*Returns 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***	*Deletes 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 permanently removed from the repository.*

Usually you will purge folders in /okm:trash/userId - the personal trash user locations - but it is possible to directly purge any folder from the repository.



When a folder is purged, it can only be restored from a previous repository backup. The purge action removes the folder permanently 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***	*Renames 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 a folder into another 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 another folder or record.*
The values of the dstId parameter should be a folder or record UUID.		

Only the security grants are copied to the destination; the metadata, keywords, etc. of the folder are not copied.

*See "**extendedCopy**" method for this feature.*

Example:

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

```
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 the newName parameter value is null, the folder will preserve 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 another 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 are copied; the metadata, keywords, etc. of the folder are not copied.

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 the newName parameter value is null, the folder will preserve the same name.

By default only the binary data and the security grants are copied; the metadata, keywords, etc. of the folder are not copied.

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 whose 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***	*Returns a ContentInfo object with information about the folder.*

The ContentInfo object retrives information about:

- *The number of folders in the folder.*
- *The number of documents in the folder.*
- *The number of records in the folder.*
- *The number of mails in the folder.*
- *The size in bytes of all objects in 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 whether the node is 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);
            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***	*Converts a folder UUID to a 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***	*Sets 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***	*Creates missing folders based on all the path nodes.*

> *The method checks all subfolders and when it detects that 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***	*Adds 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***	*Sets 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

*In most methods you'll see a parameter named "**parentUuid**". The value of this parameter can be a valid document **UUID** node.*

Example of docId:

- *Using UUID -> "**c41f9ea0-0d6c-45da-bae4-d72b66f42d0f**"*

*Also, in all methods you'll see a parameter named "**token**". Because the Cron task is 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 the "**null**" value, which indicates that the application must use the "**user session**".*

*In special cases you might be "**promoted to 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;

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();
        }
    }
}
```

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:

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

```
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 a local folder hierarchy.*

*The parameter **parentUuid** must be a **valid folder or record UUID**.*

*The value **threads** sets the **number of concurrent threads** used by the import feature. This value should **never exceed the number of server cores**. If you have 20 cores, a good starting point might be to use 10 threads; take into consideration that other software, like the database and the OS, 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();
        }
    }
}
```

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

OKMMail

Basics

*In almost all methods you'll see a parameter named "**mailId**". The value of this parameter can be a valid mail **UUID**.*

Example of mailId:

- *Using UUID -> "**0747c0cb-5e4a-4088-92a9-ab97b401af75**";*

*Regarding the parameter named "**fldId**", the value of this parameter can be a valid folder **UUID** or record node.*

Example of fldId:

- *Using UUID -> "**b67fe3ae-bcca-4947-b8ea-0cc7aedd4474**";*

*Also, in all methods you'll see a parameter named "**token**". Because the Cron Task is 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 the "**null**" value, which indicates that the application must use the "**user session**".*

*In special cases you might be "**promoted to 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 an attachment to 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 an attachment from 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 for 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 permanently removed from the repository.

Usually you will purge mails into /okm:trash/userId - the personal trash user location - 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 previous repository backup. The purge action permanently removes the mail 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 a 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 the newName parameter value is null, the mail will preserve 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 are copied; the metadata, keywords, etc. of the mail are not copied.

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 the newName parameter value is null, the mail will preserve the same name.

By default only the binary data and the security grants are copied; the metadata, keywords, etc. of the mail are not copied.

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 whose parent is fldId.*
The parameter fldId 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 indicating whether 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***	*Converts a folder UUID to a 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***	*Sends a mail message with attachment.*

The values of the dstId parameter should be a folder or record ***UUID****. The dstId parameter indicates where the mail will be stored in the repository after it is sent.*

Other parameters:

- *toRecipients is a list of destination mail accounts.*
- *subject is the mail subject.*
- *ccRecipients and bccRecipients are lists of destination mail accounts (optional)*
- *replyToMails is a list of mail accounts for replying (optional)*
- *docsId is a list of valid document UUIDs already in OpenKM that will be sent as attachments in the 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 fldId parameter indicates where the mail will be stored in the repository after it 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 fldId parameter indicates where the mail will be stored in the repository after it 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*
getContent(String token, String mailId)	***InputStream***	*Retrieves mail content (binary data).*

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
```

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

```
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 that 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 a String with the extracted text from the text extractor process.*

When a document is uploaded to OpenKM, it goes into the text extraction queue. There's a crontab job that periodically processes this queue and extracts document contents.



Unfortunately there is no direct way to know if a document has been processed or not from the API, because this information is only stored at the database level.

Despite this restriction, only administrator users can perform database queries from the API to retrieve this information. Check Repository samples for accessing database-level information.



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***	*Forwards an email to a list of users, roles, or external email addresses.*

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

*In almost all methods you'll see a parameter named "**nodeId**". The value of this parameter can be a valid node **UUID**.*

Example of docId:

- *Using UUID -> "**c41f9ea0-0d6c-45da-bae4-d72b66f42d0f**";*

*About the parameter named "**fldId**", the value of this parameter can be a valid folder **UUID or** a record node.*

Example of fldId:

- *Using UUID -> "**c6785440-0d6c-45da-1234-d9874563d0f**";*

*About the parameter named "**dstId**", the value of this parameter can be a valid folder **UUID or** a record node.*

Example of dstId:

- *Using UUID -> "**c4455667-0d6c-45da-1234-d98564323e0**";*

*Also, in all methods you'll see a parameter named "**token**". Because 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 the "**null**" value, which indicates that the application must use the "**user session**".*

*In special cases you might be "**promoted to 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:

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

```
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 cannot be promoted to the "record" status, the result of the evaluation contains information about the reason why it cannot 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
```

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

```
            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 can only 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 in an electronic record.*



Returns 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 node's 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 child nodes, paginated.*

*The parameters "**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" specifies how many results to skip before beginning to return results.*

*The parameter "**orderByField**" 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 child nodes by category, paginated.*

*The parameters "**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" specifies how many results to skip before beginning to return results.*

*The parameter "**orderByField**" 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 the 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**	*Deletes 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 a 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);
```

```
        } 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>***	*Returns 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 whether a ZIP file can be downloaded in real time or in the 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 the trash, the application keeps the original location. When possible, the application will try to restore it to the same original location (sometimes this is not possible because the original no longer exists or security settings have changed, 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)	***boolean***	*Check if the node and its descendants have been locked by other users.*

This method is used to evaluate if it is possible to lock a hierarchy. For example, in the case of a record, the lock affects all nodes within it.

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.

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

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 locking any already locked node.



This action can only be done by users with administrator privileges.

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 that the same user who previously executed the checkout lock action perform this action.



This action can only be done by a user with administrator privileges.

Example:

```
package com.openkm;
```

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

```
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

*In 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 a nodeId:

- *Using UUID -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**"*

*Also, in all methods you'll see a parameter named "**token**". Because 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 the "**null**" value, which indicates 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

add

Description:

Method	*Return values*	*Description*
add(String token, String nodeId, String text)	***Note***	*Adds a note to a node and returns a Note object.*

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 a UUID.*
>
> *The Node object has a variable named path; in that case, the path contains a 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***	*Deletes a note.*

> *The noteId is a UUID.*
>
> *The Node object has a variable named path; in that case, the path contains a 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's text.*

> *The noteId is a UUID.*
>
> *The Node object has a variable named path; in that case, the path contains a 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 for 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>***	*Returns 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

*In 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**"*

*Also, in all methods you'll see a parameter named "**token**". Because 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 the "**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

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***	*Deletes 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 subscribers 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***	*Sends a mail notification.*

The parameter users is a set of OpenKM users to be notified.

The parameter mails is a set of mails - usually external mails - to be notified.

The parameter message is the body of the mail.

When the attachment value is true, the node is attached to 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();
        }
    }
}
```

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

OKMProperty

Basics

*In 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**";*

*About the parameter named "**catId**": the value of this parameter can be the UUID of a categories folder node.*

Example of nodeId:

- *Using UUID -> "**5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f**";*

*In all methods you'll see a parameter named "**token**". Because 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 the "**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

addCategory

Description:

Method	*Return values*	*Description*
addCategory(String token, String nodeId, String catId)	***void***	*Sets a relation between a category and a node.*
The value of the catId parameter should be a category folder UUID.		

Example:

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

```
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***	*Adds a keyword to a node.*

The keyword should be a single word without spaces, formats allowed:

- *"test"*
- *"two_words" (the character "_" is used as the separator).*



We also suggest that you add keywords in lowercase, 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 encrypted binary data in the repository.*

The parameter nodeId should be a document node.

The parameter cipherName saves information about the encryption mechanism.



This method does not perform any kind of encryption; it simply marks 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 as normal binary data in the*

nodeId)		*repository.*

The parameter nodeId should be a document node.



This method does not perform any kind of encryption; it simply marks in the database that a document is unencrypted.

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 in the repository.*

The parameter nodeId should be a document node.



This method does not perform any kind of digital signature process; it simply marks in 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 whether 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

*In older OpenKM versions we called "**Metadata Groups**" "**Property Groups**".*

Although we understand this name does not help much to identify these methods as metadata ones, for historical reasons, we continue to maintain the nomenclature.

For more information about Metadata.

*In 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. Metadata field values of type date are stored in the application in ISO-8601 basic format.

To convert the retrieved metadata 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, in all methods you'll see a parameter named "**token**". Because Cron Tasks are executed in the background without authentication, the methods used in this scenario might use the token parameter. In the default application execution context you must use the "**null**" value, which indicates that the application must use the "**user session**".*

*In special cases you might be "**promoted to 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 a metadata group and its values to a node.*

The grpName should be a valid Metadata group name.

It is not mandatory to set all field values in the properties parameter; it is enough to include the fields whose values you wish to change.

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 to 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 the character "**,**" is used as the 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	***void***	*Removes a metadata group from a*

grpName)		*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

Description:

Method	*Return values*	*Description*
getAllGroups(String token)	***List<PropertyGroup>***	*Retrieves a list of all metadata groups in 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();
        }
    }
}
```

getAllGroups

Description:

Method	*Return values*	*Description*
getAllGroups(String token, String nodeId)	***List<PropertyGroup>***	*Retrieves a list of all metadata groups in the application.*



Available since version 8.1.19.

This method allows, through the "Get all metadata groups" automation task event, filtering the list of groups allowed for a specific 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.getAllGroups(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338")) {
                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 in 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,	***Map<String,***	*Retrieves a list of all metadata group elements and*

String grpName)	***String>***	*their node values.*

The grpName should be a valid Metadata group name.

> *The method is usually used to display form elements with their values for viewing or modification by users.*

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, Map<String, String> properties)	***void***	*Changes the metadata group values of a node.*

The grpName should be a valid Metadata group name.

> *It is not mandatory to set all field values in the properties parameter; it is enough to include the fields whose values you wish to change.*

> *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 to 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 the character "**,**" is used as the 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;
```

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

```
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 element definitions.*

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:

```
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 indicating whether the node has 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

Description:

Method	*Return values*	*Description*
getSuggestions(String token, String nodeId, String grpName, String propName)	***List<String>***	*Retrieves a list of suggested metadata field values.*

> *The propName parameter should be of the 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***	*Sets the XML Metadata groups definition into the repository.*



The method can only be executed by a superuser (a user with the ROLE_ADMIN role).

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 Metadata groups definition.*



The method can only be executed by a superuser (a user with the ROLE_ADMIN role).

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 group 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, String versionName)	***Map<String, String>***	*Retrieves a map (key, value pairs) of 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, String key)	***KeyValue***	*Retrieves an object that contains the key and value of the suggest box.*

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 contain the key and value from the suggest box, filtered by value.*

Example:

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

```
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

*In most methods you'll see a parameter named "**recId**". The value of this parameter can be a valid record **UUID**.*

Example of recId:

- *Using UUID -> "**7f867772-7476-4176-b86f-bdeb8bce1c5f**";*

*Also, in all methods you'll see a parameter named "**token**". Because 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 the "**null**" value, which indicates 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

create

Description:

Method	*Return values*	*Description*
create(String token, Record rec)	***Record***	*Creates a new record and returns a Record object.*

*The variable **path** in the parameter **rec** must be initialized. It indicates the folder path in OpenKM.*

```
Record rec = new Record();
rec.setPath("/okm:root/PKI-100200");
```

The other variables of the Record (rec) will not affect the record creation.

We suggest using the method below to create the record simply rather than this one.

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 = 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***	*Returns 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***	*Deletes 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 permanently removed from the repository.*

Usually you will purge folders in /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 previous repository backup. The purge action removes the record permanently 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***	*Renames 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 a record into a folder or another record.*
The value 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 the destination; the metadata, keywords, etc., of the document are not copied.

*See the "**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 the newName parameter value is null, the record will preserve the same name.



Only the binary data and the security grants are copied to the destination; the metadata, keywords, etc., of the document are not copied.

*See the "**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 value of the dstId parameter should be a folder or 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 are copied; 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 values of the notes will be copied.*
- *When the 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 whose 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 returns an object with the lock information.*



Only the user who locked the record is allowed to unlock.

A locked record cannot 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***	*Unlocks 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***	*Unlocks a locked record.*

This method allows unlocking any locked record.

It is not necessary that the same user who previously executed the checkout lock action perform this action.



This action can only be done by a superuser (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's 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***	*Sets 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***	*Sets 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***	*Adds the NodeClass of the record to the parent folders.*
Used for navigation purposes to evaluate when a folder contains a node of this node class. *The method propagates the node class value to 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***	*Sets 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***	*Creates a new record with a wizard.*

*The parameter **recPath** should be any valid folder or record **UUID***

> *The WizardNode contains a list of pending actions that should be done to complete the process of record creation. These might be:*
>
> - *Add a 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

*In 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 -> "**d168924d-e801-426d-a222-00124a1461bd**";*

*Also, in all methods you'll see a parameter named "**token**". Because 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 the value "**null**", which indicates that the application should use the "**user session**".*

*In special cases you might be "**promoted to 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 for a type.*

Available types:

- *NodeRelationType.BIDIRECTIONAL*
- *NodeRelationType.PARENT_CHILD*
- *NodeRelationType.MANY_TO_MANY*

More information at Relation types.

Example:

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

```
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.*



A relation can only be deleted when it is not used by any node; 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 for 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 for 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 to a node.*

It only makes sense to apply the relation type NodeRelationType.MANY_TO_MANY to a relation group.



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.*

It only makes sense to apply the relation type NodeRelationType.MANY_TO_MANY to a relation group.



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***	*Removes 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 report formats based on document MIME type:

Constants defined in the 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 the **active** variable is set to **true**, only the active reports are returned; otherwise all reports are returned.*		

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 a Report object.*
*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:

```
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 a parameter named "**token**". Because the Cron Task is 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 the "**null**" value which indicates the application must use the "**user session**".*

*In special cases you might be "**promoted to Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

getRootFolder

Description:

Method	Return values	Description
getRootFolder(String token)	**Folder**	Returns the Folder object for 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 Folder object for node "/okm:trash/{userId}".

The returned folder will be the user's 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 Folder object for 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 Folder object for 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 Folder object for node "/okm:personal/{userId}".*

The returned folder will be the user's 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 Folder object for 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 Folder object for node "/okm:mail/{userId}".*

The returned folder will be the user's 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 Folder object for 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 Folder object for 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 Folder object for 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***	*Permanently removes from the repository all nodes in "/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 can only be restored from a previously made repository backup. The purge action permanently removes the node 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 is 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's 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 value that indicates whether a node exists.*
The value of the nodeId parameter 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 a 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);
```

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

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodeUuid

Description:

Method	*Return values*	*Description*
getNodeUuid(String token, String path)	***String***	*Converts a node path to a 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 the 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();
        }
    }
}
```

copyAttributes

Description:

Method	*Return values*	*Description*
copyAttributes(String token, String srcId, String dstId, ExtendedAttributes extAttr)	***void***	*Copies attributes from one node to another.*

The value 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***	*Returns 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>***	*Returns the list of changes in a path and its subfolders.*

- *The method is used by the desktop synchronization application to retrieve 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***	*Returns 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

Almost all methods use QueryParams. Here are some tips about how to use 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.* *To search documents and folders, use the 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; do not use a 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.* *Can only be applied to document nodes.*
language		*No.*	*The value should be a valid language.* *Can only be applied to document nodes.*
folder		*No.*	*When empty, the "/okm:root" node is used by default.* *The value should be a valid UUID; do not use a path value.*
folderRecursive	***Boolean***	*No.*	*It only makes sense to set this variable to true when the folder variable is*
lastModifiedFrom	***Calendar***	*No.*	
lastModifiedTo	***Calendar***	*No.*	
mailSubject	***String***	*Yes.*	*Only applies 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 all.*	*Wildcards cannot be applied to metadata field values like "date".*
			
			The map of the properties is composed of pairs:
			
			('metadata_field_name','metadata_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 (f will be ignored by the OpenKM side.
			
			To filter 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 the character "**;**" is used as*



The search operation is performed using only AND logic.

Wildcard examples:

Variable	*Example*	*Description*
name	***test*.html***	*Any document that starts with the characters "test" and ends with the characters ".html".*
name	***test?.html***	*Any document that starts with the characters "test" followed by a single character and ends with the characters ".html".*
name	***?test****	*Any document 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 - the marshalling and unmarshalling process - when you are only interested in a few object variables. If that's your case you can use NodeProperties classes to retrieve only the object variables 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 parameters "**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**" is used to skip that many results before returning 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 - the marshalling and unmarshalling process - when you are only interested in a few object variables. If that's your case you can use NodeProperties classes to retrieve only the object variables 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 of the metadata group.*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - the marshalling and unmarshalling process - when you are only interested in a few object variables. If that's your case you can use NodeProperties classes to retrieve only the object variables 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 parameters "**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**" 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 - the marshalling and unmarshalling process - when you are only interested in a few object variables. If that's your case you can use NodeProperties classes to retrieve only the object variables 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" filters the field name by the word "grial".*

More information about Lucene syntax 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" filters the field name by the word "grial".*

More information about Lucene syntax 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 parameters "**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**" 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 - the marshalling and unmarshalling process - when you are only interested in a few object variables. If that's

your case you can use NodeProperties classes to retrieve only the object variables 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 parameters "**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**" 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 - the marshalling and unmarshalling process - when you are only interested in a few object variables. If that's your case you can use NodeProperties classes to retrieve only the object variables you really need.

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" filters the field name by the word "grial".*

More information about Lucene syntax 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" filters the field name by the word "grial".*

More information about Lucene syntax 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 search parameters.*



Only a saved search created by the same user who's executing the method can be updated.

```
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 a saved search created by the same user who's executing the method can be updated.

```
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 searches.



Only a saved search created by the same user who's executing the method can be updated.

```
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 a saved search created by the same user who's executing the method can be updated.*

```
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".*

Results filtered by "test" -> "one", "two", "three".

Results filtered by "one" -> "test", "two".

Results filtered by "two" -> "test", "one".

Results filtered by "three" -> "test".

Results filtered 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 to a category.*

The value 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 value 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 value 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 value 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 to 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 to 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 to 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*

The send data value format must be:

```
ISO8601.formatBasic(from)+","+ISO8601.formatBasic(to)
```

Allowed sortField values are:

- *"okm:" for no sorting.*
- *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 parameters "**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**" is used to skip that many results before the start of the returned 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 to 20, you should use these values:

- *limit=10*
- *offset=10*

```
package com.openkm;
```

```
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***	*Exports a list of results as a CSV file filtered by the values of the queryParams parameter.*

*The parameter **lang** must be ISO 639-1 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***	*Returns the calculated stamp coordinates.*

In the Expr. X and Expr. Y input fields you can enter 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 on 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 alignment*

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***	*Returns 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***	*Stamps the document.*

*The parameter **type**.*

- *Value 0 for text type*
- *Value 1 for image type*

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();
        }
    }
}
```

stampTextCustom

Description:

Method	*Return values*	*Description*
stampTextCustom(String token, String docId, long stId, String exprX, String exprY, String range, String text)	***void***	*Stamp a document with custom text.*

*The **id** is the id of an image stamp which is created by 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***	*Returns 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***	*Returns 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***	*Returns 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***	*Returns 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.*

> *Returns 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***	*Returns the processed personal seal.*

> *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 a parameter named "**token**". Because 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 the "**null**" value, which indicates the application should use the "**user session**".*

*In special cases you might be "**promoted to Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

getDocumentsByContext

Description:

Method	*Return values*	*Description*
getDocumentsByContext(String token)	***StatsInfo***	*Returns 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:

Return

Method	values	Description
getDocumentsSizeByContext(String token)	**StatsInfo**	Returns stats information about document sizes 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**	Returns 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

Description:

Method	*Return values*	*Description*
getMailsByContext(String token)	***StatsInfo***	*Returns 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***	*Returns 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.

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 end.*	*No.*
Status	***TaskStatus***	*The task status.* *You administer the list of status values.*	*Yes.*
Project	***TaskProject***	*The project related to the task.* *You administer the list of project values.*	*Yes.*

Type	***TaskType***	*The task type.* *You administer the list of task type values.*	*Yes.*
Progress	***Integer***	*The numeric progress status.* *The range of allowed values is from 0 to 100, where 100% indicates a completed task.*	*Yes.*
RepeatGroup	***Long***	*When a task is repeated over time, it is a member of a group. The group is identified by a unique repeat group id.*	*No.*
ReminderStartValue	***Integer***	*How many days before the task starts, the system might send an email notification to the user.*	*No.*
ReminderEndValue	***Integer***	*How many days before the task ends, the system might send an email notification to the user.*	*No.*
ReminderStartUnit	***String***	*Reminder start units.* *Allowed values:* • *"m" for minutes.* • *"h" for hours.* • *"d" for days.*	*Mandatory when **ReminderStartValue** is greater than 0.*
ReminderEndUnit	***String***	*Reminder end units.* *Allowed values:* • *"m" for minutes.* • *"h" for hours.* • *"d" for days.*	*Mandatory when **ReminderEndValue** is greater than 0.*

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

Users	***Set<String>***	*Collection of users assigned to the task.*	*No. But at least one user or role should be assigned.*
Roles	***Set<String>***	*Collection of assigned roles to the task.*	*No. But at least one user or role should be assigned.*
Documents	***Set<String>***	*List of UUIDs of the related documents.*	*No.*
Folders	***Set<String>***	*List of UUIDs of the related folders.*	*No.*
Mails	***Set<String>***	*List of UUIDs of the related emails.*	*No.*
Records	***Set<String>***	*List of UUIDs of the related records.*	*No.*

*In all methods, there is a parameter named "**token**". When accessing the application via REST, the login process returns a token. The token is used to identify the user for all exposed methods. From the default application execution context you must use the "**null**" value, which indicates the application must use the "**user session**".*

*In special cases, you might be "**promoted to 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 tasks related to 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 tasks related to 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 parameter must be a valid type id.*
- *The statusId parameter must be a valid status id.*
- *The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly 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 parameter must be a valid type id.*
- *The statusId parameter must be a valid status id.*
- *The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameters "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" specifies the number of results to skip before 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 to 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 parameter must be a valid type id.*
- *The statusId parameter 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 parameter of the TaskManagerTask class. Commonly 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 parameter must be a valid type id.*
- *The statusId parameter must be a valid status id.*
- *The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameters "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" specifies the number of results to skip before 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 to 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 parameter must be a valid type id.*
- *The statusId parameter 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 parameter of the TaskManagerTask class. Commonly 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 parameter must be a valid type id.*
- *The statusId parameter must be a valid status id.*
- *The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameters "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" specifies the number of results to skip before 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 to 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 statuses can be ending statuses. For example, statuses named "closed" or "cancelled". The boolean parameter "finish" indicates if your status is 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 statuses can be ending statuses. For example, statuses named "closed" or "cancelled". The boolean parameter "finish" indicates if your status is 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 whether 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 whether 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 the 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 a 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 history 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 is the topic of the task.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameter must be a valid type id.*
- *The statusId parameter must be a valid status id.*
- *The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameters "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" specifies how many results to skip before returning 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 is the topic of the task.*
- *The progress parameter is the progress of the task.*
- *The statusId parameter must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameter 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 is 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 parameter must be a valid type id.*
- *The statusId parameters must be a valid status id.*
- *The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameters "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 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 for a user.*

Filter parameters description:

- *The subject parameter is the topic of the task.*
- *The progress parameter is the progress of the task.*
- *The statusId parameter must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameter 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 is 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 parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.*
- *The orderAsc parameter orders ascending or descending.*

The parameters "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 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 is the topic of the task.*
- *The progress parameter is the progress of the task.*
- *The statusId parameter must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameter 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 parameter 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 parameter must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameter 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 parameter must be a valid type id.*
- *The statusId parameter must be a valid status id.*
- *The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly 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 for a user.*

Filter parameters description:

- *The statusId parameter must be a valid status id.*
- *The projectId parameter must be a valid project id.*
- *The typeId parameter 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 a parameter named "**token**". Because 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 the value "**null**", which indicates that the application must use the "**user session**".*

*In special cases you might be "**promoted to 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's home.*

By default, the 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:

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

Method	*Return values*	*Description*
getConfig(String token)	***UserConfig***	*Returns 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

*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. In the default application execution context you must use the "**null**" value, which indicates that the application must use the "**user session**".*

*In special cases you might be "**promoted to 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 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 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 an image of a workflow process definition.*
The parameter processDefinitionId value is a valid workflow process definition ID. *The parameter node is a node that has 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 of all workflow form definitions.*
The parameter processDefinitionId value is a valid workflow process definition ID.		

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 ID.		

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 to start the workflow (not all workflows need form values to start).

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();
        }
    }
}
```

runProcessDefinition (without node association)

Description:

Method	*Return values*	*Description*
runProcessDefinition(String token, long processDefinitionId, List<FormElement> variables)	***ProcessInstance***	*Executes a workflow without associating it to a specific node.*

The parameter processDefinitionId value is a valid workflow process definition.

The parameter values are form element values needed to start the workflow (not all workflows need form values to start).

> *This method starts a workflow without associating it to any document, mail, folder or record. Use this when you need to run a standalone workflow process.*

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.bean.workflow.ProcessInstance;
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 where form parameters are required to start workflow
            Input priority = new Input();
            priority.setName("priority");
            priority.setValue("high");
            feList.add(priority);
            TextArea comments = new TextArea();
            comments.setName("comments");
            comments.setValue("This is a standalone workflow process");
            feList.add(comments);
            ProcessInstance pi = okmWorkflow.runProcessDefinition(null, pdId, feList);
            System.out.println("Process instance created: " + pi);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendProcessInstanceSignal

Description:

Method	*Return values*	*Description*
sendProcessInstanceSignal(String token, long processInstanceId, String transitionName)	***ProcessInstance***	*Sends a signal to a process instance.*

The parameter processInstanceId value is a valid workflow process instance ID.

The parameter uuid can be any document, mail, folder or record UUID.

The parameter transitionName is the chosen transition.

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 instance.*

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***	*Deletes a process instance.*
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 for some registered workflow definitions.*
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 workflow 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 latest workflow 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 the last workflow definition of a specific workflow.*

The parameter name identifies a specific workflow definition group.

There are 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***	*Resumes 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 parameters "limit" and "offset" allow you to retrieve only 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 returning 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 parameters "limit" and "offset" allow you to retrieve only 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 returning 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 for a 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.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***	*Sets a task instance's values.*
The parameter taskInstanceId is a valid task instance ID. *The parameter transitionName is the chosen transition.* *The parameter values are form element values needed to start 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***	*Sets the actor for a task instance.*
The parameter taskInstanceId is a valid task instance ID. *The parameter actorId must be a valid OpenKM user 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.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***	*Returns 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***	*Sends a signal.*
The parameter tokenId is a valid token ID. *The parameter transitionName is a transition 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();
        }
    }
}
```

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

Common classes

OpenKM common 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 a JAR.*
FileLogger	*Utility methods to generate a log. For example, to show error, info, or warning messages.*
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 values.*
GeneralUtils	*Generic utility methods. For example, to notify a user about something.*
ISO8601	*Utility class that provides bidirectional format transformation between Java time objects and the ISO8601 format used for metadata date fields.*
MailUtils	*Utility methods to send an email. For example, to send a message or a document.*
PathUtils	*Utility methods to handle paths. For example, to get the name or the parent path.*
PDFUtils	*Utility methods to convert PDF files. For example, to stamp images or text.*
PrincipalUtils	*Utility methods for retrieving information from the security context. For example, to get the actual user ID from the security context session or to get the actual tenant of the logged-in user.*
StackTraceUtils	*Utility methods for stack traces.*
TemplateUtils	*Utility methods to handle templates. For example, to replace 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 distribute a repository across several disks.*
LegacySrv	*Utility methods to perform SQL and HQL queries.*
DbSessionManager	*Utility methods for managing user sessions.*
ConfigSrv	*Utility methods to perform configuration parameter operations.*

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

ArchiveUtils

Utility methods for archive handling. For example, creating a ZIP or JAR.

Methods

createZip

Description:

Method	*Return values*	*Description*
createZip(File path, OutputStream os)	***void***	*Create a ZIP archive from a file.*
***path**: The file to compress.* ***os**: The output stream for 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	***void***	*Recursively create a JAR archive from a*

os)		*directory.*

***path**: The file to compress.*

***root**: Text specifying the destination of the file.*

***os**: The output stream for 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 a script from a 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 a JAR from a 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 a 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 logs. For example, to show errors, info, or warnings.

Methods

info

Description:

Method	*Return values*	*Description*
info(String baseName, String message, Object... params)	***void***	*Static file logger for info messages.*
***baseName**: The name of the file to write.* ***message**: A text 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 for warnings.*
***baseName**: The name of the file to write.* ***message**: A text message for warnings.*		

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

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 for errors.*
***baseName**: The name of the file to write.* ***message**: A text message for errors.*		

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 file name or extension.

Methods

getFileName

Description:

Method	*Return values*	*Description*
getFileName(String file)	***String***	*Returns the name of the file without its 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 file name 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 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 a 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 that 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*	*Returns a date from an ISO8601 basic format string.*
***value**: An ISO8601 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*	*Returns an ISO8601 basic format string from a 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 a user.

Methods

uiNotify

Description:

Method	*Return values*	*Description*
uiNotify(String msg)	***void***	*Notify the currently authenticated user.*
***msg**: A text with the notification message.*		



The user must be logged in 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 mail. For example, to send a message or send a document.

Methods

sendMessage

Description:

Method	*Return values*	*Description*
sendMessage(String toAddress, String subject, String content)	***void***	*Send mail without a FROM address.*
***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 a FROM address.*

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

fromAddress: *The from address 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 documents to non-registered OpenKM users.*

fromAddress: *The from address 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 sent.*

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 the parent path.

Methods

getParent

Description:

Method	Return values	Description
getParent(String path)	***String***	*Gets the 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***	*Gets the node name.*
***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***	*Gets the 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 images 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 input streams for merging.* ***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 a PDF document with an image watermark.*

***input**: The input stream of the file.*

***image**: The content of the image 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 a PDF document with a text watermark.*

***input**: The input stream of the file.*

***text:** 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***	*Returns the number of pages in a PDF.*

***input**: The input stream of the file.*

***text:** 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-in user.

Methods

getUser

Description:

Method	*Return values*	*Description*
getUser()	***String***	*Get the userId of the logged-in user.*



The user must be logged in 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-in user.*



The user must be logged in 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**	*Returns 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**	*Returns the 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) {
```

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

```
    }
        } catch (Exception e) {
    System.out.println(StackTraceUtilsUtils.toString(e));
   }
  }
 }
```

TemplateUtils

Utility methods to handle templates. For example, replace 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 with the template definition.* ***model****: A map used to replace 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 for a node.*
uuid *The 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 for distributing a repository across 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 a 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 the 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 paths 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 the 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 an SQL script.*

```
SELECT NBS_UUID, NBS_AUTHOR, NBS_NAME, NBS_NOTES FROM OKM_NODE_BASE LIMIT 10;
```



The parameter named sql allows only a single SQL script.

This action can only be performed 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 an HQL script.*

```
SELECT uuid, name from NodeBase where name = 'okm:root';
```



The parameter named hql allows only a single HQL script.

This action can only be performed 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**	Gets the authorization token of the system user.

> *There's a special super-user named 'system' that is executed by OpenKM in the background.*
>
> *The authorization token of the system user is usually used to execute standard users' 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***	*Finds the config parameter (as a 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***	*Finds the config parameter (as a 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***	*Finds the config parameter (as a 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***	*Finds the config parameter (as a integer 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.getInteger("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLong

Description:

Method	*Return values*	*Description*
getLong(String key, String defaultValue)	***long***	*Finds the config parameter (as a 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***	*Finds the config parameter (as a 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***	*Finds the config parameter (as a 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>***	*Finds the config parameter (as a 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 {

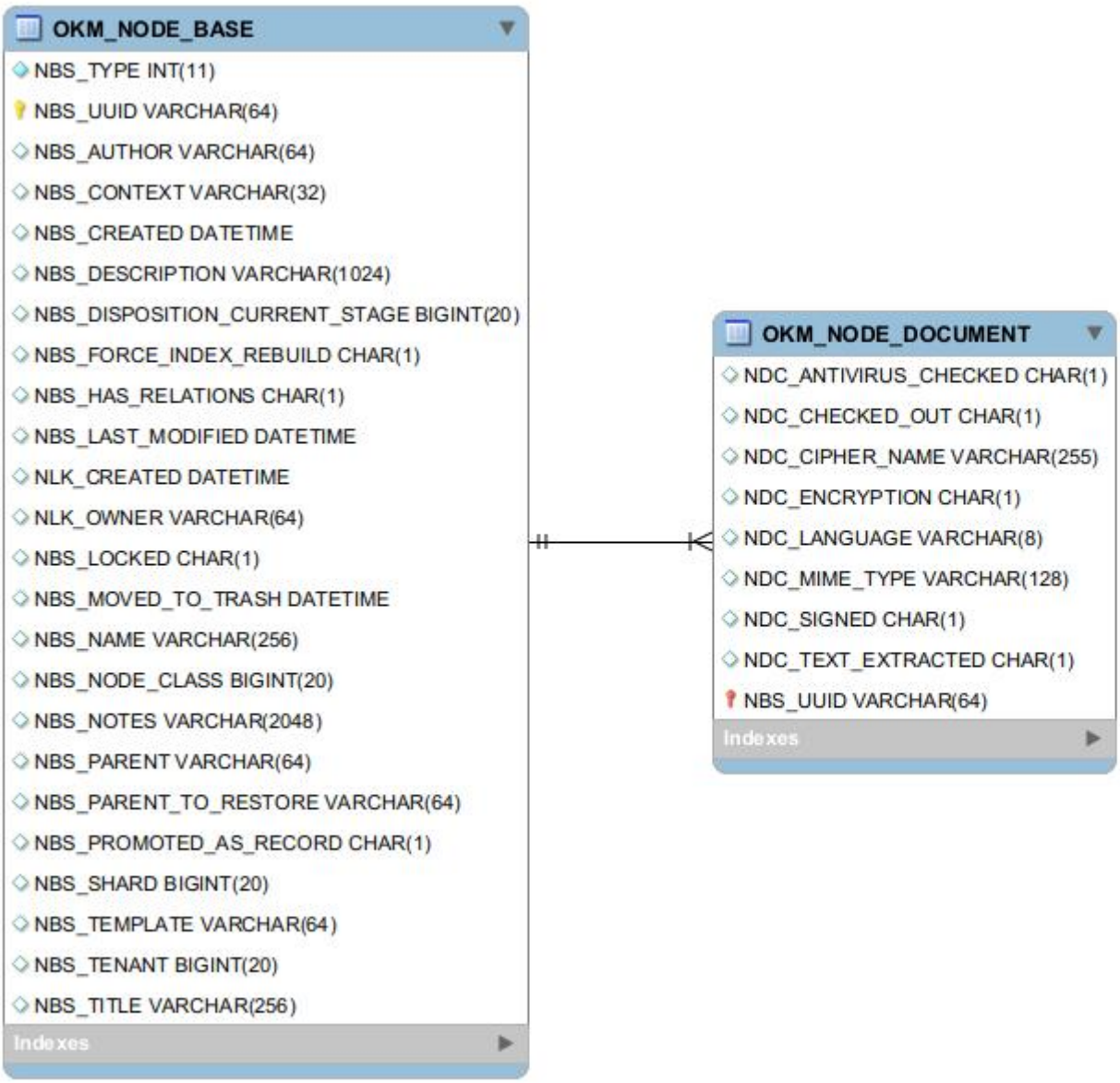
    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            for (String listElement : configSrv.getList("paramKey", "defaultValue")) {
                System.out.println(listElement);
            }
        } 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 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 the parent table (OKM_NODE_BASE) and children** (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**: Stores information about documents.*
- ***OKM_NODE_FOLDER**: Stores information about folders.*
- ***OKM_NODE_MAIL**: Stores information about mail.*
- ***OKM_NODE_RECORD**: Stores 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 whose 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, you can 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

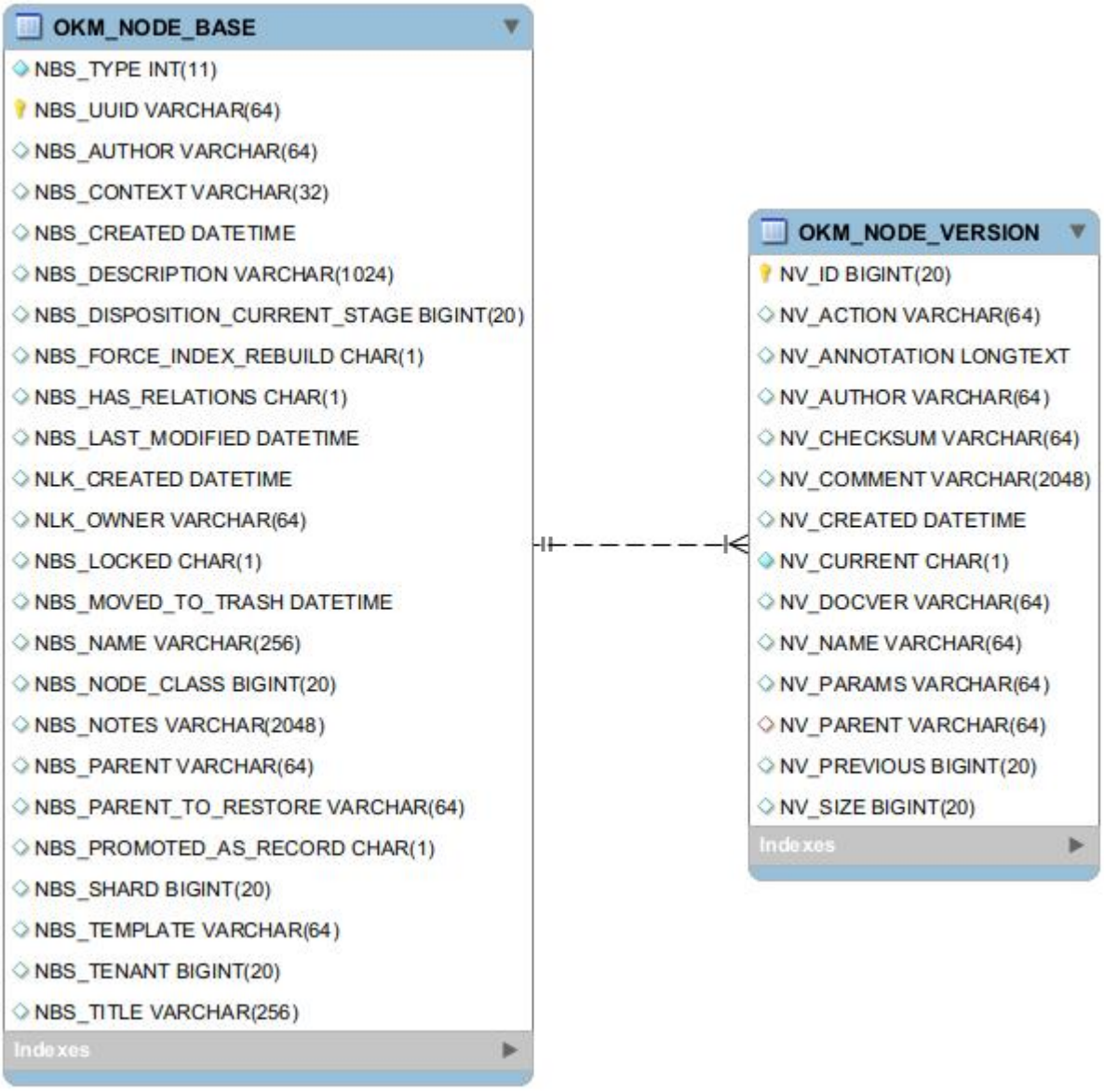
The column named NDC_TEXT_EXTRACTED in the table OKM_NODE_DOCUMENT indicates which documents have been processed by the text extraction queue and which are still pending (the value 'T' means processed and 'F' means pending). If you want to reindex the whole repository or some files, you might execute 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 find where all versions of a file named 'test.pdf' are stored in the datastore:



Only applicable to the default datastore configuration (in case all 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
```

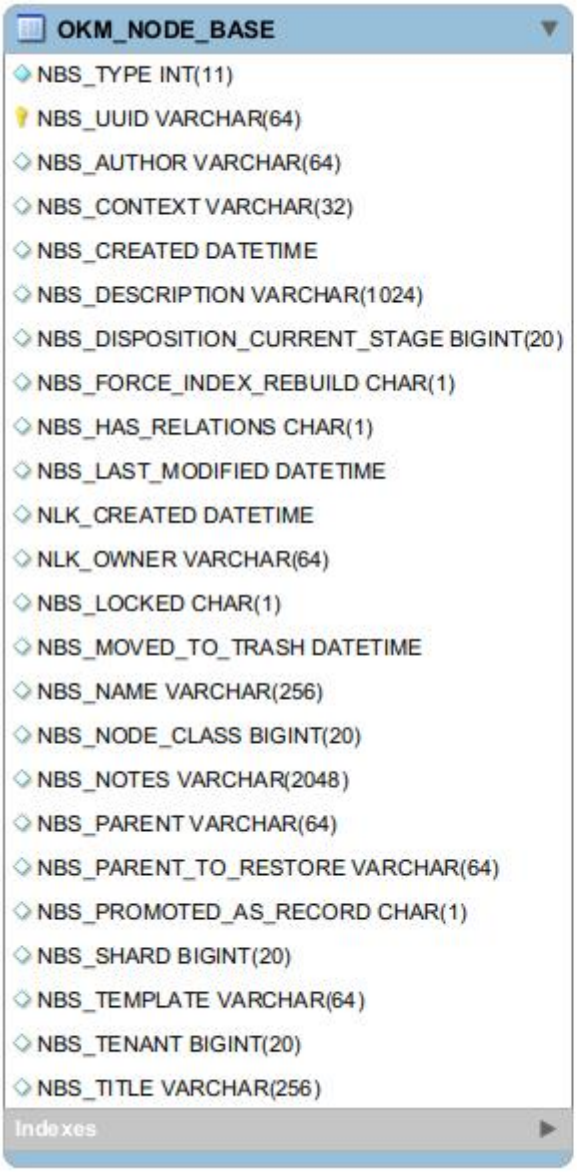
```
inner join OKM_NODE_BASE nb on nd.nbs_uuid = nb.nbs_uuid
where nb.nbs_name = 'test.pdf'
```

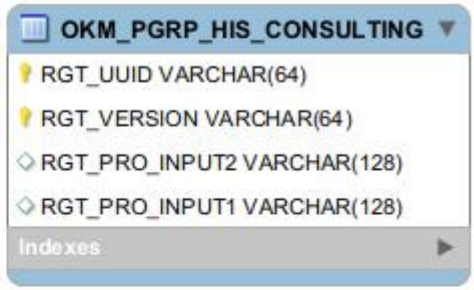
*For example '**f0d7f0a1**-b337-4917-b592-56e163b81cde'. This means your file is located on the physical drive at the path (based on the first eight characters of this ID):*

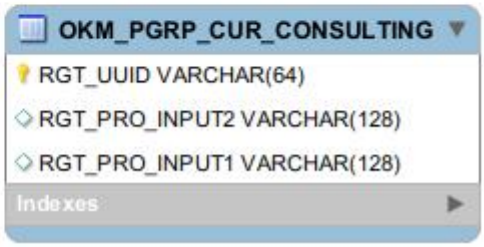
*/home/user/tomcat_home/repository/datastore/**f0/d7/f0/a1**/f0d7f0a1-b337-4917-b592-56e163b81cde*

Node metadata properties

Each metadata group uses two tables to store information; take the following 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 metadata definition above, two tables will be created 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
```



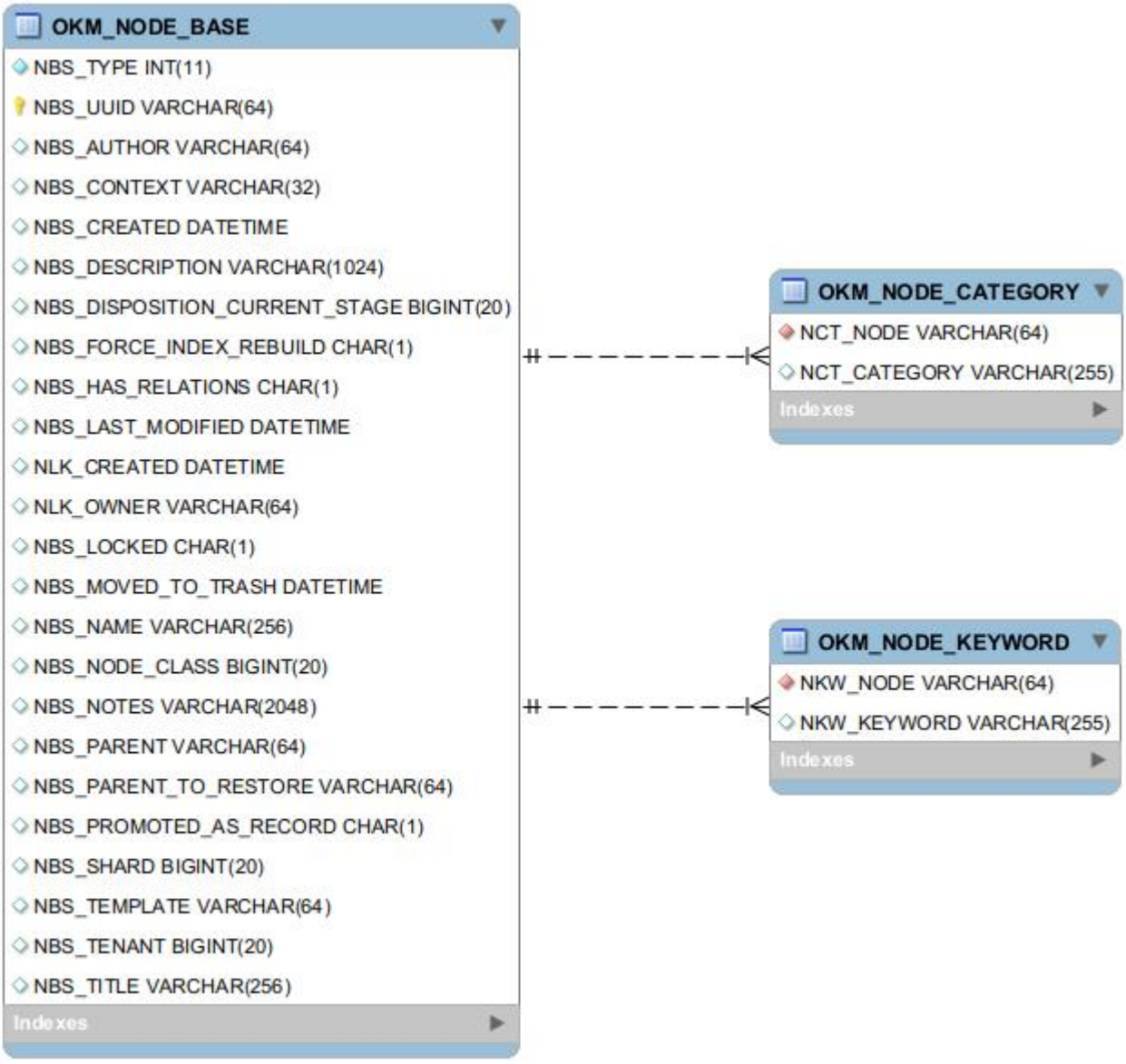
This query will print all metadata values for each version. Usually, the current version of these metadata values is the one with the largest RGT_VERSION value.

Query to get changes of a document named 'test.pdf' when its metadata properties were 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

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



*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. There is a video on YouTube that shows 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 a unique document name based on metadata	Create a unique name and a unique ID field based on metadata.
Tab UI content extension	Demonstrates how to add content to the KCenter UI using the tab extensions.
Tab UI content extension using KCenter methods	Demonstrates how to access KCenter methods from an external application using 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 on a JSP

Step 1 - Create a sample JSP file

Create the following sample JSP file and copy it to the webapps/openkm folder named ***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.*
- *A new menu option named "**Extra**" is shown in the **contextual menu** or in the **toolbar menu***

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 on 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 web services 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 has 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 value*** *or a* ***promise****, like the example 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 has the okg:consulting*** *metadata assigned.*

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

Custom UI template

Developer information for the 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 on Spring Boot at https://spring.io/guides/gs/spring-boot/

> *Please create a support ticket to ask for this source code.*

General description

Properties file

*You must create a file called **custom.properties** inside the **$CATALINA_HOME** folder.*

> *This file is located in the root folder of the project. However, you can also download it 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 access OpenKM web services if no user is provided.*
anonymous.password	*Anonymous user password*
admin.user	*Admin user name* > *Used to access OpenKM web services if no user is provided.*

admin.password	*Admin user password*
logback.file	*Logback file name*



Other properties in this file should not be modified unless you know what you are doing. More information about properties is available 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.

The OpenKM instance URL is defined in the ***openkm.url*** *configuration property.*

Log file

You must copy the file ***logback-custom.xml*** *to the* ***$CATALINA_HOME*** *folder. This file is used by the logging system.*



This file is located in the root folder of the project. However, you can download it from here: logback-custom.xml.

To change the 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 an H2 database to store remember-me data. A database file called ***custom.mv.db*** *will be created in the user's home directory.*

There is a file called ***import.sql*** *which contains SQL statements that are executed when the application starts. In this particular case, it only contains one statement to create the database table to allow the "remember me" feature.*

Roles management

In the file ***menu.jsp*** *you can see how the roles are managed by the application:*

```
<sec:authorize access="hasRole('CATALOG')">
```

Using this code, a user may or may not see this menu option, depending on their roles. However, if you do not want to allow a user to access a resource, you can configure it in the ***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 to provide a beautiful and responsive design. More information about Bootstrap is available at http://getbootstrap.com/.

Packaging

*This application can be packaged as either a **jar** or a **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 package the application as a jar, you can 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 can upload files to a folder **custom.pending.folder** (we will see what this folder is)*
- *An automation 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 will move this document to another folder*

The main classes of the catalog system are:

- ***CatalogController**: application routing for the catalog system*
- ***CatalogServiceImpl**: services for the catalog system*
- ***WSCacheDAOImpl**: manages the web services to communicate between 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 the OpenKM Download Center Download center.

> *This will download a zip file containing 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 it to the user used by the application.*

See the Manage roles section for more information.

Step 3: SQL data needed

In order to execute SQL statements, 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 statements could be different depending on your database.*

*The parameter **custom.pending.folder** contains the UUID of the folder from which to read documents for the catalog. You also need 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 types are supported: Input, TextArea, Select and Checkbox.

Step 5: Automations

You must create two automations like the following (see the Automation section for more information):

Automation create
Go to OpenKM administration, select the Automation option and New Rule, fill in the name ***"Move Consulting"****, select the eve* ***group"****, and check the 'Active' box.*
Go to OpenKM administration, select the Automation option and New Rule, fill in the name ***"Move Marketing"****, select the ever* ***group"****, and check the 'Active' box.*

Step 6: Catalog documents from UI

- *Set the URL in the browser to 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 a 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 that you configure Lombok (see https://projectlombok.org/setup/eclipse)

*This application can be packaged as a **jar** or **war**. In this POM configuration the artifact will be a WAR.*

If you need to change the 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 the "Sample templates" metadata group in the profiles.*

Step 2 - Upload template file

You can also download the template.html

- *Create a file named template.html with the following 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 the metadata group "Sample template" to the document "**/okm:templates/template.html**"*

Step 3 - Deploy the application



*There's a file named **application.properties** in the project where the OpenKM connection parameters are set.*

*By default, the application should be **run or deployed on the OpenKM server** and there should be a user named **okmAdmin** with password **admin** as 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** to your **$TOMCAT_HOME**/webapps*

Step 4 - Create a document from the UI

- *In the browser, set the URL to http://localhost:8080/forms/*
- *Fill in the fields shown on the screen and click the submit button*
- *Verify that a document has been created in OpenKM under **/okm:root** in the following format: **{creation_date}-template.html***

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

Security Provider

This section is deprecated as of version 8.1.12

Developer information for OpenKM authentication through a custom authentication provider (vía patch) and a custom Principal 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 **openkm.properties** file inside the **$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 to your Tomcat.*

Custom Principal Adapter

You must follow the steps below to configure this component correctly:

Step 1: Download

Download the application from OpenKM Download center.

This will download a zip file containing 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 through the OpenKM administration UI and search for this property ***principal.adapter*** *and modify its value to* ***com.openkm.plugin.principal.CustomPrincipalAdapter***

Step 3: Install the plugin

Once you build the Maven project (security-provider), you must copy ***security-provider-jar-with-dependencies.jar*** *into the path:* ***$CATALINA_HOME/plugins***

You can add or update users and roles by modifying the class ***com.openkm.security.utils.Users.java***

Authentication Provider

You must follow the steps below to configure this component correctly:

Step 1: Download

The application is the same one 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:* ***$CATALINA_HOME/webapps/openkm/WEB-INF/classes/com/openkm/security/provider***
- *Copy* ***com.openkm.security.utils.Users.class*** *from* ***security-provider.jar*** *into the path:* ***$CATALINA_HOME/webapps/openkm/WEB-INF/classes/com/openkm/security/utils***

If you added or updated users and roles, you also have to modify the class ***com.openkm.security.provider.TestCustomAuthProvider.java***

In addition, you have to configure a patch to preserve those classes across future OpenKM updates. To do this, create a folder called ***$TOMCAT_HOME/patch****. Create a 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 in 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 a unique document name based on metadata

Each time a new document is uploaded, the Automation action sets a unique document name based on metadata values.

- *Property **okp:data.id** stores a unique auto-incrementing 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 the metadata group is changed (event triggered), automation code is executed which generates **okp:data.id** and renames the document based on the mask **projectCode-autonumericId-clientCode-description.documentExtension**.*

The tables used in this sample are 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 an automation rule. Go to Administration > Automation and register a new rule:

- *Rule based on the "**add metadata**" event at the "**pre**" stage.*
- *Set the 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 the 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 on 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, you must configure Lombok (see https://projectlombok.org/setup/eclipse).

*This application can be packaged as either a **jar** or **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 instructions in the latest README.md in the demotab folder.

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

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 is at the External tab document extension

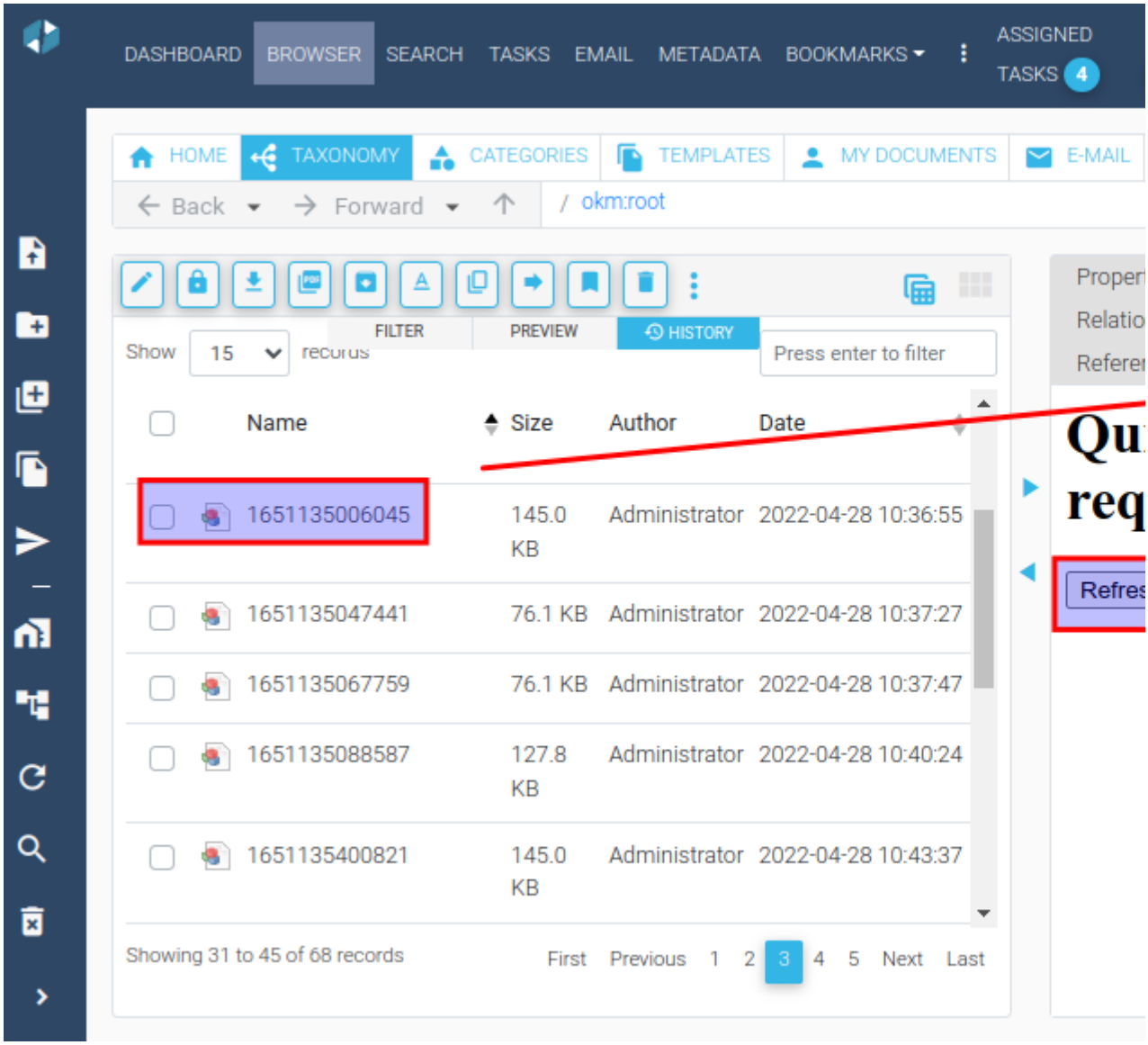
More information about enabling extensions is available at Enable extensions (remember to enable the extension in the profile and refresh the KCenter UI).

- *Configure the 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